Groupe de Recherche en Informatique,
Image, Automatique et Instrumentation de Caen
Group “Hultech”

Habilitation thesis

A contribution to Computational Linguistics and Natural Language Processing: From the Semantics of Space and Time to Annotations and Agreement Measures.

Yann Mathet

December the 5th, 2017

Committee:

Jean-Yves Antoine	Professor, Université François-Rabelais, Tours	reviewer
Delphine Battistelli	Professor, Université Paris X	reviewer
Massimo Poesio	Professor, University of Essex, UK	reviewer
Frédéric Landragin	Senior Scientist, CNRS, LATTICE, Montrouge	
Pierre Zweigenbaum	Senior Scientist, CNRS, Université Paris-Saclay, Orsay	
Marc Spaniol	Professor, Université de Caen Normandie	advisor

Yann Mathet, https://mathet.users.greyc.fr/
Group “Human Language Technology”, GREYC
Université de Caen Normandie
CNRS UMR 6072, ENSICAEN, France

Contents

Foreword 7

I Semantic Modeling of space and time 9

1 The semantics of space and motion 11

1.1 Topological models : strengths and limitations 11

1.2 An extended model of spatio-temporal semantics which considers paths as fundamental objects 12

2 The semantics of time and iteration 17

2.1 The Laurent Gosselin's theory of time 18

2.2 Definition and terminology of iteration 20

2.3 Mental spaces and Iterative models 23

2.4 Cloning and Projection 27

2.5 SdT intervals in mental spaces 29

2.6 An Object Oriented Iteration Model 30

2.7 Implementing Mental Spaces: links between model events. 39

2.8 Implementing the Iteration Model on simple cases 41

2.9 Selections: how to access to a given part of the iterates 48

2.10 Implicit Iterations 58

2.11 Framed Iteration 58

2.12 Conclusion 59

2.13 Perspectives 60

II Annotation of continuums and its assessment 65

3 Annotation of continuums with Glozz and GlozzQL 67

3.1 Introduction 67

3.2 Why another tool ? 69

3.3 Underlying model 70

3.4 Main features 74

3.5 Customizing and using Glozz 76

3.6 GlozzQL: a query language for annotation mining 78

3.7 Conclusion 81

4 Assessment of annotations - Part I:
Understanding and assessing agreement measures 83

4.1 Main concepts 84

4.2 Chance correction: a necessity and a difficulty . . . 87
4.3 Benchmarking and understanding agreement measures: the Corpus Shuffling Tool (CST) 92

5 Assessment of annotations - Part II:
The Gamma family of agreement measures for unitizing tasks 97
5.1 Introduction . . . 98
5.2 Motivations, scope and illustrations . . . 98
5.3 Introducing γ . . . 100
5.4 The implementation of γ . . . 108
5.5 benchmarking γ . . . 112
5.6 The additional coefficients γ_{cat} and γ_k . . . 112
5.7 Main requirements of a categorial measure . . . 113
5.8 How best to handle missing values ? . . . 114
5.9 The design of γ_{cat} . . . 116
5.10 The in-depth coefficient γ_k which focuses on each category . . . 120
5.11 Benchmarking γ_{cat} and γ_k . . . 120
5.12 Overview and dependencies of the gamma family . . . 121
5.13 Software . . . 122
5.14 Conclusion on the Gamma family . . . 123
5.15 Future work . . . 124

Bibliography 125

Foreword

Acknowledgments

I wish to thank Jean-Yves Antoine, Delphine Battistelli and Massimo Poesio for having agreed to review this work, and Frédéric Landragin, Marc Spaniol and Pierre Zweigenbaum for having accepted to be committee members. Marc was also my advisor, and I thank him for his confidence and his support.

This work was conducted in the GREYC laboratory, which provided me with optimal research conditions, and prolific colleagues. I have to stress a particular collaboration with Antoine Widlöcher I have worked with on many projects for years, as evidenced by the number of times our names jointly appear in our publications, but I have also to thank many other colleagues whether we officially conducted some research together or just have informal discussions. Besides, Antoine and I have both really appreciated the priceless assistance of two engineers, Jérôme Chauveau and Stéphane Bouvry.

In the recent years, I have also had the fabulous opportunity to discuss and then to collaborate with Klaus Krippendorff on questions related to agreement measures. I thank him so much for his vitality, his confidence, and the long series of e-mails we have exchanged over the years.

To finish, I have a special thought for Patrice Enjalbert I still collaborate with on some projects related to the semantics of time, but more importantly who was a primary factor for my taste for research.

Of course, I also thank my family and relatives for their great support, and in particular my wife Sophie, my parents and my parents-in-law, who provided me with a huge amount of time which was an essential requirement to write this dissertation.

Introduction

First of all, I would like to warn the reader that in this dissertation, I will use "I" rather than the royal "we" to refer to myself. This is quite unpleasant, but this is for the sake of clarity, in order to make obvious the distinction between what my personal contributions are, and what belongs to joint studies.

This study addresses two different questions in the fields of Computational Linguistics (CL) and Natural Language Processing (NLP): the question of how to model natural language semantics, especially in space and time paradigms, and the question of how to annotate corpora. These seemingly different questions are tied by the fact that when studying how to model some linguistic phenomena, for instance in semantics, it is necessary to get annotated data related to these phenomena, first to get inspiring examples of what is really studied, and second to assess our models by confronting their productions with reference annotations. Precisely, because of these ties, my research domain has progressively widened from pure semantics questions to questions about annotation.

In my PhD thesis, I addressed spatio-temporal semantics as it appears in natural language. Most available models rely on so-called topological relations, where the very questions is "in what place is X located?" These models fail to render the semantics of many expressions which cannot be described in terms of being located into a place, nor in terms of going into (or getting out of) a place. For instance, the sentence "(the road / the car) circumvents the city" involves a complex relationship between the shape of the road or of the trajectory of the car and the city (in addition to a topological relation of exteriority). I introduced the limits of these models and proposed solutions. Subsequently, my work has been focusing more and more on the semantics of time, in collaboration with other computer scientists, and also a linguist. In particular, we have addressed the question of how repetition (iterative events) is conveyed in natural language, in such examples as: "every Thursday, they played cards. The game lasted about 2 hours", and how to model it. One of the main results of this study is that natural language is able to handle an iterative event as if it were a sole generic event. This is clearly visible in the second sentence by the use of the singular "the game" which surprisingly refers to a plurality of games. We have designed a model which accounts for that, and for a wide range of related phenomena.

At the same time, several collaborations in CL and NLP research projects led us to focus more and more on annotation process. In particular, the ANNODIS project consisted in creating and providing a discourse relations corpus, and made appear the need for new methods and tools to annotate texts. Together with a colleague, Antoine Widlöcher, we designed and developed a versatile annotation platform, namely Glozz, which not only fulfills the ANNODIS requirements, but also fits a wide range of projects worldwide. Producing annotations brings another question: how to make sure that annotations are valid? Consequently, we have studied the existing methods to assess annotations, and we found that most of them do not fit CL nor NLP purposes. In particular, CL and NLP mainly refer to linguistic streams (texts, videos), whereas most used assessment methods concern sets of independent items. As a consequence, in many cases, scholars do not use relevant measures to assess their annotations, which leads to strong biases in the results. Here again, we have proposed solutions with a new set of agreement measures, namely the Gamma family. Besides, this work goes along with a more general reflection on the principles of assessment methods, which is an additional contribution.

Part I

Semantic Modeling of space and time

Chapter 1

The semantics of space and motion

Contents

1.1 Topological models : strengths and limitations 11

1.2 An extended model of spatio-temporal semantics which considers paths as fundamental objects 12

1.2.1 Spatio-temporal objects 13

1.2.2 Polymorphism: several points of view on a same thing 13

1.2.3 Main spatio-temporal relations 13

1.2.4 A short overview of the expressivity of the model through some examples . . 14

This chapter briefly recaps the main contributions of my PhD thesis (Mathet, 2000) which addressed the question of how to model space and motion as their are dealt in natural language, for both text understanding and automatic text generation.

1.1 Topological models : strengths and limitations

Most studies on space and motion in natural language rely on a model made of objects which have spatial boundaries, and which share relations based on their boundaries. For instance, "she is in Paris" corresponds to the fact that "she" is inside the boundaries of "Paris", and "the ball is on the table" to the fact that the "ball" and the "table" are externally connected. A well known and very illustrative such model is RCC8, which defines 8 possible relations between two continuous objects.

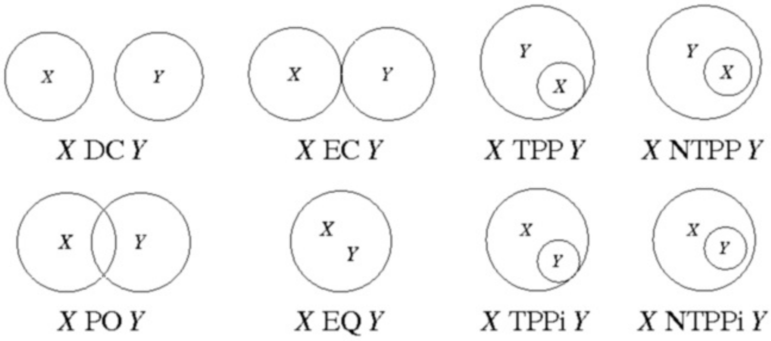


Figure 1.1: The 8 RCC (Region Connection Calculus) relations

This very basic set of relations was extended to cover proximity relations, such as "the car is near the house", with additional proximity boundaries. Hence, "near" is rendered by two relations: being out of the core region, while being into the proximity region.

Then, these static relations which initially render spatial states rather than events, are also used to render motion: A movement is described as a change of spatial relation between the moving object and some spatial reference. For instance, "he went to Paris" corresponds to a change from the relation "being out of Paris" to the relation "being in Paris".

Typically, the study by Laur (1991), relies on such kinds of relations.

Some years later, Sablayrolles (1995) proposed a model which combines proximity boundaries and change of relations to try to cover the whole semantics of motion in language, as shown in figure 1.2.

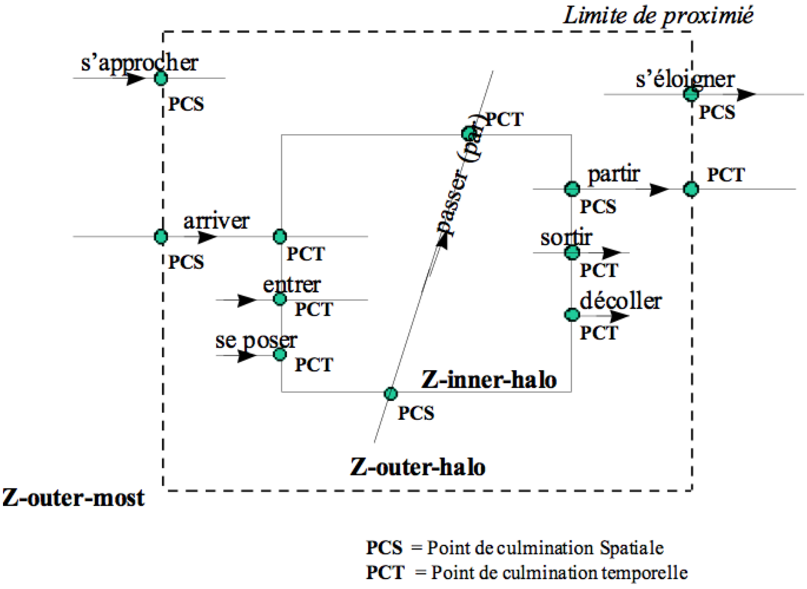


Figure 1.2: A topological model for motion by Pierre Sablayrolles

The theoretical motivation for using such kinds of models is that natural language does not convey precise spatial positions, for instance in terms of (x, y, z) coordinates, but mainly qualitative and under-specified relations between spatial objects. This assumption is partially true: most often natural language is under-specified, in the sense that nothing more than necessary is said. But it misses the capability of natural language to provide as detailed spatial descriptions as needed: "don't put anything closer than 12 inches to the heater to prevent from any risk of fire". More importantly, it misses a whole range of relations, possibly also under-specified but which do not rely on so-called topology. Examples are numerous, and the following three are very illustrative: "He is driving very fast" clearly involves a motion, but no change of relation ; "the road circumvents the city" and "the road doesn't go through the city" have different meanings but are the same in terms of topological relations ; proximity cannot be modeled through proximity regions, since it is very contextual: we can say "please get close, get closer" to anyone whatever how close she stands from us.

1.2 An extended model of spatio-temporal semantics which considers paths as fundamental objects

To overcome the mentioned limitations, my main contribution has been to propose a model where paths and trajectories are primary objects which share various kinds of relations (including but not limited to topology) with other spatial or spatio-temporal objects.

1.2.1 Spatio-temporal objects

- First, a path is defined as a continuous curve in a 2D or 3D space. It may be oriented (denoted OP) or not (denoted P). An OP provides a total order relation of its constituting points.

- Then, a trajectory (denoted T) is a pair (T_P, T_f) where T_P is a path, and T_f is a strictly growing and continuous (hence bijective) function from time to T_P (to each moment corresponds one position on the path). T_P is oriented thanks to T_f.

- A place (denoted PL) is a continuous function from time to space to associate to any moment a contiguous portion of space. Of course a place may be static (with a constant function of time).

- a plural entity (denoted PE) is a set of a finite number of entities of the same kind. For instance, a crowd is a plural entity composed of many people (each of them may be rendered by a place).

- A hybrid entity (denoted HE) is the combination of two kinds at least of primary entities. For instance, a road or a river may be seen as the hybridation of a curve (the center line of the road or the river) and of a constant place (the 2D asphalt surface, or the water of the river).

1.2.2 Polymorphism: several points of view on a same thing

In addition, polymorphism is an important mechanism which operates on these entities. It corresponds to the cognitive ability, very frequent in language, especially in space, to consider something from different points of view. By construction, PE and HE are polymorphic entities. For instance a road (considered from the sky) may be viewed as a path, or as a surface, and many soldiers may be seen as whole, namely a troop. Moreover, a trajectory T may be reduced, in some way, to its (static) path. But I have also designed some mechanisms able to transform an object of a given type to another type, which drastically extends the scope of polymorphism. For instance, a long and narrow place (PL) may be transformed, because of its shape, into a path (P), and a plural entity (PE) into a place (PL) if it is dense enough, or into a path P if its constituants are somehow disposed along a path. Hence, we get a rich and polymorphic range of objects which better correspond to how space and motion are dealt in language than sole places typically used in topological approaches.

1.2.3 Main spatio-temporal relations

- Topological relations. First kind is relations between places, and resembles the RCC8 with some extensions. Second kind is between a P or an OP (hence, a T) and a PL.

- Distance relations. First kind is between places (PL), second kind is between paths (P, OP, T). Third is within paths (P).

- Shape relations: Parallelism between two paths (P, OP, T) is an extension of the classical parallelism relation (between straight segments) to curves; Convex covering of a place (PL) by a path (P, OP, T); Covering of a place (PL) either by another place (PL), or by a path (P, OP, T) or by a plural entity (EP).

- Monodimensional relations. They may occur on oriented paths OP, thus on trajectories (T), and state that a given point of OP is before or after another point, near or far (with respect to the curve), etc.

1.2.4 A short overview of the expressivity of the model through some examples

We will not go here through all the capabilities of the proposed model, but just illustrate some of them through some examples.

Let us consider the following expressions which involve a river:

(1) *"The bird follows the river"*

(2) *"The road follows the river"*

(3) *"The trees are along the river"*

(4) *"Paul felt into the river"*

From a linguistic point of view, "river" may give rise to a place PL when associated with the verb "to fall", or to a path P when associated with verbs such as "to hug", "to follow", or "to go along". In the first three examples (1, 2 and 3), the river is considered as a path P, which shares a parallelism relation with another path: either the path associated with the trajectory T of the bird in example 1, or with the road also considered as a path in example 2, or with the path constructed over the EP corresponding to the trees in example 3. Examples 1 and 3 are shown simultaneously in figure 1.3. We can see through these examples three forms of polymorphism: river or road as a path, that is formally selecting P from a hybrid entity made of a P and a PL, trees as a path, that is formally transforming a PE into a P (thanks to its configuration), and trajectory as a path, that is formally selecting T_P from (T_P, T_f).

In example 4, there is a topological relation between Paul's trajectory (who is falling) and the place associated to the river, namely the trajectory starts from outside of the river and finishes inside the river.

An important point is that the polymorphic capabilities of the model render the fact that natural language often uses the same verbs to express both movement (example 1) or positioning (example 2), or the fact that the same "river" may be seen as a path from example 1 to 3, or a place in example 4.

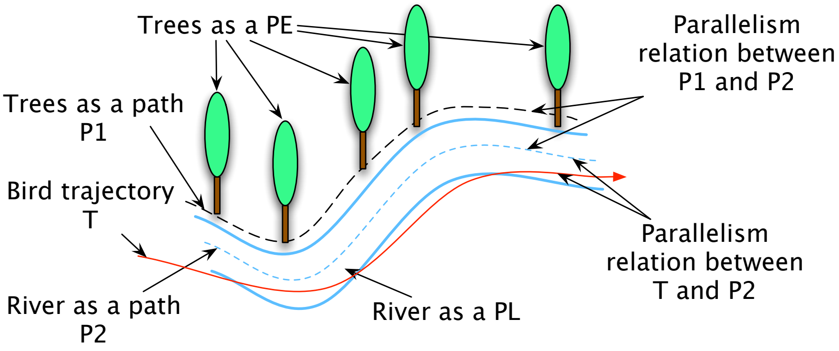


Figure 1.3: Some examples of polymorphism and parallelism in both static and motion configurations

We finish this quick overview by the semantics of "to circumvent" or "to avoid", which is particularly interesting since it combines topology and shape relations.

(5) *The car avoided the cat*

(6) *(The road / the car) circumvents the town*

In both examples 5 and 6, there is a topologic relation of exteriority, because to circumvent or to avoid means there is no contact between the verb subject and the verb object, but there is also the need for another relation which renders the fact that the verb object was kind of an obstacle to the path or the trajectory of the verb subject. We modeled the latter by the fact that the convex envelop of the path coming from the verb subject covers at least a part of the place coming from the verb object, as shown in figure 1.4. Once again, we can see that polymorphism may occur since both a positioning or a movement relation can be expressed by this complex relation.

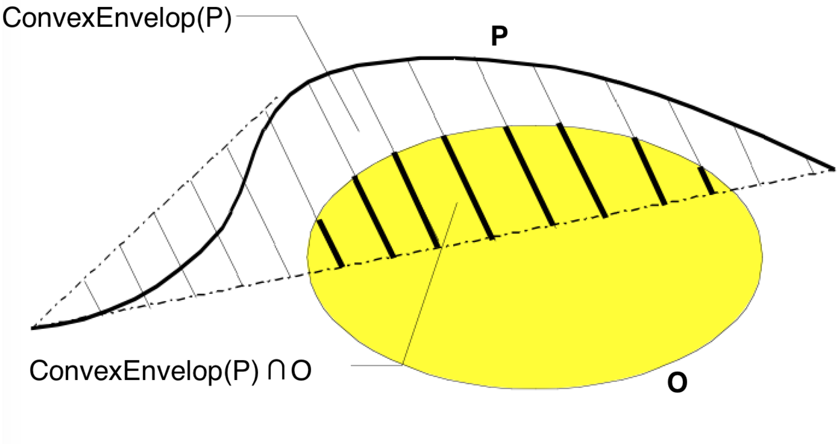


Figure 1.4: Convex covering of an object O by a path P (or a trajectory)

Chapter 2

The semantics of time and iteration

This chapter mainly relies on the joint book (Gosselin et al., 2013), which was written in french, and in particular on its second chapter of my own pp. 153 - 233.

Contents

2.1 The Laurent Gosselin's theory of time 18

2.1.1 The SdT theory in a nutshell 18

2.1.2 An implementation of the SdT interval relations construction 20

2.2 Definition and terminology of iteration 20

2.2.1 Definition of iteration 20

2.2.2 Iterator, iterating event, iterated event 21

2.3 Mental spaces and Iterative models 23

2.3.1 Mental spaces from Gilles Fauconnier 23

2.3.2 Iterative mental spaces (Model Spaces) 24

2.3.3 Iterative mental spaces in discourse 24

2.3.4 Main features of iterative mental spaces 24

2.3.5 Mental spaces and calendar 26

2.3.6 Mental Spaces and construction of iterations 26

2.4 Cloning and Projection 27

2.5 SdT intervals in mental spaces 29

2.6 An Object Oriented Iteration Model 30

2.6.1 UML in a nutshell 30

2.6.2 Iteration 31

2.6.3 Iterator 32

2.6.4 IterativeModel 34

2.6.5 ModelEvent 35

2.6.6 ModelEvent relationships 35

2.6.7 Circumstantial model intervals 35

2.6.8 Recursive constructions: Iteration as a model event 36

2.6.9 A second view on the extensional / intensional duality 37

2.6.10 A model of Events 38

2.7 Implementing Mental Spaces: links between model events. 39

2.7.1 Concomitant model events 39

2.7.2 Successive model events 40

2.7.3 General case 41

2.8 Implementing the Iteration Model on simple cases 41
2.8.1 Quantifying Iterator 41
2.8.2 Iterations with a SteadyCalendar Iterator 42
2.8.3 Iterations with a Frequency Iterator 42
2.8.4 Iteration coming from an EventDrivenIterator: "when", "each time that", etc. 45
2.8.5 Merging iterative models 47
2.9 Selections: how to access to a given part of the iterates 48
2.9.1 Introduction 48
2.9.2 Defining and modeling Selections 48
2.9.3 The purpose of Selections 50
2.9.4 Examples and analyses of Selections 51
2.10 Implicit Iterations 58
2.11 Framed Iteration 58
2.12 Conclusion 59
2.13 Perspectives 60
2.13.1 Enrichment of the model to cope with an evolutionary situation 60
2.13.2 Towards an automatic processing of Iteration in discourse 63

Following a study initiated by François Levy through the OGRE project ("Ordre de Grandeur et REpétiton", time order of magnitude and iteration) in which about 10 persons where involved, we have been collaborating since 2005 with Laurent Gosselin (linguist, University of Rouen), Patrice Enjalbert and Gérard Becher (computer scientists, University of Caen) on the specific question of how natural language handles iteration.

We call iteration the fact that a repeated activity or a repeated state is expressed through natural language. For instance, sentence 7 expresses the fact that an activity "they play cards" was repeated weekly:

(7) *Every Thursday, they played cards.*

We have conducted a multidisciplinary research where linguistics meets computer science to analyze the linguistic ground of iteration, and model it in a formal way. In this context, my contribution has been to propose an object-oriented model of iteration which is able to cope with the semantics of complex iterations even at discourse level, with a cognitive focus. I introduce this model and some of its extensions in this chapter.

2.1 The Laurent Gosselin's theory of time

This multidisciplinary study has involved a strong collaboration with Laurent Gosselin, who has been developing a theory of time semantics since 1995, and whose works are exploited here to deal with iteration. Let us introduce his SdT (for "Sémantique de la Temporalité") model in a few words.

2.1.1 The SdT theory in a nutshell

This theory is focused both on time and aspect, providing at the same time a rich description of how events are located in time (in past, present and future, but also in relation with other events), but also how they are linguistically shown (as continuous, ended, upcoming, and so on). To do so, several intervals are associated with each conjugated verb of a text. In particular, one interval corresponds to what is usually considered as time (see EI below), and another one corresponds to what is usually considered as aspect (see RI below). In more details, here are the four kinds of intervals involved:

- UI : Utterance Interval, corresponds to the time when the utterance is done (i.e. the text is written, the speech is done, etc.).

- EI : Event Interval, corresponds to the whole period of time of an event when considering the infinitive form of a verb (instead of its conjugated form).

- RI : Reference Interval, somehow corresponds to the reference point of Reichenbach (1947). It is the period of time which is shown by a conjugated verb.

- CI : Circumstantial Interval, corresponds to the period of time defined by a circumstantial complement of time.

Gosselin also proposes a set of relations between these intervals, such as "equals" (the boundaries are the same) "covers" (begins before and finishes after), "is before", "is after", and so on.

Each event of a discourse is produced at a time corresponding to its UI interval, and is given two main intervals, namely one EI and one RI, and optionally some CI.

Of course we cannot detail here, but we will just consider a very illuminating series of three examples below.

(8) *Paul was eating*

Example 8 involves past continuous tense and typically corresponds to a continuous event in the past, and may be illustrated in the Gosselin's theory by figure 2.1. The continuity is rendered by the relation "EI covers RI", which means that we focus on the period of time which is inside the event (we consider it from inside). The past is rendered by the relation "RI is before UI". It is important to understand that the past feature of the past continuous tense does not come directly from a relation between EI and UI, but from the combination of the relation between EI and RI, and RI and UI. In particular, we can infer that the event "Paul <to eat>" started in the past, but it may go beyond the UI interval (i.e. in the present or the future). In other words, it is not said that Paul has stopped eating.

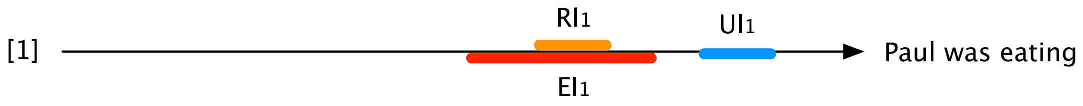


Figure 2.1: Intervals involved in example 8

(9) *Mary had done her homework*

Example 9 involves past perfect tense, and typically corresponds to anteriority in past : something is shown as already terminated at a given period of time which is located in the past. Being a combined tense ("had" + "done"), it involves two additional intervals, but the most important point is that one RI is located in the past, and one EI is located before this RI, as shown in figure 2.2

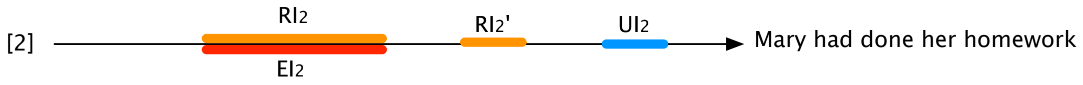


Figure 2.2: Intervals involved in example 9

The Gosselin's theory then uses a full set of rules of different levels (up to pragmatics), and ends up associating each RI with either an EI or a CI. We finally get a whole presentation of what is temporally meant, both with regard to location in time, and also with regard to how events happen and how they are linguistically shown.

To sum up this whole process with one example, let us consider example 10:

(10) *When her father came in, Paul was eating and Mary had done her homework.*

It provides the whole graph shown in figure 2.3 (with an aoristic aspect for "their father came in"), where the three UI are equal (they correspond to one and same text), and where we can see in detail that somewhere in the past, the event "father <to come> in" is surrounded by the event "Paul <eating>", and that "Mary <to do> her homework" is already over when the father comes in.

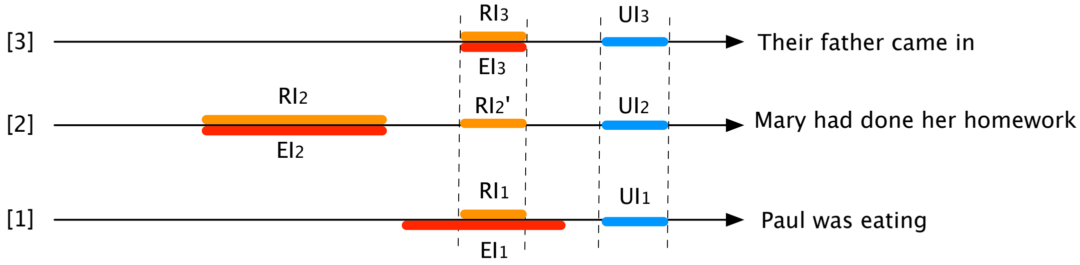


Figure 2.3: The whole set of intervals and relations corresponding to 10

2.1.2 An implementation of the SdT interval relations construction

The SdT ends up with providing a huge set of intervals and relations when applied to a whole text. Moreover, there are lots of "domino effects" because intervals are often linked to other intervals in a waterfall way. As a consequence, it is difficult to build oneself, manually, the whole final time sequence.

In addition, it is not really possible to draw a final result, since many relations between intervals are underspecified (with respect to what is linguistically said). For example, as we have already mentioned, figure 2.1 is in fact over-specified: without any additional information (from pragmatics or so), it is not possible to say if EI terminates before or after UI. It is possible that exemple 8 is followed by "and Paul is still eating while I am writing this". As a consequence, what should be shown here is not a fixed interval, but an interval whose right frontier could be moved beyond UI.

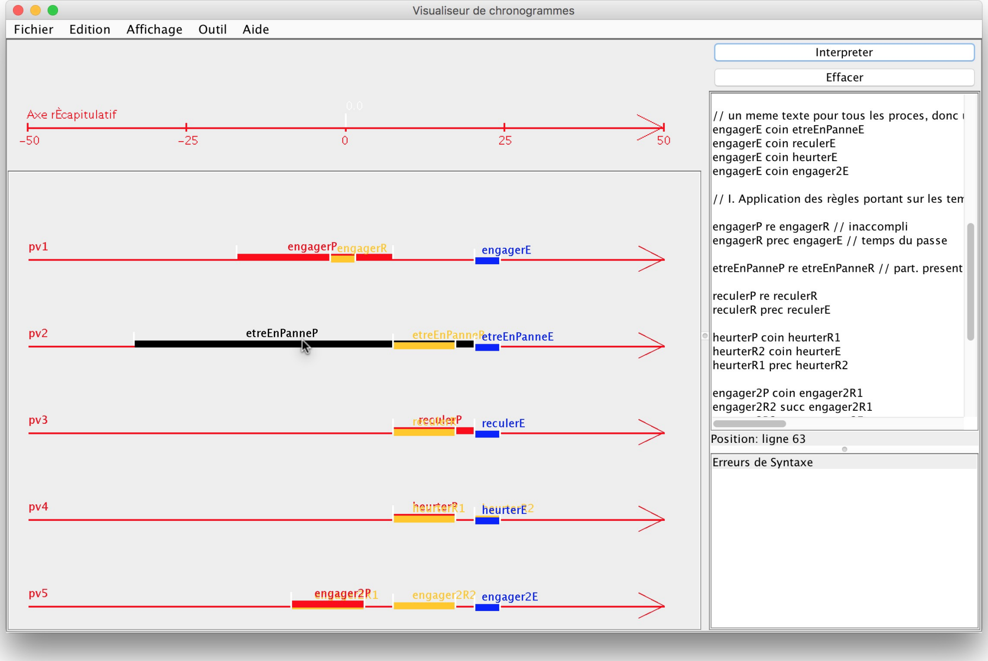
As a consequence, I designed and implemented a free Java application which receives all the intervals and relations associated with a text, once the SdT has been applied, and which constructs a complete resulting chronograph which fulfills all the relations (with respect to the "domino effects"), as shown in figure 2.4. In addition, the proposed chronograph is not static, with respect to what is computed by the SdT, and the user is able to move any frontier or interval she wishes: the system automatically applies the "domino effect" to each interval involved.

This implementation was used by Person (2004) in his PhD thesis (supervised by L. Gosselin himself and by P. Enjalbert) devoted to perform a complete automatic system which applies the SdT theory to texts.

2.2 Definition and terminology of iteration

2.2.1 Definition of iteration

We consider iteration as a linguistic object. That is, we aim to understand and to model how a language such as french or english enables us to conceive and express repetition of events (including states) or period of time, that is to say any kind of unit which has a temporal dimension, and for whose one may consider a succession of occurrences over time. Such repetitions are very frequent in our everyday



```
// un meme texte pour tous les proces, donc
engagerE coin etreEnPanneE
engagerE coin reculerE
engagerE coin heurterE
engagerE coin engager2E

// I. Application des règles portant sur les tem

engagerP re engagerR // inaccompli
engagerR prec engagerE // temps du passe

etreEnPanneP re etreEnPanneR // part. present

reculerP re reculerR
reculerR prec reculerE

heurterP coin heurterR1
heurterR2 coin heurterE
heurterR1 prec heurterR2

engager2P coin engager2R1
engager2R2 succ engager2R1
```

Position: ligne 63

Erreurs de Syntaxe

Figure 2.4: An implementation of the SdT interval relations construction

experience, for instance through the notion of hours, days, months, seasons or years, but language may also go beyond the scope of calendars, and is able to consider different events as being somehow the same phenomenon happening at different periods of time, regardless of the calendar.

First of all, in order to simplify this chapter, we use hereafter the terminology from Vendler (1957): an Event, driven by a verb, is either a State (static: "I know that", "he is here"), an Activity (dynamic and homogeneous: "He plays chess"), an Accomplishment (dynamic and with an endpoint: "he climbs the mountain"), or an Achievement (with an endpoint and which occurs instantaneously for Vendler, or which is atomic for Gosselin: "The bomb exploded"). The main notion we will use below is Event, which subsumes all other categories.

Then, we define iteration as the repetition of the same event over time:

- "Repetition over time" means that the corresponding time intervals are not coinciding, which means at least they do not start at the same time. However, overlapping is possible, since "Suzan starts a new book each year" does not necessarily mean that Suzan has finished a book before she starts the next one.

- "Same event" means that a sole infinitive verbal form (possibly complex) is able to express each of the repeated occurrences.

- To reduce the scope of this study, we have restricted our definition to the fact that these infinitive verbal forms must include an explicit subject, and this subject must steadily refer to the same entity from one occurrence to the next. For instance, in "in France, the president is elected every 5 years" does not fit this requirement since "president" may refer to different persons over time.

2.2.2 Iterator, iterating event, iterated event

Let us consider a simple iteration sentence in exemple 11.

(11) *She went seven times to Paris*

Iteration process consists in generating several events from one single event. The single event used to generate the others is called an **iterating event**, or, shortened, an **interating**, and each of the thus created events is called an **iterate**. In our exemple, "she <to go> to Paris" is the iterating, and there are seven iterates. Most of the time, an iteration is triggered by an explicit expression, "seven times" in our example, which is then called the **iterator**.

It is particularly important to point that all the iterates share a family likeness. In that respect, they are different from usual independent events which would result from seven consecutive uses of the verb "to go". This will be discussed further.

This process is illustrated in figure 2.5:

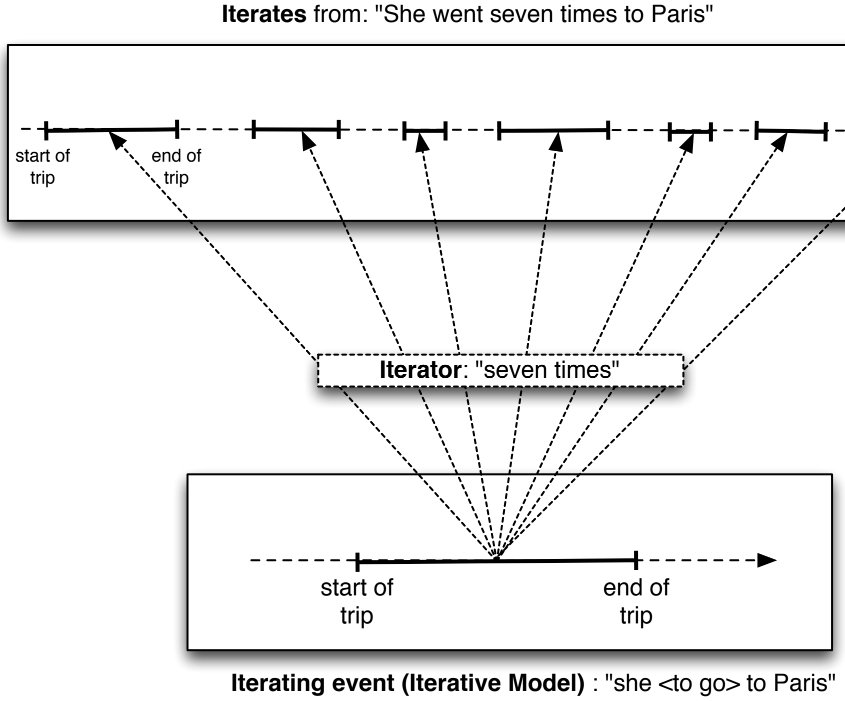

Iterates from: "She went seven times to Paris"
start of trip
end of trip
Iterator: "seven times"
start of trip
end of trip
Iterating event (Iterative Model) : "she <to go> to Paris"

Figure 2.5: Iterating and Iterate events construction

This figure shows two fundamental aspects of iteration, and how they are linked. How to integrate this duality both to linguistic context and to our model?

In this study, I have made the strong following assumption: in language, iteration may be handled from two points of view, namely extensional and intensional, as describe below, sometimes simultaneously, sometimes independently, depending on the needs of the speaker. In other words, when facing an iterated situation (either an event or a set of events which are repeated), we have the possibility to consider the whole:

- either as a set of events distributed over time (extensional point of view, corresponding to the top of figure 2.5), which may coexist with other independent events,

- or as a single situation (intensional point of view, corresponding to the bottom of figure 2.5), whose multi occurrences are known but resolutely ignored.

It is possible for the speaker to put the focus on one of these two sides, and then it is also possible to shift from one to the other. We can even observe a linguistic continuum:

(A) Focus on the extensional side : "The third of September, she went to the swimming-pool for the first time, and performed one hour of breast stroke. She went back there three weeks later for two hours, and once again yesterday morning from 8 to 10 in the morning."

(B) Focus on the intensional side : "At that time, my day started at about 7 in the morning. I went to work at 8, and back home at 6 p.m."

(C) Focus on both sides at the same time : "Every Thursday, we played cards."

In configuration A, there is a situation (she <to go> to the swimming-pool) which is repeated over time. However, the speaker introduces each individual event independently, from first to third, and specific features of each event are mentioned (the kind of swimming, the duration, etc.). This is very close to how we express non iterative situations. In this configuration, the emphasis falls on the set of iterates, which corresponds to an extensional point of view. However, the intensional point of view does not entirely disappear, since it is linguistically reflected by the use of specific words as "back" and "once again", which clearly indicates this situation has already occurred.

On the contrary, in configuration B, a generic situation is described (which we will call "Iterative Model", see after), namely the content of a day. The linguistic content is very similar to the one of non iterative situations: if we substitute "at that time" with "yesterday", then the text depicts a sole day.

Configuration C seems to be less targeted than the two others, and can be easily oriented towards configuration A by adding "overall, there has been 12 Thursdays", where the focus is on the whole occurrences from an extensional point of view, or configuration B by adding "The game started by shuffling cards, then (...)" where the focus is on one generic game, from an intentional point of view.

The two sides may be successively used, either to depict some particular occurrences, or to depict what commonly happens (which will be called an iterative model).

2.3 Mental spaces and Iterative models

2.3.1 Mental spaces from Gilles Fauconnier

Mental spaces were introduced by Fauconnier (1984) in order to model some cognitive processes. We will not detail here, but illustrate the main principles through some examples.

(12) *In the portrayal by Luc, the blue eyes girl has green eyes.*

In example 12, how the referent of girl can have "blue eyes" and "green eyes"? In fact, there are two referents, one corresponding to reality where a model girl is painted, and one corresponding to the pictural representation of this girl. In such a case, the principle of mental spaces is as follows: "the portrayal by Luc" triggers the creation of a mental space M corresponding to the pictural representations. This mental space is linked to a first space which corresponds to "real world", denoted R. R is called **parent space** of M. In addition, R is linked to M by a function F called **connector**, so that each entity in R is associated with an entity in M. In our example, an entity denoted x_R corresponds to a blue eyes girl in the real world, an entity denoted x_M corresponds to a green eyes girl in the space associated to the picture, and these entities are related by $x_M = F(x_R)$. Hence, an utterance such as "the blue eyes girl" may refer either to x_R or, through the "identification principle", to x_M, that is the "green eyes girl", and what could appear contradictory in example 12 becomes natural in the scope of mental spaces.

Fauconnier tackles example 13 which has a temporal dimension, with mental spaces connected by a temporal function.

(13) *Before she died, the head less woman went to the hair dresser.*

Even more interesting for our study is example 13 which is somehow iterative, even if it does not fit our definition (the president may be different from one occurence to the next one).

(14) *The French president changes every 5 years*

2.3.2 Iterative mental spaces (Model Spaces)

Despite my objectives differ from the ones of Fauconnier, I inspired myself of his mental spaces to propose iterative mental spaces in order to model the way iterations are handled in natural language. Indeed, the two sides of iteration may correspond to two different mental spaces. From a reference mental space M corresponding to "real time", in which all iterated events are extensionally positioned, it is possible to build a more abstract space M' in which this iterated situation is considered as a whole, that is, a one and only (not iterated) situation. Each iterate of M is connected by F_{ITER} to an iterating event in M'. There is a difference between the mental spaces from Fauconnier and the iterative mental spaces proposed here: whereas in Fauconnier, an item from M' can have only one antecedent in M for a given connector F, it is possible and natural here that a connector F_{ITER} associates to different iterate from M the same iterating event in M' (F_{ITER} is a surjective), as shown in figure 2.6.

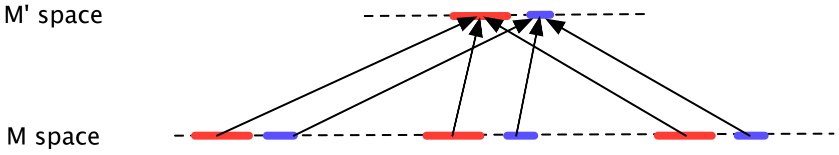


Figure 2.6: The F_{ITER} function from M to M'

M' space is designed to build and to handle the iterating situation, that is the situation that serves as a model for iterates in M. In accordance with Fauconnier's assumptions, these mental spaces do not directly correspond to specific linguistic structures, but they do correspond to constructions resulting from the interpretation of the latter (see Fauconnier, 1984, p.32). Moreover, as Fauconnier defines "introducers" as expressions which build a new mental space or which refer to a space already built, we can consider iterative expressions such as "every Thursday", "every time", etc., as mental space introducers.

2.3.3 Iterative mental spaces in discourse

We posit that a reference mental space **ME** (where letter E stands for Enunciative level) is associated with any discourse, which corresponds to the first understanding level of iterations. This level proposes a fully extensional perception of iterated events. In addition, it is possible from this level to access to enunciative intervals (IE of the SdT), which provides an absolute access to time (by opposition to relative access): an event is located in past, present or future thanks to its relations with IE.

While performing an analysis of a discourse, we build as many iterative mental spaces (henceforth **MM**, where the second M stands for Model) as the number of iterations we find, possibly in a recursive way, as detailed later.

As a consequence, for a given discourse, we get a single ME, and multiple MM. Each MM is included either in ME, or in another MM, and, possibly in a recursive way, each MM is finally included in ME.

2.3.4 Main features of iterative mental spaces

Let M and M' be two mental spaces so that M is the parent of M' (reminder: by construction, M' is necessarily a MM, and M may be either a MM in case of recursive iterations, or ME). M properties shall be deduced both from the ones of M, and from the iteration trigger. Some of these properties are temporal, and others are referential.

Temporally:

M' provides a limited access to temporal line. By "limited", we mean two aspects:

- First, M' intervals may not interact with M intervals, hence, recursively, with not any parent space of M. As a consequence, M' cannot access EI intervals (since they belong to ME, as shown in figure 2.7, hence, it does not provide access to absolute time (there is no notion of past, present, nor future).
- Second, the length of the part of the time line of a given space access to is bounded by the longer iterate it is able to generate. For instance, in an iteration based on calendar triggered by "each week", like in example 15, this length is most of the time limited to seven days (even if exceptions do exist, like in "every year, Suzan writes a new book", where some book writing may overlap, but in fact it is more "starts a new book" than "writes a new book" which is meant in such a formulation).

(15) *Every week, Paul went to the movies twice.*

Let us illustrate these principles through example 15. It is a two-level iteration, the first one being triggered by "every week", the second one, included in the first one, being triggered by "twice". Hence, there are three levels of mental spaces, as shown in figure 2.7.

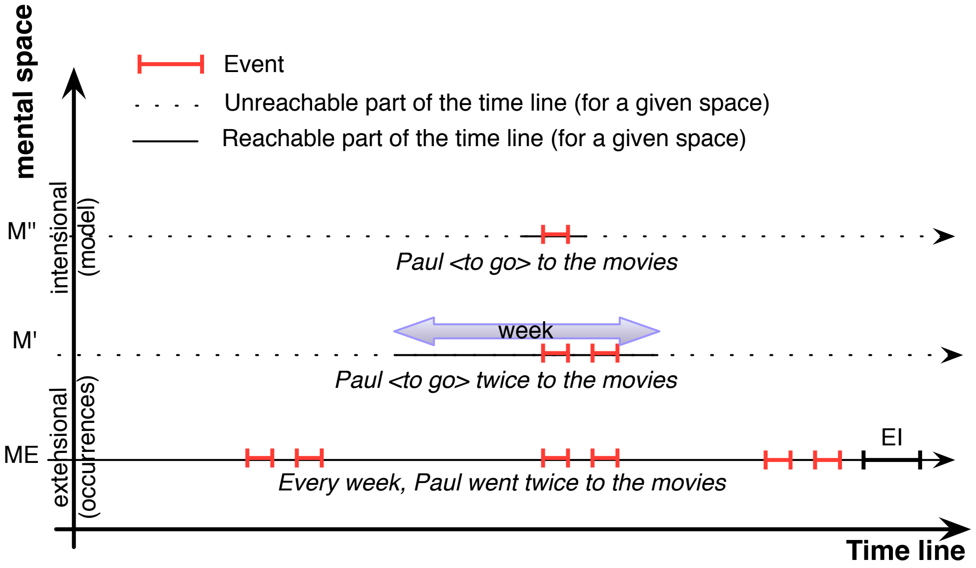

mental space
Event
Unreachable part of the time line (for a given space)
Reachable part of the time line (for a given space)
M"
intensional
(model)
Paul <to go> to the movies
M'
week
Paul <to go> twice to the movies
ME
extensional
(occurrences)
EI
Every week, Paul went twice to the movies
Time line

Figure 2.7: Mental spaces associated with a multi-level iteration

At the lowest level, ME provides a full access to the time line, where all iterates are present. At the second level, M', whose parent space is ME, provides access to a reduced part of the timeline which is one week long. Then, at the third level, M", whose parent space is M', provides access to an even more reduced part of the timeline whose length roughly corresponds to the duration of a movie.

In order to simplify, we will use the inclusion relation to mean that a mental space is built from another one: M' $\subset$ M. This relation is different from the mathematical inclusion (in particular, if x $\in$ M', one cannot conclude that x $\in$ M). It is convenient for what it stands for here since it makes clear the fact that the part of the timeline of a mental space is included in the part of the time line of its parent space. For instance, in example 15, we see that M" $\subset$ M' $\subset$ ME, and we can check that: some hours (M") < one week (M') < full time line (ME).

From a reference point of view:

All the entities which are iterated in a given space M (for instance ME in figure 2.7) are seen as a single unit in M', which keeps the features common to the set of iterated entities. For instance, in "last year, she bought a (new) pastel color dress every week", there are about 52 instances of dress in M (possibly of different pastel colors), and a single one in M', whose color property is reduced to the fact it is pastel.

However, each entity is not necessarily iterated. For instance, in "she went seven times to the mountain", there may be a single mountain, or up to seven different ones. This may or may not be linguistically specified.

2.3.5 Mental spaces and calendar

The calendar is often used to carry temporal information in natural language, and this is also true in the case of iterations. Such syntagms as "every day", "every year", "50 times per second" show us how useful calendar data are to generate repetitive events, and a part of the model I propose relies on this paradigm. This is not surprising when considering that the calendar is intrinsically built on cycles, in a semi-recursive manner (hours are repeated within days, days are repeated within weeks, and so on). Consequently, it is necessary to study how mental spaces and the calendar interoperate.

We consider that the calendar is made of different imbricated levels, each level introducing a kind of temporal period which is repeated n times within an occurence of next level. We will denote Level(i) <n< Level(i+1) the fact that an interval of level i is repeated n times in an occurence of level i+1. For instance, Second <60< Minute indicates that minute is the next level after second, and that an instance of Minute contains 60 seconds.

This system is not totally homogeneous since Months may contain from 28 to 31 days, and not exactly 4 weeks. However, we can determine that :

Second <60< Minute <60< Hours <24< Day <365.25< Year <100< Century <10< Millennium
Day <7< Week < 4< Month
Day <28-31< Month <12< Year

This system well lends itself to dating by ranking (rank of year in "in 1984", rank of day in "may the 5th"), or by naming ("Tuesday", "June"). This dating may be relative, that is, confined within a given level, and then is compliant with iteration. Or it may be absolute, that is, set with respect to an absolute point of the time line (for instance J.-C. birth for Christian calendar), and then is no compliant with iteration.

We assume that:

- A mental space may be subjected to calendar system. Then, it depends on a single calendar level (second, minute, etc.). Then, it can access to lower calendar levels, but cannot access to the upper levels.
- When two mental spaces M1 and M2 are both subjected to calendar system, and M1 $\subset$ M2, then M1 level < M2 level.
- We also consider that a mental space subjected to calendar system has access to one and only one occurrence of the concerned level. This occurrence is denoted "model occurrence". For instance, we can talk about a "Sunday model" in the case of "every Sunday". In order to simplify, we will not consider such cases as "every Sunday and Wednesday".

2.3.6 Mental Spaces and construction of iterations

To conclude this section about mental spaces, it is important to mention some cognitive facts about how iteration is built in natural language.

I think I found in Fauconnier's works a conceptual mechanism that accounts for iteration, through the propositions we have just seen. However, it should be noted that in a linguistic and maybe in a cognitive perspective, the way iterations are built may also use the opposite way to the one mental spaces work. Indeed, the latter start from an initial situation where all iterates are present, that is, a parent space, and goes toward a child space, where an iterative model (i.e. some iterating events) is obtained by subsumption of the iterates, in accordance with configuration (A) seen in section 2.2.2.

Yet, in many cases, namely the ones of configuration (B), iteration is built from a iterative model, through a process which creates iterates (in a reverse way to the one of mental spaces). This is the way which corresponds to the model I propose in this study, but this fact does not alter the compliance between Fauconnier's perspective and my approach: it simply corresponds to two reciprocal cognitive processes, namely iteration and subsumption, as shown in figure 2.8.

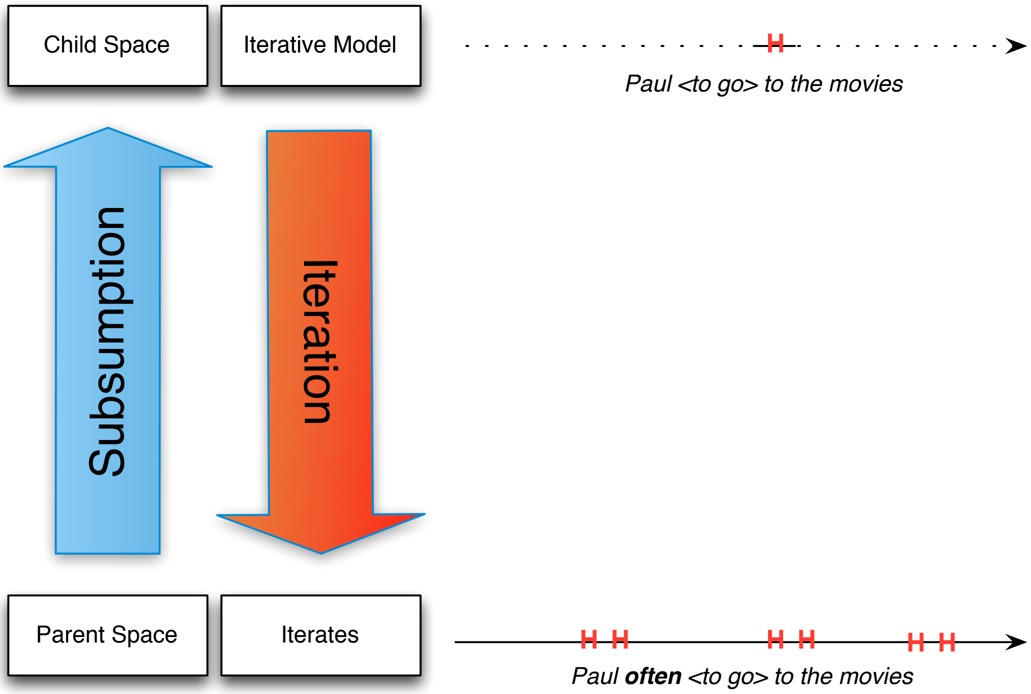


Figure 2.8: Two reciprocal cognitive processes: Iteration and Subsumption

The process of iteration cognitively consists in going from a model situation (such as "Paul <to go> to the movies") to the creation of iterates (such as in "Paul often <to go> to the movies"). Conversely, the process of subsumption consists in considering several events as belonging to a whole, that is to a model event which stands for each of them.

2.4 Cloning and Projection: the links between model event and iterates

Going from a model event to an iterate will be formalized by the notion of cloning. Cloning is quite usual in Object Oriented Programming, being a mechanism which enables to create a new object from a yet existing object, by duplicating all its features.

(16) *Every Thursday, they played cards from 8 PM to 10 PM.*

In example 16, we can consider as an initial approach that the iterator, thought as a simple mathematical function, clones the iterating event from the model space (henceforth MM) a certain number of times, and projects each of these clones in its parent space (henceforth MP). It is a steady iterator, whose period is one week.

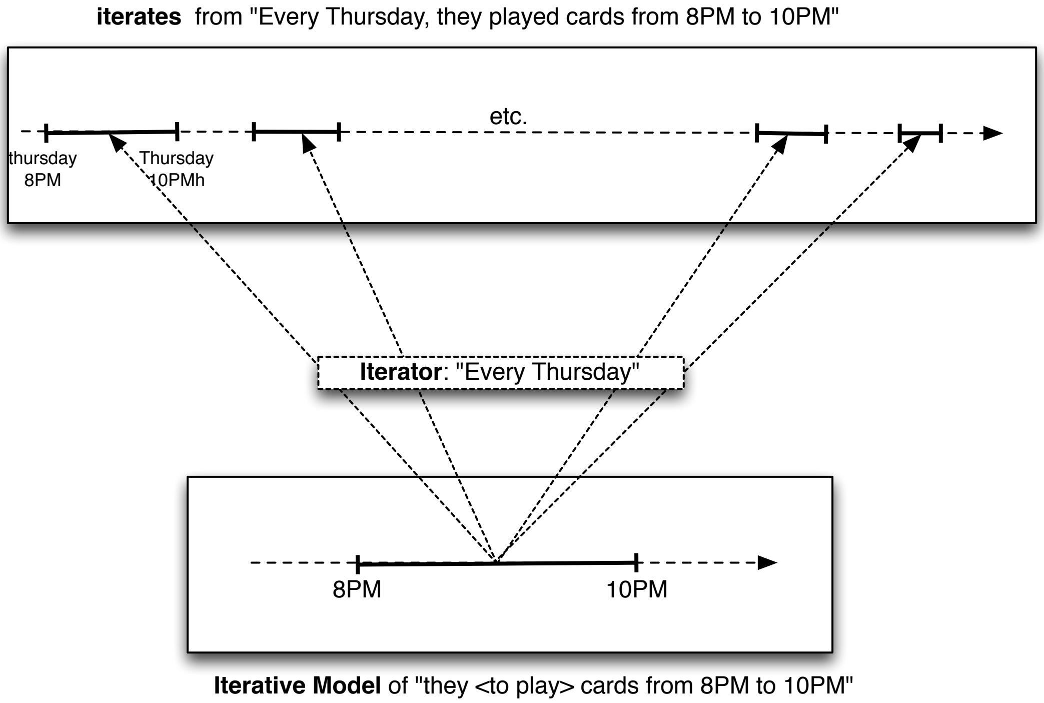

iterates from "Every Thursday, they played cards from 8PM to 10PM"
etc.
thursday 8PM
Thursday 10PMh
Iterator: "Every Thursday"
8PM
10PM
Iterative Model of "they <to play> cards from 8PM to 10PM"

Figure 2.9: Cloning and projection processes

This projection consists in two things: first, a change of mental space, from one space to its parent space. Second, a repositioning, so that each iterate is put at its particular position in this widened space. For instance, if a model event is located into the "Thursday model" of MM space built from "every Thursday", each of its clones is positioned into a different Thursday in the parent space MP.

This first approach is quite simple, but needs to be extended to fit the language reality. Indeed, its absolute regularity, that is the fact that each card game begins exactly at 8PM and stops at 10PM is clearly an over-specification of the iterates: the speaker does not necessarily assume that each card game is scheduled so precisely. Moreover, the context of card games may differ from one iterate to the next one, like in example 17.

(17) *Sometimes, they invited Suzan, which made the game lasting longer.*

In this example, the content of the iterate, that is, what happens, may change from time to time, with an additional player, and a longer game.

In addition, it is even possible to override the content of the model as much as needed, like in example 18:

(18) *Thursday the seventh of September 2017, the game started earlier, at 7PM.*

To finish, deeper derogations to the model are possible, such as in example 19 where the card game itself is changed to chess. What could appear not logical at first sight is linguistically possible thanks to the fact that the hypernym "game" subsumes both "card game" and "chess".

(19) *Occasionally, they played chess instead of cards.*

These examples show us that cloning and projection are looser than a classic mathematical mechanism, in the sense that they use the iterating event as a model to what is cloned and projected, but this model may be declined in different projected entities, namely the iterates, which inspire themselves from the model, but have their own specificities.

As a result, an iterate includes two sides :

- The model side: as the clone of a given model, it shares with other iterates some properties coming from the model.
- The singular side: this side falls within MP, and accounts for the specificities of a given iterate (for instance the fact that a game starts at 7PM instead of 8PM).

In the absence of any linguistically formulated particularity, a new iterate is given all the features of its model. For instance, in our example, the fact that a game starts at 8PM and stops at 10PM. In other words, at birth, an iterate exactly corresponds to its model (except its temporal position which is necessarily singular in MP), that is to its model side. Then, it is possible to override or to add features, which enrichies its singular side. For instance, it is the case of example 18, as illustrated in figure **??**.

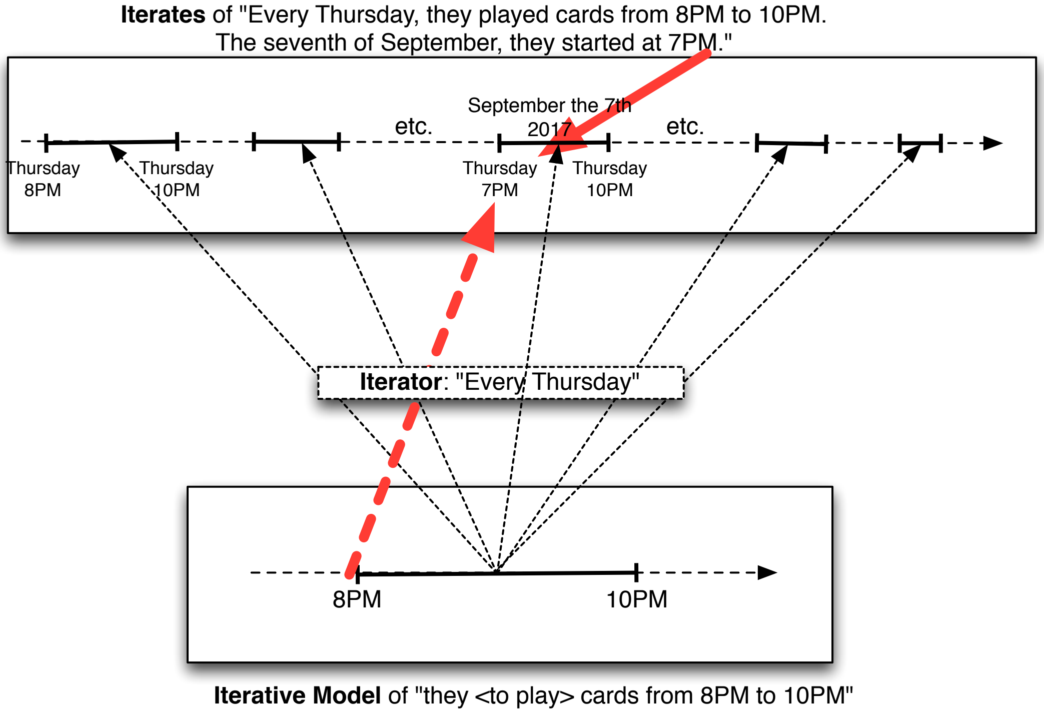

Iterates of "Every Thursday, they played cards from 8PM to 10PM.
The seventh of September, they started at 7PM."
September the 7th
2017
etc.
etc.
Thursday
8PM
Thursday
10PM
Thursday
7PM
Thursday
10PM
Iterator: "Every Thursday"
8PM
10PM
Iterative Model of "they <to play> cards from 8PM to 10PM"

Figure 2.10: The two sides of iterates

From a formal point of view, we will call **iterative model** the class of objects which describe the iterative situation, where the term "model" captures the link existing between iterative events and iterates, and in particular the fact that an iterative situation behaves as a model which can be declined in different occurrences each of them happening to be slightly different from the others.

2.5 SdT intervals in mental spaces

During this collaborative study, Laurent Gosselin has extended the SdT theory to iteration, and has created new specific intervals. I shall shortly introduce them, and see what they correspond to in our model.

The main idea, in the extended SdT, is that an iteration can be considered as a whole, and so is given intervals as it is for any regular event.

For that, first, an interval [B1s, B2s] is designed to cover the set of iterates. It somehow corresponds to the EI of a regular event. Second, a reference interval [Is, IIs] is created especially for the set of iterates. It somehow corresponds to the RI of a regular event.

In addition, Gosselin also considers the idea of an "occurrence model", which corresponds to the notion of iteration model. He designed corresponding intervals, respectively [EI1, EI2] and [RI1, RI2].

It is the same for circumstantial intervals, with [CI1s CI2s] corresponding to the set of iterates, and [CI1 CI2] corresponding to the model of what is iterated.

Let us now make a bridge from iterative models to the SdT intervals.

[EI1s, EI2s] and [RIs1, RIs2] intervals are systematically located in the parent space of another space, namely a child space in which an iterative model takes place.

[EI1, EI2] and [RI1, RI2] intervals take place in the child space (a.k.a. model space).

It is also the case, when relevant, for circumstantial intervals: [CI1 CI2] belongs to the model space, and [CI1s CI2s] belongs to the parent space. It is worth noting that some circumstantial complements concern [CI1s CI2s] (i.e. the parent space), such as "for years" in example 20, and others concern [CI1 CI2] (i.e. the model space), such as "on Sundays" still in example 20.

(20) *He came here on Sundays for years.*

The intervals resulting from example 20 are shown in figure 2.11

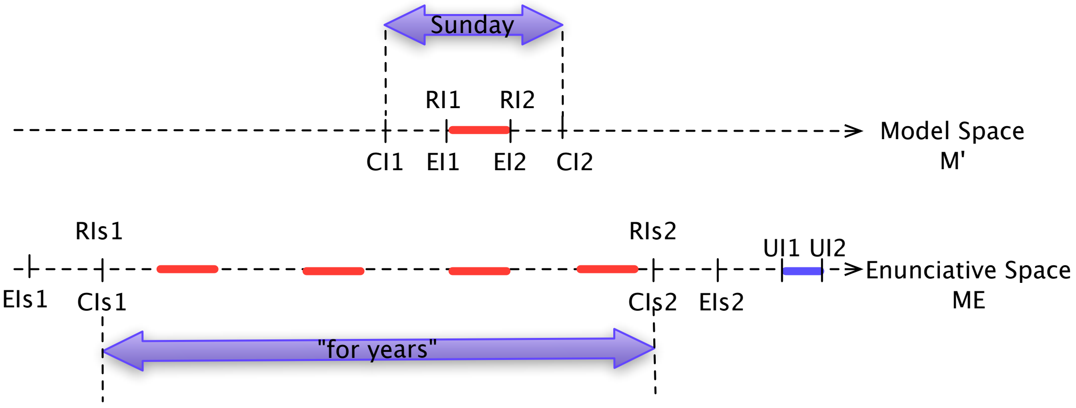


Figure 2.11: The intervals of the extended SdT in the different spaces

While in Gosselin's study, all kinds of intervals take place in the same axis, they take place in different spaces in the present model. The intervals related to the whole series of events ([EI1s, EI2s], [RI1s, RI2s], [CI1s CTIs]) take place in a given space MP parent of a child space M, whereas the intervals related to an iterative model ([EI1, EI2], [RI1, RI2], [CI1 CTI]) take place in M. This is a recursive structure, since an iterative model may come itself from another iteration, and in this case it has its own [EI1s, EI2s], [RI1s, RI2s] and [CI1s CTIs] intervals.

With regard to utterance intervals UI, in the SdT any event is related to a UI interval which corresponds to the moment when the utterance occurs (i.e. is said or is written). We take over this principle, but we put the UI intervals only in the highest level space, that is the enunciative space ME, never in the other spaces. Indeed, the latter are free of any UI, and thus have no access to absolute time. As a consequence, there is no notion of past, present or future in such spaces, but only anteriority, simultaneity and ulteriority. In other words, in MM spaces, there is no absolute time, but only relative time.

2.6 An Object Oriented Iteration Model

By now that iterative mental spaces have been introduced, and our modeling objectives defined, it is time to introduce our iteration model.

2.6.1 UML in a nutshell

In order to better understand the model introduced here, it is necessary to get some core points of UML (Unified Modeling Language, cf. Rumbaugh et al., 2004).

The model I propose is built within the "oriented object" paradigm, where we can design kinds of objects through the notion of classes. A class describes what is common to a series of objects, the latter being called instances of this class. For example, the Car class enables to create (instanciate) as many car objects as we wish. Each of these objects has its own state, independently of the one of the others (for instance, a given car can move while another one is stopped), but they all share a common structure provided by the class.

In first approach, we can distinguish two kinds of relationships between classes: association relationship, or inheritance relationship.

Association relationship stands for the fact that a class uses another one in its design. For instance, the Car class may use the Motor class to indicate that each car has a motor, and the Wheel class to indicate that each car has four wheels. In this case, there are two association relationships, one from Car to Motor, of multiplicity 1, and another one from Car to Wheel, of multiplicity 4. To finish, association may be symmetric or, in most cases, not symmetric. The corresponding UML diagram is shown in figure 2.12.

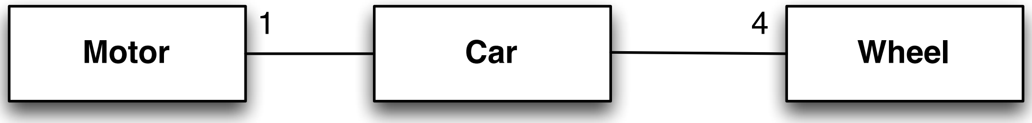


Figure 2.12: an UML diagram with association relationships

Inheritance relationship stands for the fact that a class can be the super-class of another one, which is, reversely, the child class. It is an asymmetrical relationship, being hierarchical, which indicates that the child class is a sub-type of the super-class. For instance, The Car class may be a sub-class of the Vehicle class, in the same way as hypernym and hyponym relationships. This is a very powerful feature of object-oriented modeling, and is the basis of polymorphism. The latter makes it possible to consider an object from different points of view, or, more precisely, from any of its parent types. Indeed, inheritance corresponds to a "is a" relationship, and it follows that for example an instance of Car is also an instance of Vehicle, because a Car is a Vehicle. Concretely, every time a process concerns a given type, it automatically concerns any of its sub-types, or sub-sub-types, and so on. The corresponding UML diagram is shown in figure 2.13.

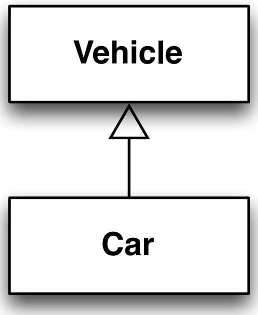


Figure 2.13: an UML diagram with an inheritance relationship

2.6.2 Iteration

Iteration is the main class of this model, and models completely any iteration of any kind.

It is composed (with association relationships) of an Iterator and an IterativeModel.

An Iterator is an object which triggers the repetition of events through time. It may rely on a set of time intervals as the ones resulting from "every Thursday" (each of them receiving one iterate), but also on other means than intervals, as for instance in "three times" (quantifying), "often" (frequency), "each time that" (event-driven).

An IterativeModel corresponds to a model of what is iterated (a.k.a. the iterative situation), and is composed of one or several IterativeEvents. The latter may be linked each other by relationships of different kinds (temporal, causative, etc.). It is shown in figure 2.15.

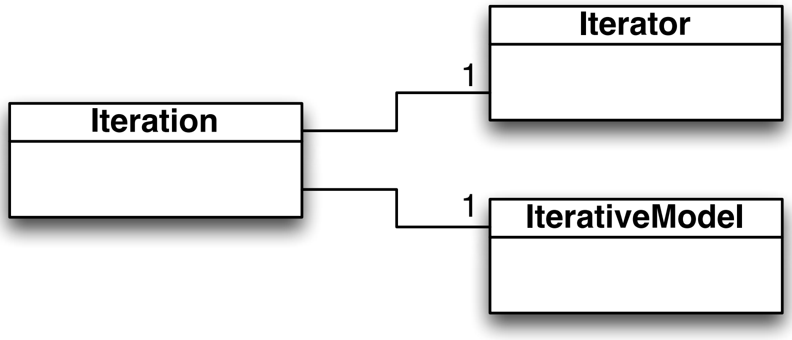


Figure 2.14: Iteration, Iterator and IterativeModel UML graph

These objects may correspond to different linguistic elements, but this is not in a systematic way.

(21) *Every Thursday, they played cards*

(22) *She often goes to Paris / to the garden*

(23) *Each time they come, we play pool*

Indeed, in example 23, the Iterator seems to come directly from the circumstantial complement "every Thursday", and the IterativeModel seems to come directly from "they played cards", but:

- In example 22, the Iterator is built not only from the adverb "often", but also from pragmatic and referential considerations, since "to Paris" induces a few times a year frequency, whereas "to the garden" induces a few times a day frequency.
- In example 23, "they come" not only constitutes a trigger of iteration, but also belongs itself to what is iterated. Hence, it is both in the Iterator and the IterativeModel.

As a conclusion, this model is devoted to render the semantics of iterations in natural language, but is not directly anchored to textual elements.

2.6.3 Iterator

Introducing Iterators

Iterators are meant to provide a series (most often, ordered) of "positions" (known or not) on the time line, each of which an iterate is positioned on. From a formal point of view, it provides from 1 to n "positions".

These positions are not necessarily temporally known. For instance, in example 24, one can say that they belong to the lifetime period.

(24) *In her whole life, she went three times to the sea.*

(25) *Before the rooster crows, you will deny me three times.*

In addition, an Iterator is given a frame, that is an interval which covers the whole set of iterates, which resembles the "reference period" from Gosselin. This frame may be contextual, or given directly in the sentence, like in examples 24 or 25.

Linguistically, this frame may be constrained by a circumstantial interval, and it covers [Bs1, Bs2] of the SdT model.

To finish, in accordance to the SdT model, a boolean attribute tells if the resulting iteration is intrinsically bounded or not. For instance, with "three times", this boolean is true, whereas it is false for "often" or "sometimes".

Iterator Model

Different classes of Iterators exist:

- Interval set: "every Thursday" (steady calendar), "one Monday out of three" (frequency calendar), "last Monday and Wednesday" (calendar)
- Quantifying: "three times", "a few times", "<to eat> three apples"
- Event driven: "When they come", "as soon as they arrive", "when the weather is fine"
- Frequency: "often", "frequently", "from time to time"
- Composite (resulting from the juxtaposition of several Iterators): "3 Thursdays and another time"

There is a clear difference to be made between CalendarIterator (and its sub-classes) and the other ones. Indeed, with this class, the temporality is already provided in the iterator, outside of the iterative model (since intervals are provided from the outset), while there is no pre-existing interval with other classes. However, we have decided to propose a flat hierarchy of all the classes in order to simplify.

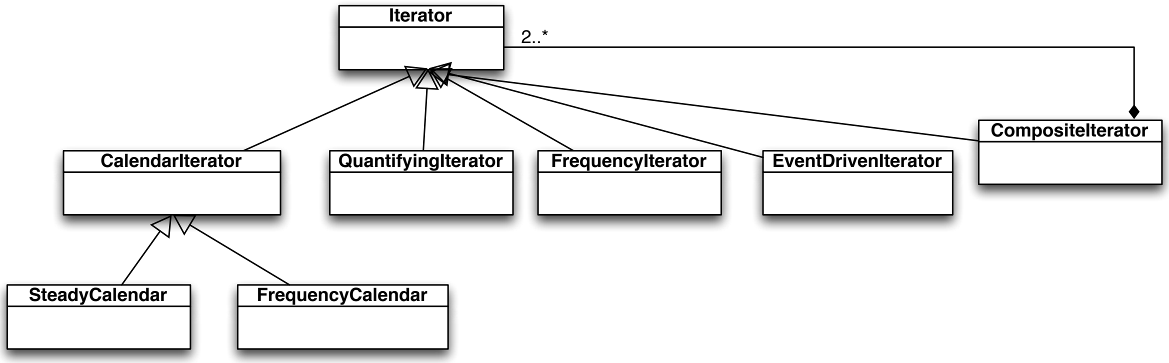


Figure 2.15: The Iterator classes

Let us now come back to the effective content of an Iterator. As already mentioned, its first goal is to create iterates, and to provide access to them. Each of the latter may be in relation with other iterates, or with other temporal elements (for instance calendar data). Let us consider two examples semantically close, but which use two different classes: (a) "three times" and (b) "three consecutive

Thursdays". Each of these Iterators provides three iterates. But, whereas (a) provides no temporal information about these iterates, (b) tells that they belong to Thursdays, and that these Thursdays are consecutive. (a) is rendered by a QuantifyingIterator, and (b) by a SteadyCalendar.

A bridge between Iterators and "iteration sources" of the SdT

Even if this study focuses on how to model iterations, it is nevertheless interesting to link some classes of Iterators to some linguistic configurations. If we take a look at the SdT "iteration sources" proposed by Gosselin, we can make the links as follows:

(a) Intrinsically iterative verbs such as "shake": they are not treated in this study

(b) Determiner in the complement of the verb such as "<to eat> two apples": QuantifyingIterator

(c) Circumstantial complements such as "Every Tuesday": SteadyCalendar

(d) frequency adverbs such as "sometimes": FrequencyIterator

(e) iterative adverbs such as "three times": QuantifyingIterator

(f) Presuppositional adverbs such as "again": iterate to be linked to a pre-existing iteration.

(g) "Conflicts" (when something seems to be contradictory or illogical at first stage of reading): may concern different Iterators. For instance, "For a long time I used to go to bed early" (conflict between "for a long time" which concerns a long period of time, and "to go to bed" which is quite short) provides a SteadyCalendar for pragmatic reasons (usually once a day), whereas "in those days, he played the piano" (one cannot play continuously during several days) should provide a FrequencyIterator.

2.6.4 IterativeModel

An IterativeModel is intended to provide a model of what is iterated. For instance, in example 16, it is "they <to play> cards". That is the most simple model, but we will see much more elaborate ones later.

Formally, an IterativeModel consists of a set of one to n ModelEvents, and of relationships between the latter.

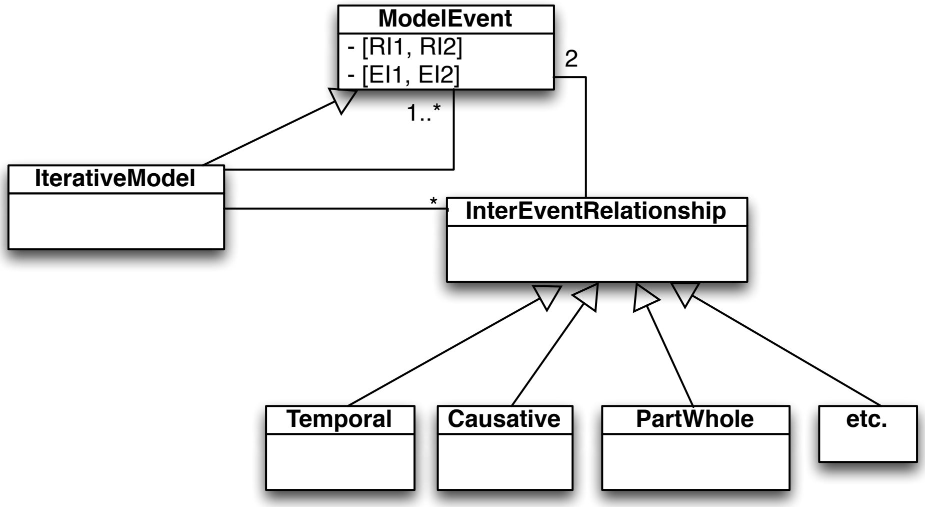


Figure 2.16: The IterativeModel class and its associate classes

Figure 2.16 represents the association between an IterativeModel and its ModelEvents, and between the IterativeModel and the InterEventRelationships. Besides, an InterEventRelationShip concerns two ModelEvents, which can be seen with the number 2 in the figure. To finish, an InterEventRelationShip may belong to different kinds, which can be seen through the inheritance relationship between classes.

The fact that an IterativeModel can contain several ModelEvents should be compared to what Gosselin calls "agglomerates". Indeed, it is the formal representation of the fact that several events may constitute a whole, and, moreover, that this whole may constitute, by inheritance, an event.

2.6.5 ModelEvent

A ModelEvent is an event which takes place in a model space. We must remember that in such a space, time access is totally focused on a generic representation of what happens. In particular, the intervals which belong to an upper level (in the parent spaces) are not available, and as a consequence, there is no absolute time (no present, past or future since enunciative intervals belong to the enunciative space).

The fact that different events are located into different spaces may evoke the "temporal referential" from Desclès (1995) or Battistelli et al. (2006). However, the goal is different. In the present case, that is to provide spaces where events from two different spaces are never temporally linked, but where it is possible from a child space to clone and project events towards a parent space.

A ModelEvent adopts the main features of normal events, in particular is three kinds of intervals: event interval, reference interval, and, sometimes, circumstantial intervals.

To finish, a ModelEvent is not given any enunciative interval. As already said, the SdT provides such intervals to any event, whether it is iterative or not. In the present case, they belong only to ME. Hence, ModelEvents, which are not in ME, are totally out of enunciation, and behave in a generic and uncoupled way. However, thanks to their event and reference intervals, they keep all the necessary features to cope with relative time and linguistic "aspect".

2.6.6 ModelEvent relationships

In the SdT, temporal and aspectual relationships are grounded on the different kinds of intervals, as already seen, and on different relationships between these intervals. In the present model, I wish to categorize the different kinds of relationships that exist between events, so that the model renders the content of texts in a more intuitive and more readable way. Since I have not worked on such relationships thoroughly, I will simply give a single picture through three cases:

- temporal relationship (see example 26)
- causative relationship (see example 27)
- part-whole relationship (see example 28)

(26) *She often arrived after Paul*

(27) *When it rained, we took our umbrellas*

(28) *The card games began with the distribution of the cards*

A temporal relationship between two events involves one of the relationships defined by Gosselin in the SdT. A non temporal relationship may have temporal consequences. For instance, a causative relationship usually implies that the consequence event is subsequent to the causative event, and a part-whole relationship implies a temporal covering of the part by the whole.

2.6.7 Circumstantial model intervals

It is necessary to broaden the iterative model so that it accounts for examples such as 29 or 30. This is done by adding circumstantial model intervals.

(29) *Each time, the cards were prepared* ***the day before***

(30) *Each time, the cards were prepared* ***in ten minutes***

These circumstantial intervals are part of a model space, and as a consequence, are not temporally connected to a parent space. In example 31, we can assume that the model space gives access to a "model year", since the iteration is triggered by "each year", and that this space contains a single "10th of July". However, this "10th of July" cannot be dated in an absolue manner, like in example 32.

(31) *Each year, they prepared the fireworks as early as the 10th of July*

(32) ** Each year, they prepared the fireworks as early as the 10th of July 1984*

According to the current study, circumstantial model intervals can belong to two kinds:

- They can provide an anchor relative to another interval from the same mental space. It is the case of "the day before".
- They can provide a partly defined calendar anchor, with a granularity level (day level in our example) strictly lower than the one of the model space it belongs to. For instance, in examples 31 and 32, the granularity level of the model space is "year", but while in example 31, "10th of July" is of level "day", and "July" is of level "month", hence compatible, in example 32, "1984" is of level "year", hence not compatible.

2.6.8 Recursive constructions: Iteration as a model event

An Iteration can itself constitute a model event, and thus can recursively belong to another Iteration. This is shown in figure 2.17 (not complete), through an inheritance relationship.

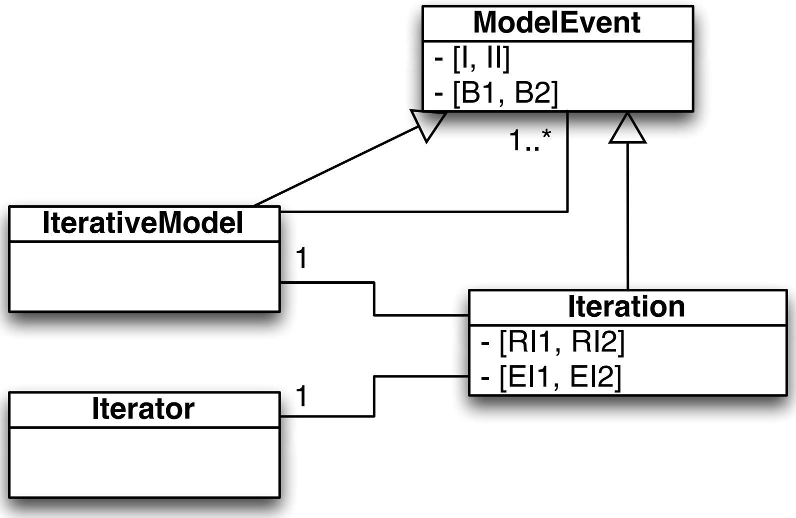


Figure 2.17: Iteration as a model event

Hence, it is possible to account for example 33.

(33) *Every Sunday, they swam twice*

However, when an Iteration serves as a ModelEvent to another Iteration, its iterates belong to model space, as any other ModelEvent of the IterativeModel it belongs to. Consequently, this Iteration should comply with the constraints relating to the affiliation to such a model space, as already formulated. Only the top level Iteration belongs to the enunciative space.

2.6.9 A second view on the extensional / intensional duality

Now that Iteration has been defined, and mental space introduced, it is wised to revisit the duality between the two possible ways to consider iterations which were introduced at the beginning of this study, and to see the related linguistic and cognitive aspects.

This model strongly relies on the construction of iterations based on the declination of an iterative model (a model of the situation that is iterated) in a certain number of singular elements called iterates, through a cloning and projection mechanism. In that sense, it is grounded on an intensional logic of iteration phenomena.

This does not mean that this study postulates that iterations in natural language always follow this intensional perspective.

First of all, we have seen with the first examples that the duality intensional / extensional is present in natural language. Moreover, the present model accounts simultaneously for this duality, since it provides both the iterative model (in an intensional perspective) and the mean to build all the iterates as well as the way to personalize any of the iterates (in an extensional perspective). Consequently, this is a dual model. To finish, it is important to distinguish between the phenomenon we want to account for, and the particular way we model it: we do not posit that any iteration in natural language comes first from an iterative model. On the contrary, it seems that the way iterations are cognitively built is a rich and various process, that we propose here to revisit.

Let us consider a excerpt of the beginning of "À la recherche du temps perdu" from Marcel Proust :

(34) *(Translated from the French by C. K. Scott Moncrieff). For a long time I used to go to bed early. Sometimes, when I had put out my candle, my eyes would close so quickly that I had not even time to say "I'm going to sleep." And half an hour later the thought that it was time to go to sleep would awaken me; I would try to put away the book which, I imagined, was still in my hands, and to blow out the light; I had been thinking all the time, while I was asleep, of what I had just been reading, but my thoughts had run into a channel of their own, until I myself seemed actually to have become the subject of my book: a church, a quartet, the rivalry between François I and Charles V. This impression would persist for some moments after I was awake; it did not disturb my mind, but it lay [sic] like scales upon my eyes and prevented them from registering the fact that the candle was no longer burning. Then it would begin to seem unintelligible.*

Example 34 puts its reader almost instantly in a model space. A long series of sentences (using the "imparfait" tense in french) creates a situation which is obviously iterated, but it is built in the same way as a non iterative situation. We only need to remove "longtemps" ("for a long time" in the translated version) to get a single (i.e. non iterative) situation.

Moreover, this iteration comes from a conflict rather than from an iteration trigger, which hides even more the extensional aspect.

An interesting clue of iteration lies in the fact that there is an enumeration of three items: "a church, a quartet, the rivalry between François I and Charles V" which would not be possible in a single situation. Each of these items corresponds to one iterate, that is, to one declination of the model. Interestingly, these different situations are subsumed by the singular "this situation" in the next sentence, which corresponds again to the model, i.e. the intensional point of view.

Except from that particularity, this text is a canonical example of an intensional presentation of iteration: from the enunciative space ME, we go into the past (thanks to the "passé composé" and then a series of "imparfaits" in the original version, which are past tenses), from which we consider a repetitive situation which occurs through a long period of time. Then, we incorporate a model space MM included in ME, which gives access to about a 24 hours period of time, from which one can build an iterative model by adding progressively different model events.

In a reverse way, instead of building an iterative situation in MM in order to clone and project it in ME, it is possible to build an iteration from ME, by describing, first, one of its iterates, which then behaves both as an iterate and as a model, like in example[1] 35.

(35) *Today, like every day, you come to the shooting studio for a new episode (# 783) in the series "Salama Bay" in which you have struggled to get a key role.*

Just removing "like every day" transforms this text in a single situation. Moreover, it is interesting that the number 783 only concerns one of the iterates. However, the phrase "like every day" indicates that this situation serves simultaneously as an iterative model for other iterates (in the past, but also possibly in the future). This example shows how close model events and single events are, both semantically and in the present model, and that their differences mostly come from the space from which we consider them. Regarding the present case, it must be noted that a single event can be seen as a model event by erasing some of its features (in accordance to what is allowed in a MM space, like absolute time, or, here, the number 783).

In a very similar manner, it is possible to build an iteration *a posteriori* from a past situation, like in the presuppositional "again" in example 36:

(36) *Yesterday, it rained all day long, and today, it is happening again.*

These linguistic considerations show that overall, model events and single events are of the same nature, and that it is possible to go from one to the other in the two ways. Building iterates from a model is what we have thoroughly detailed in the previous sections. The reciprocal process, that is, which consists in building a model from a single event, should also be provided. It is what we will see in the next section.

Let us finish with an example which goes beyond the scope of this study, but which shows some possible extensions to the present model.

(37) *Yesterday, John went to the fair. So will Paul today.*

In example 37, a single situation serves as a model for another situation where the subject is different (Paul instead of John).

2.6.10 A model of Events

An Event takes place in a temporal MentalSpace. It bears a TemporalLocation, which is relative to a MentalSpace. It owns one to several protagonists. It can be cloned as is (i.e. in the same MentalSpace), or in order to conform to another MentalSpace.

The main contribution of this model is to make no difference between what we have previously called normal events and model events. In this model, both kinds of events come from the same class, and only differ on the fact that normal events take place in ME, whereas model events take place in child spaces. According to me, this is consistent with some cognitive mechanisms if we judge by how easy it is to go from one kind of event to the other. We have already observed this phenomenon, and two new examples are provided in 38 and 39.

(38) *Yesterday, John went to the swimming pool at 2 PM.*

In example 38, we create an event e1 which is related to protagonist John, and which depends on ME space. The corresponding temporal location is absolute, provided by the deictic "yesterday", and specified by "2 PM".

[1] translated from `http://www.scenariotheque.org/Document/info_doc.php?id_doc=4411`

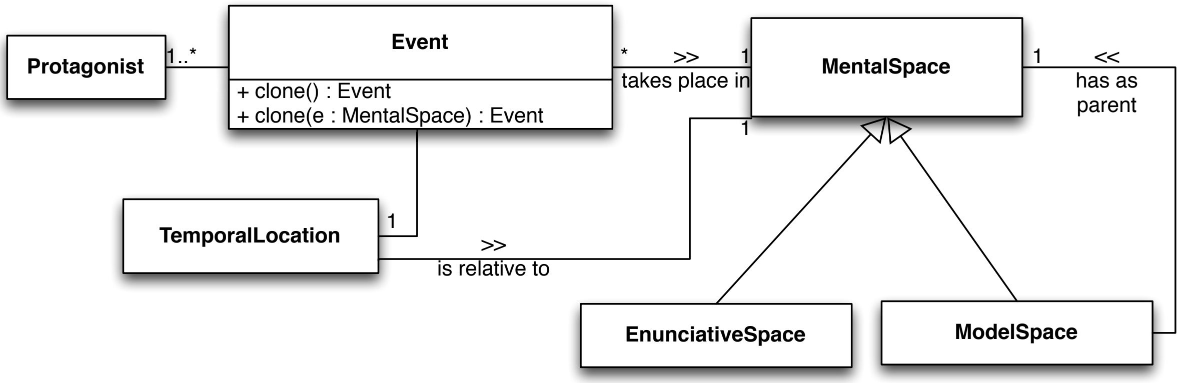


Figure 2.18: The Event class and its associate classes

(39) *He returns there tonight at 8 PM.*

In example 39, we should process in two steps. First, we have to create an Iteration, because we go from a single event e1 to an iterated situation clearly triggered by "<to return>" (which bears a presupposition). To do so, it is necessary to create a ModelEvent. Then, it is necessary to create a second event e2 corresponding to this second sentence:

- The ModelEvent which is created, called "me", is created by cloning e1, but in a child space of the space of e1 (which is ME), called here MM. Its temporal location is within MM and does not take into account the absolute location of e1, but keeps its temporal specification "2 PM").

- The second event e2 can be created either by cloning e1, staying in the same mental space, or by cloning me, with a projection into ME. In both cases, its temporal location is given in ME, and is specified by the deictic "tonight" and by "8 PM".

(40) *In fact, he has made a habit of going there regularly since the beginning of the year, and usually swims for one hour.*

In example 40, the speaker refers to the whole iteration from the beginning of the year, and gives an additional piece of information concerning the duration, and another piece of information about when the iteration has started. Thanks to this model, the event e1 and e2 get retrospective information (the one hour duration), and so we know that e1 lasts from 2 PM to (probably) 3 PM, and that e2 lasts from 8 PM to (probably) 9 PM.

To sum up, iteration in natural langage may take very various ways to be built, and the present model is able to account for any of them.

2.7 Implementing Mental Spaces: links between model events.

Mental spaces make it possible to get away from difficulties to manipulate and to render iterations. It is then possible, in such spaces, that events are no longer iterated, and the way we consider iterative temporality resembles the way we consider usual temporality.

2.7.1 Concomitant model events

(41) *Since he was married, every Sunday, Paul was baking a cake when his mother-in-law arrived.*

Let us analyse example 41. This is an Iteration whose Iterator is a SteadyCalendar which repeats Sundays and so which provides a one week long duration space.

Besides, the frame of this Iteration starts with Paul's wedding, and has no specified end.

The iterative model, induced by "Paul was baking a cake when his mother-in-law arrived", is composed of two ModelEvents. We have to move into the associated model space to figure how they work together. The available timeline is one week long, and so contains a Sunday model (a Sunday which has no absolute location). In this space, it is possible to apply the regular SdT theory: "his mother-in-law arrived" is an aoristic, which provides a coincidence relationship between the reference interval and the event interval. "Paul was baking a cake" is here a progressive, which provides a covering relationship between the event interval and the reference interval. Moreover, "when" provides a coincidence between the two reference intervals. Overall, we get the whole representation shown in figure 2.19.

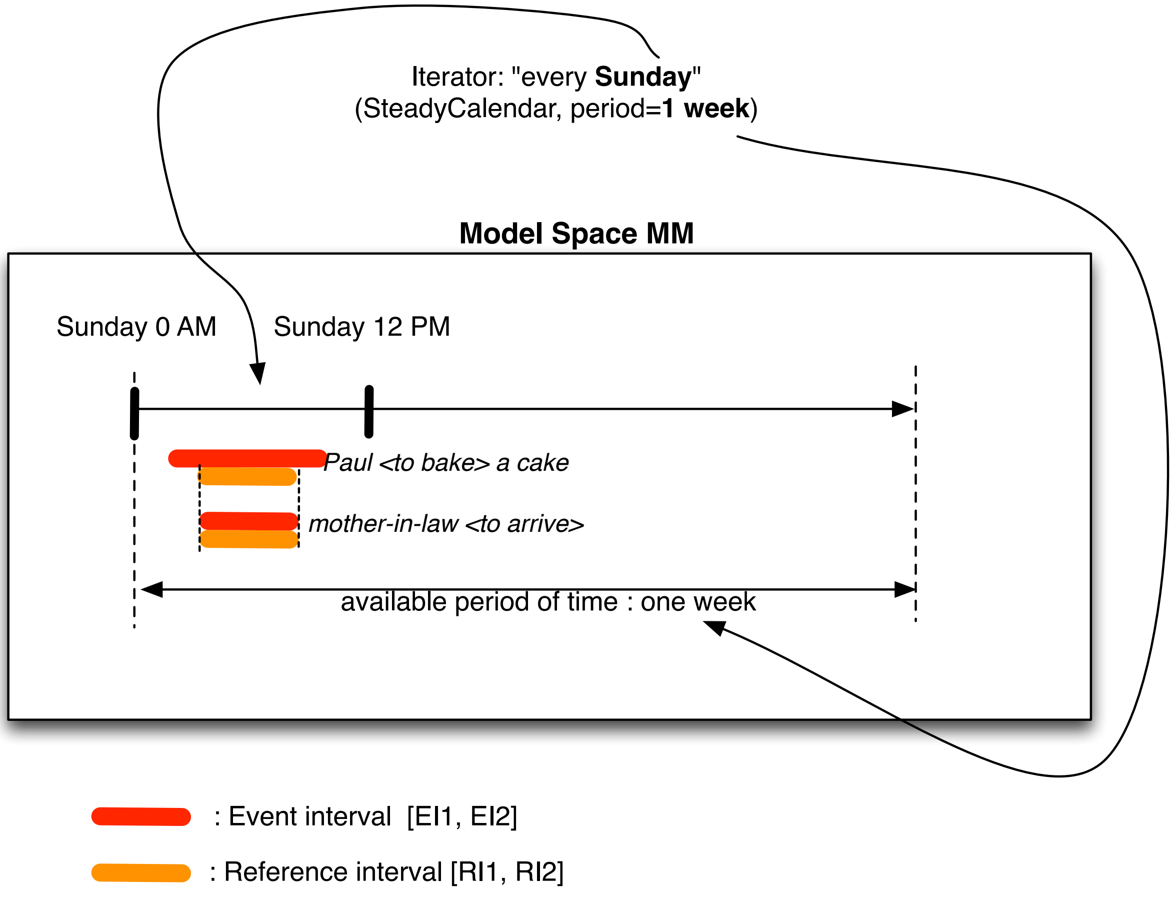

Iterator: "every **Sunday**"
(SteadyCalendar, period=**1 week**)
Model Space MM
Sunday 0 AM
Sunday 12 PM
Paul <to bake> a cake
mother-in-law <to arrive>
available period of time : one week
: Event interval [EI1, EI2]
: Reference interval [RI1, RI2]

Figure 2.19: An example of concomitant model events

2.7.2 Successive model events

In the same way as concomitant model events, it is also possible to have successive model events, which correspond to what Gosselin calls "iterative series of events". It is a particular case of model events where events occur one after the other, like in example 42.

(42) *(74 – Chapter 1) At eight o'clock in the morning he [Regimbart] was descending from the heights of Montmartre to drink white wine in the Rue Notre-Dame des Victoires. His lunch, which was followed by several games of billiards, led him till three o'clock. He was then heading towards the Passage des Panoramas [...]. (Flaubert, L'Education sentimentale, Gallimard, 1965: 57)*

In this example, there are several successive events, corresponding to "to descend", "to drink", "to have lunch", and so on.

2.7.3 General case

We have just seen two particular cases of relations between model events. In the general case, we just have to build the relationships exactly as we do with the SdT with normal events, which may lead to some quite complex structures, like in example 43:

(43) *Every Thursday at 8 pm, the game started. At that time, the house was generally quiet for a long time. The games kept rolling. The evening ended at about 10 pm with an aperitif.*

In this example, there are successive model events with "to start", "to keep rolling", "to end", while there is a concomitance (more precisely, a covering relationship) between "to be quiet" and "to start", as shown in figure 2.20.

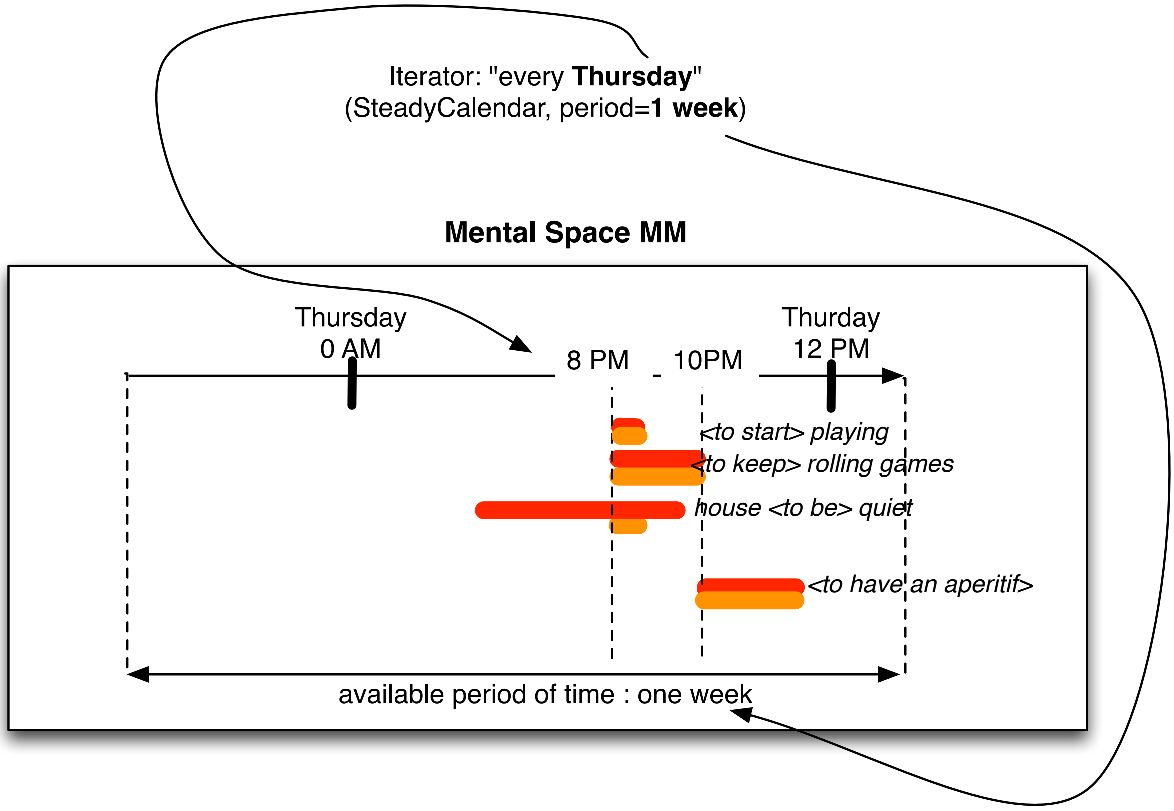


Figure 2.20: A rich example of model event relationships

2.8 Implementing the Iteration Model on simple cases

In this section, we show how the model can be implemented in simple cases through some examples.

2.8.1 Quantifying Iterator

Let us go back to one of the first examples of this study, reproduced as example 44 below:

(44) *She went seven times to Paris*

This simple iteration relies on the QuantifyingIterator coming from "seven times". There is no specified frame for this iteration, and we only know that this iteration takes place in the past (taking account of the past tense), and for pragmatic reasons that it starts after the birth of the subject. There is only one ModelEvent coming from "she <to go> to Paris", so the iteration model is simple. The resulting graph is provided in figure 2.21.

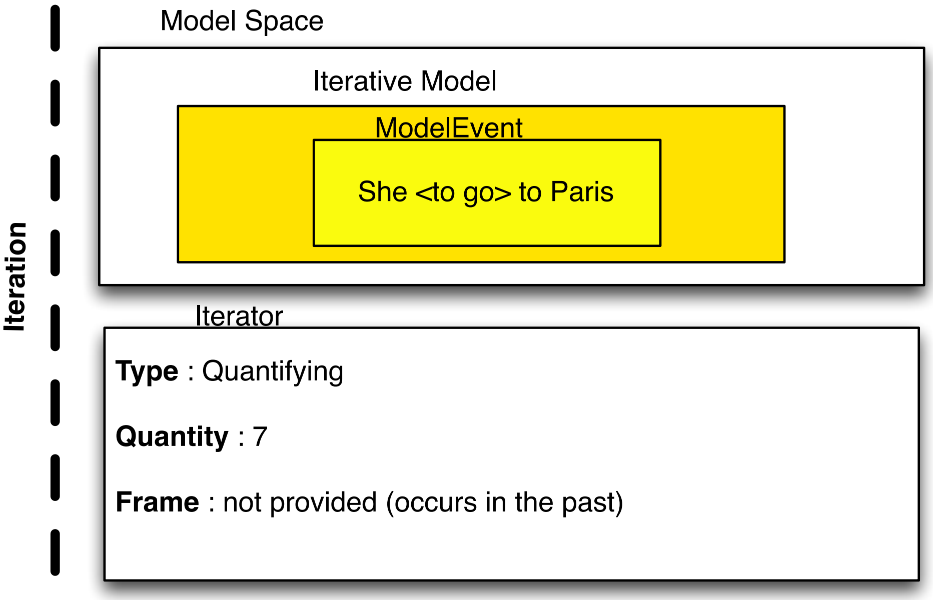


Figure 2.21: An iteration coming from a quantifying iterator

The two white frames correspond on one side to the content of the model space, in which the iterative model takes place (and its ModelEvents), and on the other side to the description of the associate Iterator, whose content depends of the type (here Quantifying). A vertical dotted line (named Iteration) merges these two frames to represent the whole iteration. For the sake of brevity, the temporal and aspectual information previously mentioned are not reported here (but would take place in the iterative model).

2.8.2 Iterations with a SteadyCalendar Iterator

We go back now to another example, reproduced as example 45:

(45) *Every Thursday, they played cards from 8 PM to 10 PM.*

In this case, there are some time specifications provided by "8 PM to 10 PM", which concern the only ModelEvent, as reported in figure 2.22.

At this stage of our study, it is important to note that the iterative constructions we have built are quite generic. Indeed, if the proposed model clearly distinguishes between Iterative Model (in the top of the figures) and Iterator (in the bottom of the figures), it is because these two parts are really independent.

It is possible to reassemble elements coming from different iterations, and get representations consistent with the linguistic correspondant reassembly, such as in "Every Thursday, she went to Paris" which mixes examples 44 and 45, or "They played cards 8 times from 8 PM to 10 PM". This reveals the compositional aspect of iterations and of the present model.

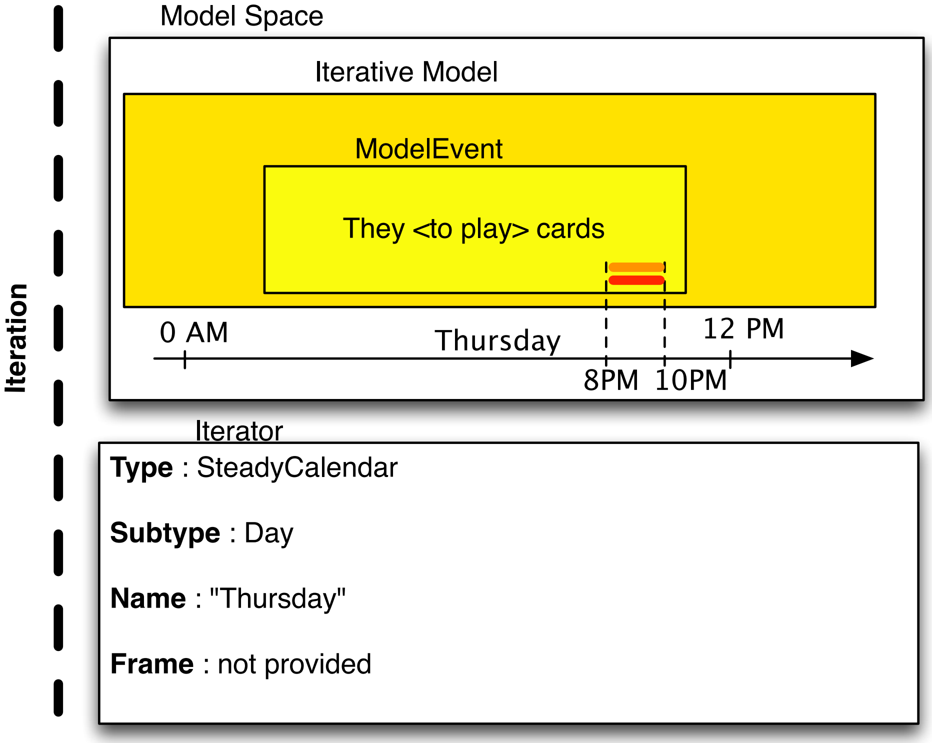


Figure 2.22: An iteration coming from a SteadyCalendar iterator

2.8.3 Iterations with a Frequency Iterator

Frequency Iterators are more difficult to handle for two reasons.

First, they bear a frequency value which is mainly given by extra-linguistic data. For instance, the obvious difference of frequency between "he (quite) often goes to the kitchen" and "he (very) often goes to New York" lies in the interpretation of the situations (we cannot go as often to New York as to the kitchen). For this reason, "quite often" in the first case may provide an higher frequency than "very often" despite the fact that linguistically, the adverbs say the contrary. This is due to the notion of "norm" provided by Gosselin.

Second, if in some cases frequency iterators make sense with respect to absolute temporality (case A), like in the examples we have just seen, they can also make sense with respect to an event driven temporality (case B) in other cases, that is, with respect to iterated events. For instance, "Paul often goes to Cabourg, more rarely to Deauville", the frequency may be related to each time Paul has a week-end trip. This is thoroughly studied by Gosselin, relying on De Swart.

For this reason, we tackle these two cases in two different ways. Case A is tackled in this section, whereas case B is tackled in section 2.9 which requires some additional developments, namely Selectors, which will be introduced later.

To make the distinction between cases A and B, we will call FrequencyIterator the one corresponding to case A, and FrequencySelector the one corresponding to case B. Besides, some adverbs like "often" in example 46 can have a double meaning, the one of case A in example 47, and the one of case B in example 48.

(46) *Often, after doing our homework, we were watching a movie*

(47) *We often did our homework, and then went (each time) to the movies*

(48) *Our homework was often followed by a movie.*

It is possible to disambiguate example 46 by supplementing it by "it happened two or three times a week" (acceptation A) or with "it happened three times out of four".

We now focus exclusively on case A, that is on FrequencyIterators, in the rest of this section, by analyzing example 47. This is an opportunity to address a more complex iterative model than previously, since it involves two ModelEvents : "we <to do> homework", and "we <to watch> a movie". These two events are linked by a succession relationship, as shown in figure 2.23.

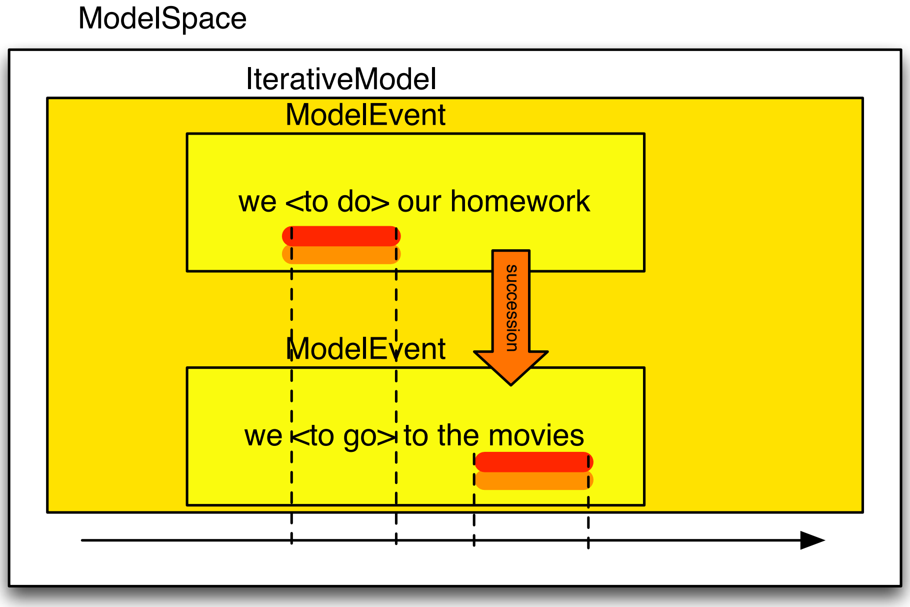


Figure 2.23: A complex iterative model

In the rest of this study, we will not provide the detail of temporal relationships, and we will only provide, at most, the name of the relationships between ModelEvents (like "succession" in this example).

The Iterator is quite simple, and is devoted to provide a certain number of temporal locations corresponding to the position of the iterates. It is very under-specified at this stage because it requires contextual or pragmatic informations in order to get better specifications (consider for instance "he changes his shirt very often" versus "he changes his house very often").

From an object point of view, we have designed a class hierarchy of different FrequencyIterators.

A FrequencyIterator may be "numeric" (NumericFI) of "fuzzy" (FuzzyFI : "we often go to the movies") depending whether or not it quantifies the frequency.

Among NumericFI iterators, we distinguish between the steady ones (SteadyNFI : "we go to the movies every three days") and others (UnsteadyNFI : "we go to the movies one week-end out of four") depending whether or not they provide a strict period of time between two iterates.

The inheritance graph is provided in figure 2.26.

There is an obvious proximity between NumericFI and calendar iterators, in particular as far as the SteadyNFI class is concerned. This is because the numeric content of the latter can be projected on a calendar. Indeed, it is possible to build very similar iterations coming from these different kinds of iterators.

(49) *We recycle our bottles every Monday (SteadyCalendar)*

(50) *We recycle our bottles every seven days (SteadyNFI)*

(51) *We recycle our bottles once a week (UnsteadyNFI)*

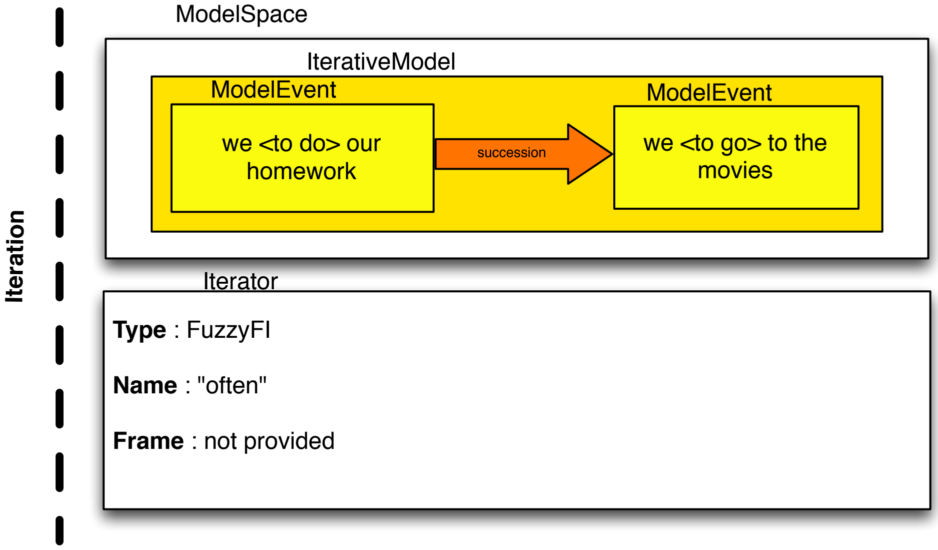


Figure 2.24: Iteration coming from a FrenquencyIterator

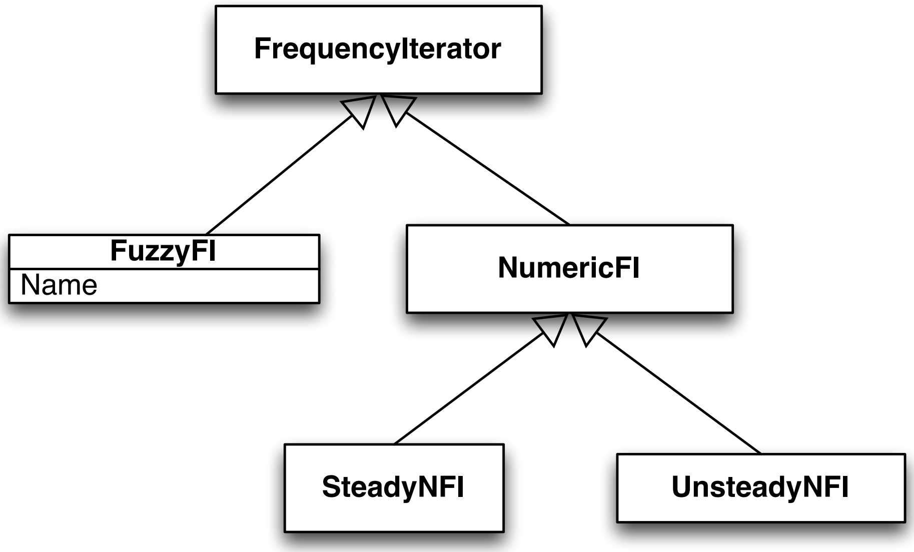


Figure 2.25: Iteration coming from a FrenquencyIterator

However, the third example (51) is clearly distinct from the first two of them (49 and 50) insofar as the concerned day may change from one week to the next one (hence the unsteady behavior). There is still to debate about the tenuous difference between SteadyCalendar and SteadyNFI. My opinion is that SteadyNFI relying first on frequency rather than on the calendar, it may easily provide temporal drifts. For instance, if we supplement example 50 by "Sometimes, it is 6 days only because the container is full", then there will be long term temporal drifts, whereas if we supplement 49 by "Sometimes, we need to do it as soon as on Sunday", the Mondays should then come back again because of the semantics of "every Monday"...

2.8.4 Iteration coming from an EventDrivenIterator: "when", "each time that", etc.

We address now a very different kind of iterators from the ones we have just seen, insofar as it is interwoven with the iterative model of which it accounts for.

(52) *When they came, we played cards.*

Indeed, in example 52, which relies on an EventDrivenIterator, the ModelEvent "they <to come>" serves both to build the IterativeModel, that is, what is being iterated, but also to build the Iterator, that is, what generates the iteration. In other words, this means that the ModelEvent "they <to come>" determines all iterates, i.e. each time the event "they <to come>" occurs, then an iterate should correspond to it. This also means that this triggering event belongs to the IterativeModel.

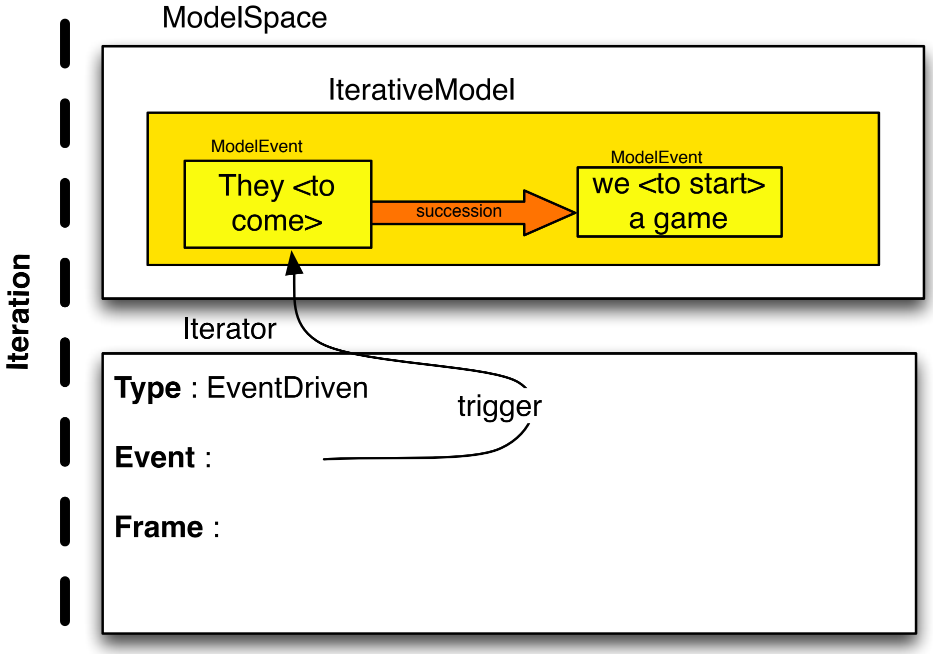


Figure 2.26: Frequency Iterators hierarchy

This model is quite different from what is proposed by the other authors of the book (Gosselin et al., 2013) than I, insofar as it does not rely on a set of time intervals as a prerequisite to build an iteration, but relies directly on a ModelEvent.

These two points of view are not incompatible, since it is possible to associate a time interval to each event coming from "they <to come>", but they do correspond to two different ways to consider this kind of iteration. In the present model, the main objectives are expressivity and factorizing information. And it turns out that the series of intervals that could come from the events "they <to come>" is the consequence of the iteration, not its cause: while a SteadyCalendar such as "every Thursday" cuts the timeline in a series of disjoint intervals so that, in a second time, iteration is build over this series, according to me, the process seems to be the contrary here: all the information linguistically provided by this iteration is present in our model. Moreover, as far as expressivity is concerned, the proposed model renders directly the fact that each event "we <to play> cards" is linked to an event "they <to come>", what would be somehow hidden by a representation relying on a series of intervals.

This model also accounts for the description of recurring states, as in example 53, in a quite natural manner.

(53) *He always come empty handed*

In this example, it would be weird to build intervals coming from the events "he <to come>", and then say that in each of these intervals, there is one event "he <to come> empty handed": there would be twice as many events as needed, which would be unclear and redundant information.

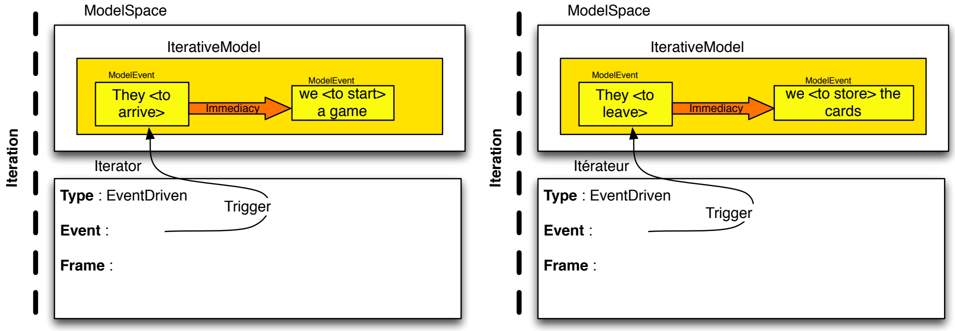
2.8.5 Merging iterative models

Such phrases as "when", "each time", "as soon as", may be iteration triggers. However, they are often used several times within a given iteration, as in examples 54 and 55. A local analysis may lead to several separate iterations while only one iteration is concerned.

(54) *As soon as they arrived, we started a game*

(55) *As soon as they left, we stored the cards*

These examples can be represented by figure 2.27.



(a) Context 1

(b) Context 2

Figure 2.27: Two iterative models before merging

However, discourse analysis of such examples should make it possible to say that example 55 is only supplementing what is said in example 54, and that there is one single iteration. Note that in the french original version of example 55, the verb "repartir" provides a clear evidence of the link between the two events, namely the "re-" prefix of "repartir".

Hence, I propose the formal way to merge two iterative models:

- let i1 and i2 be two event driven iterations
- let $me_{trigger}$ the ModelEvent which triggers i2
- if i2 finally turns out to complement what is asserted by i1, then it exists an Event $e1$ in i1 which is linked to $me_{trigger}$ by a given relation r.
- then, i1 and i2 can be merged in i=merge(i1, i2), whose Iterator is the one of i1, and whose IterativeModel is the union of the IterativeModels of i1 and i2, complemented by the additional relation r:

  $IterativeModel(i) = IterativeModel(i1) \cup IterativeModel(i2) \cup \{r\}$.

In the present case, linguistic and pragmatic analyses should determine that what is asserted in example 55 is after what is asserted in example 54. Hence, we get $me_{trigger}$="they <to leave>", $e1$="we <to start> a game", and r=Succession, which provides the merge shown in figure 2.28.

More and more complex constructions are possible, including, for example, "elaborations" from N. Asher's SDRT, resulting in increasingly rich iterative models.

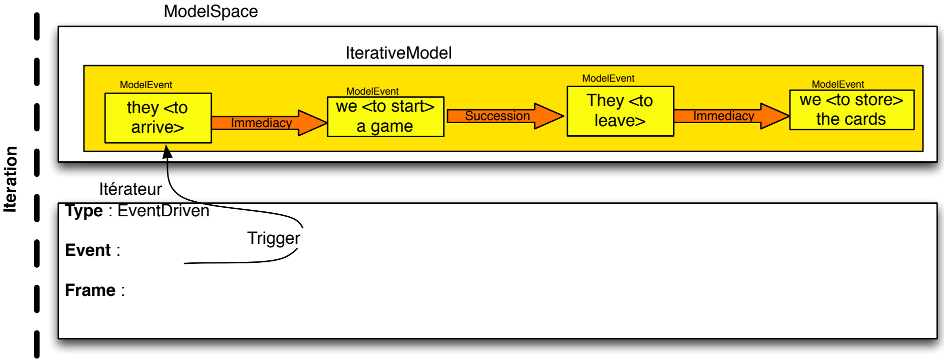


Figure 2.28: Fusion of two iterative models

2.9 Selections: how to access to a given part of the iterates

2.9.1 Introduction

As we have seen in the previous sections, an iteration makes it possible to generate iterates from an iterating situation. An additional mechanism, that I will call Selection, makes it possible, once an iteration has been built, to select a subset of iterates.

From a linguistic point of view, it corresponds to such examples as the following:

- "She throws an ace once in four times"
- "The first times that she came, ..."
- "The third time she came, ..."
- "Often, when she came, ..."

Let us go back again to one of the first examples of this study, reproduced as example 56 below:

(56) *She went seven times to Paris.*

And let us complement it by example 57:

(57) *The first three times, it was to meet her sister.*

Sentence 57 creates a selection over iterates coming from sentence 56, in the sense that "the first three times" refers to the first three iterates.

2.9.2 Defining and modeling Selections

A Selection is intended to create a subset of iterates coming from a given Iteration. From a formal point of view, a Selection bearing a certain (reduced) number of iterates, it is itself an Iteration (we will not discuss here the difficult case of singleton selection as in "one single time").

Formally, it is possible to build the Iteration corresponding to a given Selection in the usual way: take the same IterativeModel as the one of the initial Iteration (the one the Selection relies on), and create the Iterator as the composed function fOg (i.e. f(g(...))) so that g is the Iterator of the first Iteration, and f is the selecting function.

However, we will prefer the first way to define a selection (i.e. relying on a given Iteration) to the second (i.e. building a new Iteration with a combined Iterator function), because it clearly shows that a Selection relies on an Iteration, and it does not create (duplicate) any new iterate (what the second way might suggest).

As a consequence, Selection is a subclass of Iteration. It has two main attributes:

- the Iteration it relies on
- the Selector, that is, the restriction function which chooses a subset of iterates

The corresponding diagram is shown in figure 2.29 which evokes, in the Design Pattern paradigm, "Decorator". A Selection somehow decorates an Iteration in the sense that it complements the latter by a certain number of modifications, as we will see later.

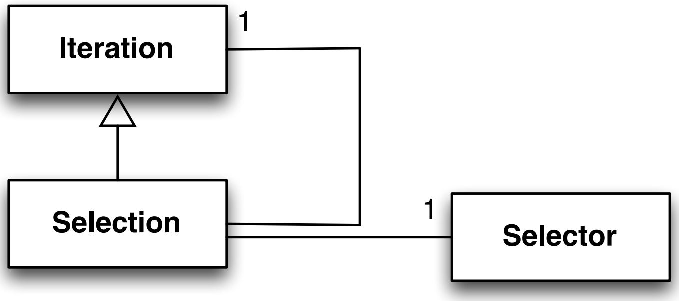


Figure 2.29: The Selection inheriting class

The different ways it is linguistically possible to make a Selection lead to the class hierarchy shown in figure 2.30.

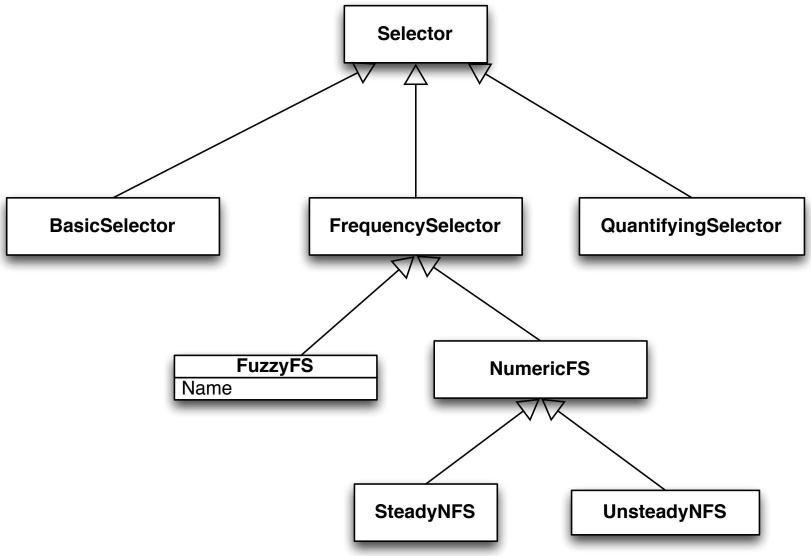


Figure 2.30: Selectors Hierarchy

A Selector is either "basic", when it directly provides a certain number of iterates through their rank (for instance "the first two times"), or "frequency", or even "quantifying".

Among FrequencyIterators, there are FuzzyFS ("often, after doing our homework") which do not provide any numeric information, and NumericFS. The latter may be SteadyNFS ("toutes les deux fois" in french), or UnsteadyNFS ("three times out of four"). We have to mention that the choice between SteadyNFS and UnsteadyNFS is not necessarily linguistically driven, and may depend on pragmatic criteria.

It is important to have a second look at the frequency notion, and see thoroughly the difference that exists between a frequency selector and a frequency iterator. Whereas a FrequencyIterator is intended to indicate, with respect to the timeline, what density of events we can infer, a FrequencySelector does not rely directly on the timeline, but relies on a discrete set of iterates. Hence, it is a frequency with respect to a discrete set, not to a temporal continuum.

Finally, a QuantifyingSelector selects a certain number of iterates, without specifying which ones, as in "She went seven times to Paris. Peter accompanied her three times". It resembles a BasicSelector, but does not provide any specification on which iterates are selected.

2.9.3 The purpose of Selections

Obviously, Selections are not intended just to select a certain number of iterates, which would be worthless. It is always intended to modify or enrich the selected iterates. Let us take an example through a short text beginning with a classic Iteration in example 58, and then using Selections to operate some modifications in examples 59 and 60.

(58) *When they come, they use to offer us gifts.*

(59) *One out of three times, they go through Montélimar.*

(60) *Each of these times, they bring us nougat.*

First of all, sentence 58 generates an Iteration. Then, sentence 59 creates a Selection among the times when "they <to come>", at the rate of one time out of three, with the amendment that "they <to go> through Montélimar". All that is conveyed by the iteration is inherited at Selection level, and then amended (that is, modified or enriched). Hence, "they <to offer> gifts" is still true in the iterates selected by example 59, and is enriched by two facts, first that "they <to go> through Montélimar", and second, that "they <to bring> us nougat."

Why to reify the Selection operation? We could settle to integrate what is selected within the Iterations instead of creating Selection objects. Indeed, we can legitimately consider that such operations do not create new Iterations, but simply modify existing ones.

However, there are two strong reasons why I have chosen to create Selections as independent objects.

First, it would be a mess to integrate amendments within a given Iteration. It would force to have an IterativeModel which includes different possibilities, which quantifies each of them, and so on. Moreover, this would probably become near impossible to tackle when there are Selections of Selections, as we will see below.

Second, and probably the most important reason, is that according to me, Selections are linguistic and cognitive real entities, and so it is important this model renders them as independent objects. For instance, "each of these times" from example 60 clearly refers to the Selection created in example 59, not to the Iteration created in 58. Besides, it would be very difficult to tackle example 60 if what is said by example 59 was directly put in the Iteration coming from 58.

Addressing the role of Selections bring us to the question of the dependency between Iterations and Selections. By design, Selections rely on a preexisting Iterations. Less obviously, an Iteration also relies on all its Selections (i.e. all the Selections that rely on it), insofar as each Selection acts on some of its iterates.

Back to our example, it is obvious that sentence 59 thoroughly modifies the iteration coming from sentence 58, since as much as the third of what is said by 58 is enriched by 59.

For this reason, I have enriched the Iteration class by adding to it the list of all of its Selections. Finally, the content of an Iteration is given not only by what it initially contains (an Iterator, an IterativeModel), but also by all the modifications conveyed by the additional Selections it is concerned by.

2.9.4 Examples and analyses of Selections

BasicSelector

A BasicSelector simply provides a set of iterates by their ranks.

(61) *When they came, we played cards. The first time was a poker game. Peter did not come the fifth nor the sixth time.*

Example 61 begins with an Iteration coming from an EventDrivenIterator. Then, two successive Selections occur, the first one with regard to the first iterate, and the second one with regard to the fifth and sixth ones. As already said, these selections modify the content of a part of the Iteration. For this reason, the Iteration now has links towards these Selections, as shown in figure 2.31. In this figure, we use the object notation such as "a.b" to mean "member b of object a". In the first Selection, the point is to specify the content of "<to play> cards". We access to the Iteration object via the "it" reference, and then to its IterativeModel by the member "model". Then we use two methods, one to access to the Event coming from "we <to play>", and a second to specify its content.

The same applies to the second Selection.

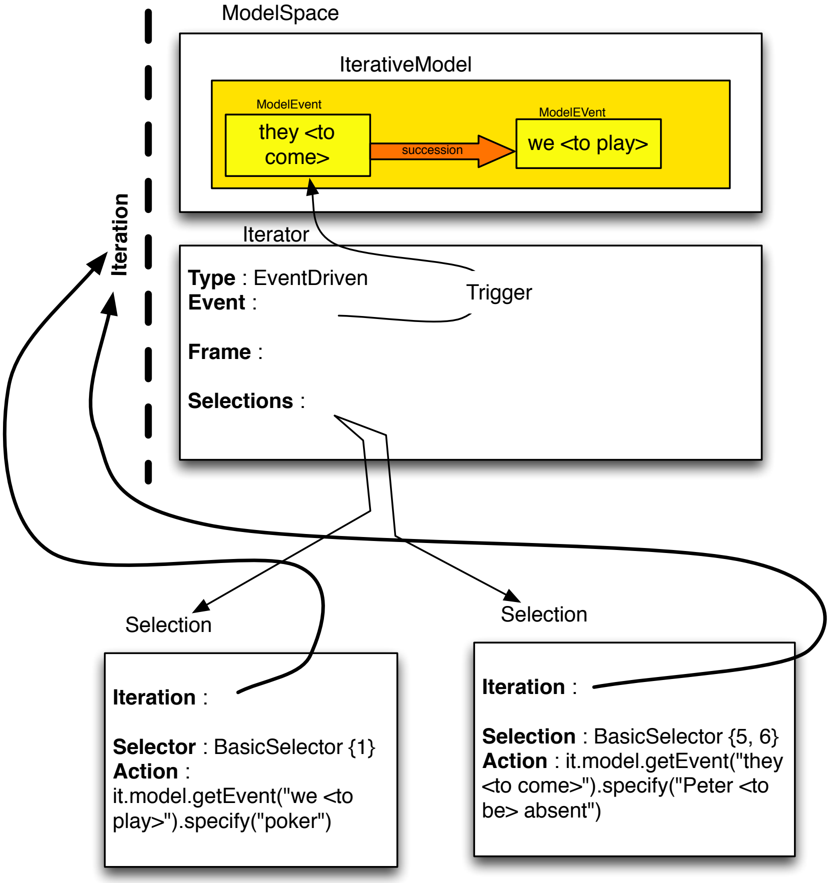

ModelSpace
IterativeModel
ModelEvent
they <to come>
succession
ModelEVent
we <to play>
Iteration
Iterator
Type : EventDriven
Event :
Trigger
Frame :
Selections :
Selection
Selection
Iteration :
Selector : BasicSelector {1}
Action : it.model.getEvent("we <to play>").specify("poker")
Iteration :
Selection : BasicSelector {5, 6}
Action : it.model.getEvent("they <to come>").specify("Peter <to be> absent")

Figure 2.31: An example of simple Selections (relying on BasicSelectors)

FrequencySelector

Now, we introduce two new points through the next example. First, the use of FrequencySelectors. Second, a more complex way to build Iterations, since the latter is implicitly triggered by the Selection, and then is complemented by the last sentence, after some Selections have occurred.

(62) *Often, when I walked, I met John. More rarely, I met Suzan. At these times, we had long conversations. My walk always ended in shopping downtown.*

Indeed, once example 62 is fully interpreted, the main Iteration (i.e. the one which concerns all iterates) contains two ModelEvents: "I <to walk>", coming from the first Selection, and "I <to do shopping>", coming from the last sentence.

Besides, another important point is the use of the anaphor "at these times", which concerns the second Selection, and confirms the great interest to have reified Selections.

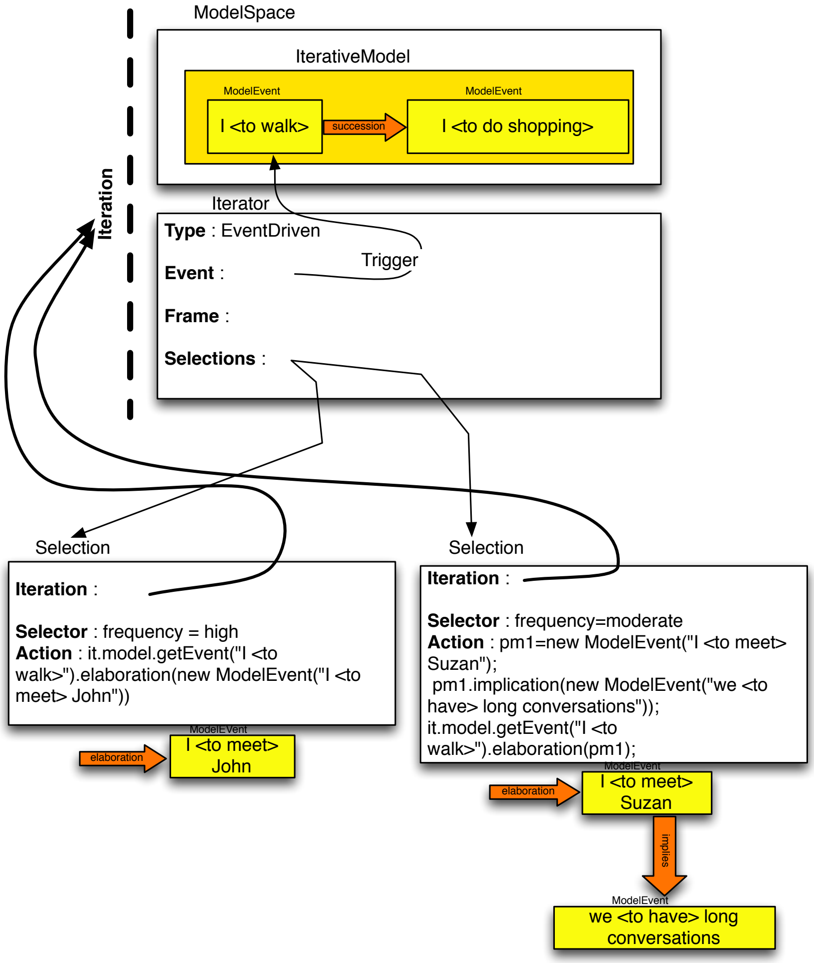

ModelSpace
IterativeModel
ModelEvent
I <to walk>
succession
ModelEvent
I <to do shopping>
Iteration
Iterator
Type : EventDriven
Event :
Trigger
Frame :
Selections :
Selection
Iteration :
Selector : frequency = high
Action : it.model.getEvent("I <to walk>").elaboration(new ModelEvent("I <to meet> John"))
elaboration
ModelEvent
I <to meet> John
Selection
Iteration :
Selector : frequency=moderate
Action : pm1=new ModelEvent("I <to meet> Suzan");
pm1.implication(new ModelEvent("we <to have> long conversations"));
it.model.getEvent("I <to walk>").elaboration(pm1);
elaboration
ModelEvent
I <to meet> Suzan
implies
ModelEvent
we <to have> long conversations

Figure 2.32: Rich selections

The final schema may become quite difficult to read when there are lots of nested elements, like in figure 2.32. In these cases, it is possible to build specific IterativeModels for each Selection. This is illustrated in figure 2.33 for the Selection coming from "at these times".

Particularisation of ModelEvents by a Selection

Let us consider the Iteration coming from example 63

(63) *On Monday, they usually had three training sessions: two of strength training, and one of stretching. Then they had an aperitif.*

This Iteration comes from a SteadyCalendar, as already seen, and we can now focus on its IterativeModel. The latter is particularly rich, embedding four ModelEvents, and has the specificity that the first three ModelEvents constitute an Iteration, through "three training sessions". As a consequence,

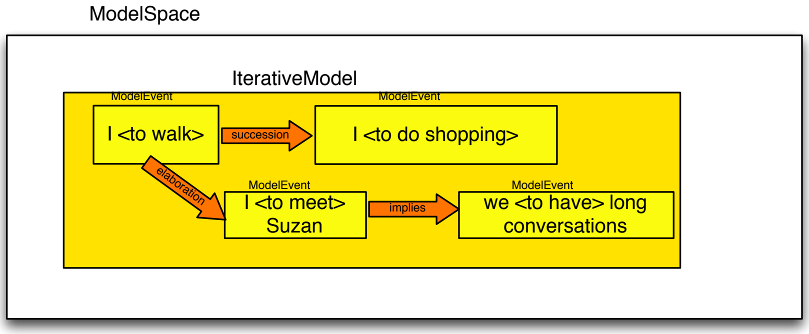


Figure 2.33: The iterative model corresponding to a given Selection

we put the corresponding Iteration in the IterativeModel, and then particularize two of its iterates with "strength training", and the other one with "stretching". To do so, we can use two QuantifyingSelectors, with the additional constraint that the two selections are mutually exclusive.

For simplicity purposes, figure 2.34 accounts only for the IterativeModel of the whole Iteration.

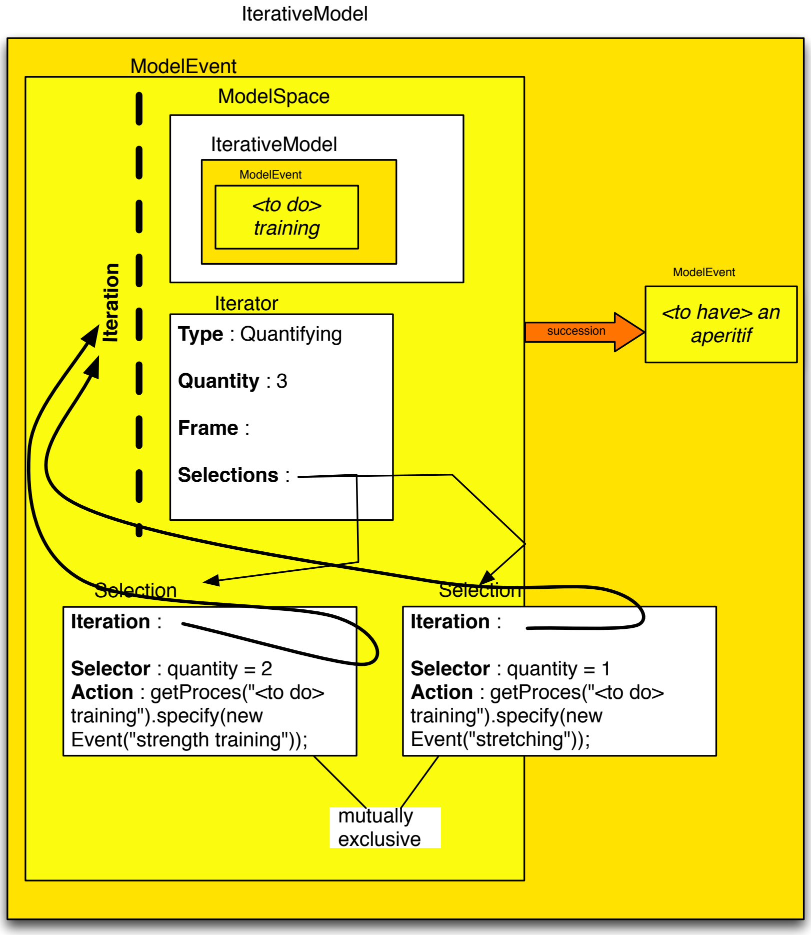


Figure 2.34: Mutually exclusive Selections

These particularizations being linguistically verified, we can conclude that strength training and stretching are sub-events of training event, which is shown in figure 2.35.

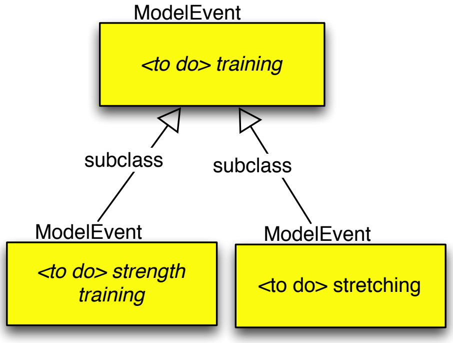


Figure 2.35: Compatibility between Events through inheritance relationships

Hypothetical ModelEvents and Conditional ModelEvents as Selection drivers

We focus now on Iterations using "if" or "when". To tackle this question, we could have relied on mental spaces from Fauconnier, which have been already a source of inspiration for this study. For sure this will be an interesting option in the future, but for now, I have limited this study to a much more modest approach, by slightly enlarging the current model.

Let us begin with a non iterative case, as reported in example 64.

(64) *Tonight, if they come, we'll go to a restaurant.*

There is here an alternative, with a first event "they <to come>", which is hypothetical, and a second event "we <to go> to a restaurant", hypothetical too, whose realization depends on the first one. As a result, we get either two successive events, or none (with the implied event "we <to have dinner> at home"). The corresponding schema is as follows :

Alternative:

- Hypothesis one: "they <to come>", then "we (including 'they') <to go> to a restaurant".
- Hypothesis two: "we (not including 'they') <to have dinner> at home".

However, there is still a missing element in this proposition. Admittedly, we finally get one of the two hypotheses, but we miss the reason for this (the fact "they <to come>), which is linguistically provided by "if they come". In order to integrate it into the model, we introduce ConditionalModelEvents which bear at once a condition event, and the alternative events, as shown if figure 2.36.

Let us now get back to Iterations with example 65, and see that ConditionalEvents apply the same.

(65) *On Sunday evenings, if they come, we all go to a restaurant. Otherwise, we eat at home. Before that, we walk on Les Champs Elysées in the afternoon.*

First, there is an Iteration relying on a SteadyRegular iterator (involving Sundays). What is iterated here is neither the fact that "they <to come>" nor the fact that "we <to go> to a restaurant", nor either the fact that "we <to eat> at home", but a kind of an alternative between different situations, depending on the fact that "they <to come>", built over the implicit hypernym event "we <to have> dinner" (which occurs every day, hence every Sunday). This dinner is available in two versions, with, in the first, the previous Event "they <to come>". As a consequence, we can build a ConditionalModelEvent which bears the two versions.

A key point of ConditionalModelEvents is that they bear both a condition, and a full Event. The whole resulting IterativeModel is shown in figure 2.37.

However, it often happens that iterations derogate somewhat from this scheme, using selections instead of alternative model processes. Indeed, in example 66, "when" does not explicitly introduce an alternative, unlike what "if" does.

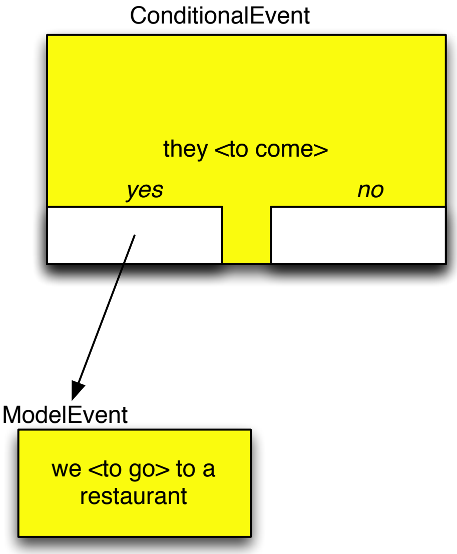


Figure 2.36: ConditionalEvent

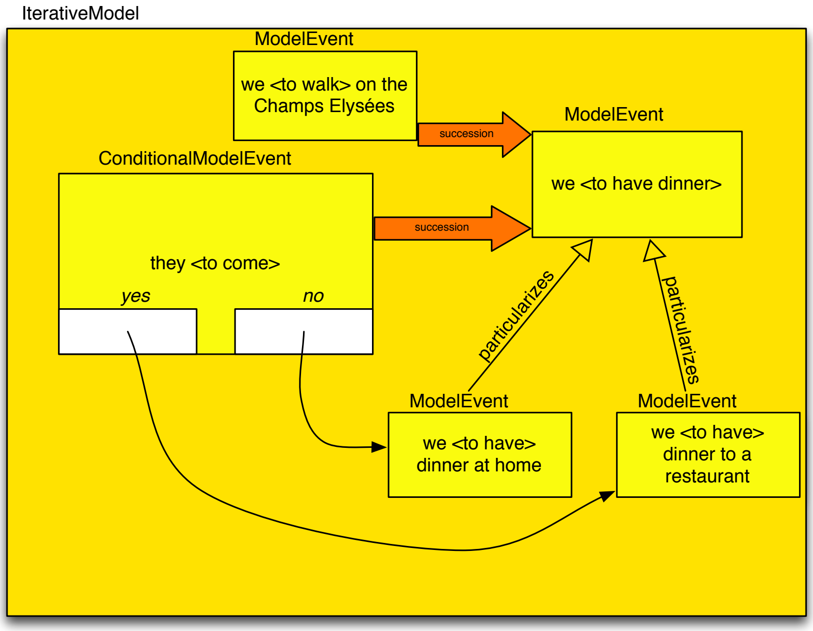


Figure 2.37: An IterativeModel containing a ConditionalEvent

(66) *On Sunday evening, when they come, we all have dinner at the restaurant. Otherwise / in other cases, we eat at home. Before that, we walk on Les Champs Elysées in the afternoon.*

First, we have to remark that "when they come" is ambiguous, since it may mean either "among all Sundays, the Sundays when they come", or "they come every Sunday, and when they arrive...". However, this ambiguity is resolved by "otherwise" which forces us to keep only the first acceptation. Reversely, the verb "to arrive" instead of "to come" would have force to keep the second acceptation because of the presupposition it bears (and the second sentence would not be relevant).

As a consequence, we face here a Selection, brought by "when they <to come>". More precisely, only the Sundays when they come are selected.

Then, a second Selection is done as the complement[2] of the first one, brought by "in other cases".

Finally, the HypotheticModelEvent "they <to come>" supports both Selections, either because it occurs, or because it does not occur. For this reason, we call it HypotheticModelEvent. Like ConditionalModelEvents, it only occurs in some iterates. The resulting schema is provided in figure 2.38.

The difference between ConditionalModelEvent and HypotheticModelEvent is tenuous, and mostly linguistic. The first one is coming from an explicit condition mark, whereas the second one accounts for a Selection. But from a semantic point of view, they are very similar, and we could imagine a process which transforms one in the other.

Lastly, we notice that HypotheticModelEvent works in a quite similar way as EventDriven selectors, which is not surprising if we consider that the associate linguistic forms are also quite similar ("when", "each time that"). When these linguistic forms occur in the context of an Iteration, it leads to a Selection based on an HypotheticModelEvent, whereas in the contrary, an Iteration is directly created.

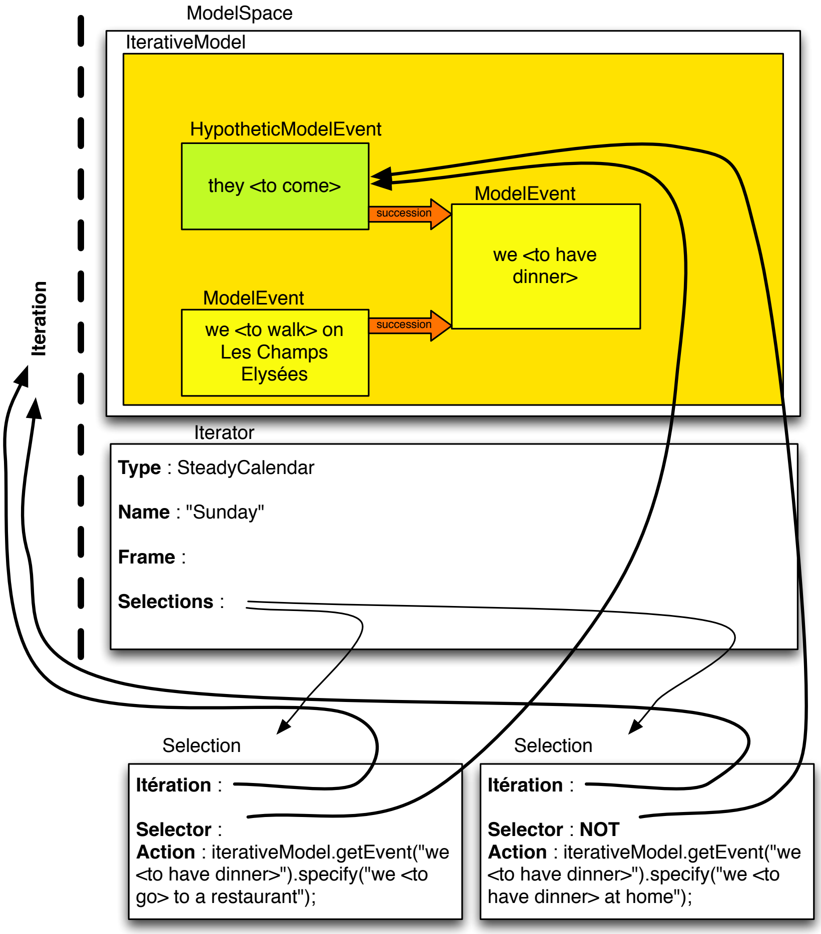


Figure 2.38: Selections relying on HypotheticModelEvents

Recursive Selections

There is little to say about recursive selections, since they are only special cases of regular selections. Indeed, a selection being itself an iteration, one can naturally apply another selection on it, and so on, recursively. We can therefore directly study example 67.

[2] Let S1 and S2 be two Selections of Iteration I. Let {X} be the set of the elements contained in X, then S1 and S2 are called complementary Selectors if {I} = {S1} $\oplus$ {S2}.

(67) *Every Monday, we go to the swimming-pool. Sometimes, Paul accompanies us. The third time he came, he learned to dive. He has already come ten times.*

A first Iteration is trigger by "Every Monday". Then, "sometimes" creates a Selection over this Iteration, which results in sub-iteration. To finish, "the third time he came" is a Selection of the third iterate of the sub-iteration.

We can notice that the information coming from "he has already come ten times" also has consequences on the first Iteration, which is shown in figure 2.39 through the cardinality constraint.

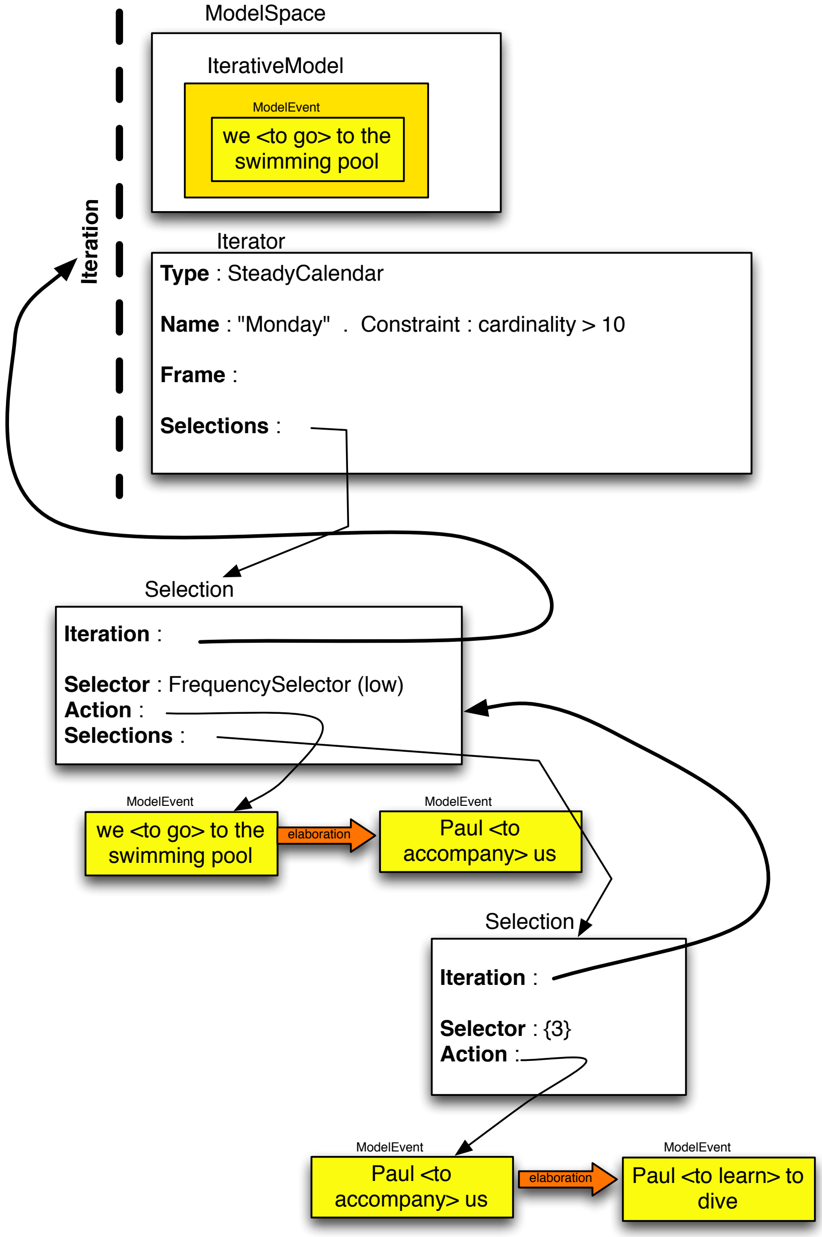


Figure 2.39: Recursive Selections

2.10 Implicit Iterations

The canonical way to get Selections is to have first an Iteration on which one can make Selections. However, this pattern is not systematic. It is common that Selections occur though no corresponding Iteration has been previously created. In such cases, I shall assume that the Selection presupposes, and consequently introduces, an Iteration on which it can be built. These are implicit iterations.

(68) *The first three times she went to France, it was to visit her sister.*

Hence, example 68 contains the same Selection as in example 57, and presupposes the same Iteration as in example 56. Of course, as an implicit Iteration, it is necessarily under-specified. Whereas example 56 provides a cardinality of seven, we do not know precisely the cardinality of example 68 and can only infer that it is above four (under three, "the first three times" would no be correct", and with three, it would be written "the three times" rather than "the **first** three times").

We can note that if the Iteration has been previously introduced, for instance by example 56, the present formulation would be "the three first times, it was...", that is, an elliptic formulation.

In addition, example 68 can be complemented by example 69:

(69) *Allover, she went there seven times.*

which specifies retrospectively the content of the implicit Iteration. Hence, with examples 68 and 69 we get a reverse ordered chronology compared to examples 56 and 57. This shows how interwoven Iteration and Selection are, to the point that we do not pay attention to it while speaking.

This interweaving goes even further insofar as a certain number of syntagms which we have previously considered as Iteration triggers could be considered as exhaustive selectors, such as "each time that". Being exhaustive, they do not differ from the associate Iteration, and so can be considered directly as iterations.

Finally, to see the variety of how the implicit iterations occur, let us observe an example built on a frequency manner.

(70) *One time out of two, he comes with his wife.*

Example 70 presupposes the existence of an Iteration of which "once in two" can be a Selector, and "he comes with his wife" be a modifier. The IterativeModel of the implicit Iteration should therefore generalize (for instance, be an hypernym) "he <to come> with his wife", for instance "he <to come>". As a consequence, we could reformulate example 70 in a more explicit manner, as given in example 71.

(71) *When he comes, on time out of two, he is accompanied by his wife.*

2.11 Framed Iteration

We address now a quite uncommon iterative phenomenon, which consists in building an Iteration the frame of which is given by another Iteration. Although this kind of iteration is very rare, I think it is important to show that the present model is able to handle all cases of Iteration we have faced in real texts.

Let us consider example 72 from "un coeur simple" ("a simple soul") from Gustave Flaubert, with a special focus on the scope of "sometimes".

(72) *She made an arrangement with a livery-stable man who drove her over to the convent every Tuesday. In the garden there was a terrace, from which the view extends to the Seine. Virginia walked in it, leaning on her mother's arm and treading the dead vine leaves. Sometimes the sun, shining through the clouds, made her blink her lids, when she gazed at the sails in the distance, and let her eyes roam over the horizon from the chateau of Tancarville to the lighthouses of Havre. Then they rested on the arbour. Her mother had bought a little cask of fine Malaga wine, and Virginia, laughing at the idea of becoming intoxicated, would drink a few drops of it, but never more.*

There is a first Iteration coming from "every Tuesday" (SteadyCalendar). Then, there is a second Iteration, coming from "sometimes", that we could mistakenly consider either as (1) a nested Frequency Iteration, or (2) as a Frequency Selection applying on the set of iterated Tuesdays. But it is not so.

In case (1), the imbrication would imply, by design, that "sometimes" applies to each Tuesday, which is not true: there are obviously some Tuesdays when the sun is not shining.

In case (2), the Selection would keep among all Tuesdays those, and only those, within which the sun shines through the clouds. This interpretation is better than the first one, but is still not convenient. Indeed, it is possible that within a given Tuesday, the sun shines through the clouds (and disappears) several times, which leads to as many iterates, whereas the selection would create a single iterate for the whole Tuesday (assuming the the sun is shining all day long).

In fact, the Iteration coming from "sometimes", even if it is linked to the first Iteration, is in a much more loose relationship with it than in the case of a sub-iteration or of a selection. As we have just seen, Tuesdays are not all involved, contrary to what would assume a nested Iteration, and for a given Tuesday, there can be several iterates, contrary to what would assume a Selection. This seems to argue in favor of the independency of the resulting Iteration, but there is a third important point not to be missed: what is involved by "sometimes" only concerns the Tuesdays iterated by the first Iteration (we do not talk about the times when the sun shines another day than a Tuesday). Consequently, it is a new configuration, but that can be easily tackled with the present model as follows:

- a first Iteration, called main Iteration, iterates a series of Tuesdays,
- a second Iteration, called Framed Iteration, is built with its own Iterator, in our example the FrequencyIterator coming from "sometimes", and has as Frame the set of iterates coming from the main Iteration.

Hence, there is no need to amend the model to tackle this complex configuration. Whereas until now he had to deal with simple frames (i.e. simple intervals), in Framed Iterations the frame is a set of intervals coming from a set of iterates.

As for Selections, Framed Iterations are meant to enrich the main Iteration: what is said through the IterativeModel of the Framed Iteration concerns (and so, enriches) a part of the main iterates.

2.12 Conclusion

According to the various observed phenomena and to the proposed model, we can now conclude on how an iteration can be constituted. We finally distinguished four levels:

1. Simple Iteration. It is the simplest case, the IterativeModel of which consists in a single ModelEvent.
2. Enriched Iteration. In this case, the IterativeModel is enriched as much as needed, with additional ModelEvents with as many relationships as needed, with respect to the the SdT theory.

3. Supplementations coming from a Selection. We have seen that thanks to Selections, it is possible to amend or supplement any subset of iterates of any Iteration (if needed, recursively).

4. Framed Iteration. We have seen in the previous section an additional mechanism which also makes it possible to amend the content of certain iterates.

Finally, an Iteration may result of the combination of these four processes which work in a very economical way from a linguistic perspective, and provide a rich semantic scope. To give a final illuminating example, let us consider another excerpt from Proust reported in example 73 which involves the first three levels out of the four we have just enumerated.

(73) *(Translated from the French by C. K. Scott Moncrieff). We used always to return from our walks in good time to pay aunt Léonie a visit before dinner. In the first weeks of our Combray holidays, when the days ended early, we would still be able to see, as we turned into the Rue du Saint-Esprit, a reflection of the western sky from the windows of the house and a band of purple at the foot of the Calvary, which was mirrored further on in the pond; [...] . But in summer, when we came back to the house, the sun would not have set; and while we were upstairs paying our visit to aunt Léonie its rays, sinking until they touched and lay along her window-sill, would there be caught and held by the large inner curtains and the bands which tied them back to the wall, and split and scattered and filtered; [...]. But on some days, though very rarely, the chest-of-drawers would long since have shed its momentary adornments, there would no longer, as we turned into the Rue du Saint-Esprit, be any reflection from the western sky burning along the line of window-panes; the pond beneath the Calvary would have lost its fiery glow, sometimes indeed had changed already to an opalescent pallor [...]. Then, as we drew near the house, we would make out a figure standing upon the doorstep, and Mamma would say to me: "Good heavens! There is Françoise looking out for us; your aunt must be anxious; that means we are late.""*

Figure 2.40 shows the implementation of example 73 in the model. There is a main Iteration, containing a succession of ModelEvents such as "to walk", "to return", "to pay a visit". It constitues a common core to the rest of text, but is modified later in various ways. A first level of Selections splits the iterates in two categories depending on whether it is the beginning of the holidays, or it is summer. Then, within the second Selection, a second level of Selection is done corresponding to "very rare" days when the protagonist goes back late in the day.

Iterative phenomena abound in this excerpt, as it is in the whole Proust's works, with recursive Selections, and can be summarized in a tree depiction, as shown in figure 2.41.

Finally, there are four kinds of possible iterates in this excerpt, which cover three levels from (1) to (3):

- (1) the general case introduced in the main Iteration
- (2a) a variation of (1) concerning the beginning of the holidays
- (2b) another variation of (1) concerning the summer
- (3) a variation of (2b), hence a double variation of (1), concerning rare days of the summer

2.13 Perspectives

2.13.1 Enrichment of the model to cope with an evolutionary situation

We have seen that natural language makes it possible to assimilate many situations to a canonical situation (what we have modeled by the Iteration class), while, conversely, providing the possibility

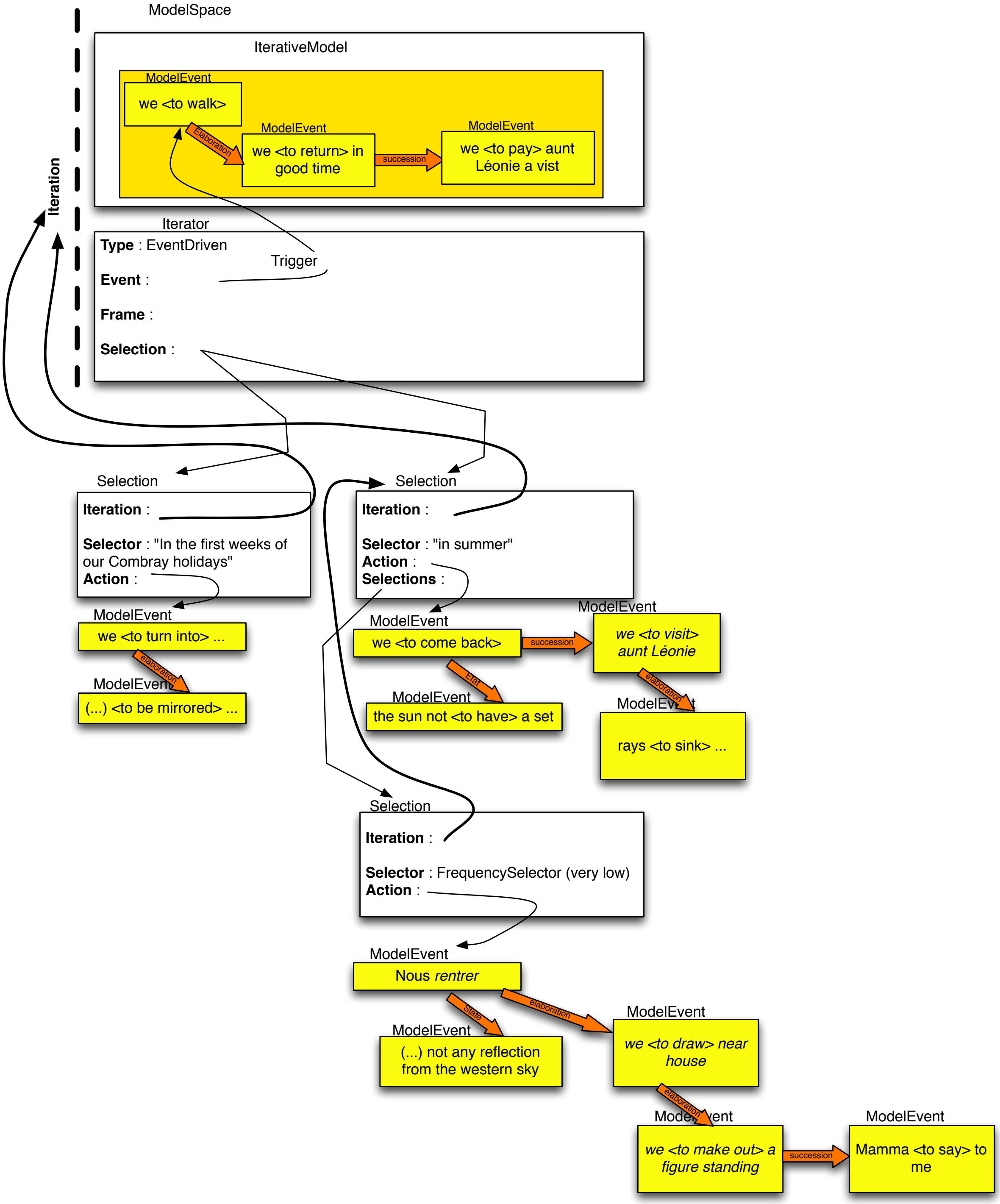


Figure 2.40: Implementing an excerpt from Proust

to give as many specificities as we wish to certain iterates over this canonical scheme (what we have modeled by the Selection class). That is the point we have just illustrated with the rich example of Proust.

Thus, example 74, at first sight, belongs to the present model, with a first Iteration corresponding to the fact that "I <to go> to bed every night", and two related Selections, one corresponding to "<to be> young," the other corresponding to the enunciative period ("now").

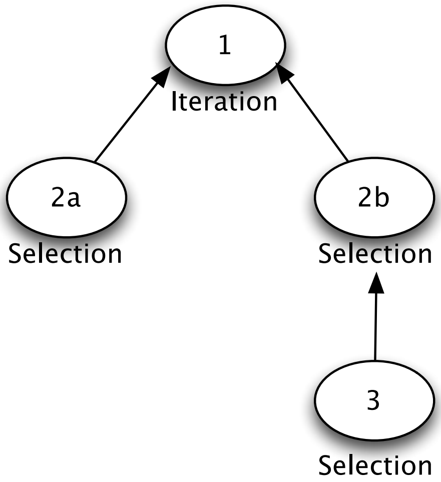


Figure 2.41: A tree depiction of an excerpt from Proust

(74) *In my younger days, I went to bed at 10 pm. Now it's never before 11 pm.*

It might be objected that such a representation is to a certain extent a too literal interpretation of the text, and that a more natural interpretation should deviate from a strict dichotomy made by the two mutually exclusive Selections. In particular, we could assume that there is a gradual drift of the bedtime from 10 pm to 11 pm through years. However, we can also assume that the first selection concerns only the strict youth period of the speaker, whereas the first selection concerns only the very present period, leaving aside the intermediary period with no Selection. Then, we get a correct while under-specified representation of the middle part of the iterates.

However, how can we face example 75 where a gradual change is linguistically expressed?

(75) *I go to bed later and later. From 10 pm when I was young, I am currently going to bed at 11 pm.*

The iterated content still relies on the same main Iteration, but it can no longer be reduced to the only addition of two selections. Indeed, it is no longer a matter of selections of iterates to which common derogations are attributed (for instance bedtime = 10 pm for some, and 11 pm for the others), but a matter of specification of a gradual change of the value of a property over iterates.

In its current stage, our model is not able to provide an entirely satisfying solution. However, I do not think that it is beyond its capabilities. Admittedly, there is a difficulty concerning the IterativeModel: how is it possible to set a value for bedtime in the model which serves as a model for all iterates while each iterate is said to have a different value. But the very notion of IterativeModel remains: we still have access to it in a linguistic way, and it is possible for instance to complement this example by "anyway, I have never stopped to read before sleeping", which refers once again to the whole IterativeModel.

As a consequence, a possible enrichment of the model could consist in providing a way to express the evolution of one or several properties over time, that is, over iterates. To my mind, this phenomenon ressembles the Selection operation, and could be modeled as an extension of it (the current Selection would be a particular case of it). Indeed, whereas a Selection chooses a subset of iterates in order to enrich or modify their content with a common amendment, the point here is to provide individual amendments to each selected iterate which may depend on their rank on the timeline. In a formal way, it would be possible, for instance, to implement the progression induced by example 75 with a function which associates a value of bedtime to each iterate, which would be increasing, starting from 10 for the first iterate, and reaching 11 for the last one. In the case of more complex examples, such as 76, it could be a holistic constraint over the set of selected iterates, which could be rendered by a probabilistic function.

(76) *However, from 1995 to 2002, my bedtime was very erratic.*

The current model of Selection would become a particular case of this enhanced version, insofar as it is possible to use a simple constant function: for a given property, each iterate is given the same value as others.

As we can see, Iteration in natural language is a rich and complex cognitive process, which seems to surf between two seemingly **contradictory sides**, but which are in fact very complementary: the assimilation of different iterates to **a same model** on one side, these iterates being all **unique** on the other side. The current version of Selection is a clear witness of the existence of these two sides, since it creates an intermediary level between them: it provides kind of singularity to a subset of iterates from the generality provided by an initial Model.

The future version of Selection we are currently discussing is a refinement that endorses the versatility provided by natural language to combine universality and singularity at the same time: in the first sentence of example 75, it is explicitly said in the same time, through a few words only, that each iterate relies on a certain IterativeModel, and also that it is different from most of the others.

In addition, we have also to remember that we have tried to reduce as much as possible our definition of iterations, with the particular point that the subject should be sufficiently steady. We have to recognize that in spite of this cautions, there is a wide range of phenomena involved by iterations. For this reason, the possible enlargement of our definition of iteration should be very cautious and progressive in the future.

2.13.2 Towards an automatic processing of Iteration in discourse

While the primary focus of the modeling work proposed in this section was above all to provide a way to represent iterations in the most expressive way, and with the broadest coverage, a longer-term objective is also to provide a contribution to automatic processing of texts containing iterations.

It is in this longer-term objective that took place the PhD work of Lebranchu (2011). This quite ambitious work consisted in studying how it is possible to account for iterative phenomena in language on real corpora while taking into account their discursive dimension (iterations spanning several sentences or even several paragraphs), from a perspective of automatic processing of the iterative representations and of their aspectual features. Two corpora were thus constituted and manually annotated via the Glozz platform (see Widlöcher and Mathet, 2012, introduced later in this dissertation): One contains novels (in extenso or excerpts) and the other contains articles from *Le Monde* (a french newspaper corpus containing 2053 articles). For each of these two genres, it was possible to determine the iterative phenomena in a quantitative way: for example, we saw that 5% of the sentences of these texts are involved in iterative phenomena, and more precisely 7% in novels, and 3% in newspaper articles; But finer observations are also proposed, such as the distribution of iterations according to their number of sentences (is it an iteration carried by one, two, or more sentences?).

This work relies on the present study, and in particular on the representation of the iterative content, with respect to the proposed object model. It reveals the great diversity of textual configurations of iterative structures in corpus: textual discontinuity, imbrication of structures, and so on. A processing chain was then set up, relying on the LinguaStream platform see (see Bilhaut and Widlöcher, 2006), making it possible to annotate the different constituents of iterative structures (events, circumstantial complements, iteration triggers). This is complemented by a semi-automatic process which generates the corresponding data in the object model and their aspectual features with respect to the SdT theory.

This study paves the way, in the longer term, for practical contributions in information retrieval (what are the repeated situations, how often are they repeated, with what variations, and at what times), but also, from a linguistic perspective, to an observation of the iterative corpus (highlighting the iterative passages, the linguistic configurations that were found, etc.) which notably enlarges the size of what can be humanly observed.

Part II

Annotation of continuums and its assessment

Chapter 3

Annotation of continuums with Glozz and GlozzQL

This chapter strongly relies on two articles, in collaboration with Antoine Widlöcher: (Mathet and Widlöcher, 2011a) and (Widlöcher and Mathet, 2012).

Contents

3.1 Introduction **67**
3.1.1 Prerequisites 68
3.1.2 Continuum: first overview 68
3.1.3 Glozz and GlozzQL overview 68
3.2 Why another tool ? **69**
3.3 Underlying model **70**
3.3.1 The Unit-Relation-Schema metamodel 70
3.3.2 Metamodel scope 73
3.3.3 Granularity and topology 73
3.3.4 Standoff representation of annotations 73
3.4 Main features **74**
3.4.1 Polymorphism and heterogeneity of input 74
3.4.2 Several representation paradigms 74
3.5 Customizing and using Glozz **76**
3.5.1 Annotation campaign 77
3.5.2 Annotation model 77
3.5.3 Customizing display 77
3.6 GlozzQL: a query language for annotation mining **78**
3.6.1 Introducing GlozzQL 78
3.6.2 Annotating and querying simultaneously 79
3.6.3 Advanced concepts 80
3.7 Conclusion **81**

3.1 Introduction

A growing number of studies in linguistics, computational linguistics (CL) or Natural Language Processing (NLP) manifest an increasing interest for corpus studies. Through a wide range of approaches, the need for a confrontation between models and corpora makes it necessary to have reference annotations to which linguistic models can be compared. Such reference corpora are also useful for machine learning, to automatically learn models, and for evaluation tasks, to assess the results of NLP systems.

The elaboration of such annotations is a complex process which requires adequate formal grounds, encoding standards and dedicated applications. Despite the availability of several annotation tools, different requirements, especially in terms of abstraction, genericity and ergonomics, were overall not satisfied, as detailed in (Widlöcher and Mathet, 2012, section 2).

3.1.1 Prerequisites

For this reason, we have been studying since 2008 how to improve the annotation process, which has resulted in the design and the development of a whole annotation platform, namely Glozz, that features three fundamental aspects:

- A high level of customization, to fit most annotation campaigns
- Different and simultaneous views on annotated data, because the annotation process is often multifaceted
- Annotating and querying (i.e. creating annotations and looking for what has been already annotated) should be possible at the same time, because we often rely on previous annotations to annotate new ones.

The Glozz platform[1] (Widlöcher and Mathet, 2009), takes these constraints into account and provides a highly configurable environment, usable for corpus annotation and mining of various linguistic phenomena.

3.1.2 Continuum: first overview

For historical reasons, Glozz is designed to tackle texts, that is, an ordered series of characters with respect to a settable encoding. Each character is naturally given an index, in increasing order, which creates what we will call a **continuum**. The main point is that a continuum is not just a set of items (here, items are characters), but, first of all, a mono-dimensional stream which relies on continuous positions. This notion will be further defined in the section about assessment.

An important point is that even though Glozz, in its current form, relies on textual data, all its principles could be extended to any other kind of continuums such as audio or video streams.

3.1.3 Glozz and GlozzQL overview

In order to satisfy the requirements of genericity and to support consequently the annotation of heterogeneous linguistic objects (in terms of structure, granularity...) as reported in section 3.2, Glozz relies on an abstract metamodel presented in section 3.3.

Given a specific linguistic model conforming to this metamodel, locating, identifying and describing linguistic objects in texts require adequate annotation tools. The incremental annotation process, Glozz GUI (presented in figure 3.5), as well as its main annotation features and tools, will be presented in section 3.4 and 3.5.

The annotation process, as well as the subsequent use of annotated data, require the ability to access information featured by the corpus. Glozz provides an easy access to this information through different "navigation" tools and, in particular, by the mean of a powerful query language called GlozzQL, which will be presented in section 3.6.

[1] Glozz was initially developed within the Annodis project citetaln09annodis, supported by the french Agence Nationale de la Recherche (ANR). Glozz has also been supported by the french Contrat de Projet Etat-Région (CPER) and the Région Basse-Normandie. The URL address of its website is: `http://www.glozz.org`.

3.2 Why another tool ?

Despite the availability of several annotation tools (see Widlöcher and Mathet, 2012, section 2), and even if some of them are highly customizable, it must be noted that they do not meet the needs of all varieties of linguistic annotations. In a way, it is necessary to bridge the gap between broad enough formats and too specialized tools. From this point of view, the following limits have to be emphasized:

1. Priority is globally given to the annotation of objects at quite local granularity levels, making it difficult to represent and then to explore, for example, structures at discourse level.

2. Available tools are often restricted to a particular theory or to a specific class of linguistic structures (segments, relations, chains...). Annotation tasks involving heterogeneous structures are then made difficult.

3. When a class of structure (segment, relation...) is available, ergonomic limits may nonetheless make its annotation process uneasy. For example, it is uneasy to express and visualise relations between textual segments, when annotation only consists in the attribution of a same ID to linked elements. A graphical artefact is necessary.

4. Strong constraints often restrict the usage of the available classes of structures. For example, in the case of annotation of textual segments, embedded or overlapped structures are often not allowed or not adequately represented.

5. It may be impossible to represent complex structures using only segments and relations (in particular if relations can only link segments). For example, annotation of enumerative structures (Ho-Dac et al., 2010) or complex discourse units (Asher et al., 2011) makes it necessary to link sub-structures and not only primary textual data. Moreover, linking non-adjacent elements in sub-structures is often needed, and can not be represented by embedding segments.

6. Annotations (segments or relations) are "labeled" to state the relevant information concerning identified objects. Simple tag-sets are not expressive enough to meet requirements of rich annotation models, and it is necessary to implement richer labelling possibilities, using for example feature structure-based models.

The limits mentioned above mainly concern the expressive power of the data model. Other important limits concern the annotation process and the annotation environment:

1. Multiple views (*in situ*, concordancers, trees, graphs, etc.) on the same data are often required. And any of these views should notify any change to other views. However, available tools often give priority to one of these paradigms or feature multiple views which do not "observe" each others.

2. Querying/Mining of annotations should not be considered as a post annotation possibility. Indeed, at any stage of an annotation process, annotators need to rely on some specific existing configurations to which query languages can give access.

Most of these requirements, in terms of abstraction, genericity and ergonomics, had already been implemented in other tools, but not, to our knowledge, in a same tool. The general-purpose Glozz platform takes these constraints into account and provides a graphical and highly configurable annotation environment, usable for corpus annotation and exploration of various linguistic phenomena.

3.3 Underlying model

Due to the diversity of linguistic phenomena, corpus linguistics and NLP studies lead to a variety of models, theories and formalisms. This diversity often results in heterogenous description formats and annotation tools, each approach developing its own framework.

However, deep interactions between the different kinds of linguistic phenomena and paradigms make it necessary to define common frameworks and standards where most kinds of objects, resulting of heterogenous models or paradigms, can be described, in order to compare or combine various approaches.

3.3.1 The Unit-Relation-Schema metamodel

Glozz relies on an abstract metamodel, called URS (for Unit-Relation-Schema), originally coming from Widlöcher (2008), which provides an adequate framework, unrestricted to a particular theory or to a specific class of objects, allowing description of existing or future linguistic models.

This metamodel, represented by the figure 3.1, relies on three abstract categories of *elements*: *units*, *relations* and *schemas* which will be described below.

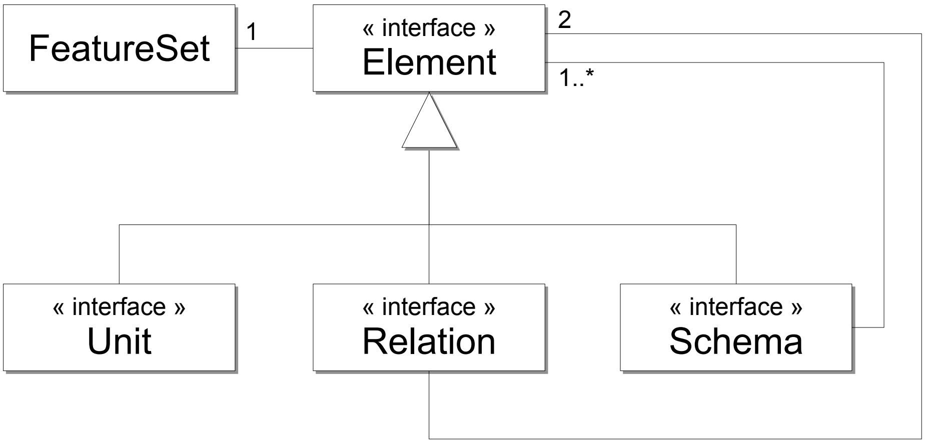


Figure 3.1: UML class diagram of URS

Metamodel and models

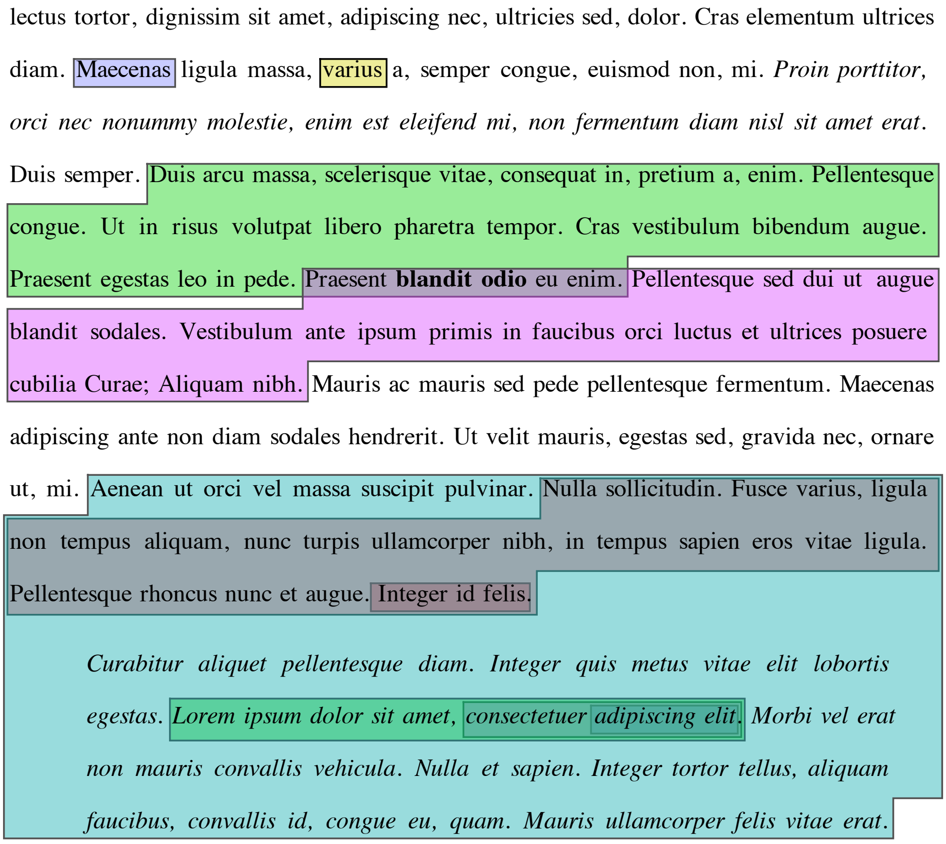
Within the general framework defined by the meta-model, specific models can be expressed, depending on the linguistic theory or approach. Each specific model declares available linguistic object types (identified by the theory) and explicits the way their instances have to be characterized (or labeled). The specialization of URS for a specific campaign will be presented in section 3.5.

Element

All available linguistic objects, called *elements*, may be *units*, *relations* or *schemas*. All of them are characterized by a *type name*, which explicits their linguistic category, and a *feature set*, representing their properties. Type names, expected features for a given type and possible values for these features depend on the specific user-defined model designed for a campaign.

Unit

Units, illustrated by figure 3.2, are textual segments, sequences or spans, of any size.



lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. *Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat.* Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent **blandit odio** eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis.

Curabitur aliquet pellentesque diam. Integer quis metus vitae elit lobortis egestas. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi vel erat non mauris convallis vehicula. Nulla et sapien. Integer tortor tellus, aliquam faucibus, convallis id, congue eu, quam. Mauris ullamcorper felis vitae erat.

Figure 3.2: Units

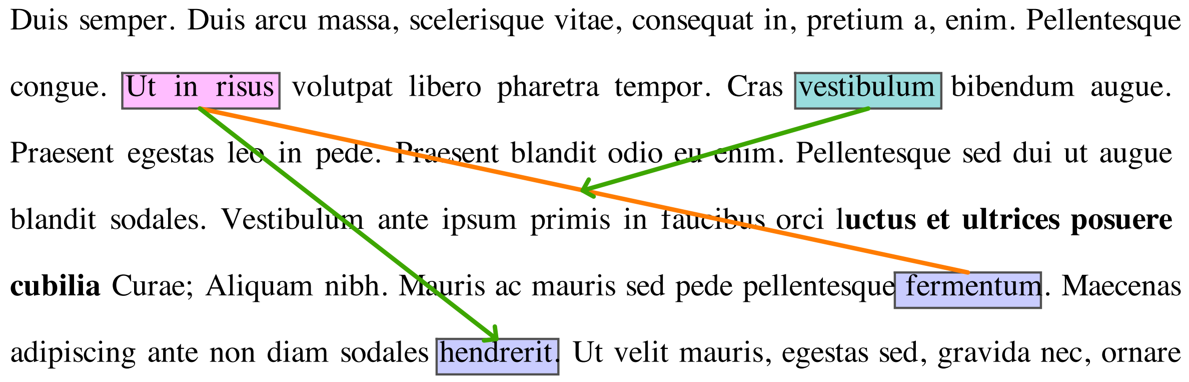
A part of speech annotation task could for example define a unit type *word*, and two features to represent its morpho-syntactic tag and its lemma (the former having a predefined set of possible values). An annotator could then annotate all words of a text, each of them, instance of the type *word* (derived from the abstract meta-type *unit*), having its own tag and lemma values. Named entities, propositions, sentences, topical units, argumentative segments, sections or the whole document give other examples of possible units, at higher granularity levels.

Relation

Relations, illustrated by figure 3.3, designate links (directed or not) between two *elements*.

If relations between units are widely used, it must be noted that the possibility of relations linking whatever elements, including schemas or relations, significantly improves the expressive power of the metamodel.

At a syntactic level, *dependancies* could, for example, be represented by directed relations. At a higher granularity level, a rhetorical annotation task would make use of directed relations to represent *causality* and benefit from symmetric relations to represent *contrast*, between propositions delimited as *units*, or between more complex patterns represented by *schemas*.

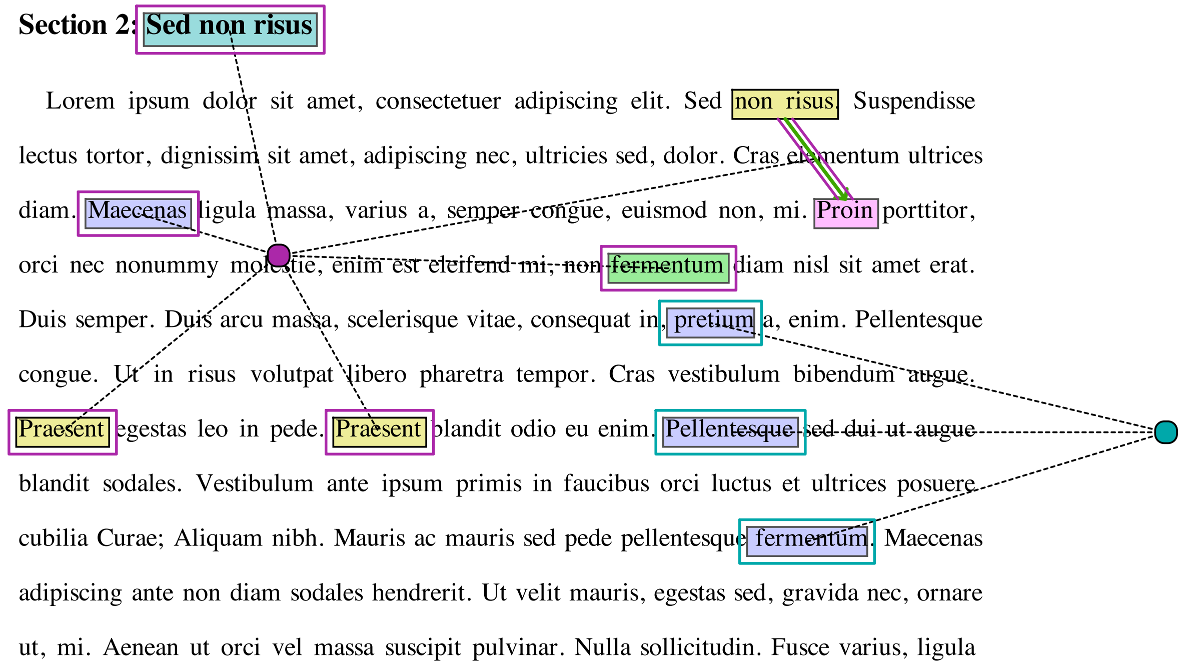


Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci l**uctus et ultrices posuere cubilia** Curae; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare

Figure 3.3: Relations

Schema

If both previous elements are quite common (even if designated otherwise), the schema category, illustrated by figure 3.4, is more original. Schemas are used to represent complex configurations or patterns involving any number of elements (units, relations or sub-schemas).

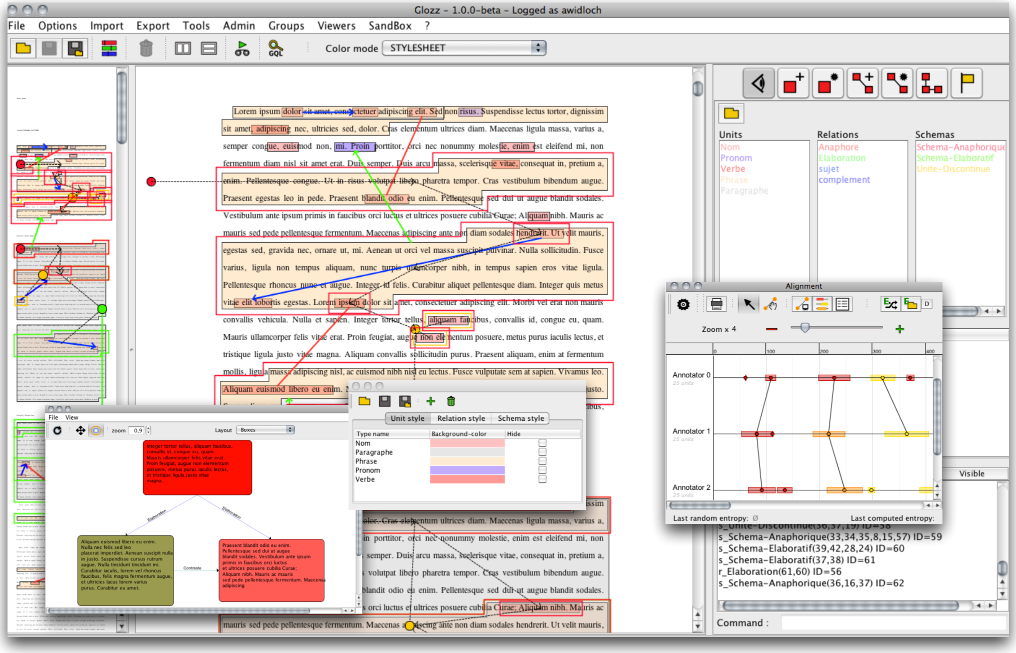


Section 2: Sed non risus

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula

Figure 3.4: Schemas

Coreference chains could, for example, be represented by a set (or a path) of binary relations, all grouped in a schema, whose features could describe the common reference. *Enumerative structures* provide a more complete example. Composed of a set of consecutive *item* units, the enumeration is usually embedded in a larger structure (an *enumerative structure*), introduced by a *header* which is thereby in an *introduction relation* with items. In addition, *inheritance* relations between header and items, and *similarity* or *contrast relations* between items, often complete this quite frequent textual configuration.



s_Unite-Discontinue(36,37,19) ID=58
s_Schema-Anaphorique(33,34,35,8,15,57) ID=59
s_Schema-Elaboratif(39,42,28,24) ID=60
s_Schema-Elaboratif(37,38) ID=61
r_Elaboration(61,60) ID=56
s_Schema-Anaphorique(36,16,37) ID=62

Figure 3.5: The Glozz main GUI

3.3.2 Metamodel scope

This very abstract metamodel enables the representation of very many (if not all) configurations. If linguistic objects can then often be representend within the URS framework, it must be noted that "non-linguistic" information can also be encoded in this way.

For example, Glozz also uses the URS metamodel to represent document structure (titles, section titles...) and typographical information (ordered lists, emphasis...). Thus, with Glozz, everything but the raw text, is an annotation.

Represented in a unified way, all the available information can be used or mined in a unified way, as we will see in section 3.6.

3.3.3 Granularity and topology

The proposed meta-model makes no hypothesis on granularity level of elements or on distance between elements involved in relations or schemas. In particular, it meets the requirements of annotation at discourse level, which are overall not satisfied by multipurpose tools.

Furthermore, this data model accepts embedded and overlapping structures, as illustrated by the figure 3.2. More difficult to represent, the latter are often not well supported by annotation tools.

3.3.4 Standoff representation of annotations

Glozz uses standoff annotations. Units are linked to textual data using position offsets. Relations and schemas refer to objects they link or group.

3.4 Main features

3.4.1 Polymorphism and heterogeneity of input

Annotated texts involve heterogenous data. In particular, in a same text, annotations may: come from different annotators; belong to various granularity levels (word, sentence, paragraph, text, etc.); be related to various linguistic paradigms (syntax, semantics, discourse, coreference, etc.).

An annotation environment should allow such an heterogeneity, and provide adequate ways to deal with it.

Several annotators

For a given annotation campaign, several annotators may add annotations to a same document. In Glozz, each annotator is authenticated, and each annotation is stamped with its creator's identifier. This prevents collisions, and makes it possible to allow or disallow modifications by others, and to filter annotations by authors afterwards.

Granularity

Annotations attached to a same document may concern several levels of granularity. It may be a problem since each granularity level needs a specific modality to work with, particularly in terms of display.

Glozz proposes two simultaneous text views of the annotated document which are respectively set to "macro" and "micro" granularities. Thus, it is possible to have a global view of the document, where macro structures appear, and, at the same time, to have a focused local view, where micro structures can be well represented.

Several linguistic paradigms

It is necessary that several linguistic paradigms (syntax, semantics, etc.) can combine in a same document, because some paradigms may depend on others. However, too many structures at the same time make interpretation uneasy.

With Glozz it is possible to focus on one or several specific paradigms, and to hide annotations that do not belong to them. Indeed, annotation models can define groups of types, and each type can belong to one or several groups (see section 3.5). Users can hide as many groups as necessary.

Several linguistic types

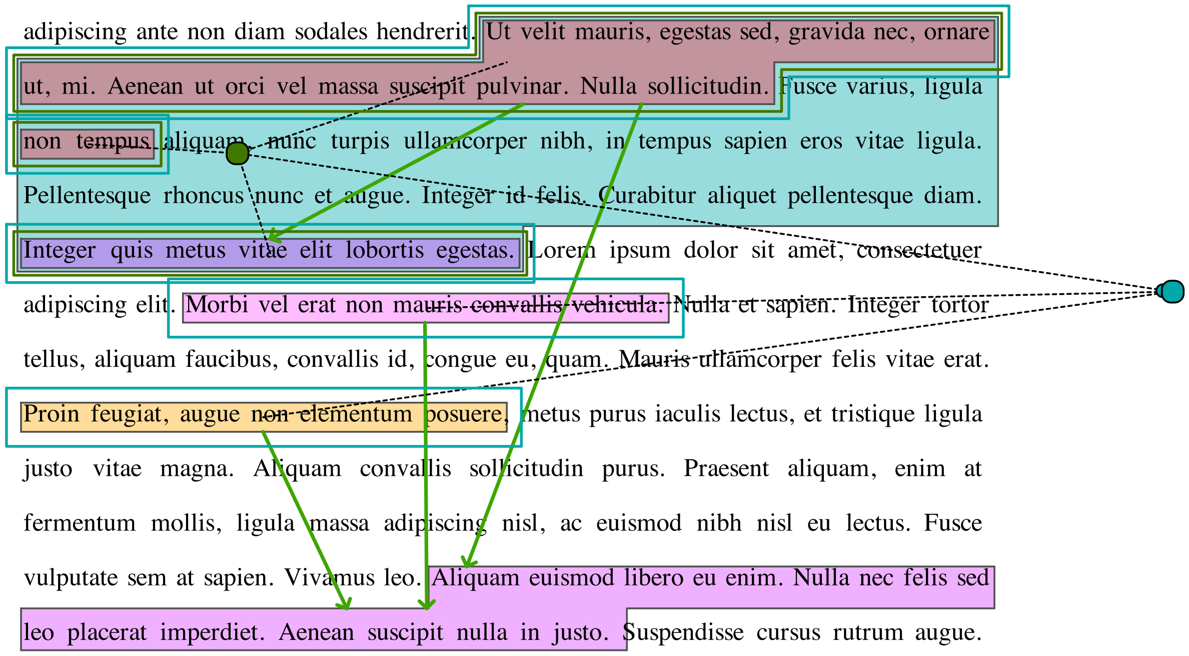
Annotated items, instances of units, relations or schemas, are grouped in types (each type belonging to one or several paradigms). For example, in order to annotate the argumentative structure of scientific texts (this is an annotation paradigm), we could annotate unit objects having *types*: *introduction*, *background*, *state of the art*, *own work*, *experiment*, *evaluation*, *future works* or *conclusion*.

Annotation display, in Glozz, uses a stylesheet. This stylesheet makes it possible to define visual properties for each type, and, if necessary, to hide all instances of a given type.

3.4.2 Several representation paradigms

Different annotation paradigms (coreference chains, argumentative structures, rhetorical relations, etc.) often require different representation paradigms, called here *views*.

Nonetheless, we often need different paradigms at a same time, in a same campaign, hence in a same tool.



adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis. Curabitur aliquet pellentesque diam. Integer quis metus vitae elit lobortis egestas. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi vel erat non mauris convallis vehicula. Nulla et sapien. Integer tortor tellus, aliquam faucibus, convallis id, congue eu, quam. Mauris ullamcorper felis vitae erat. Proin feugiat, augue non elementum posuere, metus purus iaculis lectus, et tristique ligula justo vitae magna. Aliquam convallis sollicitudin purus. Praesent aliquam, enim at fermentum mollis, ligula massa adipiscing nisl, ac euismod nibh nisl eu lectus. Fusce vulputate sem at sapien. Vivamus leo. Aliquam euismod libero eu enim. Nulla nec felis sed leo placerat imperdiet. Aenean suscipit nulla in justo. Suspendisse cursus rutrum augue.

Figure 3.6: Annotations over the text

Annotations over the text, in situ annotation

Of course, the most usual way to annotate texts is to process directly upon them, in order to select and "highlight" identified objects. As shown in figure 3.6, it is possible to add or modify units, relations and schemas through a WYSIWYG[2] interface.

Annotations as a graph

If relations and schemas are integrated in complex constructions, or for specific annotation paradigms, a graph representation is obviously a good way to reveal what would be confused on the flat view of the text. Hence, in figure 3.7, the annotations of figure 3.6 are represented by a graph, where the relations (of *elaboration*) and their interaction with schemas more clearly appear. In this configuration, units are represented by circled numbers and schemas by boxes.

Annotations as predicates

It may be convenient, as well, to read and create annotations directly as predicates, straight expressing, for example, that a relation should exist from annotation 1 to annotation 2, and so on. In Glozz, a module (illustrated by figure 3.8) permanently shows the list of all existing annotations in this way. A prompt may also be used to create new objects, with the help of auto-completion for element metatypes and type names, as well as for syntax checking.

Several simultaneous views

Moreover, a real strength of Glozz is its ability to make all its representation paradigms working at the same time, and together. Indeed, Glozz keeps central control of what is being selected through any

[2] What You See Is What You Get

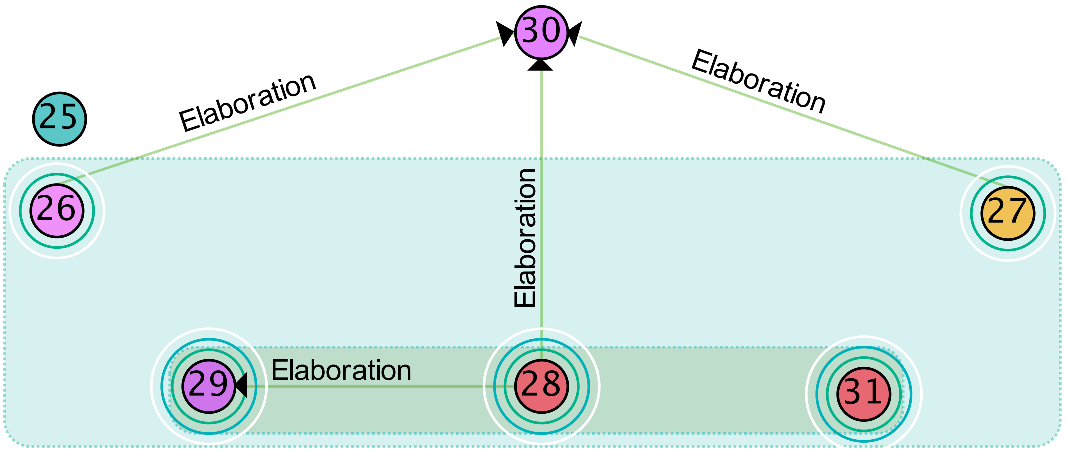


Figure 3.7: Annotations as a graph

```
...
u_Thesis(13260,13627) ID=25
u_Antithesis(13686,13731) ID=26
u_Synthesis(13858,13900) ID=27
s_Thesis/Antithesis/Synthesis(26,27,40) ID=39
u_Introduction(13260,13381) ID=28
u_Exemplification(13580,13627) ID=29
u_Exemplification(14157,14265) ID=30
r_Elaboration(28,29) ID=35
r_Elaboration(27,30) ID=36
s_Elaboration(28,29,44) ID=40
r_Elaboration(26,30) ID=41
r_Elaboration(28,30) ID=43
...
```

Figure 3.8: Annotations as predicates

view, and transmits the selection to all other views. This interaction is illustrated by figure 3.9, where the selection of an object in any of the 3 views selects it in the two other views.

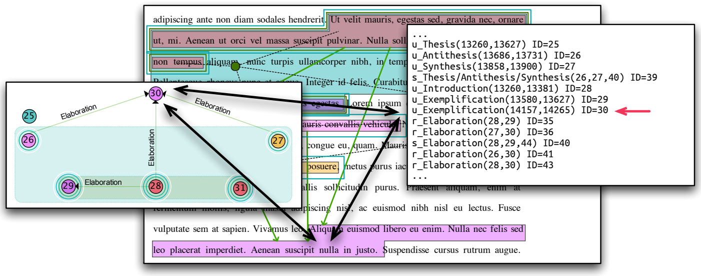
This way, Glozz enhances the annotation process in two ways. Indeed, it makes it possible:

- to observe a same annotation from different points of view, in order to consider, for instance, its exact position in the text on the one hand, as well as its hierarchical position among other annotations, on the other hand;

- to select an annotation by using a first view (the most adequate one to detect the searched object), and modify it by using another one (more adequate to edit its properties).

3.5 Customizing and using Glozz

For a given annotation task, it may be necessary to configure the annotation environment of Glozz.



```
...
u_Thesis(13260,13627) ID=25
u_Antithesis(13686,13731) ID=26
u_Synthesis(13858,13900) ID=27
s_Thesis/Antithesis/Synthesis(26,27,40) ID=39
u_Introduction(13260,13381) ID=28
u_Exemplification(13580,13627) ID=29
u_Exemplification(14157,14265) ID=30
r_Elaboration(28,29) ID=35
r_Elaboration(27,30) ID=36
s_Elaboration(28,29,44) ID=40
r_Elaboration(26,30) ID=41
r_Elaboration(28,30) ID=43
...
```

Figure 3.9: Several views on a same annotation

3.5.1 Annotation campaign

An annotation platform should conform to the requirements of collaborative work. The concept of *annotation campaign* refers to an annotation task involving several annotators sharing resources.

Such resources, built by the *campaign managers*, and distributed to each annotator, include:

1. texts to be annotated;
2. one or several annotation models, to work with different paradigms;
3. one or several stylesheets to configure "points of view" on the data;
4. filters (see section 3.6.1), either to check the reliability of the current annotation, or to bootstrap higher order annotations.

3.5.2 Annotation model

For a given annotation task, a specific annotation model is defined, conforming to the metamodel URS, which declares available types of units, relations and schemas. This *ad hoc* model also specifies the way of describing each instance of these types, by means of a feature set, and expresses constraints on the possible values for each feature. Available element types may also be grouped in categories or levels (see below and section 3.4). Relations may be declared oriented or not.

Conforming to this specific model, annotators can locate instances of these types in corpora, and feed or select adequate feature values.

3.5.3 Customizing display

There are several ways in Glozz 1) to select annotations to be shown, and 2) to configure the way they are represented:

Filtering. Three options are given: 1) using style setting of types, since one of the style properties is visibility; 2) switching visibility of a whole group declared by the annotation model; 3) (temporary) switching visibility of individual annotation instances.

View settings. Type styling consists in choosing a color (for background or edge), and, for schemas, a shape. Several stylesheets can be defined for a same annotation model, in order to adopt successively different points of view on a same data.

3.6 GlozzQL: a query language for annotation mining

The annotation process, as well as the subsequent use of annotated data, requires the ability to access information featured by the corpus. Of course, this information concerns raw textual content, but it is also necessary to give a convenient access to linguistic (morphologic, syntactic, semantic, etc.) or infra-linguistic (document structure, typographical data, etc.) information, which may result from preliminary annotation (manual or computational) steps.

Annotating is an incremental process, which requires to take current annotations into account, in order to produce new ones. At a given stage of an annotation task, annotators need to find some specific configurations of annotations from previous steps (e.g. relations of a given type, units linked to other units by a relation of a given type, and so on).

Besides, it may be very helpful to check that all annotations conform to the annotation directives. Hence, the ability to locate non-valid configurations while annotating is a very convenient way to do so.

Glozz provides such facilities within annotation tasks by the means of some basic tools (not presented here), and a more advanced one, called GlozzQL.

3.6.1 Introducing GlozzQL

Principles

GlozzQL (Glozz Query Language) is a language dedicated to Glozz annotations, and comes with an associated engine.

It is designed to select in a corpus each instance of element (unit, relation or schema) that satisfies expressed constraints. Requests are built piece after piece, in an incremental manner, using two interdependent concepts: *Constraint* and *Constrained-Annotation*.

Constraint: A constraint expresses one condition an annotation must satisfy in order to be selected. They are classified in 4 categories, depending on their domain, i.e. the kind(s) of element they concern (units, relations, schemas, any of those).

Constrained-Annotation: This simple concept refers to a set of annotations (of a given corpus) that all satisfy a given constraint. For a given text, and depending on its associated constraint, a Constrained-Annotation contains 0 to n entities.

Examples

To get an idea of GlozzQL expressivity, let us mention some incremental possible queries:

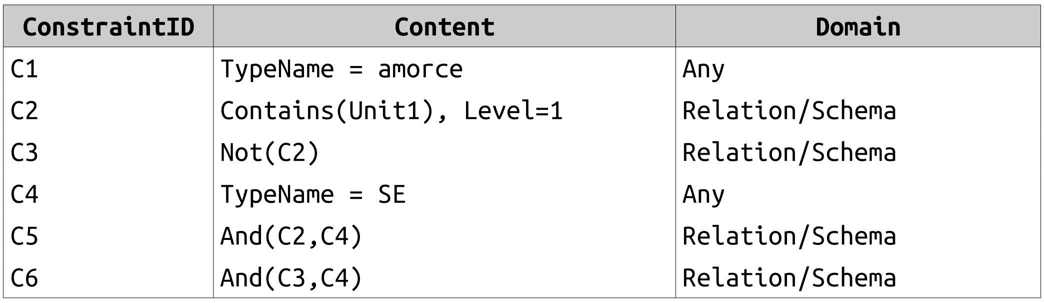
- getting all Units of a given type (this set is called U1)
- getting all Units from U1, with a given value for a given feature (this set is called U2)
- getting all Relations whose target is an element of U2 (this set is called R1)
- getting all Schemas containing a relation among R1, with a maximum depth of 3 (this set is called S1)
- getting all Schemas belonging to S1 and from a given annotator

How it works

A more complete explanation of this system is provided in (Mathet and Widlöcher, 2011a). Let us only introduce here one real-world example coming from the Annodis project (Péry-Woodley et al., 2009), in order to give an overview of the principles.

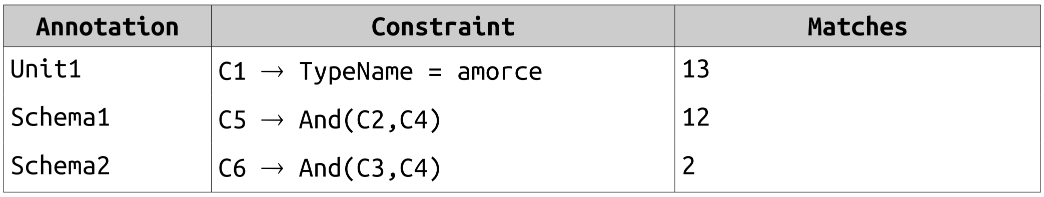
We need to find all schemas having *SE* type (french acronym of Enumerative Structures) embedding a unit having *amorce* type (french name of the header of an enumerative structure), as well as all the *SE* not embedding one.

To do so, we have first to define a Constrained-Unit which represents all *amorce* units. This is done by `Unit1`, in figure 3.11, which relies on `C1` Constraint, in figure 3.10.



ConstraintID	Content	Domain
C1	TypeName = amorce	Any
C2	Contains(Unit1), Level=1	Relation/Schema
C3	Not(C2)	Relation/Schema
C4	TypeName = SE	Any
C5	And(C2,C4)	Relation/Schema
C6	And(C3,C4)	Relation/Schema

Figure 3.10: Constraints



Annotation	Constraint	Matches
Unit1	C1 → TypeName = amorce	13
Schema1	C5 → And(C2,C4)	12
Schema2	C6 → And(C3,C4)	2

Figure 3.11: Constrained annotations

`C1` is a constraint which concerns the type name, which must be *amorce*. This constraint may be used with any kind of annotation (which is mentioned by its domain `Any`), hence with units. At this stage, having declared `Unit1` already implies a mining process : as stated in figure 3.11, 13 units fitting `C1` were found.

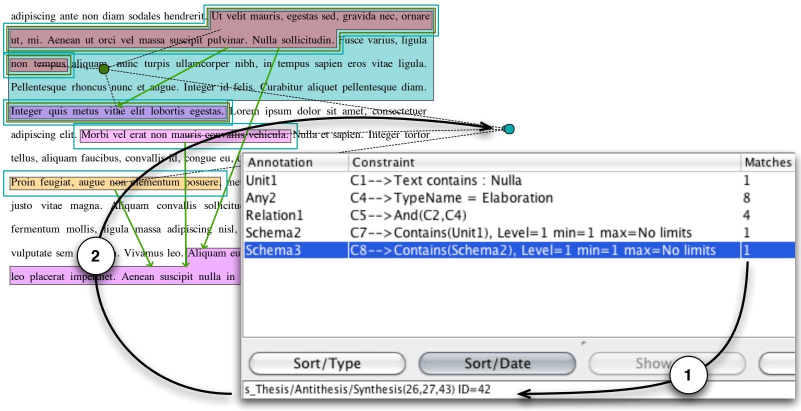
The second step consists in building `Schema1`, which represents all utterances of the first kind of searched schemas, that is to say schemas a) containing an utterance of `Unit1`, and b) having *SE* type. We express two preliminary constraints `C2` and `C4` respectively for a) and b). Then, a logical `And` constraint named `C5` is built over `C2` and `C4`. As a consequence, we get, with `Schema1`, 12 utterances of *SE* containing an *amorce*.

Then, we do the same to find all utterances of *SE* not containing any *amorce*, through `C6` built over `C3=not(C2)` and `C4`. We get 2 utterances.

Hence, we have discovered that in the annotated text, 12 *amorce* units out of 13 are contained in a *SE* schema, and that 2 *SE* schemas out of 14 do not contain any *amorce*.

3.6.2 Annotating and querying simultaneously

As already mentioned, it is helpful to query what is currently being annotated, and to be able to go from query results to current annotations.



adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare
ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula
non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula.
Pellentesque rhoncus nunc et augue. Integer id felis. Curabitur aliquet pellentesque diam.
Integer quis metus vitae elit lobortis egestas. Lorem ipsum dolor sit amet, consectetuer
adipiscing elit. Morbi vel erat non mauris convallis vehicula. Nulla et sapien. Integer tortor
tellus, aliquam faucibus, convallis id, congue eu,
Proin feugiat, augue non elementum posuere, me
justo vitae magna. Aliquam convallis sollicitu
fermentum mollis, ligula massa adipiscing nisl,
vulputate sem . Vivamus leo. Aliquam eu
leo placerat imperdiet. Aenean suscipit nulla in

Annotation	Constraint	Matches
Unit1	C1-->Text contains : Nulla	1
Any2	C4-->TypeName = Elaboration	8
Relation1	C5-->And(C2,C4)	4
Schema2	C7-->Contains(Unit1), Level=1 min=1 max=No limits	1
Schema3	C8-->Contains(Schema2), Level=1 min=1 max=No limits	1

s_Thesis/Antithesis/Synthesis(26,27,43) ID=42

Figure 3.12: Querying while annotating

It's the reason why GlozzQL is integrated to Glozz, and interacts with it as shown in figure 3.12. On the right of the figure, is the list of all queries. We can click on any of them to make the list of its results appear, just below (arrow 1). Then, a click on one result will select it in the main interface of Glozz (arrow 2). We can immediately see this object in its context and we are able to modify or delete it if needed.

3.6.3 Advanced concepts

The main principles we've just introduced make it possible to build requests as complex as needed, recursively. Additionally, two advanced constraints concepts enable to do more restrictive selections just activating relevant options.

Double way constraints First advanced option consists in providing some of the constraints with enhanced semantics. To make it short, let us consider an informal example as shown in fig.3.13 with: `Unit:R1=is-a-circle` and `Schema1:R2=Contains(Unit1).`

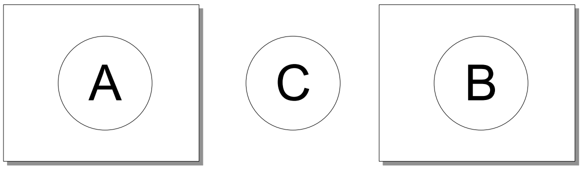


Figure 3.13: Double way constraints

With standard semantics, Unit1={A,B,C}, whereas with double-way semantics, Unit1={A,B}. Indeed, with this option, not only Unit is constrained by its own constraint R1, but also by the one of Schema1 for the reason it is embedded in it : to belong to Unit1, it is also necessary to be contained in (at least) one utterance of Schema1. Hence, it is possible to double the semantics of the constraints with no additional writings. To illustrate this option, let's mention that in fig.3.11, the matches of `Unit1` would drop from 13 to 12.

Unification mechanism The constraints semantics of GlozzQL is originally built on sets : for a given utterance of schema `s`, `s` belongs to `Schema1` if it does exist (at least) one `u` that belongs to `Unit1` such that `s contains u`. Consequently, when several ConstrainedAnnotations are linked via their constraints, only sets are linked, not their utterances. Let us assume that we are looking for a configuration of 3 units linked in a triangle pattern. Assume that A, B, C, D, E and F in fig.3.14 all belong to Unit1, Unit2 and Unit3 sets.

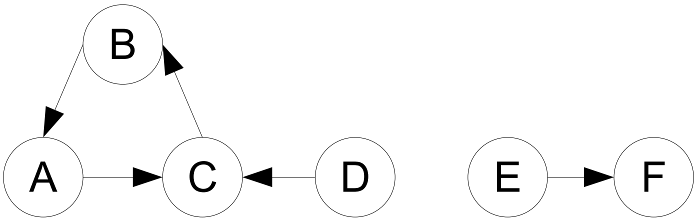


Figure 3.14: Unification

If we write (here, in an informal way): `Unit1 is-linked-to Unit2, Unit2 is-linked-to Unit3, Unit3 is-linked-to Unit1`, this will result in selecting not only A, B and C, but also D, because D is linked to C which is linked to B which is linked to A, and both D and A belong to Unit1. At this stage, these constraints are not relevant to find triangle patterns only (even if E and F were rejected, D wasn't).

However, when using "unification mechanism" option, all ConstrainedAnnotations (Unit1, Unit2 and Unit3 in our example) are considered as mathematical variables involved in equations. Hence, D will no longer be considered as a relevant unit, and the whole semantics of the constraints is strongly enhanced.

Basket(s) Using GlozzQL results in getting a certain number of sets of annotations, as many sets as ConstrainedAnnotations are defined. A special feature makes it possible to collect annotations from any of these sets in a new set called "basket" (in fact, one of the several available baskets). The same basket may collect results from different sets. At any moment, the current basket can be stored as a new annotation file, or reversely it is possible to require that the content of the basket is removed from the current annotations. In order to illustrate this feature, let us just mention that it is possible, in a very simple way, to separate a multi-annotated text (from several annotators) to as many mono-annotated texts as the number of annotators. It is also possible to add or to modify some features to each annotation of the whole content of a basket through a single request.

3.7 Conclusion

It is not possible to introduce the whole content of Glozz and GlozzQL in this dissertation. In addition to the two articles cited at the beginning of this chapter, the reader can refer to three documents I have written which describe Glozz, GlozzQL and a third related tool not mentioned here, the Concordancer. They are available at `http://www.glozz.org/`.

In its current state, this platform has a main limitation: It can handle only a single text at the same time. In the future, it would be helpful to have the opportunity to work on a set of texts, in particular in order to make requests whose scope is the whole set of texts rather than a single text.

This tool in widely used in the community, worldwide. At the date of June the first 2017, it has been downloaded 1189 times, and we have been required 267 logins, corresponding to active users (who create annotations). Interestingly, the designers of ANALEC (Landragin et al., 2012), a powerfull tool devoted to the dynamic annotation of textual data, decided to rely on the Glozz annotation model, and to use the same storage format. Moreover, the delivery of this tool has been a way to create links with other research teams, and last but not least, has triggered our involvement in assessment of annotations which will be introduced in the next sections.

Chapter 4

Assessment of annotations - Part I: Understanding and assessing agreement measures

This chapter mainly relies on two articles: (Mathet et al., 2012), which provides a method to compare and assess agreement measures, and (Mathet and Widlöcher, 2016), a french written article which is a methodological study about assessing annotations.

Contents

4.1 Main concepts **84**
4.1.1 Continuum, annotation, localization, characterization 84
4.1.2 Gold standard, or reference 85
4.1.3 Multi manual annotations 85
4.1.4 Agreement measure 85
4.1.5 Validity measure 86
4.1.6 Links between the different concepts 86
4.2 Chance correction: a necessity and a difficulty **87**
4.2.1 An undeniable necessity 88
4.2.2 First principle: perform a change of referential 88
4.2.3 Estimating chance agreement: several conceptions, and a legitimate debate . 89
4.2.4 An anti-distribution-driven conception of chance 91
4.2.5 Discussion about models of chance 91
4.3 Benchmarking and understanding agreement measures: the Corpus Shuffling Tool (CST) **92**
4.3.1 Introducing the CST 93
4.3.2 Protocol 93
4.3.3 Interlaced paradigms 94
4.3.4 Errors versus discrepancies 95
4.3.5 Results 95

In CL and NLP, the question of assessment of annotations has been receiving a growing interest for more than 15 years. It is now out of the question to publish any annotated resource without some figures supposed to render its validity. For this purpose, agreement measures are often employed, mainly coming from other domains than NLP, and their use in our fields of study has been largely described and commented, notably in the survey article by Artstein and Poesio (2008).

However, even if people increasingly make use of assessment measures, they do not always know which measures, if any, are appropriate to assess their annotations, and do not always understand what their results really mean. Moreover, the question of how to assess the quality of annotations is still subject of debate, when not simply ignored.

We have addressed these questions since 2009 in quite a particular way. It started with the need to get agreement measures to assess Glozz annotations in the context of a research project, namely (Labadié et al., 2010). Then, we have started to design a new family of measures, the Gammas, within the GREYC laboratory, and soon after started a joint reflection about agreement measures at a larger scale with all the authors of the article (Mathet et al., 2012). A collaboration with Klaus Krippendorff also started in 2013, which has resulted in mutual comparisons between our approaches, and in the joint article (Krippendorff et al., 2016). During this period, we have gained a solid experience on how agreement measures work and what their current limitations are, which we have shared in the article (Mathet and Widlöcher, 2016).

In this chapter, we will first see what the main concepts are (in particular what the nature of annotations can be), what the main principles of agreement measures are, including the so-called "chance correction", the main methodological flaws one can encounter and we will introduce a method and a tool (the CorpusShufflingTool) to improve the understanding of what the different agreement measures really do. We will introduce the new family of Gamma measures in the next chapter for more clarity.

4.1 Main concepts

4.1.1 Continuum, annotation, localization, characterization

In Computational Linguistics (CL) and Natural Language Processing (NLP), the initial data we are concerned with is, most of the time, a mono-dimensional **continuum**, whether it is textual, audio or video data.

Texts are made of contiguous characters, that is, a character stream which forms words, then sentences, and so on, characters upon which one can define positions, either from the positions of the characters themselves (from position one to the number of characters), or from the frontiers between characters (position zero is just before the first character, position one is between first and second character, and so on).

Speech involves an audio stream where one can recognize phonemes, words, sentences, and also silences.

In video, we may hear people speaking, watch their corresponding facial expressions and gestures over time, in a rich multi-layer stream (since audio generally goes along with video). Basically, the video stream itself is composed of frames which are organized through a timeline. Whatever the way we consider this stream, discrete (which is necessarily the case with computers) or not, steady or not (i.e. whether all the frames have the same duration or not), the point is that frames are temporally contiguous, and their temporal bounds constitute ordered positions.

Hence, we decide to define a **continuum** as a mono-dimensional stream, whatever the matter it conveys, this matter being discrete or not. The main point for any continuum is that it bears positions, ordered and contiguous. This definition is different from the mathematical definition (which relies on the difference between discrete and non discrete data, for instance reals versus integers), and relies on the etymology "continuous", which bears the notion of contiguity of elements.

This continuum constitutes the context in which annotators have to identify occurrences of the phenomenon under study, and their characterization. The **annotation** process hence corresponds to two steps :

1. **locating** occurrences on the continuum

2. associating to occurrences representations meant to **characterize** them.

These two steps are not necessarily the responsibility of annotators. In some cases, step one is provided, that is to say occurrences are already located on the continuum, and annotators only have to characterize each of them. For instance, verbs have already been identified and located, and the task of annotators is to choose a semantic value for each of them. But in other cases, annotators have to fulfill both steps, which makes both the annotation task and its assessment more complex.

Then, a **unit** is defined as a continuous part of a continuum, that is, formally, by a pair of positions (in increasing order) stating the beginning and the end of a part of a continuum. It corresponds to any element of interest with respect to an annotation task. We make absolutely no assumption about the granularity of units. In addition, a unit is given a characterization (a category, possibly accompanied by feature sets, etc.).

Now that units have been introduced, and following the terminology of Krippendorff (2013a), we will call step one **unitizing**. This task is very different from usual categorization of predefined items, since the annotator is free to identify as many units as she wants, and to choose their corresponding positions. This somehow corresponds to what a reader does with a highlighter pen over a clean text when she decides to highlight some parts (words, sentences) of particular interest. However, since we consider that units have not only a position but also a category, we enlarge the definition of unitizing so that it includes not only the identification of units, but also the categorization of each of them (a reader may use highlighter pens of different colors).

4.1.2 Gold standard, or reference

Computational Linguistics and NLP need trusted data in order to test linguistic hypotheses or to assess automatic systems. These annotated data, which serve as a **reference**, are called **gold standard**. This reference is meant to reflect the "reality" of the phenomena under study, or, if this reality cannot be achieved, at least to reflect the consensual understanding of the phenomena by the community, at a given time. It is by reference to a gold standard that annotations are validated, with the underlying idea that validity means adequacy to "reality". As mentioned by Krippendorff (2013a), it is important that a gold standard accounts for the whole set of phenomena under study, which is difficult to ascertain when facing an entirely new task.

4.1.3 Multi manual annotations

As we have just seen, assessing annotations requires the availability of a gold standard. However, when facing a new annotation task, we do not get any, nor, sometimes, any expert to build one. Moreover, it is not sure at this stage that the task is formally well described (possible problem of fuzzy categories for instance) and consistent (for a given input, only one possible output) so that a reference could be built.

An usual practice to get at the same time both the evidence that the task is consistent and the corresponding gold standard, is to do a multi manual annotation of the same corpus: each human annotator is provided with the same instructions and the same corpus, and annotates independently from others, with the underlying assumption: if annotators all produce (almost) the same annotations, i.e. largely agree, their annotations are likely to be valid, and so, constitute a reference (provided that there is a final correction). This assumption needs to be discussed, as we will further.

4.1.4 Agreement measure

An agreement measure is meant to measure the degree of consensus reached by several annotators who annotate the same document. Such a measure compares annotations whose degree of validity is unknown to other annotations whose degree of validity is also unknown. By not including reference data in its computation, an agreement measure cannot, by design, ensure that annotations are valid. On the other hand, it is generally agreed that a high agreement value ensures a high level of reproducibility ((see for instance Krippendorff, 2013a)), which means that if annotators agree on a part of the data, they should agree on the whole data. As a consequence of reproducibility, once a high agreement is achieved on a part of the data: It is no longer necessary to have a multi-annotation of the rest of the data. We can have each part of the remaining data annotated by a single annotator. As we will see later, it is important that such measures take into account the part of agreement resulting of "chance" (defined later).

The survey article (Artstein and Poesio, 2008) provides a very wide picture of agreement measures for CL, and explains in details how they work.

4.1.5 Validity measure

Once a gold standard is available, it is possible to assess the validity of what is produced by a system or by humans by means of different measures, kind of distances between the gold standard and the productions. This can be the percentage of correct responses in the case of categorization of predefined items, or measures such as recall, precision and f-measure in the case of identification of elements, or some more complex measures in such cases as topic segmentation, discourse relations identification, or reference chains. When such measures provide a score of 100%, this means that the system perfectly does what it is meant to do. Besides, a system which gets a better score than another one can be considered as better than it.

Contrary to agreement measures, validity measures have no need to use any "chance" correction : since the comparison is done with a reference, the obtained score is the one we may expect on other data, and really accounts for the performances of the system. This is one great difference between reproducibility and validity. However, it is desirable to have a baseline which indicates what is the gain offered by the system compared to a random system, a naive system, or, better, already existing other systems.

4.1.6 Links between the different concepts

Now that the main concepts have been introduced, it is important to understand how they relate together. First of all, figure 4.1, freely inspired by Krippendorff (2013a), where the center of a target is a metaphor of the correct annotation, depicts the difference between agreement and validity: disagreement limits validity (cf. B *versus* D), but agreement does not ensure validity (cf. C). It also illustrates the fact that the absence of reference (in the right sub-figure) hides the differences between very different configurations (A *versus* B and C *versus* D).

Generally speaking, 4.2 depicts the links between the four concepts. In order to obtain a reference, it is possible and usual to process to a multi-annotation. The reproducibility value of the latter is assessed by an agreement measure. If it is high, the multi-annotations will serve as a basis to build a reference annotation, by means of a given strategy[1]. This reference is then used jointly with a validity measure to assess the output of a system. It is important to bear in mind that agreement and validity are two distinct elements in terms of both their inputs and how they work.

[1] (1) majority strategy: we retain the category which gets the most choices, with the problem of possible equality, and the problem of poor majority. (2) unanimity strategy: we keep only items which get full agreement, with the problem that only easy items are kept, but creating a biased reference only composed of "easy items". (3) collegial revision: each item not reaching unanimity is collectively revised, probably one the best strategies, used for instance in (Péry-Woodley et al., 2009)

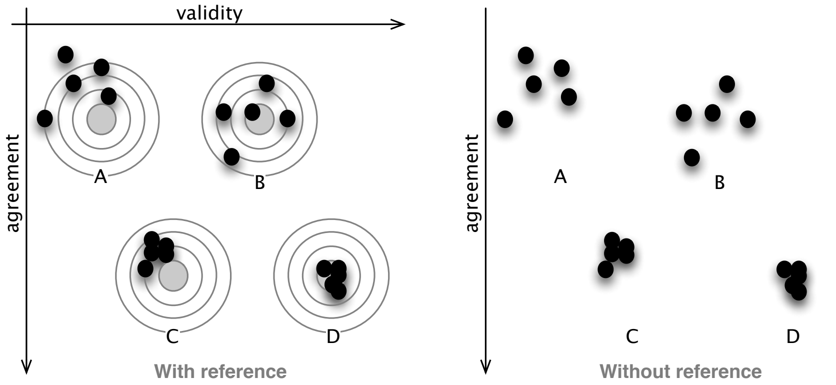


Figure 4.1: Validity *versus* agreement

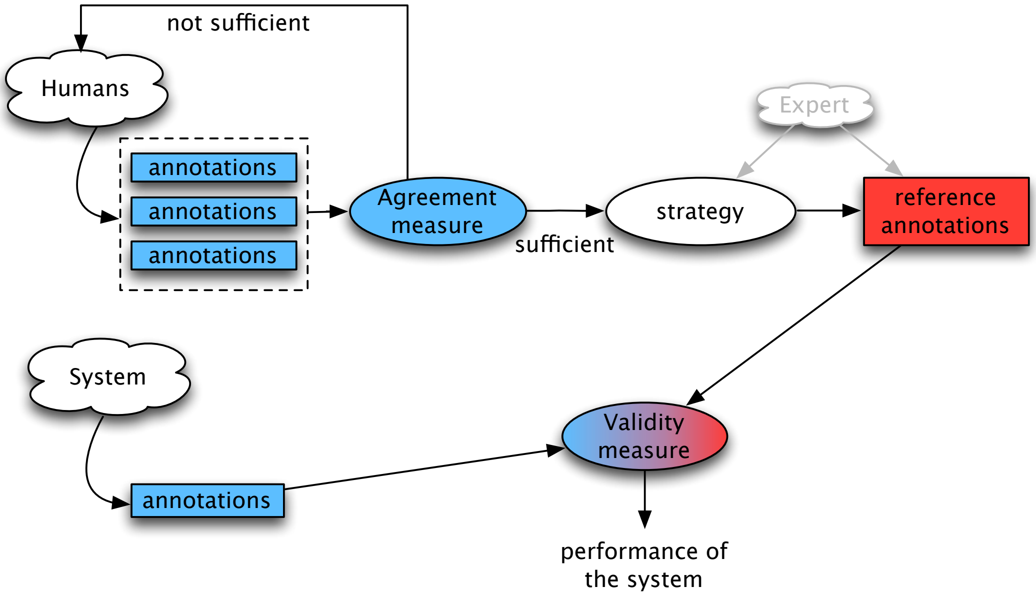


Figure 4.2: Links between the different concepts

4.2 Chance correction: a necessity and a difficulty

Agreement measures should take into account the part of agreement coming from "chance" (we usually talk about "chance corrected" measures). This is a concept used by many measures, but rarely clearly defined. This is not just a refinement of the computation, but a necessary requirement in order to know to what extent multi-annotations are reproducible.

We will consider that "chance" corresponds to an agreement between two annotators which results from an uncontrolled action of at least one of the two annotators. For instance, one of them selects accidentally another category than the one he has chosen (which accidentally corresponds to the one of the other annotator, whether it is a correct choice or not), or both annotators make the same mistake which results in an agreement (sometimes, the mistake may lead to the correct annotation, but for bad reasons, which should be considered also as chance, kind of correct agreement by chance). We will use the term "fortuitous[2]", or the expression "obtained by chance".

What we have to compute is the agreement exclusive of chance, that is, the part of agreement resulting from the feasibility of the task, the correct understanding of the instructions, and the correct

[2] in its regular acceptation, that is here, with no apparent logic with respect to the annotation task.

interpretation of the data to annotate. Indeed, if we consider two annotators whose raw agreement is not better than if they were annotating randomly (for instance by rolling the dice), their behaviour would be totally independent from the annotation task they are supposed to accomplish. An agreement measure should rate them zero. This is a tricky aspect often discussed in the literature, from the first reservations from Feinstein and Cicchetti (1990) to the recent controversy between Zhao et al. (2013) and Krippendorff (2013b), that we will investigate in detail. As we will see, the theoretical level of raw agreement corresponding to chance is usually called "expected agreement", which bears a strong mathematical connotation, suggests that in such a case one can assimilate the annotators' behavior to random variables. This is a strong assumption, which may be interesting to try to find a way to compute the chance agreement value, but which does not entirely account for what agreement by chance really is: an uncontrolled action cannot necessarily be described in terms of randomness. As far as I know, this point is not discussed in the literature.

4.2.1 An undeniable necessity

Often discussed, this classical conception of agreement is vigorously contested by Zhao et al. (2013) who claim that it deliberately assumes that annotators are dishonest and partly annotate at random. Before we address the question of chance, let us first observe an example where the raw agreement value cannot be interpreted straightforwardly, since it cannot achieve zero: three annotators annotate a series of items with two possible categories A and B. Even if annotators made collusion to try to never agree, they would automatically get at least 33.3% of agreement, because if two annotators disagree, the third one has no choice to agree either with the first or with the second. Hence, the degree zero of agreement really starts here at value 33.3%. But more generally, taking into account the part of agreement coming from chance is a necessity in order to interpret the results: As soon as annotators do not perfectly agree, we can conclude that at least a part of their annotations is not produced in a controlled way, because they cannot disagree and all be consistent with what is expected. Henceforth, each time they do not have total control, there is a probability that they agree accidentally, for instance (but not exclusively) because they make the same mistake simultaneously. Clearly, if there are two annotators and two categories, and assuming that the theoretical degree where annotators would annotate blindly, they would achieve a 50% agreement value. In this context, a raw agreement value of 75% would be only halfway from chaos to perfect agreement. As a consequence, taking chance into account is a necessity: an annotation campaign cannot rely on raw agreement value.

4.2.2 First principle: perform a change of referential

Agreement measures first compute a raw measure, which is usually called "observed agreement", denoted A_O. In the simplest context of categorization of predefined items by two annotators, it may be the percentage of items for which the annotators have chosen the same category. As such, except it is 100%, this raw value is not informative about the reproducibility of the annotations, as we have just seen.

Taking chance into account consists in estimating the value A_e (for *expected agreement*) intended to correspond to the part of chance, and to perform a change of referential so that value zero means no correlation between what annotators do and what they are supposed to do, and 1 means a total agreement. This change of referential is provided by equation 4.1 common to most agreement measures:

$$A = \frac{A_o - A_e}{1 - A_e} \tag{4.1}$$

Let us mention that some agreement measures, such as α from Krippendorff (2013a) or γ from Mathet et al. (2015) are built over computation of disagreements instead of agreements, and so rely on equation 4.2 which is equivalent to the previous one, by considering that disagreement values denoted D_o and

D_e are complementary values of A_o and A_e. Given this equivalence, our discussion will rely only on values A_o and A_e.

$$\left.\begin{array}{l} A_o + D_o = 1 \\ A_e + D_e = 1 \end{array}\right\} \Rightarrow A = 1 - \frac{D_o}{D_e} \tag{4.2}$$

This change of referential is illustrated by figure 4.3. We observe that this new referential leaves unchanged value 1 corresponding to perfect agreement, but shifts value 0 to the point allegedly corresponding to chance, that is, formally, A_e. In this figure, which corresponds to our last example (two categories), we observe that despite $A_o = 0.75$, A is only 0.5 (i.e. the middle of the new referential). In addition, let us mention that negative values are possible as soon as annotators perform worse than chance, which means a part of systematic disagreement, and may occur for instance when one annotator inverts two categories.

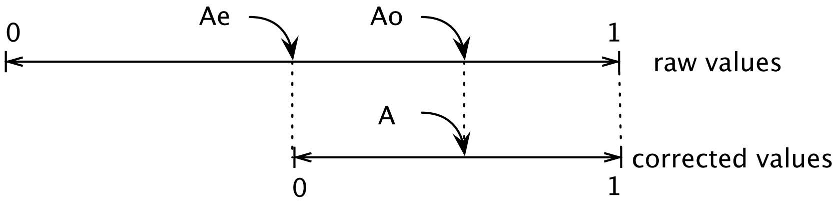


Figure 4.3: Chance correction seen as a change of referential

In response to the criticism by Zhao et al. (2013), let us mention that chance corrected measures do not assume a lack of ethics from annotators: it is important to note that chance correction leaves as is value 1, i.e. 100% agreement, which means that these measures consider annotators achieve such a result without chance. A part of chance is estimated only when there is not a total agreement, without necessarily assuming that annotators roll the dice, as we will see below.

4.2.3 Estimating chance agreement: several conceptions, and a legitimate debate

We address now a crucial point regarding how to compute an agreement value, and probably the most sensitive one. If we can demonstrate, as we have just done, that it is necessary to take chance into account, there is however no certainty about how to do so. To our knowledge, there is no available model, to this day, of annotators' behavior, and we even question whether it is possible to design a model which fits the diversity of annotation campaigns. Though, providing a model of chance is implicitly making assumptions about the annotators' behavior. Let us observe and discuss the main models of chance. We notice that the article (Krippendorff, 2011) provides a very interesting analysis, while promoting the conception corresponding to his own alpha measures.

Equiprobable model : what random can do

This model, used by Bennett et al. (1954) for his S measure, considers that there is an urn containing one ball for each available category. For each item to annotate, we draw one ball to select the category, and put the ball back into the urn. This leads to 50% agreement in the case of two categories. The main objection to this model is its dependency to the number of categories, with the debatable point that the more the number of categories, the less the chance. For instance, if we add a third unused category to our initial two categories, the chance level decreases from 50% to 33.3%. According to us, this fact results from a conceptual flaw of this model, which assumes a schizophrenic behavior of annotators: for each item, either they meaningfully annotate and never make mistake in this case, either they roll the dice to randomly choose a category. In other words, annotators are supposed to oscillate between perfect and worst annotators.

Distribution driven models: chance is different from random

In this conception of chance, that we will call "distribution driven", the main principle is the same as previously, with the difference that the urn (or the urns, see later) includes as balls all the annotations that were made by the annotators. For instance, if annotators make use of category A more often than category B, there will be more balls A than balls B. According to us, this better corresponds to a plausible annotator's behavior: an annotator has her own weaknesses, such as fatigue, lassitude, habits (for instance most of items are A rather than B), or simply mistake in how to interpret data, but she tries to annotate correctly. As a consequence, when she fails, one can assume that what she does is not totally uncorrelated from the task, but that it comes from all the mechanisms she performs in order to try to find the correct annotation. And we may assume that these mechanisms lead to the observed distribution.

This conception of chance is used by most of the current agreement measures, but there are two opposed versions. The one from Cohen (1960) for his famous κ considers that each annotator has her own urn, while the other one considers there is a single urn containing all annotations from all annotators, supported for instance by Scott (1955) for his π and Krippendorff (1980) for his α.

We cannot go into details here, but the reader can have a thorough analysis of the differences in (Mathet and Widlöcher, 2016). In a few words, our conclusion is the same as Zwick (1988) and Krippendorff (2011) who argue that for a given observed agreement, κ (i.e. one urn per annotator) rewards the fact that annotators disagree on the distribution of categories, which is not desirable.

Dependency to prevalence: a difficult debate

However, there is a long term debate concerning the dependence of distribution driven models to prevalence of categories, which has started as early as in 1990. Indeed, in the medical filed, Feinstein and Cicchetti (1990) have mentioned the possibility of a paradoxical low kappa value despite a strong observed agreement, mainly because of an unbalanced category distribution. This paradox has then been introduced in the CL domain by Di Eugenio and Glass (2004). Let us consider the illuminating medical example of a rare disease, mentioned for instance by Krippendorff (2013a), which concerns one person out of 1000. Two doctors compare their diagnostics: they agree on 99.85% of the patients, as detailed in table 4.1. The resulting kappa value is only 0.499, which is not very good. What value should be trusted, 0.9985 or 0.499?

	+	−	
+	5	5	10
−	5	9 985	9 990
	10	9 990	10 000

Table 4.1: High prevalence of a category in the case of a rare disease

In the detailed data provided by table 4.1, we can see that doctors agree on 9985 patients whom they have evaluated healthy, 5 that they have evaluated diseased, but there are 10 patients that are evaluated healthy by a doctor and diseased by the other.

We can argue that good doctors are those who are able to identify the rare disease cases, which is obviously not the case in our example. A second argument is that a fake doctor (relying only on the fact that it is a rare disease) who would evaluate all patients as healthy would get 99.9% of correct responses, which confirms that the very question here concerns the diseased patients rather than the healthy ones. In a few words, what does chance correction here is to remove from computation what is known, that is, here, that most patients are healthy. Hence, the 0.499 kappa computed here corresponds to the fact that only one diseased patient out of two is correctly identified.

On the other hand, we cannot dismiss this delicate question. A very illuminating experiment has been conducted by Bonnardel (1996), which shows that for a given "sensitivity" (the propensity to make real positives) and a given "specificity" (the propensity to make true negatives) of doctors, that is, for a given capability of doctors, the kappa values do fluctuate depending on the prevalence of categories.

For these reasons, it is important to be aware about this dependency, and to understand it. From our point of view, it may be desirable to identify obvious flaws (such the doctors concerning a rare disease), but in the case where the fact that a category is most rare than others is not important, this may be unfair. It is the reason why we should not take into account only the final agreement value, but also the details of its computation, and in particular the detailed agreement for each individual category as proposed by Krippendorff et al. (2016).

4.2.4 An anti-distribution-driven conception of chance

A third kind of approach has been driven by Aickin (1990) and next by Gwet (2012). It is intended to take the opposite view from distribution driven approaches, with the goal to inverse the behavior of chance corrected measures in the case of unbalanced distributions. Even if such measures are more rarely used, it is important we discuss them in the present analysis. This approach considers that: (1) each item is either easy (denoted E), or difficult (denoted H) to annotate; (2) an annotator never makes mistakes with E items; (3) she rolls the dice with H items (with the difference that E and H are common to all annotators for Aickin, whereas they depend on annotators for Gwet). Since, obviously, we ignore *a priori* if a given item is E or H, a quite complex probabilist model is used which is intended to estimate the respective quantities of them (without identifying them) from the observed annotations. Not only the relevance of such a model is not proved, but also the initial assumption appears debatable. On one side, the dichotomy E *vs* H is simplistic (why no item is moderately "difficult" ?), and on the other side the resulting annotator's behavior is, as it is for the equiprobable model, schizophrenic: for a E, she never fails, for a H, she behaves the worse one can expect.

More precisely, without going into computation details, and taking a simple example with two categories with respective distributions p = x and q = 1 – x, coefficient AC_1 from Gwet considers that A_e = 0.5 when x = 0.5, and A_e reaches zero when x reaches 0 or 1 (p or q equals 1 and the other one 0). That is to say that AC_1 corresponds to other approaches when categories are balanced, but refutes any possibility of chance agreement with very unbalanced categories, and then behaves the opposite way of other approaches. Gwet argues that if annotators' choice goes toward a particular category, it is necessarily for a deterministic reason (and that, if they behaved at random, they would get a balanced distribution p = q = 0,5). However, it is not possible, in the case where annotators would choose always the same category, to distinguish whether it is conform to the annotation task, or if they behave as broken thermometers which would always indicate the same temperature.

4.2.5 Discussion about models of chance

Our main contribution in this section will be to open a discussion rather than to provide answers. We have already given some arguments in favor of the distribution driven (with a single urn) model, and recommend it at the present time, but we also question its limitations, with some points not addressed in the literature to our knowledge.

First limitation: This model is less and less justifiable as the agreement decreases[3]. To the extreme, when $A = 0$, which implies $A_o = A_e$, what annotators do is totally uncorrelated from the annotation task. In these conditions, their annotations cannot depict to the real distribution of categories, contrary to what the computation of A_e assumes (this assumption is made for instance in (Krippendorff, 2011): "Without knowledge of the correct valuation of units, this conception takes the distribution of values that all coders collectively use to describe a given set of units as the best estimate of what the population of units is like").

[3] high agreement values are the most important since they give the green light to consider annotations as reproducible, and so should be the more accurate. That counterbalances this first limitation which concerns lower values.

Second limitation: Based on observed distributions, this assumption excludes the cases (difficult to quantify) where annotators made an input mistake (for instance they click on another category than their choice), since these cases have no reason to follow the real distribution. In such cases, it is more the way the software interface is designed which may affect the false choices (for instance an inaccurate click will lead to choose the next category in presentation order).

Third limitation: This conception of chance implicitly assumes that distributions are homogeneous within a given corpus. This is a problem if such homogeneity is not ensured. Let us assume a textual annotation campaign where items are predefined and where two categories A and B are available. Let us assume that a given annotated text contains two pages, both having the same number of items, and the same observed agreement $A_o = 0,9$. By contrast, we assume that distributions are very different between the two pages : 50% A and 50% B for page one, and 10% A and 90% B for page 2, which results in 30% A and 70% B for the whole text. It follows that A_e values are respectively 0.5, 0.82 and 0.58, and so chance corrected A values are respectively 0.8, 0.44 and 0.76 (we assume that annotators have the same distribution to avoid to debate about differences between κ, π et α). From these values, on can conclude that: (1) page 2 is considered much easier to annotate that page 1 (A_e is 0.82 *vs* 0.5); (2) from a high agreement on the first half of the document (0.8 for page 1) and a poor agreement on the second half (0.44) results a high allover agreement (0.76). Conclusion (1) is not straightforward: annotators, for page 1, use half the time category A, and half the time category B, and then, without any warning, use by themselves 10% of A and 90% of B for page 2. Why, when doing mistakes on page 2, do annotators use a different distribution? That is the very hub of the problem concerning this conception: distribution of categories results, in a interlaced way, partly from the annotation task (for instance, in a given task, category A is allover more frequent than category B, see the previous example of a rare disease), and partly from the part of the corpus being currently annotated (from a part of text to the next, category distribution may change for instance because of citations, change of topic, and so forth). Consequently, we are facing a double level, respectively local and global, whereas this conception assumes there exists a single level. Conclusion (2) seems to confirm a lack of consistency of such measures when facing local variations of distribution. To assess that the whole work of annotators is reproducible from what we observe on a document, while assessing the contrary from what we observe on half of the same document, is debatable. Our own works around Gamma measures in (Mathet et al., 2015)[4] (introduced later) address this problem, but the question remains open. From our point of view, it is still necessary to enhance this conception in the case of unsteady distributions. At least, measures should provide additional information about the homogeneity degree of distribution over a given corpus, to warn the campaign leader about a possible flaw of the measure.

To conclude, chance is a difficult question, but we cannot dismiss this concept for the reason it is still an open question: relying only on raw agreement (i.e. observed agreement) provides artificially high results. From current knowledge, we advise to use distribution driven (with a single urn) approaches, such as π or α, while being warned about their possible limitations. Even if the results they provide may seem unfavorable in case of unbalanced distributions, the precautionary principle leads us to prefer false warning rather than a false guarantee.

4.3 Benchmarking and understanding agreement measures: the Corpus Shuffling Tool (CST)

The domain lacked a tool providing a clear and generic picture of the agreement coefficients behavior, allowing to better qualify the obtained agreement results. Why several measures do exist, providing different results, for the same annotation tasks? Is a given agreement value high enough to ensure

[4] By proposing to compute A_e either at local level (*i.e.* for each single document), either at global level when it is possible, that is, for a whole set of documents from the same corpus

the reproducibility of annotations? The already mentioned survey article (Artstein and Poesio, 2008) contains a section dedicated to various attempts at providing an interpretation scale for the Kappa family coefficients and how they failed to converge. Works such as (Gwet, 2012) are also to be mentioned. They present various inter-rater reliability coefficients and insist on benchmarking issues related to their interpretation.

Our contribution in this domain has been to design a method and a tool in order to compare the behavior of all existing measures, and to better interpret the values they provide.

4.3.1 Introducing the CST

Manual annotation, by essence, is subject to human errors. These errors may involve several paradigms. Indeed, in the quite general case of unitizing, each manually annotated element may diverge from what it should be (the reference, or gold standard), in one or multiple ways, including: (i) the location is not correct (the frontiers of an element do not exactly match those of the reference); (ii) the characterization is not correct (wrong category, or wrong feature value); (iii) the annotation does not belong to the reference (false positive); or (iv), on the contrary, a reference element is missing (false negative). All of these error paradigms tend to damage the annotations, so each of them should be taken into account by agreement measures. We propose here to apply each measure to a set of corpora, each of which embeds errors from one or more paradigms, and with a certain magnitude (the higher the magnitude, the higher the number of errors). This experiment allows us to observe how the measures behave w.r.t. the different paradigms, and with a full range of magnitudes. The idea of creating artificial damaged corpora is inspired by Pevzner and Hearst (2002b), and then by Bestgen (2009) in thematic segmentation, but our goal (giving meaning to measures) and our method (e.g. applying progressive magnitudes) are very different.

4.3.2 Protocol

We cannot describe here the whole protocol, which is detailed in (Mathet et al., 2012), and will just sum up it below.

The main principle of this tool is as follows. A reference corpus is built, with respect to a statistical model which defines the number of categories, their prevalence, the minimum and maximum length for each category, etc. Then, this reference is used by the shuffling tool to generate a multi-annotator corpus, simulating the fact that each annotator makes mistakes of a certain type, and of a certain magnitude. It is important to remark that the generated corpus does not include the reference it is built from.

The magnitude m is the strength of the shuffling, that is to say the severity of the mistakes the annotators make with respect to the reference. It can be set from 0 which means no damage is applied (and the annotators are perfect), to the extreme value 1 which means annotators are assumed to behave in the worst possible way (but still being independent each other), namely at random.

A very important point is that magnitude is not a degree of disagreement, but a degree of annotators' weakness, which, of course, fatally results in more and more disagreement.

Figure 4.4 illustrates the way such a corpus is built: from the reference containing some categorized units, 3 new sets of annotations are built, simulating 3 annotators who are assumed to have the same annotating skill level, which is set in this example at magnitude 0.1. The applied error type is position only, that is to say that each annotator makes mistakes only when positioning boundaries, but does not make any other mistake (the units are reproduced in the same order, with the correct category, and in the same number). At this low magnitude, the positions are still close to those of the reference, but often vary a little. Hence, we obtain here a slightly shuffled multi-annotator corpus. Let us sum up the way error types (or "paradigms") are currently designed in the CST.

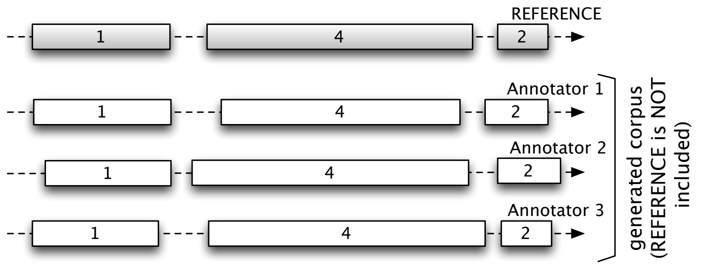


Figure 4.4: The Shuffling Tool generating 3 annotations with position errors at magnitude $m = 0.1$

Position At magnitude m, for a given unit, we define a value $shift_{max}$ which is proportional to m and to the length of the unit, and each boundary of the unit is shifted by a value randomly chosen between $-shift_{max}$ and $shift_{max}$ (note: at magnitude 0, since $shift_{max} = 0$, units are not shifted).

Category This shuffling cannot be described in a few words (see Mathet et al. [2012] for details). It uses special matrices to simulate, using conditional probabilities, progressive confusion between categories, and can be configured to take into account overlapping of categories. The higher the magnitude, the more frequent and severe the confusions.

False negatives At magnitude m, each unit has the probability m to be forgotten. For instance, at magnitude $m = 0.5$, each annotator misses (on average) half of the units from the reference (but not necessarily the same units as the other annotators).

False positives At magnitude m, each annotator adds a certain number of units (proportional to m) to the ones of the reference.

Splits At magnitude m, each annotator splits a certain number of units (proportional to m). A split unit may be re-split, and so on.

Combination It is also possible to combine several error types at the same time, for instance Position and Category, to see how dedicated measures behave overall.

4.3.3 Interlaced paradigms

A difficulty has to be mentioned: error paradigms may interlace. This is not a weakness of the CST, but an inherent difficulty of how to interpret several dimension annotation errors, such as unitizing. For instance, if we consider the annotations done by two annotators named "annotator 1" and "annotator 2", in the dotted parts of figure 4.5, and even if we have access to the reference (shown in the top of the figure), at least two interpretations are possible:

1. In the left and right parts of the figure, we consider that annotator 1 has annotated the element of category A exactly as in the reference (arrows A1 and A1') while s/he has forgotten the element of category B (arrows B1 et B1').

2. However, if we consider in the left part of the figure that annotator 2 has forgotten the reference element of category A (arrow A2), and has done a category mistake with element of category B (arrow B2), we consider in the right part of the figure that annotator 2 has shifted the reference element of category A, for a result being the same!

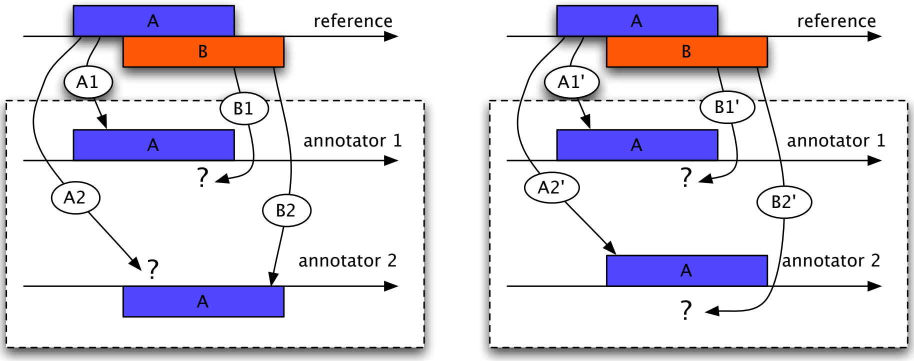


Figure 4.5: Two ways of interpreting a same error in a multi-paradigms error context

As a consequence, it is not possible, at high magnitude, to fully separate all paradigms of errors. For instance, high shifts may confound with false negatives of false positives.

4.3.4 Errors versus discrepancies

We have to put the emphasis on the fundamental difference between errors and discrepancies. The CST method is sometimes misunderstood because these two notions are not correctly understood.

As already explained, agreement measure are only concerned with discrepancies between annotators, since they do not have access, by design, to a reference. In the same time, the CST does use a reference, which may seem contradictory with the fact it is designed to assess agreement measures. The point is that the reference used by the shuffling tool is a starting point in a process which simulates, in a controlled manner, the behavior of different annotators who should all, ideally, produce what is in the reference.

In particular, figure 4.4 shows that the reference is not submitted as an input to the different measures. This way, the assessed measures are confronted to discrepancies, not to errors.

As a consequence, the magnitude really corresponds to a degree of weakness of the annotators (the divergence between what they should do and what they produce).

4.3.5 Results

This method provides an immediate visualization of the behavior of a measure for a progressive damaging, for one or several paradigms. At magnitude 0, any measure should be at 1 on the y-axis, since all annotators fully agree. At the other extremity, that is, at magnitude 1, measures should reach 0, since annotations are no more linked to the reference (so, to the annotation task). Between these two extremities, measures should ideally strictly decrease, not reaching a plateau nor being asymptotic.

Moreover, up to now, it was quite difficult to get a precise idea of the meaning of a given score provided by an agreement measure. If one could consider, for instance, than 0.8 is a “good” agreement, what does it really mean in terms of mistakes done by the annotators? With the proposed method, it is possible to read on the graphics, for a given score on the y-axis, what is the corresponding magnitude on the x-axis. Hence, this magnitude has an accurate signification in each paradigm (for instance, for the false positive paradigm, it means a particular probability of adding an annotation for each existing one).

Figure 4.6 shows an experiment involving three measures and a segmentation task. We can see that only one measure goes from agreement 1 at magnitude 0 to agreement 0 at magnitude 1, as it should, whereas the two others fail to go under 0.4, simply because they are not chance corrected. More results will be provided in the next section about the Gamma family.

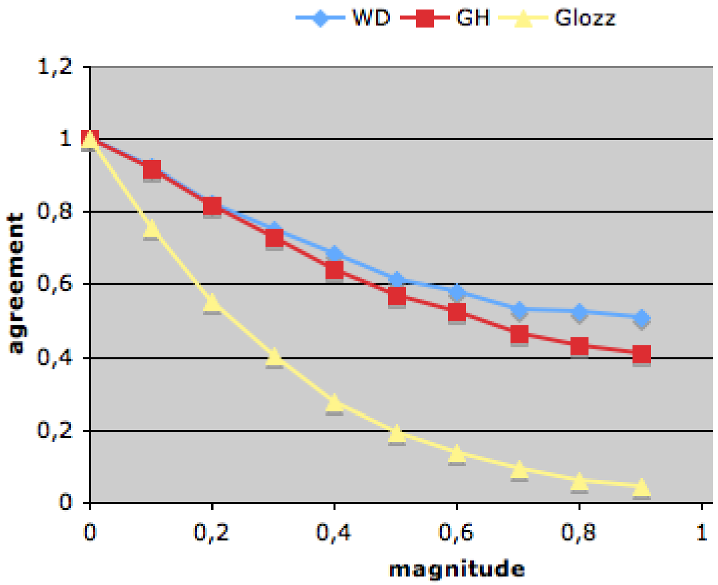


Figure 4.6: An example of output of the Corpus Shuffling Tool

In addition to comparison between different measures, this tool can be used in order to design or parametrize new measures: it is possible to see almost instantly the consequences of any modification of the design or of the parameters of a new measure, as we have been doing through the whole conception of the Gamma family.

Chapter 5

Assessment of annotations - Part II: The Gamma family of agreement measures for unitizing tasks

This chapter relies on two journal articles published in *Computational Linguistics*, MIT Press: (Mathet et al., 2015) and (Mathet, 2017).

Contents

5.1 Introduction 98

5.2 Motivations, scope and illustrations 98

5.3 Introducing γ 100

5.3.1 Our proposal 100

5.3.2 Main principles of γ 100

5.3.3 Definitions : unit, annotator, annotation set 102

5.3.4 Dissimilarity between two units 103

5.3.5 Unitary alignment, Alignment 105

5.3.6 Alignment and Disorder 106

5.3.7 Best alignment, disorder of an annotation set 106

5.3.8 Expected disorder for chance correction 106

5.3.9 The agreement measure γ 107

5.4 The implementation of γ 108

5.4.1 Computing the disorder 108

5.4.2 Implementation of the expected disorder 110

5.4.3 Computing an average value: a sampling question 111

5.5 benchmarking γ 112

5.6 The additional coefficients γ_{cat} and γ_k 112

5.7 Main requirements of a categorial measure 113

5.7.1 Positional discrepancies should not impact categorial agreement 113

5.7.2 False negatives/positives should not impact categorial agreement 114

5.7.3 Length of units should not be taken into account 114

5.8 How best to handle missing values ? 114

5.9 The design of γ_{cat} 116

5.9.1 Getting an alignment of the units from γ 117

5.9.2 Value weight: giving the same importance to each value 117
5.9.3 Confidence weight: enhancing the accuracy of δ 117
5.9.4 Total weight of a pair of units 118
5.9.5 The algorithm to compute the disorder of γ_{cat} 118
5.9.6 Discussion: Why not to use a naive two step method 119
5.10 The in-depth coefficient γ_k which focuses on each category 120
5.11 Benchmarking γ_{cat} and γ_k 120
5.12 Overview and dependencies of the gamma family 121
5.13 Software 122
5.14 Conclusion on the Gamma family 123
5.15 Future work 124

Important notice: we cannot expose in this thesis the whole content of our works on the Gamma family, for which we refer the reader to (Mathet et al., 2015) and (Mathet, 2017), in particular for the state of the art and for the benchmarking of our methods, and will mainly focus in the current presentation on the design of our measures.

5.1 Introduction

Admittedly, much work has already been done concerning agreement measures for some kinds of annotation efforts, namely when annotators have to choose a category for previously identified entities. This approach, which we will call **pure categorization**, has led to several well known and widely discussed coefficients such as κ, π or α, as of the 1950s. Some more recent efforts have been made in the domain of unitizing. However, studies are scarce, as Krippendorff pointed out: “Measuring the reliability of unitizing has been largely ignored in favor of coding predefined units” (Krippendorff, 2013a, page 310). This scarcity concerns either **segmentation**, where annotators simply have to mark boundaries in texts to separate contiguous segments, or more generally **unitizing**, where gaps may exist between units. Moreover, some even more complex configurations may occur (overlapping or embedding units), which are more rarely taken into account.

The γ family, currently γ, γ_{cat} and γ_k, is a set of agreement measures concerning the joint tasks of unit locating and unit labeling at the same time, i.e. *unitizing*. The main measure γ relies on an **alignment** of units between different annotators, with penalties associated with each positional and categorial discrepancy. The alignment itself is chosen to minimize the overall discrepancy in a **holistic** way, considering the full continuum to make choices, rather than making local choices. The proposed method is **unified** because the computation of γ and the selection of the best alignment are interdependent: the computed measure depends on the chosen alignment, whose selection depends on the measure.

This method and the principles proposed in this thesis have been built up since 2010, and were first presented to the French community in a very early version in (Mathet and Widlöcher, 2011b). The initial motivation for their development was the lack of dedicated agreement measures for annotations at the discourse level, and more specifically for annotation tasks related to TOPIC TRANSITION phenomena.

5.2 Motivations, scope and illustrations

In the present work, we focus on both categorizing and locating, and consider therefore annotation tasks where annotators are not provided with pre-selected units, but have to locate them and to categorize them at the same time. An example of a multi-annotated continuum (as a reminder, this continuum may be a text or, for example, an audio or a video recording) is provided in Figure 5.1, where each line represents the annotations of a given annotator, from left to right, respecting the continuum order.

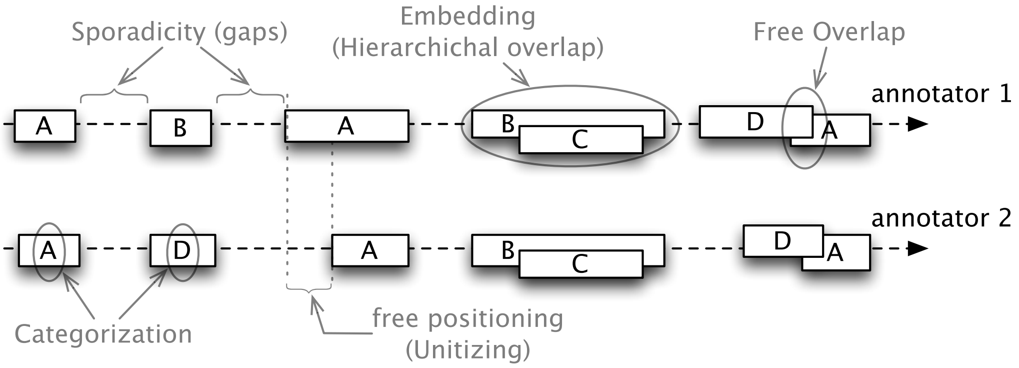


Figure 5.1: Multi-annotation including positioning and categorizing

Properties of annotation tasks and annotated items

In order to characterize the annotation efforts focusing on specific linguistic objects, we consider the following properties, illustrated in Figure 5.1.

Categorization occurs when the annotator is required to label (predefined or not) units.

Unitizing occurs when the annotator is asked to identify the units in the continuum: she has to determine each of them (and the number of units that she wants) and to locate them by positioning their boundaries. As already said, in our acceptation, unitizing may include categorization.

Embedding (hierarchical overlap) may occur if units may be embedded in larger ones (of the same type, or not).

Free overlap may occur when guidelines tolerate the partial overlap of elements (mainly of different types). Embedding is a special case of overlapping.

Full-covering (*vs* sporadicity) applies when all parts of the continuum are to be annotated. For other tasks, parts of the continuum are selected.

Two specific cases

We call **pure segmentation** (illustrated by Figure 5.2) the special case of unitizing with full-covering and without categorization, and we call **pure categorization** categorization without unitizing.

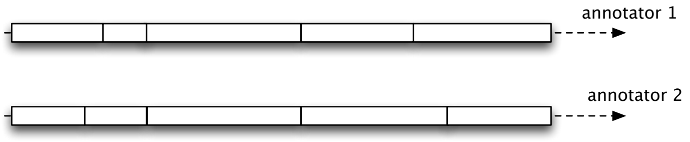


Figure 5.2: The particular case of pure segmentation

Examples of annotation tasks

To make our presentation more concrete, it is useful to mention examples of linguistic objects and annotation tasks for which agreement measures may be required. Small caps are used to refer to the names of these tasks.

Annotation tasks	Categorization	Unitizing	Embedding	Free Overlap	Sporadicity
Part-of-Speech	■	□			
Gene Renaming	■				■
Word sense	■	□			■
Named Entity	■	■	□	□	■
Argumentative Zoning	■	□			■
Discourse Framing	■	■	□	□	■
Communicative Behavior	■	■	□	□	□
Dialog Act	■	■	□	□	□
Topic Segmentation		■			
Hierarchical Topic Segmentation		■	■		
Topic Transition	■	■	■	■	□
Enumerative Structures	■	■	■	□	■

Table 5.1: Properties associated to some examples of annotation tasks. ■: mandatory, □: possible.

Table 5.1 summarizes the properties of the linguistic objects and some typical annotation tasks.

5.3 Introducing γ

5.3.1 Our proposal

The basic idea of this new coefficient is as follows: all local disagreements (called **disorders**) between units from different annotators are averaged to compute an overall disorder. However, these local disorders can be computed only if we know for each unit of a given annotator, which units, if any, from the other annotators it should be compared with (via what is called **unitary alignment**), that is to say if we can rely on a suitable alignment of the whole (called **alignment**). Since it is not possible to get a reliable preconceived alignment (as explained in Section 5.3.2) , γ considers all possible ones, and computes for each of them the associated overall disorder. Then, γ retains as the best alignment the one which minimizes the overall disorder, and the latter value is retained as the correct disorder. To obtain the final agreement, as with the familiar kappa and alpha coefficients, this disorder is then chance corrected by a so-called expected disorder which is calculated by randomly resampling existing annotations.

First of all, we introduce three main principles of γ in Section 5.3.2. We introduce in Section 5.3.3 the basic definitions. The comparison of two units (depending on their relative positions and categories) relies on the concept of **dissimilarity** (Section 5.3.4). A **unitary alignment** groups at most one unit of each annotator and a set of unitary alignments covering all units of all annotators is called an **alignment** (Section 5.3.5). The **disorder** associated to a unitary alignment results from dissimilarities between all its pairs of units, and the disorder associated to an alignment depends on those of its unitary alignments (Section 5.3.6). The alignment having the minimal disorder (Section 5.3.7) is used to compute the **agreement** value, taking **chance correction** into account (Section 5.3.8).

5.3.2 Main principles of γ

Measuring and aligning at the same time: γ is unified.

For a given phenomenon identified by several annotators, it is necessary to provide an agreement measure permissive enough to cope with a double discrepancy concerning (1) its position in the continuum, (2) the category attributed to the phenomenon.

Because of discrepancies in positioning, it is necessary to provide an agreement measure with an inter-annotator alignment, which shows which unit of a given annotator corresponds, if any, to which unit of another annotator. If such an alignment is provided, it becomes possible, for each phenomenon identified by annotators, to determine to what extent the annotators agree both on its categorization and its positioning. This quantification relies on a certain measure (called **dissimilarity** hereafter) between annotated units: the more the units are considered as similar, the lesser the dissimilarity.

But how can such an alignment be achieved? For instance, in Figure 5.3, aligning unit $A1$ of annotator A with unit $B1$ of annotator B consists in considering that their properties are similar enough to be associated: annotator A and annotator B have accounted for the same phenomenon, even if in a slightly different manner. Consequently, to operate, the alignment method should rely on a measure of distance (in location, in category assignment, or both) between units.

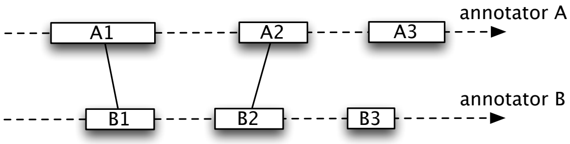


Figure 5.3: An example of alignment choices: two pairs of units are aligned ($A1$ with $B1$, $A2$ with $B2$), one is not.

Therefore, agreement measure and aligning are interdependent: it is not possible to correctly measure without aligning, and it is not possible to align units without measuring their distances. In that respect, measuring and aligning cannot constitute two successive stages, but must be considered as a whole process. This interdependence reflects the unity of the objective: establishing to what extent some elements, possibly different, may be considered as similar enough, either to quantify their differences (when measuring agreement), or to associate them (when aligning).

Interestingly, Reidsma et al. (2006, page 1119), not really satisfied by the use of the *discretizing measure* as already mentioned, "have developed an extra method of comparison in which [they] try to align the various segments". This attempt highlights the necessity to rely on an alignment. Unfortunately, the way the alignment is computed, adapted from Kuper et al. (2003), is disconnected from the measure itself, being an *ad hoc* procedure on which other measures are applied.

Aligning globally: γ is holistic.

Let us consider two annotators A and B having respectively produced unit $A5$, and units $B4$ and $B5$, as shown in Figure 5.4. When considering this configuration at a local level, we may consider, based for instance on the overlapping area, that $A5$ fits $B5$ slightly better than $B4$. However, this local consideration may be misleading. Indeed, Figure 5.5 shows two larger configurations, where $A5$, $B4$ and $B5$ are unchanged from Figure 5.4. With a larger view, the choice of alignment of $A5$ may be driven by the whole configuration, possibly leading to an alignment with $B4$ in Figure 5.5(a), and with $B5$ in Figure 5.5(b): alignment choices depend on the whole system and the method should consequently be **holistic**.

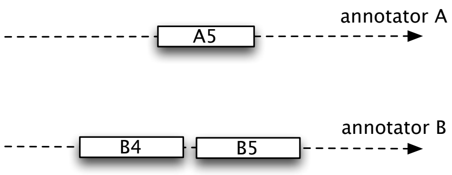
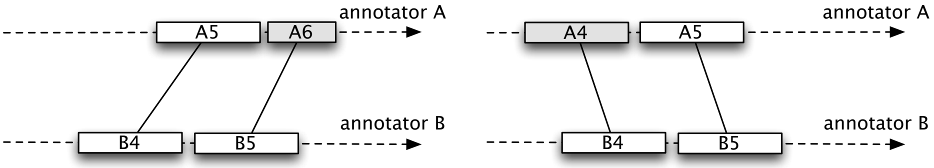


Figure 5.4: Inter-annotator configuration observed at local level



(a) Context 1

(b) Context 2

Figure 5.5: Alignment choices depend on the whole system

Accounting for different severity rates of errors: positional and categorial permissiveness of γ.

As far as positional discrepancies between annotators are concerned, it is important for a measure to rely on a progressive error count, not on a binary one: two positions from two annotators may be more or less close to each other but still concern the same phenomenon (partial agreement), or may be too far to be considered as related to the same phenomenon (no possible alignment). For instance, for segmentation, specific measures such as WindowDiff (see Pevzner and Hearst, 2002a) rely on a progressive error count for positions, with an upper limit being half the average size of the segments. For unitizing, Krippendorff considers with $_u\alpha$ that units can be compared as long as they overlap. However, γ considers that in some cases, units by different annotators may correspond to the same phenomenon though they do not intersect. We base this claim on two grounds. First, if we observe the configuration given in Figure 5.6, annotator 2 and 3 have both annotated part of the NAMED ENTITY which has been annotated by annotator 1. Consequently, though they do not overlap, their units refer to the same phenomenon. In addition, we find a direct echo of this assumption in (Reidsma, 2008, pages 16-17) where, in a video corpus concerning COMMUNICATIVE BEHAVIOR, "different timing (non-overlapping) [of the same episode] was assigned by [...] two annotators." Regarding categorization, some available measures consider all disagreements between all pairs of categories as equal. Other coefficients, called *weighted* coefficients (see Artstein and Poesio, 2008), as well as γ, consider on the opposite that mismatches may not all have the same weight, some pairs of categories being closer than others. This closeness is often referred to as *overlap*.

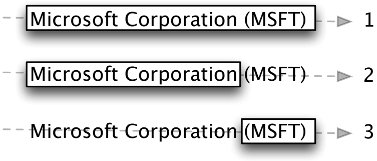

Microsoft Corporation (MSFT)
1
Microsoft Corporation (MSFT)
2
Microsoft Corporation (MSFT)
3

Figure 5.6: Alignments with no necessary intersection

In our terminology, we call **category-overlapping** this closeness between categories, and **overlap** means **positional overlap**. For example, within annotation efforts related to WORD SENSE or DIALOG

ACTS, it is clear that disagreements on labels are not all alike.

5.3.3 Definitions : unit, annotator, annotation set

Given a multi-annotated continuum t:

- let $\mathcal{A} = \{a_1, ..., a_n\}$ be the set of annotators
- let $n = |\mathcal{A}|$ be the number of annotators
- let $\mathcal{U}$ be the set of units from all annotators
- $\forall i \in [\![1, n]\!]$, let x_i be the number of units by annotator a_i for t
- let $\overline{x} = \frac{\sum\limits_{i=1}^{n} x_i}{n}$ be the average number of annotations per annotator
- for annotator $a = a_i$, $\forall j \in [\![1, x_i]\!]$, we note u_j^a unit from a of rank j

Annotation set : an annotation set s is a set of units attached to the same continuum and produced by a given set of annotators.

Corpus : a corpus c is defined with respect to a given annotation effort, and is composed of a set of continua, and of the set of annotations related to these continua.

Unit : a unit u bears a category denoted $cat(u)$, and a location given by its two boundaries, each of them corresponding to a position in the continuum, respectively denoted $start(u)$ and $end(u)$, $start$ and end being functions from $\mathcal{U}$ to $\mathbb{N}^+$.

Equality between units is defined as follows:

$$\forall (u, v) \in \mathcal{U}^2, u = v \Leftrightarrow \begin{cases} cat(u) = cat(v) \\ start(u) = start(v) \\ end(u) = end(v) \end{cases}$$

5.3.4 Dissimilarity between two units

We introduce here the first brick to build the notion of disorder, which works at a very local level, between two units. A dissimilarity tells to what degree two units should be considered as different, taking into account such features as their positions, their categories, or a combination of the two.

A dissimilarity is a function $d\ : \mathcal{U}^2 \rightarrow \mathbb{R}^+$, so that :

$$\forall (u, v) \in \mathcal{U}^2, \begin{cases} d(u, v) = d(v, u) \text{ (d is symmetric)} \\ u = v \Rightarrow d(u, v) = 0 \end{cases}$$

A dissimilarity is not necessarily a distance in the mathematical sense of the term specially because triangular inequality is not mandatory (for instance, in Figure 5.7, $d(A1, B2) > d(A1, C1) + d(C1, B2)$).

Empty unit $u_\emptyset$, empty dissimilarity $\Delta_\emptyset$

As we will see, γ relies on an alignment of units by different annotators. In particular, this alignment indicates for unit u_i^{a1} of annotator a_1, to which unit u_j^{a2} of annotator a_2 it corresponds, in order to compute the associated dissimilarity. In some cases, though, the method will choose not to align u_i^{a1} with any unit of annotator a_2 (none corresponds sufficiently). We define the empty pseudo unit, denoted

$u_\emptyset$, which corresponds to the realization of this phenomenon: ultimately, a pseudo unit $u_\emptyset$ is added to the annotations of a_2, and $u_i^{a_1}$ is aligned with it.

We also define the associated cost $\Delta_\emptyset$:

$$\begin{aligned} \forall u \in \mathcal{U}, d(u, u_\emptyset) = d(u_\emptyset, u) &= \Delta_\emptyset \\ \text{and } d(u_\emptyset, u_\emptyset) &= \Delta_\emptyset \end{aligned}$$

Dissimilarities should be calibrated so that $\Delta_\emptyset$ is the value beyond which two compared units are considered critically different. Consequently, it constitutes a reference, and dissimilarities will be expressed as multiples of $\Delta_\emptyset$ for better clarity. It is not a parameter of γ, but a constant (which is set to 1 in our implementation).

Positional dissimilarity d_{pos}

Different positional dissimilarities may be created, in order to deal with different annotation tasks. We will use the dissimilarity shown below in Equation 5.1, which is very versatile.

$$d_{pos-sporadic}(u, v) = \left(\frac{|start(u) - start(v)| + |end(u) - end(v)|}{(end(u) - start(u)) + (end(v) - start(v))} \right)^2 \cdot \Delta_\emptyset \tag{5.1}$$

Equation 5.1 sums the differences between the right and left boundaries of both units in its numerator. Its denominator sums the lengths of both units, so that this dissimilarity is not scale dependent. Squaring the value is an option used here to accelerate dissimilarity when differences of positions increase. It is illustrated in Figure 5.7 with different configurations and their associated values, from 0 for the perfectly aligned pair of units $(A1, B1)$ to $22.2 \cdot \Delta_\emptyset$ for the worst pair $(A1, C2)$.

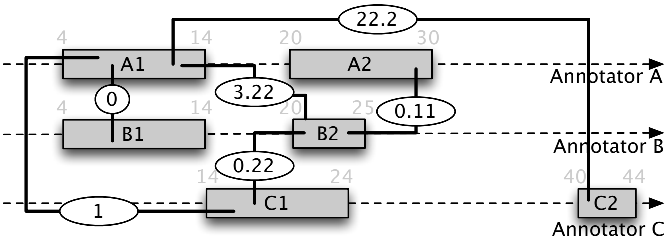


Figure 5.7: Real examples of $d_{pos-sporadic}$ values (divided by $\Delta_\emptyset$)

Categorial dissimilarity d_{cat}

Let K be the set of categories. For a given annotation effort, $|K|$ different categories are defined. For more convenience, we first define **categorial distance** between categories $dist_{cat}$ via a square matrix of size $|K|$, with each category appearing both in row titles and column titles. Each cell gives the distance between two categories through a value in $[0, 1]$. Value 0 means perfect equality, whereas the maximum value 1 means that the categories are considered as totally different. As $dist_{cat}$ is symmetric, such a matrix is necessarily symmetric, and bears 0 in each diagonal cell. Table 5.2 gives an example for 3 categories, and shows that an association between a unit in category cat_1 with one in category cat_3 is the worst possible (distance=1), whereas it is half as much between cat_1 with cat_2 (distance=0.5). This makes it possible to take into account so-called **category-overlapping** (in our example, cat_1 and cat_2 are said to overlap, which means they are not completely different), as **weighted** coefficients such as κ_w or α already do. Note that in the case of so-called "nominal categories", the matrix will be full of 1

Table 5.2: Categorial matrix of $dist_{cat}$ for 3 categories

	cat_1	cat_2	cat_3
cat_1	0	0.5	1
cat_2	0.5	0	1
cat_3	1	1	0

outside the diagonal, and full of 0 in the diagonal (different categories are considered as not matching at all).

This categorial distance matrix is then used to build the categorial dissimilarities, taking into account the $\Delta_\emptyset$ value. We define categorial dissimilarity between two units by:

$$d_{cat}(u,v) = f_{cat}(dist_{cat}\,(cat(u), cat(v))) \cdot \Delta_\emptyset \tag{5.2}$$

Function f_{cat} can be used to adjust the way the dissimilarity grows with respect to the categorial distance values. The standard option[1] (used in this study) is to simply consider $f_{cat}(x) = x$, with which d_{cat} naturally increases gradually from zero when categories match, to $\Delta_\emptyset$ when categories are totally different ($dist_{cat}(cat(u), cat(v)) = 1 \implies d_{cat}(u,v) = \Delta_\emptyset$).

Combined dissimilarity d_{combi}

Since in some annotation tasks units may differ both in position and in category, it is necessary to combine the associated dissimilarities so that all costs are cumulated. This is provided by a combined dissimilarity.

Let d_1 and d_2 be two dissimilarities. We define:

$$d^{\alpha,\beta}_{combi(d_1,d_2)}(u,v) = \alpha.d_1(u,v) + \beta.d_2(u,v) \tag{5.3}$$

It is easy to demonstrate that this linear combination of dissimilarities is itself a dissimilarity (if $(\alpha,\beta) \neq (0,0)$). It enables the same weight to be assigned to positions and categories using $d^{1,1}_{combi(d_{pos},d_{cat})}$, which is currently used for γ.

Then, we can note that it is the same cost $\Delta_\emptyset$ for a unit either not to be aligned with any other one, or to be aligned with a unit in the same configuration as $(A1, C1)$ of Figure 5.7 (if they have the same category), or to be aligned with a unit having an incompatible category (if they occupy the same position).

5.3.5 Unitary alignment, Alignment

Unitary alignment $\breve{a}$

A unitary alignment $\breve{a}$ is an i-tuple, i belonging to $[\![1, n]\!]$ (n being the number of annotators), containing at most one unit by each annotator: it represents the hypothesis that i annotators agree to some extent on a given phenomenon to be unitized. In order to make all unitary alignments homogenous, we eventually complete any unitary alignment that is an i-tuple with $n-i$ empty units $u_\emptyset$, so that all unitary alignments are ultimately n-tuples. Figure 5.8 p. 108 illustrates unitary alignments with some $u_\emptyset$ units.

[1] Another option is for example to use $f_{cat}(x) = -ln(1-x)x^{30} + x$ which is a function almost similar to $f_{cat}(x) = x$ on the $[0, 0.9]$ range, and reaches ∞ near 1. Then, when the categorial distance is equal to 1, the categorial dissimilarity reaches infinity, which guarantees that the units cannot be aligned.

Alignment $\bar{a}$

For a given annotation set, an alignment $\bar{a}$ is defined as a set of unitary alignments such that each unit of each annotator belongs to one and only one of its unitary alignments. Mathematically, it constitutes a **partition** of the set of units (if we do not take $u_\emptyset$ into account).

5.3.6 Alignment and Disorder

Disorder of a unitary alignment

The disorder of a unitary alignment $\breve{a}$, denoted $\breve{\delta}(\breve{a})$, is defined for a given dissimilarity d as the average of the one-to-one dissimilarities of its units:

$$\breve{\delta}(\breve{a}) = \frac{1}{C_n^2} \cdot \sum_{(u,v) \in \breve{a}^2} d(u, v) \tag{5.4}$$

Averaging dissimilarities rather than summing them makes the result independent of the number of annotators.

Disorder of an alignment

The disorder of an alignment $\bar{a}$, denoted $\bar{\delta}(\bar{a})$, is the sum of the disorders of all its unitary alignments divided by the mean number of units per annotator:

$$\bar{\delta}(\bar{a}) = \frac{1}{\bar{x}} \cdot \sum_{i=1}^{|\bar{a}|} \breve{\delta}(\breve{a}_i) \tag{5.5}$$

We chose to consider the average value rather than the sum so that the disorder does not depend on the size of the continuum.

5.3.7 Best alignment, disorder of an annotation set

Best alignment $\hat{a}$. An alignment $\bar{a}$ of the annotation set s is considered as the best (w.r.t. a dissimilarity) if it minimizes its disorder among all possible alignments of s. It is denoted $\hat{a}$. The proposed method is holistic in that it is necessary to take into account the whole set of annotations in order to determine each unitary alignment.

Disorder of an annotation set $\delta(s)$. The disorder of the annotation set s, denoted $\delta(s)$, is defined as the disorder of its best alignment(s) $\bar{\delta}(\hat{a})$. Note that it may happen that several alignments produce the lowest disorder.

We have just presented the two crucial definitions of our new method, which make it “unified”. Indeed, the best alignment is chosen w.r.t. the disorder, therefore w.r.t. to what computes the agreement measure, and, conversely and simultaneously, the resulting agreement value (see below) is given by the best alignment: agreement computation and alignment are fully intertwined, whereas in most agreement metrics, the alignment is fixed a priori or no alignment is used.

5.3.8 Expected disorder for chance correction

The model of chance of γ

As we have already mentioned in this dissertation, it is necessary for an inter-annotator agreement measure to provide chance correction. We have also seen that there are several chance correction models, and that it is a controversial question. However, we compte γ on the basis of the average distribution of observed annotations of the several annotators for the reasons already given.

More precisely, we define the expected (chance) disorder as the average disorder of a multi randomly annotated continuum where:

- the random annotations **fulfill the observed annotation distributions** for the following features:
  - the distribution of the number of units per annotator
  - the distribution of categories
  - the distribution of unit length per category
  - the distribution of the length of gaps
  - the distribution of overlapping or covering between each pair of categories (for instance, units of categories A and B may never intersect, 7% of the units of category A may cover one unit of category C, and so on...).
- the number of random annotators is the same as the number of annotators in the observed data

Two possible sources to build chance: local data versus corpus data

In addition, whereas other studies systematically compute the expected value on the data also used to compute the observed value, we consider that it should be computed, when possible (that is to say when several continua have been annotated with the same set of categories and the same instructions), from the distribution observed in all continua of the annotation effort the evaluated continuum comes from: if distribution changes from one continuum to another one, it is more because of the content of each continuum than because of chance.

As a consequence, γ provides two ways to compute the expected values. One which considers only the data of the continuum being evaluated, as do every other coefficients. A second one, which considers the data from all continua of the annotation effort the evaluated continuum comes from. When available, we recommend to use the second one, for the reasons already expressed.

Using sampling to compute the expected value

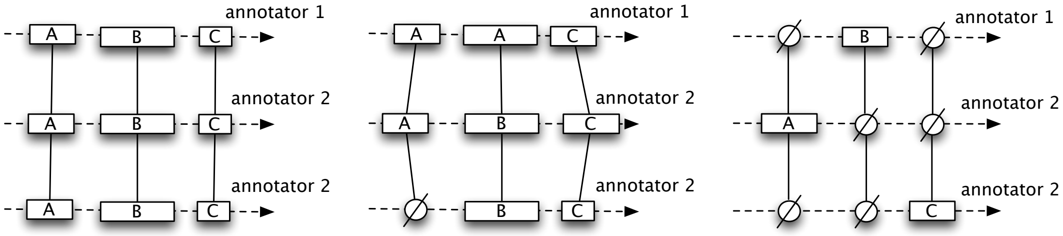
Expected agreement (or disagreement) is the expected value of a random variable. But which random variable? For coefficients like kappa and alpha, observed agreement (or disagreement) is the mean agreement (or disagreement) on all pairs of instances, so the random variable can be as simple as a random pair of instances (however we interpret "random"). This value can be readily computed. For gamma, however, observed disagreement is determined on a whole annotation, so the random variable needs to be a whole random annotation. The expected value of such a complicated variable is much more difficult to determine analytically. Instead, gamma uses sampling, as introduced in Section 5.4.

5.3.9 The agreement measure γ

Now that the disorder and the expected disorder have been introduced, we can define the agreement measure (of annotation set s belonging to corpus c, with $c = \{s\}$ if s is a sole annotation set) by Equation 5.6 which comes from Equation 4.2 p. 88:

$$\forall s \in c, \gamma = 1 - \frac{\delta(s)}{\delta_e(c)} \tag{5.6}$$

If all annotators perfectly agree (Figure 5.8(a)), $\gamma = 1$. Figure 5.8(c) corresponds to the worst case, where the annotators are worse than annotating at random, with $\gamma < 0$. Figure 5.8(b) shows an intermediate situation.



(a) best

(b) intermediate

(c) worst

Figure 5.8: Examples of best, intermediate and worst possible disorders

5.4 The implementation of γ

In this section, we first propose an efficient solution to compute the disorder of an annotated continuum, which relies on linear programming. Second, we propose two ways to generate random annotated continua (with respect to the observed distributions) to compute the expected disorder, one relying on a single continuum, the other one relying on a corpus (i.e. several continua). Third, we determine the number of random data sets that we must generate (and compute the disorder of) to get an accurate value of the expected disorder.

5.4.1 Computing the disorder

In order to simplify the discussion and the demonstrations, we consider in this section that n annotators all made the same number of annotations p.

The proposed method has now been fully described on a theoretical level, but, being holistic, its software implementation leads to a major problem of complexity. One can demonstrate that there are theoretically $(p!)^{n-1}$ possible alignments. However, we will (1) show how to reduce the initial complexity, and (2) provide an efficient linear programming solution.

Reducing the initial complexity

The initial number of possible unitary alignments (which are used to build a possible alignment) is p^n. Fortunately, Theorem 5.7 states that any unitary alignment with a cost beyond the value $n \cdot \Delta_\emptyset$ cannot belong to the best alignment, and so can be discarded. Indeed, any unitary alignment with a cost above $\Delta_\emptyset$ can be replaced by creating a separate unitary alignment for each unit (of cost $\Delta_\emptyset$ per unitary alignment, so of total cost $n \cdot \Delta_\emptyset$).

Demonstration Consider the best alignment $\hat{a}$, of cardinality m. Let $\breve{a}$ be any of its unitary alignments. For convenience, we attribute to it the index 1 ($\breve{a} = \breve{a}_1$), while the others are indexed from 2 to m. This unitary alignment $\breve{a}$ contains n units (either real or $u_\emptyset$). For each of these units u_i ($1 \leq i \leq n$), we create the unitary alignment $\breve{a}_{m+i} = (u_i, u_\emptyset, ..., u_\emptyset)$ of cardinality n. It is possible to create an alignment $\bar{a}$ made up of the set of unitary alignments of $\hat{a} \setminus \{\breve{a}\}$, to which we add the unitary alignments $\breve{a}_{m+1}$ to $\breve{a}_{m+n}$ that we have just created.[2] It is of cardinality $m + n - 1$. Since $\hat{a}$ minimizes the disorder, we get:

$$\begin{aligned}\bar{\delta}(\hat{a}) \leq \bar{\delta}(\bar{a}) &\Rightarrow \frac{1}{x}\sum_{i=1}^{m}\breve{\delta}(\breve{a}_i) \leq \frac{1}{x}\sum_{i=2}^{m+n}\breve{\delta}(\breve{a}_i)\\ &\Rightarrow \sum_{i=1}^{m}\breve{\delta}(\breve{a}_i) \leq \sum_{i=2}^{m+n}\breve{\delta}(\breve{a}_i)\\ &\Rightarrow \breve{\delta}(\breve{a}_1) \leq \sum_{i=m+1}^{m+n}\breve{\delta}(\breve{a}_i)\end{aligned}$$

since $\forall i > m, \breve{\delta}(\breve{a}_i) = \frac{1}{C_n^2}(C_n^2\Delta_\emptyset) = \Delta_\emptyset$, and since we have denoted $\breve{a} = \breve{a}_1$,

$$\Rightarrow \breve{\delta}(\breve{a}) \leq n \cdot \Delta_\emptyset \tag{5.7}$$

Experiments have shown that this theorem allows us to discard about 90% of the unitary alignments.

Finding the best alignment: a linear programming solution

Finding the best alignment consists in minimizing the global disorder. A collaboration with J.-P. Métivier has shown that such a problem may be described as a linear programming problem, so that the solution can be computed by a linear programming solver. For more convenience, we introduce two new definitions:

- Let $\mathcal{UA}$ be the set of all unitary alignments.
- Let $\mathcal{UA}_u$ bet the set of the unitary alignments which contain unit u.

The description of the problem in linear programming terms is threefold.

First, for a given alignment $\bar{a}$, for each possible unitary alignment $\breve{a}_i$, we define the **boolean variable** $X_{\breve{a}_i}$ which indicates if this unitary alignment belongs or not to the alignment:

$$\forall \breve{a}_i \in \mathcal{UA}, X_{\breve{a}_i}^{\bar{a}} = \begin{cases} 0 \text{ iff } \breve{a}_i \notin \bar{a} \\ 1 \text{ iff } \breve{a}_i \in \bar{a} \end{cases}$$

Second, we have to express the fact that, by definition, each unit u (of each annotator), should **belong to one and only one unitary alignment** of the alignment $\bar{a}$, that is to say that among all unitary alignments containing u, exactly one $X_{\breve{a}_i}$ equals 1, and all the others equals 0:

$$\forall u \in \mathcal{U}, \sum_{\breve{a}_i \in \mathcal{UA}_u} X_{\breve{a}_i}^{\bar{a}} = 1$$

Third, the **goal** is to minimize the global disorder $\bar{\delta}(\bar{a})$ associated to $\bar{a}$, among all possible alignments $\bar{a}$:

$$Minimize\ \bar{\delta}(\bar{a}) = \sum_{\breve{a}_i \in \mathcal{UA}} \breve{\delta}(\breve{a}_i) \cdot X_{\breve{a}_i}^{\bar{a}}$$

The `LPSolve` solver[3] finds the best solution in less than one second with $n = 3$ annotators and $p = 100$ annotations per annotator on a current laptop (once the initial complexity has been reduced thanks to the previous theorem), which is fast enough to be practical.

[2] $\bar{a}$ is indeed an alignment, since each of its units appears in one and only one unitary alignment.

[3] `http://lpsolve.sourceforge.net`

5.4.2 Implementation of the expected disorder

The first two following subsections detail two strategies to generate randomly annotated continua with respect to the definition of the expected disorder of γ, and the third subsection explains how to choose the number of expected disorder samples to generate so that their average is an accurate enough value of the theoretical expected value. The two strategies correspond to the need expressed in Section 5.3.8 to compute the expected value on the largest set of available data, either a single continuum, or, when available, several continua from the same corpus.

A strategy to compute the expected disorder using a single continuum

When the annotation effort is limited to a single continuum, we can only rely on the annotated continuum itself to compute the expected value. To create random annotations which fulfill the observed distributions, the implemented strategy is as follows: we take the real annotated continuum of an annotator (such as the example shown on the left in Figure 5.9), choose at random a position in this continuum, split the continuum at this position, and permute the two parts of the split continuum. Three examples of split and permutation are shown in the right part of the figure, for split positions of respectively 15, 24 and 38, all coming from the same real continuum, with units that are no longer aligned (except by chance). However, we have to address the fact that some units may intersect with themselves, generating some part of agreement beyond chance. For instance, in Figure 5.9, unit 3 intersects with itself between #15 and #24, because the length of the unit, 12, is higher than the difference of shifts $24 - 15 = 9$. To limit this phenomenon, we do not accept the distance between two shifts to be below the average length of units.

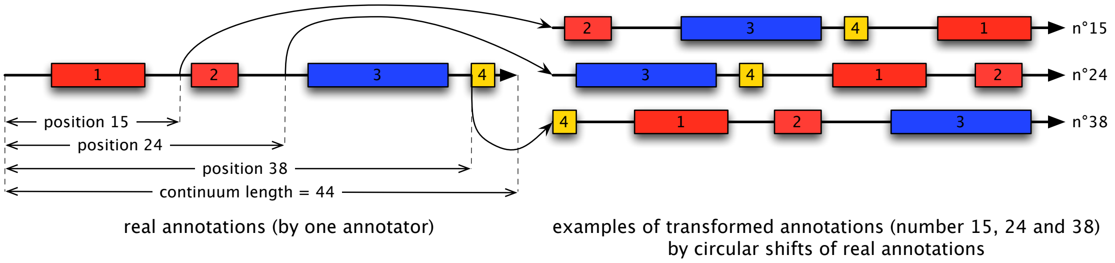

2
3
4
1
n°15
1
2
3
4
3
4
1
2
n°24
4
1
2
3
n°38
position 15
position 24
position 38
continuum length = 44
real annotations (by one annotator)
examples of transformed annotations (number 15, 24 and 38)
by circular shifts of real annotations

Figure 5.9: Principle of circular shift for creating random annotations

A strategy to compute the expected disorder using several continua (from the same corpus)

This strategy consists in mixing annotations coming from different continua, so that their units may align only by chance. To create a random annotation of n annotators, we randomly choose n different continua of the corpus, and pick the annotations of one annotator (randomly chosen) of each of these continua. When different continua are of different lengths, each of them is adjusted to the longest one by duplicating as many times as necessary (like a mosaic).

This is shown in Figure 5.10 for $n = 3$ annotators. We assume the corpus contains 8 continua, each annotated by 3 annotators. To generate a random set of 3 annotations, we have randomly selected a combination of 3 values between 1 and 8, here $(2, 4, 7)$ to select 3 different continua among the 8 available ones of the corpus. Then, for each of these selected continua, we choose one annotator, here annotator 2 for continuum 2, annotator 3 for continuum 4, and annotator 1 for continuum 7. We combine the associated annotations as shown in the right part of the figure, and obtain a set of random annotations which fulfill (on average) the observed distributions. The (very limited) extent of resulting

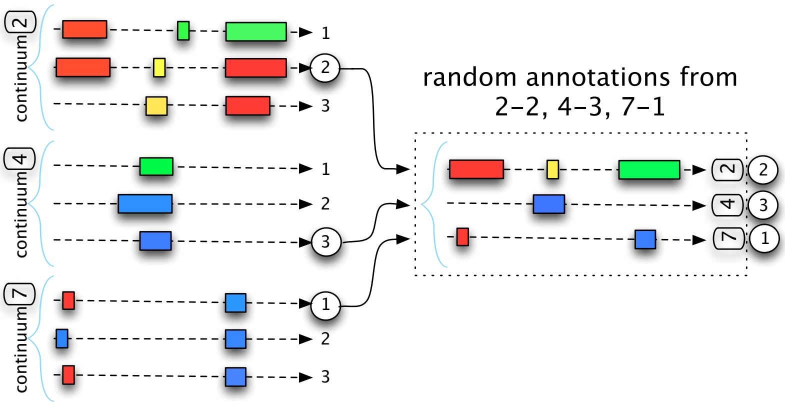


Figure 5.10: A corpus level strategy for creating random annotations (3 annotators)

agreement we can see in this example (only two units have matching categories, but with discrepancies in position) is only by chance since the compared annotations come from different continua.

In addition, it is possible to create a great number of random sets of annotations with this strategy: with n annotators and m continua ($m \geq n$), it is possible to generate up to $C_m^n \cdot n^n$ different combinations. For instance, in our example which assumes $n = 3$ and $m = 8$, there are $56 \cdot 3^3 = 1512$ combinations to create random annotations.

5.4.3 Computing an average value: a sampling question

Since the expected disorder is by definition randomly obtained **on average**, and since there is virtually an infinite number of possible random annotations (with a discrete and finite continuum, it is not really infinite, but still too big to be computed), we can only compute **a reduced but sufficient number of experiments** and get an approximate value of the expected disorder. This is a sampling problem as described for example by Israel (1992). What statistics provide is a way to determine the minimal number n_0 of experiments to do (and to average) so that we get an approximate result of **a given precision** with **a given confidence level**. It consists in: first, taking a small sample to estimate the mean and standard deviation; then, using these estimates to determine the sample size n_0 that is needed.

We follow the strategy provided by Olivero (2001) to compute a disorder value which differs less than $e = 2\%$ from the real value with a $(1-\alpha) = 95\%$ confidence (the software distribution we provide is set by default with these values).

First, we consider a sample of chance disorder values of size $n = 30$. Let μ be the sample mean, and σ' be its standard deviation. μ is directly an unbiased estimator of the population mean, and $\sigma = \sqrt{(\frac{n}{n-1})} \cdot \sigma'$ is an unbiased estimator of the real standard deviation.

Let $C_v = \frac{\sigma}{\mu}$ be the coefficient of variation (i.e. the relative standard deviation).

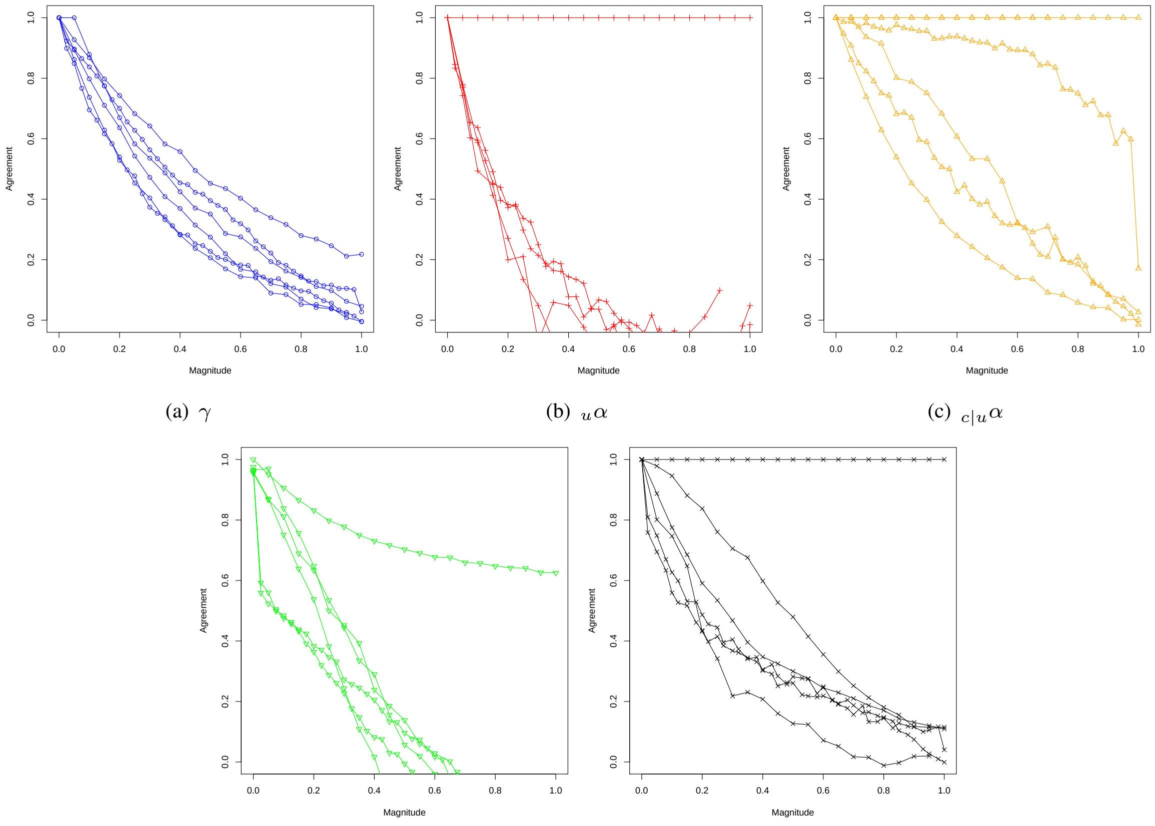
Let $U_{1-\frac{\alpha}{2}}$ be the abscissa of the normal curve that cuts off an area α at the tails. This value is provided in statistical tables. We get n_0 by the following equation:

$$n_0 = \left(\frac{C_v \cdot U_{1-\frac{\alpha}{2}}}{e} \right)^2$$

Let us consider a real example. We generate a sample of random disorders of size $n = 30$. We compute its mean $\mu = 3.49$, its standard deviation $\sigma' = 0.1379$, hence $\sigma = 0.1403$, and $C_v = 0.040188$. We get $U_{1-\frac{0.05}{2}} = 1.96$ from the corresponding available table, hence we get $n_0 = 15.5$. This means that a sample of 16 disorder values gives 2% of precision with 95% confidence. The mean we have already computed with 30 values fulfills this condition, and is a good approximation of the real expected disorder. If we wish to get a high precision of 1%, we get $n_0 = 62$. It is beyond the initial size of our sample (which is 30), and we will have to generate an additional set of 32 values in order to reach the required number.

5.5 benchmarking γ

We cannot go into the details of the benchmarking of γ in this thesis, and we refer the reader to the full presentation provided in (Mathet et al., 2015). However, to get a first idea of the benefits of γ over the other available measures, figure 5.11 depicts the response of different measures, including γ, to different error types via to the Corpus Shuffling Tool.



(a) γ

(b) ${}_u\alpha$

(c) ${}_{c|u}\alpha$

(d) *SER*

(e) κ_d

Figure 5.11: Overviews : γ, ${}_u\alpha$, ${}_{c|u}\alpha$, *SER* and κ_d

Briefly, γ shows a steady behavior for all error types, almost strictly decreasing from 1 to 0, which is the case of no other measure. Some measures have increasing parts, negative values, no response at all (stuck at 1), and so on.

Moreover, γ is the only measure which copes with overlapping units (intersecting or even nested units).

5.6 The additional coefficients γ_{cat} and γ_k

γ is an overall coefficient for all unitizing discrepancies at the same time. When its value is close to 1, annotations can be trusted as reliable, but when it is not the case, this coefficient does not provide an insight of the kind(s) of discrepancy(ies) between annotators: We know the annotations are not reliable, but we do not know what to focus on to improve them.

I have designed an additional coefficient to γ, named γ_{cat}, which focuses on the categorizing part of disagreement between annotators, leaving aside, as much as possible, the unitizing part (in particular positional discrepancies). In simple words, γ_{cat} tries to answer the question: **If annotators had not had to unitize the continuum (put units by themselves and categorize them), but only to categorize predefined units on the continuum, what would have been their agreement?** It shares the same goal as ${}_{cu}\alpha$, the measure belonging to the αs from Krippendorff dedicated to categorization of a continuum, but relies on the same assumptions as γ. In particular, it shares the same alignment method, before it does a specific computation focused on categories.

In addition, an even more in-depth coefficient, named γ_k, is provided which focuses on the agreement on each individual category. It helps to know if a low or moderate γ_{cat} value comes from discrepancies on some particular categories, and so may be useful in order to modify the annotation model or to enhance the annotation instructions. This additional coefficient somehow corresponds to the recent ${}_{k}\alpha$ from Krippendorff et al. (2016), which replaces a first attempt (Krippendorff, 2004).

5.7 Main requirements: what should a categorial measure account for?

In this section, we will see how γ_{cat} should complement γ. The very objective is that γ_{cat} be insensitive to disagreements which involve other aspects of unitizing than categorization (positions, lengths, etc.), contrary to γ. All the points introduced below are benchmarked in section 5.

5.7.1 Positional discrepancies should not impact categorial agreement

Since γ_{cat} aims at providing the agreement on categorization only, it is important that it does not take disagreements on positioning into account. This sounds obvious, but it is not straightforward with unitizing. Figure 5.12 shows a case of perfect agreement on categories which comes with some disagreement on positions: both annotators have identified 3 units at about, but not exactly, the same positions, and totally agree on categorizing these units (respectively with category “1,” “4” and “2”). Hence, a measure focused on categorization should provide a total agreement in such a configuration.

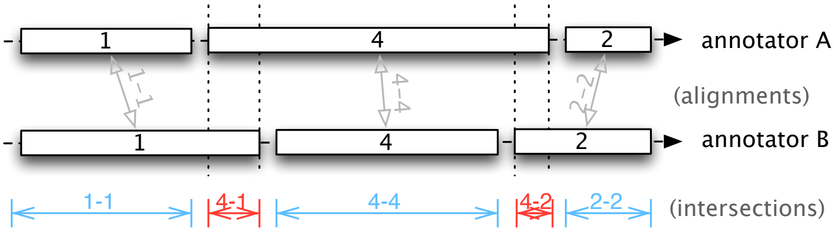


Figure 5.12: Positional discrepancies but perfect agreement on categories

However, measures based on intersections, like the αs, or on atomization of the continuum (a workaround method discussed later), will find a part of the continuum with categorial disagreement, since there are an intersection between units of categories “4” and “1,” and between units of categories “4” and “2.” This is reported in the bottom of Figure 5.12: there are 5 intersections, 3 of them corresponding to correct comparisons, and 2 of them corresponding to unfortunate comparisons. This leads here to about 20% of fake categorial disagreement, according to corresponding intersection lengths.

What a categorial measure should do here is to compare each unit from annotator A to the corresponding one from annotator B (if any), as reported in Figure 5.12 by the 3 gray arrows “1-1,” “4-4” and “2-2,” and assess here a total agreement. This is typically what the γ family is designed to do, thanks to its alignment capability.

5.7.2 False negatives/positives should not impact categorial agreement

We have to be cautious concerning the terminology. A false negative occurs when an annotator fails to put a unit where she should, i.e. where the reference (if any) tells us there should be a unit, and a false positive is the opposite situation. However, no reference exists in the case of agreement measures, and there is a symmetry between false positives and false negatives: if annotator 1 puts a unit where annotator 2 doesn't, it is a false positive if we consider annotator 2 as the reference, or a false negative if we consider annotator 1 as the reference. However, to make this discussion more simple, we will extend the meaning of false positives/negatives to the field of agreement measures.

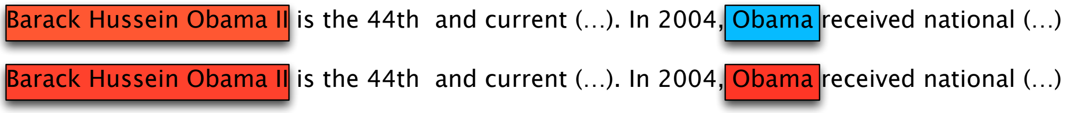
Here again, such disagreements should not be taken into account by a measure focused on categorization. For instance, such a measure should provide a total agreement if annotator A identifies and categorizes 100 units, and annotator B identifies only 50 of them but agrees with A on categories.

5.7.3 Length of units should not be taken into account

For categorization of predefined units, all known coefficients (κ, α, and so on) give the same importance to each item. Why should it be different for unitizing?

In Figure 5.13 (text from Wikipedia), which is an example of a Named Entity Recognition effort, both annotators identified two units, containing respectively "Barack Hussein Obama II" and "Obama." They agree on the category of the first one, and disagree on the category of the second one. This leads to an observed categorial agreement of 50% if we consider, as measures do with predefined units, that all units have the same importance. However, if we rely on unit lengths, the first unit counts 4 times as much as the second one (if we work at word level), and the observed agreement would artificially reach 80% (instead of 50%). This does not make sense for most CL annotation tasks. In this example, it is the same entity which is referred to by a long or a short expression, which confirms, if necessary, that the annotations are of the same importance.

In the same manner, in Sentiment Analysis, it is as important to correctly assess the short "Yes" answer as the twice as long "For sure" one, or as the even longer one "I am absolutely convinced of that".



Barack Hussein Obama II is the 44th and current (...). In 2004, Obama received national (...)

Barack Hussein Obama II is the 44th and current (...). In 2004, Obama received national (...)

Figure 5.13: Units of different lengths, but of the same importance (in Named Entity Recognition)

5.8 How best to handle missing values ?

In categorization tasks, there is a so-called "missing value" (a.k.a. "missing data") when an annotator does not provide a **value** to a given **item**, like a "no opinion" answer. They are inherently and frequently present in unitizing: since annotators have to put units by themselves on a continuum, it is part of the game that they do not put units where others do. However, this question goes beyond the scope of unitizing, and the results of this section concern any categorization measure.

The conceptualization problem here is how to handle the fact that the number of values may differ from one item to another. It is hardly addressed in the literature: Not only do annotation softwares and annotating formats not always provide this possibility to annotators, but many popular coefficients simply cannot handle such data, and even in the reference survey by Artstein and Poesio (2008), this notion is mentioned once but never discussed. As a precursor, Krippendorff's α coefficient was inherently conceived to cope with missing values as early as in 1980 (Krippendorff, 1980); More recently, Gwet (2012) wrote the third version of his handbook specifically to give answers to this question. Each of them provides solutions as we will see below, but as far as I know the present study is the first attempt to compare different approaches.

I will consider in this study that the observed agreement is computed from **pairwise comparisons**. This is the way most coefficients work, including kappas, alphas, and also gamma, even if exceptions exist, like Lotus (Fretwurst, 2015). For instance, if a given predefined item is categorized respectively A, A and B by 3 annotators, the resulting observed agreement is 33.3%. Indeed, there are 3 combinatory pairs : A-A (1 with 2), A-B (1 with 3) and A-B (2 with 3), and so there is one agreement for two disagreements (on the contrary, Lotus would consider that the "most commonly coded value" is A, and that two annotators agree with this value, hence 66.6% of agreement, but a discussion would be out of the scope of this paper).

There are in the literature three very different ways to natively consider missing values in agreement coefficients, and a workaround method introduced just after:

1) **item** level (**IL**). In this conception, all items are given the same weight. Consequently, item 4 from table 5.3 is given the same weight as item 1, which is equivalent to consider that Suzan, Jack and Robert said "noun" for item 4 though they did not say anything.

2) **value** level (**VL**) . This intermediate conception gives the same importance to any pairable value. Since in an item having n_v values, each value can be paired with $n_v - 1$ other values, each pair is weighted $\frac{1}{n_v-1}$ so that the total weight of the value is 1.

3) **pair** level (**PL**). At the extreme opposite end of **IL**, this conception considers any pair of values as having the same weight as any other, whatever the item they belong to. For instance, when Mary says "noun" for item 1 (giving rise to 4 pairs), this weighs four times as much as when she says "noun" for item 4 (giving rise to one pair).

To better understand the differences between these 3 conceptions, table 5.3 shows[4] for each of them the **item** weight w_u, the **value** weight w_v, and the **pair** weight w_p.

Table 5.3: Item, value and pair weight comparisons in the case of 5 annotators with missing values (denoted by ".")

	item 1	item 2	item 3	item 4
Mary	noun	noun	noun	noun
Paul	noun	noun	noun	noun
Suzan	noun	noun	verb	.
Jack	noun	noun	.	.
Robert	noun	.	.	.
n_v = number of values	5	4	3	2
n_p = number of pairs	10	6	3	1
w_u(**IL**) : **item** weight in **IL**	1	1	1	1
w_u(**VL**) : **item** weight in **VL**	5/2	2	3/2	1
w_u(**PL**) : **item** weight in **PL**	10	6	3	1
$w_v(\mathbf{IL}) = 2w_u(\mathbf{IL})/n_v$: **value** weight in **IL**	2/5	1/2	2/3	1
$w_v(\mathbf{VL}) = 2w_u(\mathbf{VL})/n_v$: **value** weight in **VL**	1	1	1	1
$w_v(\mathbf{PL}) = 2w_u(\mathbf{PL})/n_v$: **value** weight in **PL**	4	3	2	1
$w_p(\mathbf{IL}) = w_u(\mathbf{IL})/n_p$: **pair** weight in **IL**	1/10	1/6	1/3	1
$w_p(\mathbf{VL}) = w_u(\mathbf{VL})/n_p$: **pair** weight in **VL**	1/4	1/3	1/2	1
$w_p(\mathbf{PL}) = w_u(\mathbf{PL})/n_p$: **pair** weight in **PL**	1	1	1	1

Key facts are: (1) w_u is steady for **IL** by design, whereas it grows linearly with n_v for **VL**, and with $\frac{n_v(n_v-1)}{2}$ for **PL**. (2) w_v reveals the opposite conceptions of **IL** and **PL**, the first decreasing and

[4] Notice that the values being relative weights, they are comparable only within a given row (since rows have different sums). Accordingly, w_v values are multiplied by 2 for better readability.

the second increasing with n_v, while w_v is steady by design. (3) Since agreement measures rely on pairwise comparisons, w_p discloses the very differences between them. There is up to a ratio of 1 to 10 between the different conceptions, which shows the importance to make the best choice among them.

Besides, a workaround method (rather than a real conception of missing values) to use measures such as κ on such data, which is called **RM** (for "ReMove") hereafter, is simply to remove items which are not valued by all the annotators. In our example, items 2 to 4 would simply be discarded before computation by a standard measure.

In addition to these comparisons, to make an objective choice between these different conceptions (and the workaround method), I have designed a specific experiment reported in Table 5.4. Consider a set of items fully annotated by $n \geq 3$ annotators (column 1). This leads to a given observed pairwise agreement (column 2). Now consider the same initial set of items but with some randomly chosen missing values (w.r.t. the percentage shown in column 3), and apply the different conceptions of missing values to these data. **The better the conceptualization of missing data, the lesser the results should diverge from complete data**. The standard deviation of each conception is reported[5] from column 4 to 7 for 1,000,000 tests from a given set of data, each row corresponding to a certain initial data. Obviously, **VL** steadily shows less deviation than all other conceptions, which makes this conception the best (known) choice under any circumstances. At the opposite, **RM**, i.e. removing the whole item when value(s) is (are) missing is the worst choice. To finish, **IL** and **PL** rank differently depending on the number of annotators and the initial observed agreement.

Table 5.4: Standard deviation of different methods when coping with missing values

# annotators	observed	missing %	σ(**VL**)	σ(**IL**)	σ(**PL**)	σ(**RM**)
6	0.567	25%	0.073	0.077	0.089	0.285
6	0.567	12.5%	0.050	0.051	0.061	0.230
6	0.567	4%	0.028	0.029	0.034	0.100
3	0.905	10%	0.031	0.036	0.038	0.058
3	0.476	5%	0.040	0.051	0.041	0.058

The α measure for predefined units was natively designed to cope with missing values according to **VL**, as explained by Krippendorff (2013a, page 284): "The number of pairs of values from the values-by-units matrix [is] weighted by $\frac{1}{(n_{u.}-1)}$ so that each pairable value in the reliability data adds exactly one to its total count." As a consequence, it is the measure of choice for predefined units with missing values. As a matter of fact, Krippendorff wished to have ${}_{cu}\alpha$ behave as a generalization of α for a continuum, but he failed on this point because ${}_{cu}\alpha$ deeply relies on independent pairwise comparisons of (intersections of) units with no notion corresponding to **items**: "While ${}_{cu}\alpha$ ignores gaps between units, it does it unlike how α ignores missing values." More precisely, ${}_{cu}\alpha$ unfortunately relies on **PL**, whereas α relies on **VL**. Finally, Gwet, in his attempt to adapt classical coefficients to missing values, uses **IL**, as we can see in equation 2.9 of (Gwet, 2012, page 31).

Of course, γ_{cat} relies on the same conception of missing values as α, namely **VL**, since we have just seen it is the best known choice. This is made possible, as we will see, thanks to its alignment process.

5.9 The design of γ_{cat}

As already said, we focus here only on the computation of the disorder of γ_{cat}, which is used to calculate the observed and the expected values. To do so, γ_{cat} uses a four step process, as detailed in the following sections. The main idea is to rely on an alignment of units (provided by γ) to compare the categories used by different annotators to assess the same items. In addition, γ_{cat} uses special

[5] The average result of each method is not reported because interestingly, they all provide the exact initial result in average.

features to cope with **VL** conception of missing values and to improve its accuracy thanks to statistical considerations.

In this introduction to γ_{cat}, we will use the following example of unitized annotations:

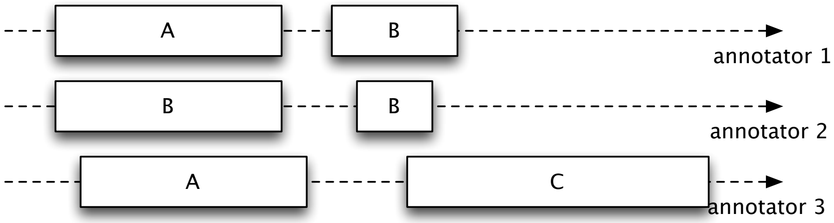
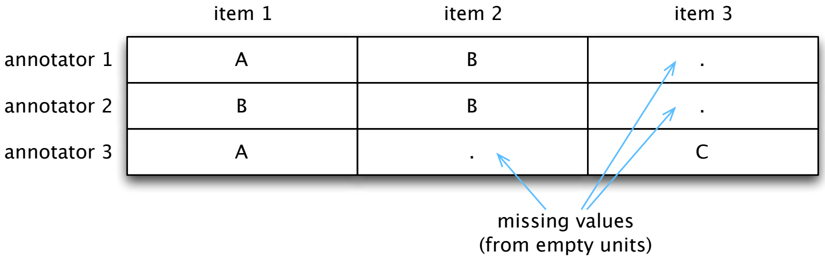


Figure 5.14: Free unitizing by 3 annotators

5.9.1 Getting an alignment of the units from γ

For its first step, γ_{cat} uses the alignment provided by γ[6], which seemingly transforms a difficult unitizing problem into a more simple categorizing of predefined items question, as shown in Figure 5.15.



	item 1	item 2	item 3
annotator 1	A	B	.
annotator 2	B	B	.
annotator 3	A	.	C

missing values
(from empty units)

Figure 5.15: Resulting units/values matrix from unitary alignments

Each unitary alignment translates into a column of a matrix, and each unit belonging to this unitary alignment gives its category as a value in this column. Hence we get a very usual matrix similar to those used for predefined items. To sum up, what is usually called an “item” (and sometimes a “unit”) with predefined items corresponds here to a unitary alignment, and what is usually called a “value” corresponds here to the category of a unit. Of course, empty units generated by γ translate into missing values. The remaining work of γ_{cat} resembles what the usual α (which copes with missing values) does, but there are two important differences as pointed in section 5.9.6.

5.9.2 Value weight: giving the same importance to each value

As we have seen in section 3, it is important that γ_{cat} relies on **VL** conception of missing values. This is done in a simple way here, now that unitary alignments have been translated in kind of predefined items with possibly missing values: we just have to count the number n_v of values in a column, and weight $\frac{1}{n_v - 1}$ each of its pairs.

5.9.3 Confidence weight: enhancing the accuracy of δ

γ_{cat} relies on an alignment, but aligning is a bet, and even if γ was designed to get the most likely overall alignment, it cannot ensure that a particular given pair of aligned units from two annotators

[6] The reason for using the alignment from γ instead of creating an alignment which maximizes the score of γ_{cat} is that a correct alignment relies both on positions and categories.

really corresponds to the same intent of both of them. More precisely, some pairs are aligned with a great confidence (because they correspond both in position and category) whereas others are hardly aligned (γ hesitates to align them). Given this, how to get the most accurate value of the (categorial) disorder δ from our data? We could think about two opposite methods: (1) keeping only pairs of total confidence, hence relying on a trusted but very reduced set of data, or (2) considering that all pairs are of the same importance, and so relying on fake data as much as on trusted data.

However, statistics provide a third method, through the notion of conditional expectation, which takes the best from these two naive methods. To simplify the problem, let us put aside missing values, just addressed in the previous section, and consider we have full alignments with no empty units. In these conditions, **VL**, **IL** and **PL** conceptions are equivalent, and if we had predefined units, the categorial disorder would correspond to the average categorial dissimilarity between all pairs of units.

In the context of unitizing, let $\{pair_i\}$ be the set of pairs of units aligned by γ, let $\delta_i = d_{cat}(pair_i)$ be the categorial dissimilarity of $pair_i$, and let p_i be the probability of the event called "$truePair$" that $pair_i$ really corresponds to a same intent of both annotators. Let D be the random variable defined as the function of dissimilarity between pairs of units from different annotators. The categorial disorder δ we want to estimate is the average value taken by D **for true pairs only**, which formally corresponds to the conditional expectation of D given the event $truePair$, and is provided by equation 5.8:

$$\delta = E(D|truePair) = \frac{1}{\sum_i(p_i)} \cdot \sum_i (p_i \cdot \delta_i) \tag{5.8}$$

In other words, what we really get from an alignment is a "fuzzy set" of true pairs rather than a classical set, and the best estimate of δ we can get from this data is the weighted (by p_i) average value of $\{\delta_i\}$.

For our purpose, I built the probability p_i on positional ground only, because taking categories into account would bias the results: agreements (on categories) would be more weighted than disagreements, which would lead to a lowered overall disorder value. Consequently, the probability p_i is designed so that it equals 1 for two units positioned at the exact same location ($d_{pos} = 0$), and so that it reaches 0 when γ begins to prefer not aligning them because of too much difference in positions (that is to say when d_{pos} reaches 1): for $pair_i = (u_j, u_k)$, $p_i = max(0, 1 - d_{pos}(u_j, u_k))$.

I call this value "pairing confidence", and it is a second weight which will be taken into account in the global computation. Experiments with the Corpus Shuffling Tool (introduced later) have confirmed the benefits from using the notion of confidence weight, which provides an agreement value of 0 with random annotations, which is correct, whereas when not using it, agreement may be slightly below 0, which is not desirable.

5.9.4 Total weight of a pair of units

Figure 5.16 illustrates both the value weight and the pairing confidence weight for each pair of units (i.e. for each pair of values in the table) still for the data coming from Figure 5.14. The total weight for a given pair of units is the product of its value weight and its confidence weight. For instance, the total weight for the pair annotator 1 with annotator 3 of item 1 is 0.5 (because there are 3 values for this item) multiplied by 0.98 (because of the slight positional discrepancy), which is 0.49.

5.9.5 The algorithm to compute the disorder of γ_{cat}

We can now formally define all the steps of the computation of the disorder of γ_{cat}. The detailed procedure is provided in Algorithm 1.

First of all, let us recap the γ terminology: $\hat{a}$ is the best possible alignment computed by γ, i.e. which minimizes the total disorder of its unitary alignments. The unitary alignments are denoted $\breve{a}$, and each of them contains one or zero unit from each annotator, denoted u_1 to u_{n_v}.

The first step, at line 1, is to get $\hat{a}$ exactly as γ does.

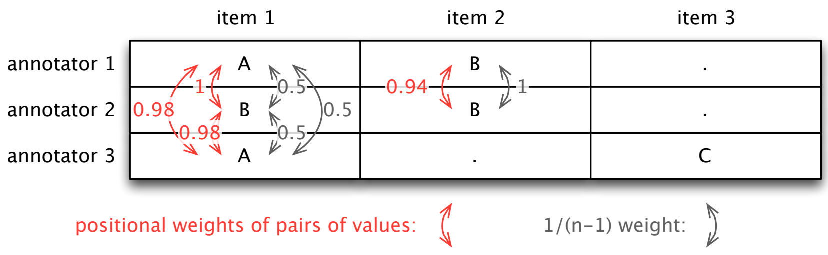


Figure 5.16: Adding weights for **VL** conception and for pairing confidence

Then, a loop, from line 4 to line 14, computes the contribution of each unitary alignement to the total disorder. To do so, it considers the number of true units (i.e. not empty ones) contained in the unitary alignment, and then computes the $\frac{1}{n_v-1}$ weight shared by all pairs of units. Then, it uses a sub loop to enumerate each possible pair of units of the unitary alignment. For each of them, it computes its (categorial) dissimilarity, its own confidence weight, and so gets its resulting weight (product of the shared weight and the confidence weight) and its disorder contribution.

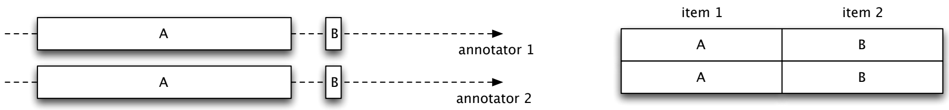
At the end of the main loop, we get the total disorder contribution and the total weight, hence the total disorder.

Algorithm 1 Computation of the total disorder

```
 1: Compute an alignment $\hat{a}$ by using the normal $\gamma$ dissimilarity $d_{combi} = d_{pos} + d_{cat}$
 2: $disorder_{total} \leftarrow 0$
 3: $weight_{total} \leftarrow 0$
 4: for all $\breve{a} \in \hat{a}$ do
 5:     $n_v \leftarrow$ number of real units in $\breve{a}$ (exluding $u_\emptyset$)
 6:     $weight_{base} \leftarrow \frac{1}{n_v-1}$
 7:     for all $(u_i, u_j) \in \breve{a}$ do
 8:         $weight_{confidence} \leftarrow max(0, 1 - d_{pos}(u_i, u_j))$
 9:         $weight \leftarrow weight_{base} \times weight_{confidence}$
10:         $dissimilarity \leftarrow d_{cat}(u_i, u_j)$
11:         $disorder_{total} \leftarrow disorder_{total} + dissimilarity \times weight$
12:         $weight_{total} \leftarrow weight_{total} + weight$
13:     end for
14: end for
15: return $disorder_{total}/weight_{total}$
```

5.9.6 Discussion: Why not to use a naive two step method

Of course, to build such a coefficient, one might think to use a naive method which consists, first, in generating an alignment thanks to γ, and second in applying the α measure (for predefined units) to the resulting matrix (as the one shown in Figure 5.15). However, doing so, we would miss two important points: (1) Obviously, we would not benefit from the statistical enhancement provided by the confidence weight; (2) A more hidden problem is that the expected value computed by α would be biased. Indeed, when units are of different lengths, mixing tabulated values coming from an alignment is not the same as resampling unitized units and then aligning them. For instance, in the example of Figure 5.17 (left: unitizing, right: resulting matrix), the naive method would provide an expected value $\delta_e = 0.5$ (what we get in average from 50% of A and 50% of B), whereas γ_{cat} would provide $\delta_e = 1$, since A and B would never be aligned because of too much differences in lengths, and so only A-A and B-B pairs would occur when resampling unitized units.



item 1	item 2
A	B
A	B

Figure 5.17: Computing the expected value is not the same for unitizing (left) and for predefined items (right)

5.10 The in-depth coefficient γ_k which focuses on each category

γ_k works the same way as γ_{cat} does, except the fact that it focuses on each particular category, and so provides not just one agreement value, but as many agreement values as the number of categories. For instance, if there are 3 categories A, B and C in the annotations, γ_k will provide 3 agreements, namely $\gamma_{k(A)}$, $\gamma_{k(B)}$ and $\gamma_{k(C)}$. I have chosen the letter "k" by reference to $_k\alpha$ from Krippendorff which aims at the same goal.

By focusing on a given category, for instance A, this measure will only look at what a unit of type A is combined with: in our example, A with A, A with B, A with C, but not B with C. Hence, it is γ_{cat} reduced to a subset of pairs of units, only the ones which contain at least one unit of type A.

It is very simple to design γ_k from γ_{cat}: we just have to add one condition in Algorithm 1 so that we keep only relevant pairs of units: more precisely, we add the condition $(cat(u_i) = k) \vee (cat(u_j) = k)$ to line 7 to focus on pairs which concern (at least) one unit of category k only:

```
7:        for all $(u_i, u_j) \in \breve{a} \mid (cat(u_i) = k \vee cat(u_j) = k)$ do
```

Of course, the computation is done as many times as the number of categories, since several agreement values are provided: γ_k is in fact a set of measures.

5.11 Benchmarking γ_{cat} and γ_k

Here again, we refer the reader to the dedicated article (Mathet, 2017) for a comprehensive benchmark of these additional coefficients. Let us just list below the main results provided in the latter article:

- γ_{cat} has been demonstrated to behave the same way as α (the coefficient from Krippendorff that we consider as the best currently available for pure categorization) when we reduce unitizing to pure categorization (with a special set of annotations where positions are steady), which proves that this coefficient has reached the goal to extend to unitizing what works the best for pure categorization (and it is the only unitizing coefficient to do so at present time).

- When facing positional discrepancies via the shuffling tool, γ_{cat} remains steady at 1 up to high magnitudes, which was one of the most important requirements

- The same occurs, as expected, with false positives and false negatives

- γ and γ_{cat} are complementary when facing pure categorial discrepancies (when annotators still agree on positions) with the shuffling tool, since γ_{cat} goes from 1 to 0 as it should, whereas γ is stuck at 0.35 because of agreement on positions, as shown in figure 5.18.

- Reversely, when combining positional and categorial discrepancies, γ_{cat} is now above γ, which shows once again that the two coefficients are complementary, as shown in figure 5.19.

- The corpus shuffling tool has also demonstrated that γ_{cat} is not sensitive to variation of units length, as expected (and contrary to the α family).

- γ_k was tested with a special experiment of the shuffling tool, where disagreement concerned only 3 categories out of 4, and it succeeded in showing on which categories annotators disagree, which makes it a fundamental complement to γ_{cat} (which only provides an overall agreement), as reported in figure 5.20.

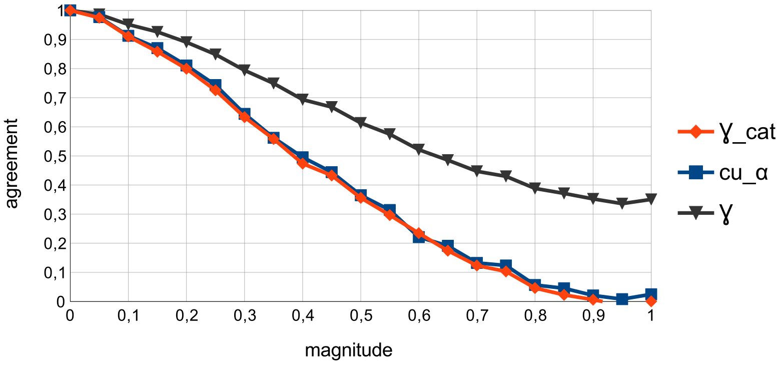


Figure 5.18: Categorial discrepancies

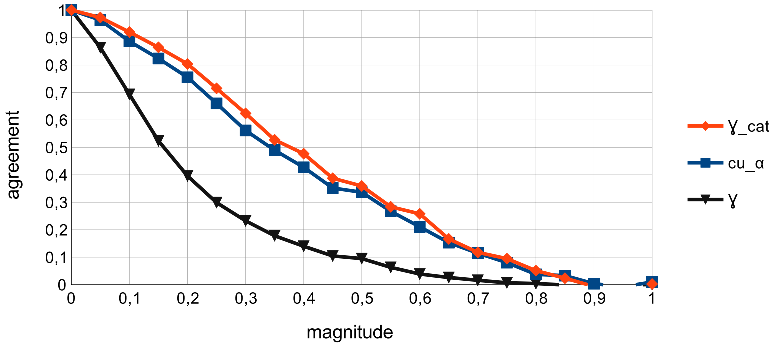


Figure 5.19: Categorial plus positional discrepancies

5.12 Overview and dependencies of the gamma family

Now that γ_{cat} and γ_k have been introduced on the basis of γ, let us recap the links between the three measures, as illustrated in Figure 5.21.

From the multi-annotators annotations, the unified and holist method is used to compute γ and to generate an alignment at the same time. This process relies on an overall dissimilarity which combines positional and categorial dissimilarities. Then, from the alignment and the confidence weights which have been computed by γ, and using only the categorial dissimilarity from the previous step, γ_{cat} and γ_k are computed.

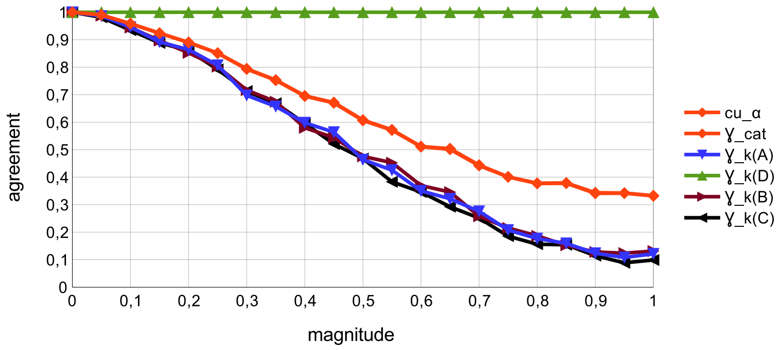


Figure 5.20: Benchmarking γ_k

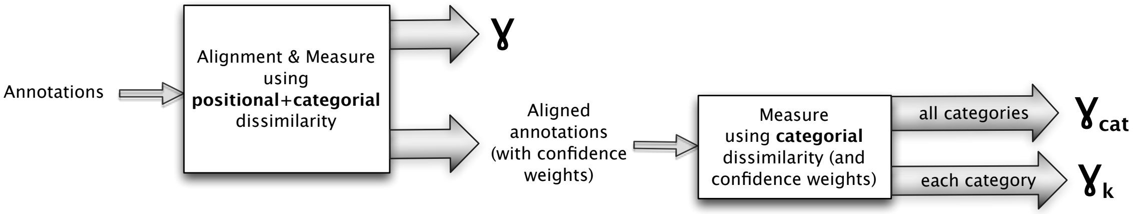

Annotations
Alignment & Measure using positional+categorial dissimilarity
Ɣ
Aligned annotations (with confidence weights)
Measure using categorial dissimilarity (and confidence weights)
all categories
each category
Ɣcat
Ɣk

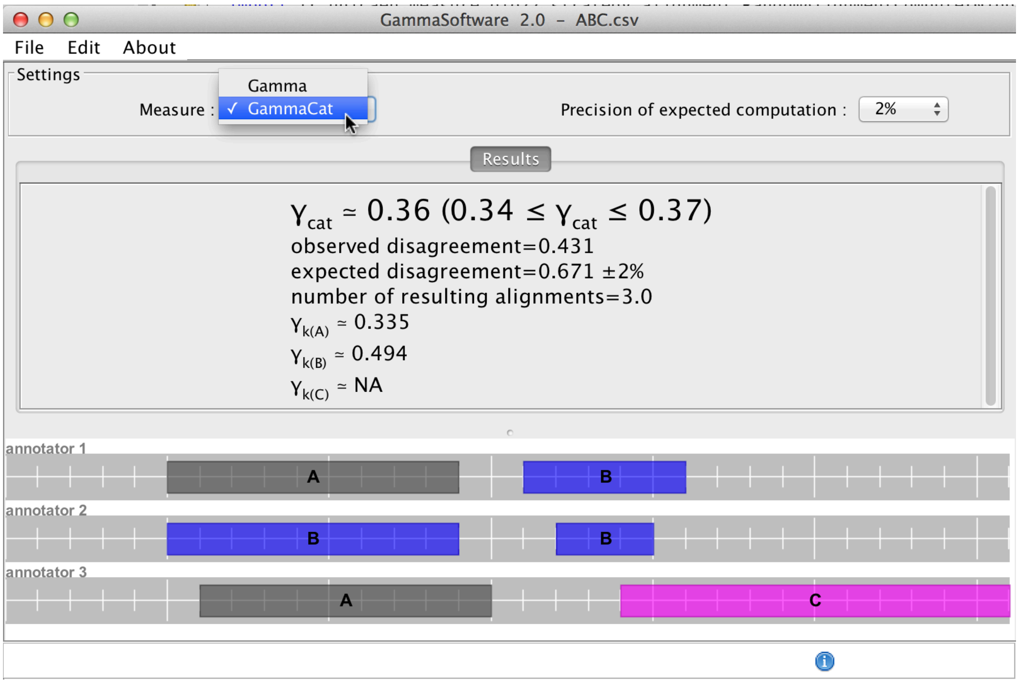
Figure 5.21: Overview of the γ family

5.13 Software

The full implementation of the γ family (γ, γ_{cat} and γ_k) is provided as a free software on the `http://gamma.greyc.fr` website. It is a standalone application written in Java, which runs on any platform, and successfully tested on Mac Os X, Windows and Linux. It is also available as a web service for people who do not want to install it. It is compatible with annotations created with the Glozz Annotation Platform (Widlöcher and Mathet, 2012), and with annotations generated by the Corpus Shuffling Tool (Mathet et al., 2012). Since these formats rely on simple and public CSV specifications, it is easy to translate other formats to these ones.

The application comes with a graphical user interface, as shown in the screenshot of Figure 5.22. The window is divided in 3 panels, respectively, from top to bottom, settings, results, and annotations: In the setting panel, one can choose the measure(s) to apply, either γ, or both γ_{cat} and γ_k. One may also set the desired precision to compute the expected value, since, as explained in (Mathet et al., 2015, page 460), the latter is computed by sampling. In the results panel, all the results are detailed: the agreement, observed and expected values, and also the number of unitary alignments found. In the example, the user chose 2% of precision for the expected value, hence γ_{cat} is known to be between 0.34 and 0.37 with a 95% degree of confidence. Also, the values of γ_k are provided for the 3 categories, and $\gamma_{k(C)}$ is not available (NA) because there is no pair of units containing at least one unit of category C. When the user loads a new file of annotations, or when she changes a setting, the computation is automatically relaunched, so that the results always correspond to what is shown in the interface.

In our example, γ_{cat} is quite low at 0.36, because of confusions between categories A and B, since category C is not contributing to the result as we have just seen. To go deeper into details, γ_k tells that this low agreement is due more to category A ($\gamma_{k(A)} = 0.335$) than to category B ($\gamma_{k(B)} = 0.494$). Moreover, $\gamma = 0.29$ (not visible in the screenshot since one have to click on “Gamma” to make it appearing), is quite close to γ_{cat}, which tells that the annotators have to improve both unitizing and categorization.



Measure : ✓ GammaCat

Precision of expected computation : 2%

$\gamma_{cat} \simeq 0.36$ $(0.34 \leq \gamma_{cat} \leq 0.37)$
observed disagreement=0.431
expected disagreement=0.671 ±2%
number of resulting alignments=3.0
$\gamma_{k(A)} \simeq 0.335$
$\gamma_{k(B)} \simeq 0.494$
$\gamma_{k(C)} \simeq$ NA

Figure 5.22: The γ familly software

5.14 Conclusion on the Gamma family

In computational linguistics, when annotation efforts are relative to a continuum (which implies the combination of positioning and categorizing) rather than to predefined items, researchers are not provided very much with methods and tools to assess the agreement among several annotators.

In the end, only Krippendorff's coefficients ${}_u\alpha$ and ${}_{c|u}\alpha$ was specifically addressing unitizing, but they do not fulfill all the needs of CL, in particular by giving much importance to size of units, and with the restriction that they are natively limited to non-overlapping units.

The main reason why research on this topic is sparse, and why it may be difficult to enlarge Krippendorff's coefficients to overlapping units, probably results from the fact that we are facing here a major difficulty: the simultaneous double discrepancy between annotators, with annotations possibly differing both in positioning relevant units anywhere on a continuum, and in categorizing each of these free units. Consequently, it is difficult for a method to choose precisely which features to compare between different annotators (unlike pure categorizing, where we know exactly what each annotator says for each predefined element to be categorized), and this problem is exacerbated when overlapping units (within an annotator) occur.

To cope with this critical point, we advocate the use of an alignment which ultimately expresses which unit from one annotator should be compared to which unit, if any, from another one, and consequently makes it natural and easier to compute the agreement. Moreover, we have shown that this alignment cannot be done in an independent way, but is part of the measure method itself. This is the "unified" aspect of our approach. We have also shown that in order to be relevant, this alignment cannot be done at a local level (unit by unit), but should consider the whole set of annotations at the same time, which is the "holistic" aspect.

Recently, we designed a set of new coefficients, the Gamma family, for that purpose. γ proposes an overall solution which takes into account all kinds of discrepancies (categories, positions, false positives and negatives) in order to assess whether the multi-annotations are reliable or not.

However, when the agreement is not as good as wished, the researchers would like to have more details about the discrepancies, in order to better understand the difficulties and so to enhance the annotation model or the annotation manual. In particular, is a given overall low agreement due to poor specifications of categories? Or even of some particular categories? For that purpose, I also provided such complements to γ, with two additional coefficients γ_{cat} and γ_k which focus on the categorization part of the agreement, with the wish they also fulfill three important requirements for computational linguistics: (1) Positional discrepancies should not impact categorial agreement; (2) Length of units should not be taken into account; (3) Missing values should be tackled the best way. γ_{cat} was designed not only as a complement to γ, but also with the same conception of how to handle unitizing, and with a common alignment process. Relying on an alignment, it compares genuine units and so ensures requirements (1) and (2).

Since the aim of γ_{cat} is somehow to extend what agreement measures do for predefined items to the case of unitizing a continuum, it was important that γ_{cat} performs as well as the best specialized measures. Moreover, the context of free unitizing leads to a great number of so called “missing values”, when some annotators put units where others do not, which led me to frontally study this other neglected question for requirement (3): how a measure should natively handle missing values? I made a thorough analysis of the question and formulated a clear answer: the best solution is to do as the classic α measure does (and what $_{cu}\alpha$ unfortunately fails to do). This is also a result which goes beyond the scope of this paper focused on unitizing. γ_{cat} manages to do (almost) exactly the same as α when restrained to the simpler case of predefined units, which constitutes a strong basis.

Finally, the Gamma family fulfills all the requirements expressed for computational linguistics. Experiments with the shuffling tool confirm all these capabilities, and also that the three coefficients γ, γ_{cat} and γ_k are complementary.

These coefficients are already implemented, ready to use, and freely available.

5.15 Future work

If we have reached a first important step for CL and NLP agreement measures with the Gamma family, there are still many other points to address in the future. The three main goals we intend to achieve are :

1. Improving the understanding of the part of chance in agreement. As we have seen, this is an important and difficult point which deserves further work and probably some experiments. In particular, we wish to find a way to correctly account for the fact that distribution of categories may change through a given continuum, which is not taken into account by any measure at present time.

2. An important problem with assessment of unitizing is that the way we measure the agreement depends on the nature of the task. For certain tasks, a little disagreement in position is important, whereas for others it may be almost negligible. Currently, the Gamma family is set with what we consider as the most versatile measures of dissimilarity, but we wish in the future to propose special settings for each kind of annotation task.

3. A recurrent need expressed by scholars is also the possibility for measures to take into account feature-sets in the computation in the agreement, in addition to mere categories. For instance, if there are two categories Noun and Verb, and if these categories come with feature sets saying for instance what is the "gender" of a given Noun, or what are the "tense" and "aspect" of a given Verb, the current measures only take into account the agreement on Noun and Verb, not on the gender, tense nor aspect. We have ongoing developments to fulfill this need.

4. Of course, we also wish in the future to provide new coefficients for even more difficult tasks, in particular the case where unitizing is complemented by relationships (i.e. when units can be linked, like in discourse annotation). We think that γ is an interesting start point for such a

measure, but we will probably face complexity problems when computing an alignment which relies not only on the units but also on the relationships between them.

Bibliography

Aickin, M. (1990). Maximum likelihood estimation of agreement in the constant predictive probability model, and its relation to cohen's kappa. *Biometrics*, 46:293–302.

Artstein, R. and Poesio, M. (2008). Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Asher, N., Venant, A., Muller, P., and Afantenos, S. (2011). Complex discourse units and their semantics. In *Proceedings of Constraints in Discourse*.

Battistelli, D., Chagnoux, M., and Desclès, J.-P. (2006). Référentiels et ordonnancements temporels dans les textes, information temporelle, procédures et ordre discursif. In *Cahier Chronos 18*.

Bennett, E. M., Alpert, R., and C.Goldstein, A. (1954). Communications through limited questioning. *Public Opinion Quarterly*, 18(3):303–308.

Bestgen, Y. (2009). Quels indices pour mesurer l'efficacité en segmentation thématique? In *Actes de TALN'09*, Senlis (France).

Bilhaut, F. and Widlöcher, A. (2006). Linguastream: An integrated environment for computational linguistics experimentation. In *Proceedings of the 11th Conference of the European Chapter of the Association of Computational Linguistics (EACL'06)*, pages 95–98.

Bonnardel, P. (1996). *Test statistique Kappa : programmation informatique et applications pratiques*. Phd thesis, Université de Paris V.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Desclès, J.-P. (1995). Les référentiels temporels pour le temps linguistique. In *Modèles Linguistiques XVI*, pages 9–36.

Di Eugenio, B. and Glass, M. (2004). The kappa statistic: a second look. *Computational Linguistics*, 30(1):95–101.

Fauconnier, G. (1984). *Espaces mentaux : Aspects de la construction du sens dans les langues naturelles*. Les éditions de Minuit.

Feinstein, A. and Cicchetti, D. (1990). High agreement but low kappa : The problems of two paradoxes. *Clin. Epidemiol.*, 43:543–548.

Fretwurst, B. (2015). Reliability and accuracy with lotus. In *Proc. of the 65'th ICA Annual Conference*, San Juan, Puerto Rico. International communication association, ICA.

Gosselin, L., Mathet, Y., Enjalbert, P., and Bécher, G. (2013). *Aspects de l'Itération, L'expression de la répétition en français : analyse linguistique et formalisation*, volume 106. Peter Lang éditions, Sciences pour la communication, Laurent Gosselin, Yann Mathet, Patrice Enjalbert, Gérard Bécher.

Gwet, K. L. (2012). *Handbook of Inter-rater Reliability*. Advanced Analytics, LLC, third edition.

Ho-Dac, L.-M., Péry-Woodley, M.-P., and Tanguy, L. (2010). Anatomie des Structures Énumératives. In *Actes de la conférence TALN 2010*, Montréal, Canada.

Israel, G. D. (1992). Determining Sample Size. *Agricultural Education and Communication Department, University of Florida, IFAS Extension, PEOD6 (Reviewed 2013).*

Krippendorff, K. (1980). *Content Analysis : An Introduction to Its Methodology*, chapter 12. Sage : Beverly Hills, CA.

Krippendorff, K. (2004). *Content Analysis : An Introduction to Its Methodology*, chapter 11. Sage : Thousand Oaks, CA., 2nd edition.

Krippendorff, K. (2011). Agreement and Information in the Reliability of Coding. *Communication Methods and Measures*, (5.2):1–20.

Krippendorff, K. (2013a). *Content Analysis : An Introduction to Its Methodology*, chapter 11. Sage : Thousand Oaks, CA., 3rd edition.

Krippendorff, K. (2013b). A dissenting view on so-called paradoxes of reliability coefficients. *C. T. Salmon (ed.), Communication Yearbook*, 36:481–499.

Krippendorff, K., Mathet, Y., Bouvry, S., and Widlöcher, A. (2016). On the reliability of unitizing textual continua: Further developments. *Quality and Quantity*, 50(6):2347–2364.

Kuper, J., Saggion, H., Cunningham, H., Declerck, T., de Jong, F., Reidsma, D., Wilks, Y., and Wittenburg, P. (2003). Intelligent multimedia indexing and retrieval through multi-source information extraction and merging. In *IJCAI*, pages 409–414.

Labadié, A., Enjalbert, P., Mathet, Y., and Widlöcher, A. (2010). Discourse structure annotation : Creating reference corpora. In *Workshop on Language Resource and Language Technology Standards - state of the art, emerging needs, and future developments*, La Valetta, Malta. Conference LREC 2010.

Landragin, F., Poibeau, T., and Victorri, B. (2012). Analec: A new tool for the dynamic annotation of textual data. In *Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, pages 357–362.

Laur, D. (1991). *Sémantique du déplacement et de la localisation en français : une étude des verbes, des prépositions et de leurs relations dans la phrase simple*. PhD thesis, University of Toulouse II.

Lebranchu, J. (2011). *Étude des phénomènes itératifs en langue : Inscription discursive et Calcul aspectuo-temporel, Vers un traitement automatisé*. PhD thesis, University of Caen, France.

Mathet, Y. (2000). *Etude de l'expression en langue de l'espace et du déplacement : analyse linguistique, modélisation cognitive, et leur expérimentation informatique*. PhD thesis, University of Caen, France.

Mathet, Y. (2017). The agreement measure gamma-cat (γ_{cat}), a complement to gamma focused on categorization of a continuum. *Computational Linguistics*, 43(3):661–681.

Mathet, Y. and Widlöcher, A. (2011a). Stratégie d'exploration de corpus multi-annotés avec GlozzQL. In *Traitement Automatique des Langues Naturelles 2011 (TALN 2011)*, Montpellier, France.

Mathet, Y. and Widlöcher, A. (2011b). Une approche holiste et unifiée de l'alignement et de la mesure d'accord inter-annotateurs. In *Traitement Automatique des Langues Naturelles 2011 (TALN 2011)*, Montpellier, France.

Mathet, Y., Widlöcher, A., Fort, K., Francois, C., Galibert, O., Grouin, C., Kahn, J., Rosset, S., and Zweigenbaum, P. (2012). Manual corpus annotation: Giving meaning to the evaluation metrics. In *COLING 2012*, Mumbai, India.

Mathet, Y., Widlöcher, A., and Métivier, J.-P. (2015). The unified and holistic method gamma (γ) for inter-annotator agreement measure and alignment. *Computational Linguistics*, 41(3):437–479.

Mathet, Y. and Widlöcher, A. (2016). Évaluation des annotations : ses principes et ses pièges. *Revue T.A.L.*, 52(2):73–98.

Olivero, P. (2001). Calcul de la taille des Échantillons. *CETE du Sud-Ouest / DAT / ZELT*.

Person, C. (2004). *Traitement automatique de la temporalité du récit : implémentation du modèle linguistique SdT*. PhD thesis, University of Caen.

Péry-Woodley, M.-P., Asher, N., Enjalbert, P., Benamara, F., Bras, M., Fabre, C., Ferrari, S., Ho-Dac, L.-M., Le Draoulec, A., Mathet, Y., Muller, P., Prévot, L., Rebeyrolle, J., Tanguy, L., Vergez-Couret, M., Vieu, L., and Widlöcher, A. (2009). ANNODIS: une approche outillée de l'annotation de structures discursives. In *Actes de la 16e Conférence Traitement Automatique des Langues Naturelles (TALN'09), session poster*, Senlis, France.

Pevzner, L. and Hearst, M. (2002a). A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Pevzner, L. and Hearst, M. A. (2002b). A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Reichenbach, H. (1947). *Elements of Symbolic Logic*. Macmillan & Co, New York.

Reidsma, D. (2008). *Annotations and Subjective Machines of Annotators, Embodied Agents, Users, and Other Humans*. Phd thesis, University of Twente. publisher: Twente University Press, publisherlocation: Enschede, ISSN: 1381-3617, ISBN: 90-365-2726-0.

Reidsma, D., Heylen, D., and Ordelman, R. (2006). Annotating emotions in meetings. In *Proc. of the fifth international conference on Language Resources and Evaluation, LREC 2006*, pages 1117–1122, Paris. ELRA. ISBN=2-9517408-2-4.

Rumbaugh, J., Jacobson, I., and Booch, G. (2004). *Unified Modeling Language Reference Manual*. Pearson Higher Education, 2nd edition.

Sablayrolles, P. (1995). *Sémantique formelle de l'expression du mouvement. De la sémantique lexicale au calcul de la structure du discours en français*. PhD thesis, University Paul Sabatier, Toulouse.

Scott, W. (1955). Reliability of content analysis: The case of nominal scale coding. *Public Opinion Quarterly*, 19(3):321–325.

Vendler, Z. (1957). Verbs and times. *The Philosophical Review*, 66(2):143–160.

Widlöcher, A. (2008). *Analyse macro-sémantique des structures rhétoriques du discours - Cadre théorique et modèle opératoire*. PhD thesis, Université de Caen Basse-Normandie.

Widlöcher, A. and Mathet, Y. (2009). La plate-forme Glozz: environnement d'annotation et d'exploration de corpus. In *Actes de la 16e Conférence Traitement Automatique des Langues Naturelles (TALN'09), session posters*, Senlis, France.

Widlöcher, A. and Mathet, Y. (2012). The glozz platform: a corpus annotation and mining tool. In Concolato, C. and Schmitz, P., editors, *ACM Symposium on Document Engineering (DocEng'12)*, pages 171–180, Paris, France. ACM.

Zhao, X., Liu, J., and Deng, K. (2013). Assumptions behind inter-coder reliability indices. *C. T. Salmon (ed.), Communication Yearbook*, 36:418–480.

Zwick, R. (1988). Another look at interrater agreement. *Psychological Bulletin*, (103):347–387.