14

Discourse

14.1 Introduction

The grammatical concepts we have seen so far apply mostly to isolated words, phrases, or sentences. Texts and conversations, either full or partial, are out of their scope. Yet to us, human readers, writers, and speakers, language goes beyond the simple sentence. It is now time to describe models and processing techniques to deal with a succession of sentences. Although analyzing texts or conversations often requires syntactic and semantic treatments, it goes further. In this chapter, we shall make an excursion to the discourse side, that is, paragraphs, texts, and documents. In the next chapter, we shall consider dialogue, that is, a spoken or written interaction between a user and a machine.

Most basically, a discourse is made of **referring expressions**, i.e., words or phrases that refer to real – or possibly imaginary – things: the **discourse entities** or **discourse referents**. A first objective of discourse processing techniques is then to identify and track sets of referring expressions – phrases or words – along with sentences and to relate them to entities – real-world objects.

A discourse normally links the entities together to address topics, issues throughout the sentences, paragraphs, chapters such as, for instance, the quality of food in restaurants, the life of hedgehogs and toads, and so on. At a local level, i.e., within a single sentence, grammatical functions such as the subject, the verb, and the object provide a model of relations between entities. A model of discourse should extend and elaborate relations that apply not to an isolated sentence but to a sequence and hence to the entities that this sequence of sentences covers.

Models of discourse structures are still a subject of controversy. As for semantics, discourse has spurred many theories, and it seems relatively far off to produce a synthesis of them. In consequence, we will merely adopt a bottom-up and pragmatic approach. We will start from what can be a shallow-level processing of discourse and application examples; we will then introduce theories, namely centering, rhetoric, and temporal organization, which provide hints for a discourse structure.

14.2 Discourse: A Minimalist Definition

14.2.1 A Description of Discourse

Intuitively what defines a discourse, and what differentiates it from unstructured pieces of text, is its coherence. A discourse is a set of more or less explicit topics addressed in a sequence of sentences: what the discourse is about at a given time. Of course, there can be digressions, parentheses, interruptions, etc., but these are understood as exceptions in the flow of a normal discourse. Distinctive qualities of a discourse are clarity, expressiveness, or articulation, which all relate to the ease of identification of discourse topics and their logical treatment. Discourse coherence ideally takes the shape of a succession of stable subjects (or contexts) that are chained rationally along with the flow of sentences.

More formally, we describe a discourse as a sequence of utterances or segments, $S_1, S_2, S_3, ..., S_n$, so that each of these segments is mapped onto a stationary context. Segments are related to sentences, but they are not equivalent. A segment can span one or more sentences, and conversely a sentence can also contain several segments. Segments can be produced by a unique source, which is the case in most texts, or by more interacting participants, in the case of a dialogue.

14.2.2 Discourse Entities

Discourse entities – or discourse referents – are the real, abstract, or imaginary objects introduced by the discourse. Usually they are not directly accessible to a language processing system because it would require sensors to "see" or "feel" them. In a language like Prolog, discourse entities are represented as a set of facts stored in a database. Referring expressions are mentions of the discourse entities along with the text. Table 14.1 shows entities and references of sentences adapted from Suri and McCoy (1994):

1. *Susan drives a Ferrari*
2. *She drives too fast*
3. *Lyn races her on weekends*
4. *She often beats her*
5. *She wins a lot of trophies*

Table 14.1. Discourse entities and referring expressions.

Referring expressions	Discourse entities (or referents)	Logic properties
Susan, she, her	`'Susan'`	`'Susan'`
Lyn, she	`'Lyn'`	`'Lyn'`
A Ferrari	`X`	`ferrari(X)`
A lot of trophies	`E`	`E` $\subset$ `{X, trophy(X)}`

Discourse entities are normally stable – constant – over a segment, and we can use them to delimit a segment's boundaries. That is, once we have identified the entities, we can delimit the segment boundaries. Let us come back to our example. There are two sets of relatively stable entities that we can relate to two segments. The first one is about Susan and her car. It consists of sentences 1 and 2. The second one is about Susan and Lyn, and it extends from 3 to 6 (Table 14.2).

Table 14.2. Context segmentation.

Contexts	Sentences	Entities
C1	1. *Susan drives a Ferrari*	Susan, Ferrari
	2. *She drives too fast*	
C2	3. *Lyn races her on weekends*	Lyn, Susan, trophies
	4. *She often beats her*	
	5. *She wins a lot of trophies*	

14.3 References: An Application-Oriented View

As a starting point of discourse processing, we will focus on referring expressions, i.e., words or phrases that correspond to the discourse entities. This treatment can be done fairly independently without any comprehensive treatment of the text. In addition, the identification of discourse entities is interesting in itself and has an industrial significance in applications such as information extraction.

In this section, we will take examples from the Message Understanding Conferences (MUCs) that we already saw in Chap. 9. We will learn how to track the entities along with sentences and detect sets of phrases or words that refer to the same thing in a sentence, a paragraph, or a text.

14.3.1 References and Noun Phrases

In MUC, information extraction consists in converting a text under the form of a file card. Cards are predefined templates whose entries are formatted tabular slots that represent the information to be extracted: persons, events, or things. For each text, information extraction systems have to generate a corresponding card whose slots are filled with the appropriate entities.

Detecting – generating – the entities is a fundamental step of information extraction; a system could not fill the templates properly otherwise. To carry it out, the basic idea is that references to real-world objects are equivalent to noun groups or noun phrases of the text. So detecting the entities comes down to recognizing the nominal expressions.

To realize in concrete terms what it means, let us take an example from Hobbs et al. (1997) and identify the entities. We just have to bracket the noun groups and to assign them with a number that we increment with each new group:

[$_{entity1}$ Garcia Alvarado], 56, was killed when [$_{entity2}$ a bomb] placed by [$_{entity3}$ urban guerrillas] on [$_{entity4}$ his vehicle] exploded as [$_{entity5}$ it] came to [$_{entity6}$ a halt] at [$_{entity7}$ an intersection] in [$_{entity8}$ downtown] [$_{entity9}$ San Salvador].

We have detected nine nominal expressions and hence nine candidates to be references that we represent in Table 14.3.

Table 14.3. References in the sentence: *Garcia Alvarado, 56, was killed when a bomb placed by urban guerrillas on his vehicle exploded as it came to a halt at an intersection in downtown San Salvador*.

Entities	Noun groups
Entity 1	*Garcia Alvarado*
Entity 2	*a bomb*
Entity 3	*urban guerrillas*
Entity 4	*his vehicle*
Entity 5	*it*
Entity 6	*a halt*
Entity 7	*an intersection*
Entity 8	*downtown*
Entity 9	*San Salvador*

Typical discourse analyzers integrate modules into an architecture that they apply on each sentence. Depending on applications, they use a full-fledged parser or a combination of part-of-speech tagger, group detector, semantic role identifier, or ontological classifier. Here, we could have easily created these entities automatically with the help of a noun group detector. A few lines more of Prolog to our noun group detector (Chap. 9) would have numbered and added each noun group to an entity database.

14.3.2 Finding Names – Proper Nouns

In unrestricted texts, in addition to common nouns, many references correspond to names (or proper nouns). Their detection is then central to a proper reference processing. Names include:

- Persons: Mrs. Smith, François Arouet, Dottore Graziani, Wolfgang A. Mozart, H.C. Andersen, Sammy Davis, Jr.
- Companies or organizations: IBM Corp., Fiat SpA, BT Limited, Banque National de Paris, Siemens GMBH, United Nations, Nations unies
- Countries, nations, or provinces: England, France, Deutchland, Romagna, Vlanderen
- Cities or geographical places: Paris, The Hague, Berlin, le Mont Blanc, la Città del Vaticano, the English Channel, la Manche, der Rhein

Name recognition frequently uses a dedicated database and the help of some heuristics. Such name databases can sometimes be downloaded from the Internet. However, for many applications, they have to be compiled manually or bought from specialized companies. A name recognition system can then be implemented with local DCG rules and a word spotting program (see Chap. 9).

However, name databases are rarely complete or up-to-date. Peoples' names particularly are tricky and may sometimes be confused with common names. The same can be said of names of companies, which are created every day, and those of countries, which appear and disappear with revolutions and wars. If we admit that there will be names missing in the database, we have to design the word spotter to cope with it and to implement some rules to guess them.

Guessing a person's name is often done through titles and capitalization. A few rules of thumb attempt to match:

- A first name, a possible initial, and a surname that is two strings of characters with a capitalized first letter followed by lower case letters:
  Robert Merryhill, Brigitte Joyard, Max Hübnisch
  A possible enhancement is to try to match the first string to common first names.
- A title, possible first names or initials and a surname where common titles have to be itemized:
  Sir Robert Merryhill, Dr. B. K. Joyard, Herr Hübnisch
- A person's name and a suffix:
  R. Merryhill Sr., Louis XXII, Herr Hübnisch d. med.

These heuristics can be implemented with a word spotter and DCG rules to match titles and first names. A short piece of Prolog code will also have to test the case of certain characters. We can also use regular expressions or a stochastic classifer.

14.4 Coreference

14.4.1 Anaphora

In the example of the previous section, we have numbered eight objects corresponding to noun groups and one corresponding to the pronoun *it*. Such a pronoun is generally related to a previous expression in the text and depends on this expression to be interpreted. Here, the reader can easily guess that *it* and the noun group *his vehicle* designate the same entity. This means that entities 5 and 4 in Table 14.3 are equal and that nominal expressions *his vehicle* and *it* **corefer** to a same thing (Fig. 14.1).

The pair of expressions *his vehicle* and *it* form an **anaphora** where the first reference to the object – *his vehicle* – is the **antecedent** and subsequent references – here *it* – are **anaphors**. Antecedent and anaphors are then a set of references to a same entity in a text. The antecedent acts as the semantic source of the set and enables the understanding of the anaphors (Tesnière 1966).

Third-person and relative pronouns (*he/she/it/who*) are typical examples of anaphors. In addition to them, anaphora uses demonstrative pronouns such as *this/that*

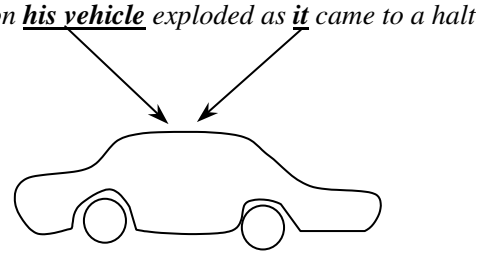


Fig. 14.1. Coreferencing an entity with a noun group and a pronoun.

in *He did that*, or location adverbs such as *here*/*there* in *He was there*. Demonstrative pronouns (or adjectives) can be used as determiners as in *this vehicle*. The possessive pronouns (or adjectives) are similar and also denote an anaphora as *his* in *his vehicle* that is tied to the vehicle's possessor.

While normally anaphors have their antecedent before they occur, there are sometimes examples of forward references or **cataphora**. For example, in the sentence:

I just wanted to touch ***it****, this stupid animal.*

It refers to the *stupid animal*. It has been shown that in most structured discourses, cataphoras occur in the same sentence. However, this is not always the case, and sometimes the referent is never mentioned, either because it is obvious given the context, because it is unknown, or for some other reasons:

They have stolen my bicycle.

14.4.2 Solving Coreferences in an Example

Although we had no difficulty recognizing the identity of the two expressions, *his vehicle* and *it* in the example above, coreference resolution is not as straightforward as it may appear at a first sight. Let us come back to our example in Table 14.3 to show this, and let us make our method explicit to outline an algorithm. We will first admit that coreferences of a pronoun are always located before the pronoun occurs in the text. Then, in the example, in addition to *his vehicle* (entity 4), pronoun *it* (entity 5) has four possible candidates: *it* could be *Garcia Alvarado* (entity 1), *a bomb* (entity 2), or *urban guerrillas* (entity 3).

We can rule out entities 1 and 3 from the coreference set because they do not match the pronoun's number or gender. If entity 5 had been *Garcia Alvarado* – a man – the pronoun would have been *he*, and if it had been *urban guerrillas* – a plural – the pronoun would have been *they*. The noun group *A bomb* is more difficult to discard. We do not retain it because of a semantic incompatibility. Selectional restrictions of the verb *came* likely require that its subject is a vehicle or a person.

We saw examples of anaphora where a same entity is specified by a noun and a pronoun. Pairs of references can also consist of nouns or noun groups. They can simply be a repetition of identical expressions, (*the vehicle*, *the vehicle*). Sometimes

there might be a different determiner, a different denomination, synonyms, or aliases to refer to a same thing. For instance, in an economic wire, we can first have *Bayerische Motoren Werke*, then *BMW*, and finally *the German automaker*.

Coreference is a far-reaching concept that can prove very complex. The definition of anaphora may also vary: most authors restrain anaphors to be pronouns and certain types of adverbs. Others extend it to noun phrases, either definite or not. In the rest of the text, we will make no distinction, and we will define coreference resolution or coreference recognition as the retrieval of sets of references to identical entities in a text – what we have just come to do. We will also keep the terms antecedent and anaphor to refer to the first and second term of a coreferring pair, even if the anaphor is not a pronoun.

14.4.3 A Standard Coreference Annotation

Before we explain general methods to solve coreferences, let us first examine an annotation scheme proposed in the sixth and seventh Message Understanding Conferences (MUC-6 and MUC-7) to tag them. While there are various mark-up models, this one, based on XML tags, is widely public and can be considered as a standard. In addition, as the MUC's final objective is to extract information, these tags have an application interest.

The annotation of references and coreferences in a text consists first of identifying of the referring expressions and then assigning a unique label to expressions referring to a same entity. Hirschman and Chinchor (1997) proposed to annotate nominal expressions, that is nouns, noun phrases, and pronouns, here considered as referring expressions, and their antecedents, with the XML-defined `COREF` element. `COREF` has five possible attributes: `ID`, `REF`, `TYPE`, `MIN`, and `STAT`.

`ID` is an arbitrary integer that assigns a unique number to each nominal expression of the text. `REF` is an optional integer that links a nominal expression to a coreferring antecedent. `REF` value is then the `ID` of its antecedent. From Hirschman and Chinchor's annotated examples, the text

```
<COREF ID="100">Lawson Mardon Group Ltd.</COREF> said <COREF
ID="101" TYPE="IDENT" REF="100">it</COREF>
```

indicates that *Lawson Mardon Group Ltd.* and *it* are assigned respectively with `ID` `100` and `101`, and that *it* refers to the same entity as *Lawson Mardon Group Ltd* through `REF="100"`.

In the MUC competitions, coreference is defined as symmetric and transitive, that is, if A is coreferential with B, the reverse is also true. And if A is coreferential with B, and B is coreferential with C, then A is coreferential with C. Such a coreference set then forms an equivalence class called a **coreference chain**. This is stated with the `TYPE` attribute that specifies the link between the anaphor and its antecedent: `"IDENT"` is the only possible value of the attribute, and it indicates that coreferences are identical. One may imagine other types of coreference such as part, subset, etc.

Other attributes are `MIN` and `STAT`. Some denominations may have a variable length and yet refer to the same entity, such as *Queen Elisabeth of England* and

Queen Elisabeth. In a text where the denomination appears in full, a coreference analyzer could bracket both. The `COREF` tag `MIN` indicates the minimum valid string. From Hirschman and Chinchor's guidelines,

> `<COREF ID="100" MIN="Haden MacLellan PLC">`Haden MacLellan PLC of Surrey, England`</COREF>` ... `<COREF ID="101" TYPE= "IDENT" REF="100">`Haden MacLellan`</COREF>`

indicates that *Haden MacLellan PLC of Surrey, England* and *Haden MacLellan PLC* are both valid bracketing.

Finally, `STAT` ("status") means that the annotation is optional. It is used when coreference is tricky or doubtful. The only value for this attribute is `OPT` ("optional"). From Hirschman and Chinchor's guidelines,

> `<COREF ID="102" MIN="Board of Education">`Our Board of Education`</COREF>` budget is just too high, the Mayor said. `<COREF ID="103" STAT="OPT" TYPE="IDENT" REF="102">` Livingston Street `</COREF>` has lost control.

indicates that *Board of Education* and *Livingston Street* refers to the same entity, but that it can bewilder the reader and the annotation is left optional.

14.5 References: A More Formal View

14.5.1 Generating Discourse Entities: The Existential Quantifier

In Chap. 12, we introduced a logical notation to represent nominal expressions that differs from that of the previous section. If we take the formal semantics viewpoint, a sentence such as:

A patron ordered a meal.

exposes two new terms: *a patron* and *a meal*. These entities are tied to indefinite noun phrases and hence to logical forms headed by the existential quantifier $\exists$:

$$\exists x, patron(x)$$
$$\exists y, meal(y)$$

A discourse interpretation program should reflect them in a Prolog database and augment the database with the corresponding semantic facts:

```
patron(patron#3).
meal(meal#15).
```

We generate the entities by creating new constants – new atoms – making sure that they have a unique name, here `patron#3` or `meal#15`. Then, we can add them in the database under the form of facts using the `asserta/1` built-in predicate.

New entities are only a part of the whole logical set because the complete semantic representation of the sentence is:

```
a(X, patron(X), a(Y, meal(Y), ordered(X, Y)))
```

To be consistent with this representation, we must also add the predicate `ordered(Subject, Object)` to link the two new entities. We carry this out by asserting a last fact:

```
ordered(patron#3, meal#15).
```

14.5.2 Retrieving Discourse Entities: Definite Descriptions

While indefinite noun phrases introduce new entities, definite ones usually refer to entities created previously. A possible subsequent sentence in the discourse could be:

The patron ate the meal,

which should not create new entities. This simply declares that the patron already mentioned ate the meal he ordered. Such definite noun phrases are then anaphors.

The logic interpretation of definite descriptions usually translates as:

$$\exists!x, patron(x)$$
$$\exists!y, meal(y)$$

where properties are quantified with $\exists!$ meaning that x and y are unique. To reflect this in the Prolog database, we could identify x and y among the entities previously created and then assert the new fact:

```
ate(patron#3, meal#15).
```

An alternate processing of the `ate/2` relation – and probably a more alert one – is to first create new atoms, that is, new names:

```
patron(patron#5).
meal(meal#17).
```

to link them with `ate/2`:

```
ate(patron#5, meal#17).
```

and to assert later that some names are identical:

```
equals(patron#3, patron#5).
equals(meal#15, meal#17).
```

This method is precisely the coreference recognition that we described previously. Besides, proceeding in two steps enables a division of work. While a first task generates all potential entities, a second one resolves coreferences using techniques that we will review in Sect. 14.7.

14.5.3 Generating Discourse Entities: The Universal Quantifier

We saw that determiners can also correspond to the universal quantifier $\forall$. An example of such a sentence is:

Every patron ordered a meal.

Its corresponding logic representation is:

$$\forall x, patron(x) \Rightarrow \exists y, meal(y), ordered(x, y)$$

or in a predicate form:

```
all(X, patron(X), a(Y, meal(Y), ordered(X, Y)))
```

In such a logical form, each value of `X` should be mapped onto a specific value of `Y`: each patron has eaten his/her own and unique meal. A definition in extension of this sentence – that is, the list of all the facts it encompasses – could be:

Pierre ordered a cassoulet,
Charlotte ordered a pytt i panna, and
Dave ordered a Yorkshire pudding.

Doing so, we have defined a function linking each value of `X` with a unique value of `Y`, that is, *Pierre* with a specific *cassoulet*, *Charlotte* with a *pytt i panna*, and *Dave* with a *Yorkshire pudding*. In logic, this is called a Skolem function (Table 14.4).

Table 14.4. A Skolem function.

X	Y	Skolem function values
pierre	cassoulet#2	f(pierre) = cassoulet#2
charlotte	pytt_i_panna#4	f(charlotte) = pytt_i_panna#4
dave	yorkshire_pudding#4	f(dave) = yorkshire_pudding#4

Our Skolem function has eliminated variable y and the existential quantifier. It has replaced them by $f(x)$ in the logical form:

$$\forall x, patron(x) \Rightarrow ordered(x, f(x))$$

or

```
all(X, patron(X), a(f(X), meal(f(X)),
           ordered(X, f(X))))
```

More generally, Skolemization handles logical formulas with universally quantified variables, $x_1, x_2, ..., x_n$, and a variable existentially quantified y on its left-hand side:

$$\forall x_1, \forall x_2, ..., \forall x_n, \exists y, pred(x_1, x_2, ..., x_n, y)$$

It substitutes y by a function of the universally quantified variables:

$$y = f(x_1, x_2, ..., x_n)$$

yielding unique values for each n-tuplet $(x_1, x_2, ..., x_n)$.

Skolemization results in a new formula, where variable y has disappeared:

$$\forall x_1, \forall x_2, ..., \forall x_n, pred(x_1, x_2, ..., x_n, f(x_1, x_2, ..., x_n))$$

and where $f(x_1, x_2, ..., x_n)$ is called a Skolem function.

14.6 Centering: A Theory on Discourse Structure

Of the many theories on discourse structure, Grosz and Sidner's (1986) has been very influential in the computational linguistics community. Grosz and Sidner modeled a discourse as being a composite of three components:

- the linguistic structure of the actual sequence of utterances in the discourse
- a structure of intentions
- an attentional state

Grosz and Sidner's first assumption is that the linguistic structure of a discourse is made of segments. They substantiated this claim using psychological studies showing a relative agreement among individuals over the segmentation a text: given a text, individuals tend to fractionate it in a same way. Segments have a nonstrict embedded (hierarchical) organization (Fig. 14.2). It is roughly comparable to that of the phrase structure decomposition of a sentence. Segment boundaries are often delimited by clues and **cue phrases**, also called markers, that indicate transitions.

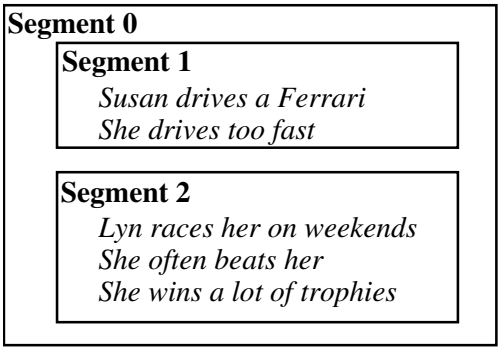


Fig. 14.2. The embedded structure of discourse. Segment 0 covers the five sentences and spans segment 1 (1 and 2) and segment 2 (3–5).

The intentional structure is what underlies a discourse. It is the key to how segments are arranged and their internal coherence. It has global and local components.

From a global viewpoint, intention relates to the **discourse purpose**, which is the main objective of the discourse and why it takes place. Within each segment there is a **discourse segment purpose** that is local and that contributes to the main purpose. Discourse segment purposes are often easier to determine than the overall discourse intention.

The attentional state is the dynamic set of objects, relations, and properties along with the discourse. The attentional state is closely related to segments. For each of them there is a focus space made of salient entities, properties of entities, and relations between entities, that is, predicates describing or linking the entities. The attentional state also contains the discourse segment purpose.

While Grosz and Sidner's general model may prove difficult to implement, Grosz et al. (1995) derived a simpler concept of **centering** from it. Centering retains the idea of segment, defined as a set of utterances, along which a limited number of dynamic centers turn up. Centers are the "useful" entities of an utterance that link it to other utterances of a segment. Since centers are a subset of entities, they are easier to detect than the intention or the whole attentional state. They provide a tentative model to explain discourse coherence and coreference organization.

Centers of an utterance are split into a set of **forward-looking centers** and a unique **backward-looking center**, except for the first utterance of the segment, which has no backward-looking center:

- The backward-looking center, or simply the center, is the entity that connects the current utterance with the previous one and hence with one of the previous forward-looking centers. It is often a pronoun.
- Forward-looking centers are roughly the other discourse entities of a segment. More precisely, they are limited to entities serving to link the utterance to other utterances.

Forward-looking centers can be ordered according to syntactic, semantic, and pragmatic factors, and the first one has great chances to become the backward-looking center of the next utterance. As examples, centers in Table 14.1 are:

- In sentence 1, *Susan* and *Ferrari* are the discourse entities and forward-looking centers.
- In sentence 2, *she* is the backward-looking center because it connects the utterance with the previous one.
- In sentence 3, *Lyn* and *weekends* are the forward-looking centers; *her* is the backward-looking center.

14.7 Solving Coreferences

Although coreferences to a same object are frequently ambiguous, they generally raise no understanding problem to a human reader, with the exception of poorly written texts. However, they represent a tricky issue for a machine. The field has long been dominated by complex linguistic theories that are difficult to implement and to

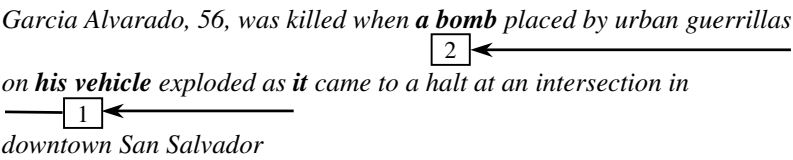
process. Fortunately, as with partial parsing, the MUCs have focused research on concrete problems and robust algorithms that revolutionized coreference resolution.

In the next sections, we will describe algorithms to automatically resolve coreferences. We will first introduce systems based on manually written rules and then describe an efficient machine-learning approach. Even if coreference algorithms do not reach the performance of POS taggers or noun group detectors, they have greatly improved recently and can now be applied to unrestricted texts.

14.7.1 A Simplistic Method: Using Syntactic and Semantic Compatibility

A basic rule that links an anaphor and its antecedent is that their number and gender are identical. This yields the idea of a simplistic method to resolve anaphoric pronouns. The algorithm first collects a list of all the discourse's referents. When an anaphor occurs, the antecedent is searched backward in this list. We set aside cataphoras here. The resolution retains the first antecedent it finds in the list – the most recent one – that agrees in gender and number.

This method may seem naïve, but in fact, most of the time the first antecedent occurring in the sentence or in the previous one with matching gender and number is the good one. This **recency** principle has been observed in many experimental studies. The methods ranks properly potential antecedents of *it* in the sentence:



We can extend this resolution method to find antecedents of definite noun phrases. The recency principle remains the same, but in addition to syntactic features such as gender and number, we add semantic constraints. We search the antecedent of a definite noun phrase, considered as an anaphor, among the entities semantically compatible. Compatibility takes the form of:

- the identity – identical noun groups indicate a same reference
- a direct ontological link between groups – generalization or specialization as in *a car* and *the vehicle*, or
- compatible modifiers – adjectives and complements as in *car*, *white car* or *police car*, but not in *police car* and *ambulance*

Huls et al. (1995) report that such a method identifies pronoun anaphor coreferences with an accuracy of 95%. Although this figure would probably degrade in some cases, it proves the power and effectiveness of this very simple model. The existence of gender for nouns in French and in German makes the search probably more accurate in these languages.

14.7.2 Solving Coreferences with Shallow Grammatical Information

Kameyama (1997) proposed an algorithm using manually written rules that produced good results in the MUC contest for the coreference resolution task. Here is a slightly modified version of his algorithm. It operates on pronouns and definite noun groups only. It sets aside others such as indefinite and possessive noun groups.

The algorithm first extracts all nominal expressions of the text. Then, it scans these expressions in left-to-right order, and for each pronoun or definite noun group, it collects preceding nominal expressions – the potential antecedents – within a definite span of a couple of sentences. The exact window size depends on the type of referring expression:

- The entire MUC text preceding the current expression for proper names.
- Narrower for definite noun phrases. Kameyama suggests 10 sentences.
- Even narrower for pronouns. Again, Kameyama suggests 3 sentences.
- The current sentence for reflexive pronouns.

The algorithm applies constraints on the collected nominal expressions to check the compatibility between the current entity E and possible antecedents:

- Number and gender consistency: both must coincide. In some cases, such as with organizations, plural pronouns may denote a singular antecedent.
- Ontological consistency: type of E must be equal to the type of the antecedent or subsume it. For instance, *the automaker* is a valid antecedent of *the company*, but not the reverse.
- Modifier consistency: modifiers such as adjectives must not contradict such as in *the British company* and *the French company*.

Then, among possible candidates, the algorithm retains the one whose **salience** is the highest. This salience is based on the prominence of certain elements in a sentence, such as subjects over objects, and on obliteration with time (or recency). It has its origin in a rough model of human memory. Memory tends to privilege recent facts or some rhetoric or syntactic forms. A linear ordering of candidates approximates salience in English because subjects have a relatively rigid location in front of the sentence. Kameyama's salience ranks candidates from:

1. the preceding part of the same sentence in left–right order (subject salience)
2. the immediately preceding sentence in left–right order (subject salience)
3. other preceding sentences within the window in right–left order (recency)

In addition, the algorithm improves the name recognition with aliases. Companies are often designated by full names, partial names, and acronyms to avoid repetitions. For example, consider *Digital Equipment Corporation*, *Digital*, *DEC*. An improvement to coreference recognition is to identify full names with substrings of them and their acronyms.

14.7.3 Salience in a Multimodal Context

EDWARD (Huls et al. 1995) is a model that extends salience to a gesture designation of entities. EDWARD is part of a system that is intended to control a graphical user interface made of windows containing icons that represent files. The interface accepts natural language and mouse commands to designate objects, that is, to name them and to point at them. This combination of modes of interaction is called **multimodality**.

The multimodal salience model keeps the idea of recency in language. The subject of the sentence is also supposed to be retained better than its object, and an object better than an adjunct. In addition, the model integrates a graphical salience and a possible interaction. It takes into account the visibility of entities and pointing gestures. Syntactic properties of an entity are called linguistic context factors, and visual ones are called perceptual context factors. All factors: subject, object, visibility, interaction, and so on, are given a numerical value. A pointed object has the highest possible mark.

The model uses a time sliding window that spans a sentence. It creates the discourse entities of the current window and assigns them a weight corresponding to their contextual importance. Computation of an entity's weight simply sums up all the factors attached to it. An entity salience is then mapped onto a number: its weight. Then the window is moved to the next sentence, and each factor weight attached to each entity is decremented by one. An entity mentioned for the first time and in the position of an object has a context factor weight – a salience – of 3. The next sentence, its worth will be 2, then 1, and finally 0 (Table 14.5).

The model sequentially processes the noun phrases of a sentence. To determine coreferring expressions of the current noun phrase, the model selects all entities semantically compatible with it that have been mentioned before. The one that has the highest salience value among them is retained as a coreference. Both salience values are then added: the factor brought by the current phrase and the accumulated salience of its coreference. All entities are assigned a value that is used to interpret the next sentence. Then, the decay algorithm is applied and the window is moved to the next sentence.

Table 14.6 shows a processing example. It indicates the salience values of *Lyn*, *Susan*, and *Ferrari*. In case of ambiguous reference, the system would ask the user to indicate which candidate is the right one.

14.7.4 Using a Machine-Learning Technique to Resolve Coreferences

Algorithms we have seen so far are based on manually engineered rules. This strategy requires a good deal of expertise and considerable clerical work to test and debug the rules. In this section, we introduce a machine learning approach where the coreference solver uses rules obtained automatically from a hand-annotated corpus (Soon et al. 2001).

The coreference solver is a decision tree. It considers pairs of noun phrases (NP_i, NP_j), where each pair is represented by a feature vector of 12 parameters.

Table 14.5. Context factors (simplified) according to Huls et al. (1995). Note that a subject appears twice in the context factor list, as a subject and as a major constituent.

Context factors (CF)	Objects in Scope	Successive weights
Linguistic CFs		
Major-constituent referents CF	Referents of subject, (in)direct object, and modifier	[3, 2, 1, 0]
Subject referent CF	Referent of the subject phrase	[2, 1, 0]
Nested-term referent CF	Referents of the noun phrase modifiers (e.g., prepositional phrase, relative clause)	[1, 0]
Perceptual CFs		
Visible referent CF	Referents visible in the current viewpoint. Typically icons visible in a window	[1, . . . , 1, 0]
Selected referent CF	Referents selected in the model world. Typically icons selected – highlighted – with the mouse or by a natural language command	[2, . . . , 2, 0]
Indicated referent CF	Referents indicated by a pointing gesture. Typically an icon currently being pointed at with a mouse	[30, 1, 0]

Table 14.6. Computation of the salience value (SV) of *Lyn*, *Susan*, and *Ferrari*.

	SV of Susan	SV of Lyn	SV of Ferrari
Initial values	0	0	0
Susan drives a Ferrari	$3+2=5$ major + subject	0 major	3
Decay after completion	$3-1+2-1=3$		$3-1=2$
She drives too fast	$3+3+2=8$ existing + major + subject	0	2
Decay after completion	$3-1-1+2-1-1+3-1+2-1=4$		$3-1-1=1$
Lyn races her on weekends	$4+3=7$ existing + major	$3+2=5$ major + subject	1
Decay after completion	$3-1-1-1+3-1-1+2-1-1+3-1=3$	$3-1+2-1=3$	$3-1-1-1=0$
She often beats her	$3+3+2=8$ existing + major + subject	$3+3=6$ existing + major	0

The solver first extracts pairs of noun phrases and computes feature vectors for each pair. It then takes the set of NP pairs as input and decides for each pair whether it corefers or not. Using the transitivity property, it identifies all the coreference chains in the text.

The ID3 learning algorithm (Quinlan 1986) automatically induces the decision tree from annotated texts using the MUC annotation standard (Sect. 14.4.3).

Noun Phrase Extraction. The engine first identifies all the noun phrases – the coreference candidates – from a text using a pipeline of language processing modules. The pipeline is similar to what we have seen in information extraction: tokenization, morphological processing, POS tagging, noun phrase identification, named entity recognition, nested noun phrase extraction, and semantic class determination (Fig. 14.3).

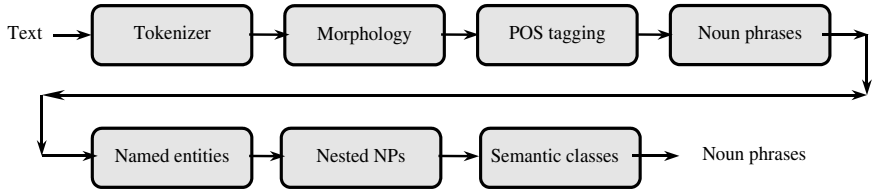


Fig. 14.3. A cascade of NL modules.

The four first modules are generic to many language processing applications. The named entities module follows the MUC style and extracts organization, person, location, date, time, money, and percent entities. When a noun phrase and a named entity overlap, they are merged to form a single noun phrase. The Nested NPs module splits some noun phrases and is more specific to coreference resolution:

1. It brackets possessive noun phrases and possessive pronouns, as in *his long-term strategy*, to form two phrases, *his* and *his long-term strategy.*
2. It also brackets modifier nouns in nominal compounds, as in *wage reductions*, to generate two noun phrases, *wage* and *wage reduction*.

Features. As input, the coreference engine takes a pair of extracted noun phrases (NP_i, NP_j), where NP_i is before NP_j in the text. The engine considers NP_i as a potential antecedent and NP_j as an anaphor and classifies the pair as positive if both NPs corefer, or negative if they do not. Each pair is described by a feature vector of 12 parameters that correspond to positional, grammatical, semantic, and lexical properties:

- Positional feature:
  1. Distance (DIST): This feature is the distance between the two noun phrases measured in sentences: 0, 1, 2, 3, ... The distance is 0 when the noun phrases are in the same sentence.

- Grammatical features:
  2. i-Pronoun (I_PRONOUN): Is NP_i a pronoun i.e. personal, reflexive, or possessive pronoun? Possible values are true or false.
  3. j-Pronoun (J_PRONOUN): Is NP_j a pronoun? Possible values are true or false.
  4. Definite noun phrase (DEF_NP): Is NP_j a definite noun phrase, i.e., that starts with *the*? Possible values are true or false.
  5. Demonstrative noun phrase (DEM_NP): Is NP_j a demonstrative noun phrase, i.e., that starts with *this, that, these, those*? Possible values are true or false.
  6. Number agreement (NUMBER): Do NP_i and NP_j agree in number? Possible values are true or false.
  7. Gender agreement (GENDER): Do NP_i and NP_j agree in gender? Possible values are true, false, or unknown.
  8. Both proper nouns (PROPER_NOUN): Are NP_i and NP_j both proper nouns? Proper nouns are determined using capitalization. Possible values are true or false.
  9. Appositive (APPOSITIVE): Is NP_j an apposition to NP_i, as *the chairman of Microsoft* in *Bill Gates, the chairman of Microsoft*, ...
- Semantic features:
  10. Semantic class agreement (SEMCLASS): Do NP_i and NP_j have the same semantic class? Possible values are true, false, or unknown. Classes are organized as a small ontology with two main parts, person and object, themselves divided respectively into male and female, and organization, location, date, time, money, and percent. The head nouns of the NPs are linked to this ontology using the WordNet hierarchy.
  11. Alias (ALIAS): Are NP_i and NP_j aliases, for instance, *IBM* and *International Business Machines*? Possible values are true or false.
- Lexical feature:
  12. String match (STR_MATCH): Are NP_i and NP_j equal after removing articles and demonstratives from both noun phrases? Possible values are true or false.

Figure 14.3 shows an example of feature vector for the pair *Frank Newman* and *vice chairman* excerpted from the next sentence (Soon et al. 2001):

> Separately, Clinton transition official said that *Frank Newman*, 50, *vice chairman* and chief financial officer of BankAmerica Corp., is expected to be nominated as assistant Treasury secretary for domestic finance.

Training Examples. The classifier is a decision tree. It is trained from positive and negative examples extracted from the annotated corpus using the ID3 algorithm:

- The training procedure generates the positive examples using pairs of adjacent coreferring noun phrases. If NP_{a1} – NP_{a2} – NP_{a3} – NP_{a4} is a coreference chain in a text, the positive examples correspond to pairs: (NP_{a1}, NP_{a2}),

Table 14.7. Feature vector of the noun phrase pair: NP_i = *Frank Newman* and NP_j = *vice chairman*. After Soon et al. (2001).

Feature type	Feature	Value	Comments
Positional	DIST	0	NP_i and NP_j are the same sentence
Grammatical	I_PRONOUN	–	NP_i is not a pronoun
	J_PRONOUN	–	NP_j is not a pronoun
	DEF_NP	–	NP_j is not a definite NP
	DEM_NP	–	NP_j is not a demonstrative NP
	NUMBER	+	NP_i and NP_j are both singular
	GENDER	1	NP_i and NP_j are both males (false = 0, true = 1, unknown = 2)
	PROPER_NOUN	–	Only NP_i is a proper noun
	APPOSITIVE	+	NP_j is not an apposition to NP_i
Semantic	SEMCLASS	1	NP_i and NP_j are both persons (false = 0, true = 1, unknown = 2)
	ALIAS	–	NP_j is not an alias of NP_i
Lexical	STR_MATCH	–	NP_i and NP_j do not match

(NP_{a2}, NP_{a3}), (NP_{a3}, NP_{a4}), where the first noun phrase is always considered to be the antecedent and the second one the anaphor.

- To create the negative examples, the training procedure considers the same adjacent pairs antecedent, anaphor (NP_i, NP_j), and the noun phrases intervening between them $NP_{i+1}, NP_{i+2}, ..., NP_{j-1}$. For each positive pair (NP_i, NP_j), the training procedure generates negative pairs, which consist of one intervening NP and the anaphor NP_j: (NP_{i+1}, NP_j), (NP_{i+2}, NP_j), ..., and (NP_{j-1}, NP_j). The intervening noun phrases can either be part of another coreference chain or not.

Extracting the Coreference Chains. Once the classifier has been trained, it is applied to the noun phrases in a text to identify the coreference chains. The engine first extracts all the noun phrases in the text. It traverses the text from left to right from the second noun phrase. For each current NP_j, the algorithm considers every NP_i before it as a possible antecedent. It then proceeds from right to left and submits the pairs (NP_i, NP_j) to the classifier until it reaches an antecedent or the start of the text.

The algorithm is as follows:

1. Let $NP_1, NP_2, ..., NP_N$ be the noun phrases.
2. For $j = 2$ to N.
   a) For each NP_j, generate all the pairs (NP_i, NP_j), where $i < j$.
   b) Compute the feature vector of each pair (NP_i, NP_j).
   c) For $i = j - 1$ to 1, submit the pair (NP_i, NP_j) to the classifier until a positive pair is found or the beginning of the text is reached.
   d) If a noun phrase returns positive, NP_j has an antecedent and is part of the corresponding coreference chain.

14.7.5 More Complex Phenomena: Ellipses

An **ellipsis** is the absence of certain words or phrases normally necessary to build a sentence. Ellipses occur frequently in the discourse to avoid tedious repetitions. For instance, the sequence:

I want to have information on caterpillars. And also on hedgehogs.

features a second sentence whose subject and verb are missing. The complete sentence would be:

I want to have information on hedgehogs.

Here the speaker avoids saying twice the same thing. Ellipses also occur with clauses linked by conjunctions where a phrase or a word is omitted as in the sentence:

I saw a hedgehog walking on the grass and another sleeping,

Everyone, however, can understand that it substitutes the complete sentence:

I saw a hedgehog walking on the grass and I saw another hedgehog sleeping.

Ellipses are rather difficult to handle. In many cases, however, maintaining a history of all the discourse's referents can help retrieve an omitted referent or verb. A referent missing in a sentence can be searched backward in the history and replaced with an adequate previous one.

14.8 Discourse and Rhetoric

Rhetoric also offers means to explain discourse coherence. Although rhetoric has a very long tradition dating from ancient times, modern linguists have tended to neglect it, favoring other models or methods. Recently however, interest has again increased. Modern rhetorical studies offer new grounds to describe and explain argumentation. Modeling argumentation complements parts of human discourse that cannot only be explained in terms of formal logic or arbitrary beliefs. The *Traité de l'argumentation* by Perelman and Olbrechts-Tyteca (1976) is a prominent example of this trend.

On a parallel road, computational linguistics also rediscovered rhetoric. Most of the renaissance in this community is due to influential papers on rhetorical structure theory (RST) by Mann and Thompson (1987, 1988). This section provides a short introduction to ancient rhetoric and then describes RST.

14.8.1 Ancient Rhetoric: An Outline

Rhetoric was studied in most schools of ancient Greece and Rome, and in universities in the Middle Ages. Rhetoric was then viewed as a way to define how best to compose ideas in a discourse, to make it attractive, to convince and persuade an audience. It was considered as a kind of discourse strategy defining the optimal arrangement or planning of arguments according to the type of audience, of speech case, etc.

According to the ancient rhetoric school, the production of discourse had to be organized around five canons – invention, arrangement, style, memory, and delivery.

- Invention (*inventio*) is related to the ideas or facts contained in a discourse: what to say or to write are the first things to identify to make a discourse exist. According to ancient Greeks, a key to invention was to answer the right questions in the right order.
- Arrangement (*dispositio*) is the discourse construction for which general patterns have been proposed. According to Cicero, a discourse should feature an introduction (*exordium*), a narrative (*narratio*) where the orator sets forth the issues of the problem, a proposition (*propositio*) where s/he states her/his arguments for the case, a refutation (*refutatio*), where s/he gives counterarguments, a confirmation (*confirmatio*) where s/he reinforces her/his arguments, and finally a conclusion (*peroratio*).
- Style (*elocutio*) concerns the transcription and the edition of ideas into words and sentences. Rules of style suggested to privilege clarity – use plain words and conform to a correct grammar. This was a guarantee to be understood by everybody. Style was also a literary art where efficiency mattered most. It was divided into three categories whose goals were to emote (*movere*), to explain (*docere*), or to please (*delectare*) according to the desired effect on the audience.
- Memory (*memoria*) was essential that the orator should retain what s/he had to say. The Ancients advised orators to sleep well, to be in good shape, to exercise memory by learning by heart, and to use images.
- Delivery (*actio*) concerned the uttering of the discourse: voice, tone, speed, and gestures.

Although current discourse strategies may not be the same as those designed and contrived in Athens or Sicily 2500 years ago, if elucidated they give keys to a discourse structure. Later, the historical definition of rhetoric has been sometimes superseded by a pejorative sense meaning empty political speeches or ranting.

14.8.2 Rhetorical Structure Theory

Rhetorical structure theory (RST) is a theory of text organization in terms of relations that occur in a text. As for Grosz and Sidner, RST identifies a hierarchical tree structure in texts. A text consists of nonoverlapping segments that define the tree nodes.

These segments are termed by Mann and Thompson as "text spans." They correspond typically to one or more clauses. Text spans may be terminal or nonterminal nodes that are linked in the tree by relations.

Rhetorical relations are sorts of dependencies between two text spans termed the **nucleus** and the **satellite**, where the satellite brings some sort of support or explanation to the nucleus, which is the prominent issue. To illustrate this concept, let us take the example of the *Justify* relation from Mann and Thompson (1987, pp. 9–11): "A justify satellite is intended to increase the reader's readiness to accept the writer's right to present the nuclear material." In the short text:

1. *The next music day is scheduled for July 21 (Saturday), noon–midnight*
2. *I'll post more details later,*
3. *but this is good time to reserve the place on your calendar.*

segments 2 and 3 justify segment 1, and they can be represented graphically by Fig. 14.4.

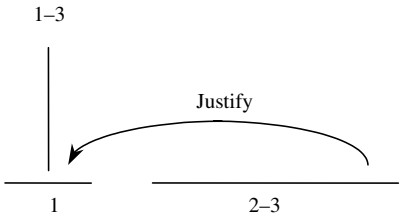


Fig. 14.4. The Justify relation.

Segments can then be further subdivided using other relations, in the example a *Concession* (Fig. 14.5).

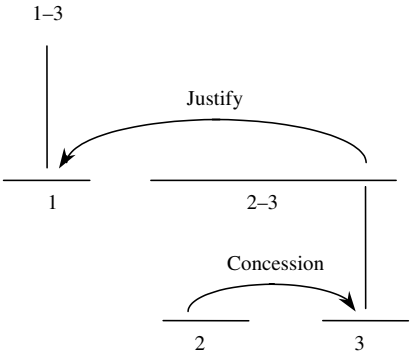


Fig. 14.5. More relations: Concession.

Relations are easy to represent in Prolog with facts `rhetorical_relation(relation_type, satellite, nucleus)`:

```
rhetorical_relation(justify, 3, 1).
rhetorical_relation(concession, 2, 3).
```

Another example is given by this funny text about dioxin (Mann and Thompson, 1987, pp. 13–15):

1. *Concern that this material is harmful to health or the environment may be misplaced.*
2. *Although it is toxic to certain animals,*
3. *evidence is lacking that it has any serious long-term effect on human beings.*

which can be analyzed with relations *Elaboration* and *Concession* in Fig. 14.6.

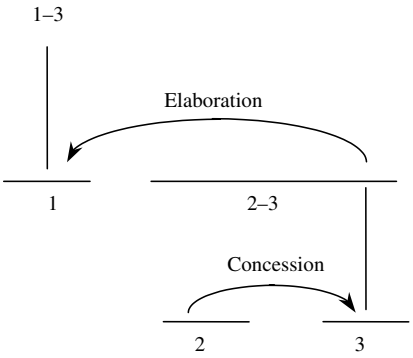


Fig. 14.6. Elaboration and Concession.

These relations are equivalent to the Prolog facts:

```
rhetorical_relation(elaboration, 3, 1).
rhetorical_relation(concession, 2, 3).
```

14.8.3 Types of Relations

The total number and the type of rhetorical relation vary much among authors and even among papers written by their creators. Their number ranges from a dozen to several hundreds. As we saw in the previous section, most relations link a nucleus and a satellite. Figure 14.7 shows a slightly simplified list of them from Mann and Thompson (1987). In some instances, relations also link two nuclei. They are shown in Fig. 14.8.

Circumstance	Evidence	Otherwise
Solutionhood	Justify	Interpretation
Elaboration	Cause	Evaluation
Background	Antithesis	Restatement
Enablement	Concession	Summary
Motivation	Condition	

Fig. 14.7. RST rhetorical relations linking a nucleus and a satellite.

Sequence Joint Contrast

Fig. 14.8. Relations linking two nuclei.

14.8.4 Implementing Rhetorical Structure Theory

Mann and Thompson gave formal definitions of rhetorical relations using constraints on the satellite, the nucleus, and both. Table 14.8 shows constraints holding for *evidence*. In addition, a rhetorical relation entails consequences that are described by an effect: here, with *evidence*, the reader's belief of the nucleus is increased.

Table 14.8. The EVIDENCE relation. After Mann and Thompson (1987).

Relation name	EVIDENCE
Constraints on the nucleus N	The reader R might not believe to a degree satisfactory to the writer W
Constraints on the satellite S	The reader believes S or will find it credible
Constraints on the $N + S$ **combination**	R's comprehending S increases R's belief of N
The effect	R's belief of N is increased
Locus of the effect	N

Such constraints are difficult – if not impossible – to implement in a computer as is because they involve knowing the thoughts of the reader and the writer. However, rhetorical relations are often indicated by a handful of specific cue words or phrases. Mann and Thompson observe that a concession is often introduced by *although*, as in the dioxin text from the previous section, or *but*. A common workaround to detect a relation is then to analyze the surface structure made of these cue phrases. They may indicate the discourse transitions, segment boundaries, and the type of relations. Many cue phrases are conjunctions, adverbial forms, or syntactic patterns (Table 14.9).

Mann and Thompson also observed that the nucleus and the satellite had typical topological orders (Table 14.10).

Recently, comprehensive works have itemized cue phrases and other constraints enabling the rhetorical parsing of a text. Marcu (1997) and Corston-Oliver (1998)

Table 14.9. Examples of cue phrases and forms.

Cues	English	French	German
Conjunctions	*Because, in fact, but, and*	*Car, en effet, puisque, et, mais*	*denn, und, aber,*
Adverbial forms	*In addition, for example*	*De plus, en particulier, particulièrement, par exemple*	*dazu, besonders, zum Beispiel*
Syntactic forms	Past participles: *given*	Present participles: *étant donné*	

Table 14.10. Typical orders for some relations.

Satellite before nucleus		
	Antithesis	Condition
	Background	Justify
	Concession	Solutionhood
Nucleus before satellite		
	Elaboration	Evidence
	Enablement	Statement

are notable examples of this trend. As an example, Corston-Oliver (1998) recognizes the *Elaboration* relation with a set of necessary criteria that must hold between two clauses, clause 1 being the nucleus and clause 2 the satellite:

1. Clause 1 precedes clause 2.
2. Clause 1 is not subordinate to clause 2.
3. Clause 2 is not subordinate to clause 1.

and cues that are ranked according to an heuristic score (Table 14.11).

Corston-Oliver (1998) applied these cues to analyze the Microsoft *Encarta* encyclopedia. With the excerpt:

1. *A stem is a portion of a plant.*
2. *Subterranean stems include the rhizomes of the iris and the runners of the strawberry;*
3. *The potato is a portion of an underground stem.*

using cue H41, he could obtain the rhetoric structure shown in Fig. 14.9.

14.9 Events and Time

In most discourses, actions, events, or situations have a **temporal** context. This context is crucial to the correct representation of actions. It involves time, which is reflected by time expressions, such as adverbs or adjuncts, *now*, *tomorrow*, *in 5 minutes*, and verb **tenses**, such as present, past, or future.

Table 14.11. Cues to recognize the *Elaboration* relation. After Corston-Oliver (1998, p. 129).

Cue	Score	Cue Name
Clause 1 is the main clause of a sentence (sentence i), and clause 2 is the main clause of a sentence (sentence j), and sentence i immediately precedes sentence j, and (a) clause 2 contains an elaboration conjunction (*also, for example*), or (b) clause 2 is in a coordinate structure whose parent contains an elaboration conjunction.	35	H24
Cue H24 applies, and clause 1 is the main clause of the first sentence in the excerpt.	15	H26
Clause 2 contains a predicate nominal whose head is in the set {*portion, component, member, type, kind, example, instance*}, or clause 2 contains a predicate whose head verb is in the set {*include, consist*}	35	H41
Clauses 1 and 2 are not coordinated, and (a) clauses 1 and 2 exhibit subject continuity, or (b) clause 1 is passive and the head of the direct object of clause 1 and the head of the direct object of clause 2 have the same base form, or (c) clause 2 contains an elaboration conjunction.	10	H25
Cue H25 applies, and clause 2 contains a habitual adverb (*sometimes, usually*, ...).	17	H25a
Cue H25 applies, and the syntactic subject of clause 2 is the pronoun *some* or contains the modifier *some*.	10	H38

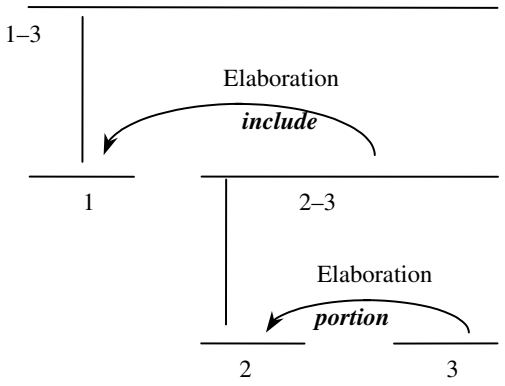


Fig. 14.9. Rhetorical structures.

Understanding temporal relations between events is difficult and may depend on the language. For instance, there is no exact correspondence for past and present tenses between French and English. In the next section, we will provide hints on theories about temporal modeling.

14.9.1 Events

Research on the representation of time, events, and temporal relations dates back to the beginning of logic. It resulted in an impressive number of formulations and models. A possible approach is to **reify** events, that is to turn them into objects, to quantify them existentially, and to connect them to other objects using predicates based on action verbs and their modifiers (Davidson 1966). The sentence *John saw Mary in London on Tuesday* is then translated into the logical form:

$$\exists\varepsilon[saw(\varepsilon, John, Mary) \wedge place(\varepsilon, London) \wedge time(\varepsilon, Tuesday),$$

where ε represents the event.

To represent the temporal context of an action sequence we can use a set of predicates. Consider:

Spring is back. Hedgehogs are waking up. Toads are still sleeping.

There are obviously three actions or events described here. These events are located in time around a reference point defined by the return of spring. From this point, the hedgehogs' waking up process extends onwards while the toads' sleeping process overlaps it (Fig. 14.10). Events have a different duration: the first sentence merely describes a single time point whereas the two last processes are defined inside intervals.

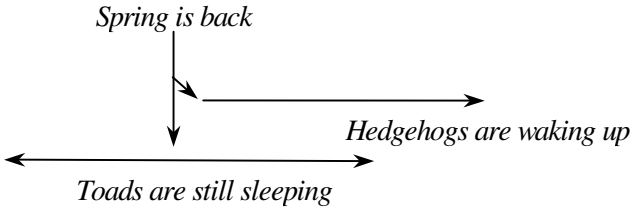


Fig. 14.10. Events.

Let us denote `e1`, `e2`, and `e3` the events in Fig. 14.10, and let us portray them in Prolog. In addition, let us use the agent semantic role that we borrow from the case grammars. We have a first representation:

```
event(e1).
is_back(e1).
agent(e1, spring).
```

```
...
event(e2).
waking_up(e2).
agent(e2, hedgehogs).
...

event(e3)
sleeping(e3).
agent(e3, toads).
```

14.9.2 Event Types

Events are closely related to sentence's main verbs, and different classifications have been proposed to associate a verb with a type of event. Vendler (1967) for English, Gosselin (1996) for French, and others came to a consensus to divide verbs into four categories, denoting:

- A state – a permanent property or a usual situation (e.g., *be*, *have*, *know*, *think*).
- An achievement – a state change, a transition, occurring at single moment (e.g., *find, realize, learn*).
- An activity – a continuous process taking place over a period of time (e.g., *work, read, sleep*). In English, activities often use the present perfect, *-ing*.
- An accomplishment – an activity with a definite endpoint completed by a result (e.g., *write a book, eat an apple*).

Some authors have associated events to verbs only. It is safer, however, to take verb phrases – predicates – and even subjects into account to link events to Vendler's categories (Table 14.12). Compare *The water ran*, which is an activity in the past, and *The hurdlers ran* (in a competition), which depicts an achievement.

14.9.3 Temporal Representation of Events

Let us now try to represent processes in a temporal chronology. In the example in Fig. 14.10, the only process that has a definite location is `e1`. It is associated to a calendar period: *spring*. Other processes are then relative to it. As for these sentences, in most discourses it is impossible to map all processes onto an absolute time. Instead, we will represent them using relative, and sometimes partial, temporal relations.

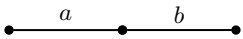
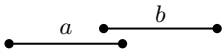
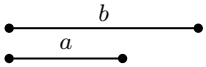
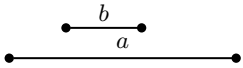
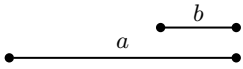
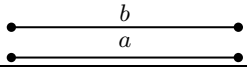
Simplifying things, we will suppose that time has a linear ordering and that each event is located in time: it has a certain beginning and a certain end. This would not be true if we had considered conditional statements. Temporal relations associate processes to time intervals and set links, constraints between them. We will adopt here a model proposed by Allen (1983, 1984), whose 13 relations are listed in Table 14.13.

Using Allen's representation, relations `before(e1, e2)`, `after(e2, e1)`, and `contains(e3, e1)` depict temporal constraints on events `e1`, `e2`, and `e3` in Sect. 14.9.1. Temporal relations result in constraints between all processes that enable a total or partial ordering of them.

Table 14.12. Vendler's verb categories.

	English	**French**	**German**
State	*The cat is sick* *I like chocolate*	*Le chat est malade* *J'aime le chocolat*	*Die Katze ist krank* *Ich esse Schokolade gern*
Activity	*She works for a company* *He is writing a book*	*Elle travaille pour une entreprise* *Il écrit un livre*	*Sie arbeitet für eine Firma* *Er schreibt ein Buch*
Accomplishment	*He wrote a book* *The dormouse ate the pears*	*il a écrit un livre* *Le loir a mangé les poires*	*Er hat ein Buch geschrieben* *Die Haselmaus hat die Birnen gegessen*
Achievement	*The sun set* *I realized I was wrong*	*Le soleil s'est couché* *Je me suis rendu compte que j'avais tort*	*Die Sonne ist untergegangen* *Ich habe eingesehen, ich nicht recht hatte*

Table 14.13. Allen's temporal relations.

#	**Relations**	**Graphical representations**
1.	`before(a, b)`	
2.	`after(b, a)`	
3.	`meets(a, b)`	
4.	`met_by(b, a)`	
5.	`overlaps(a, b)`	
6.	`overlapped_by(b, a)`	
7.	`starts(a, b)`	
8.	`started_by(b, a)`	
9.	`during(b, a)`	
10.	`contains(a, b)`	
11.	`finishes(b, a)`	
12.	`finished_by(a, b)`	
13.	`equals(a, b)`	

14.9.4 Events and Tenses

As we saw, event modeling results in time intervals and in relations between them. From event examples in Fig. 14.10, we can define two new temporal facts:

- instantaneous events, which are punctual and marking a transition
- situations, which have a duration – true over an interval

Relations as well as events or situations are not accessible directly. As for rhetorical relations or segment boundaries, we need cues or markers to track them. In the example above, we have mapped events onto verbs. This hints at detection and description methods. Although there is no definitive solution on how to detect events, many techniques rely on verbs and verb phrases to act as markers.

A first cue to create and locate an event is the verb tense. A sentence sequence defines a linear sequence of enunciation events. A basic distinction is between the moment of the enunciation and the time of the event (or situation). Figure 14.11 represents a kind of ideal time.

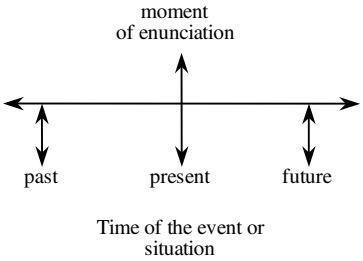


Fig. 14.11. Ideal time: past, present, and future.

The sentence

Ernest the hedgehog ate a caterpillar

creates two events; one corresponds to the processes described the sentence, `e1`, and the other, `e2`, to the time of speech. Both events are linked by the relation `before(e1, e2)`. We could have refined the model with a beginning `e1b` and an end `e1e` of *Ernest*'s dinner. New relations would be:

```
before(e1b, e1e).
before(e1b, e2).
before(e1e, e2).
```

Using a verb classification and tenses helps determine the events location or situation boundaries. We may also rely on time adverbs and time adjuncts such as *for five minutes*, *tomorrow*, etc.

The 'ideal' representation, however, is not sufficient to describe many narrative phenomena where the writer/reader viewpoint is moved relatively to temporal events.

Reichenbach (1947) elaborated a more complex representation to take this viewpoint into account. Basically, verb tenses are mapped onto a triplet representing on a linear scale the point of the event or situation denoted E, the point of speech denoted S, and a point of reference denoted R. The reference corresponds to a sort of writer/reader viewpoint.

Let us first consider the time of speech and the event. It is clear to the reader that an event described by basic tenses, past, present, and future, is respectively before, coinciding, and after the point of speech (Fig. 14.12).

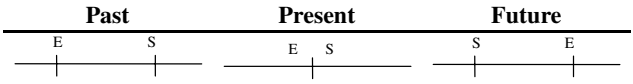


Fig. 14.12. Ideal tenses.

Reichenbach's tense model introduces the third point to position events relatively in the past or in the future. Consider the past sentence

Hedgehogs had already woken up when the sun set.

Two events are described, the hedgehogs' waking up, `ewu`, and the sunset, `ess`. Among the two events, the speaker viewpoint is focused by the clause *Hedgehogs had already woken up*: then, the action takes place. This point where the speaker moves to relate the story is the point of reference of the narrative, and the event is before it (Fig. 14.13). The point of reference of the first process enables us to locate the second one relatively to it and to order them in a sequence.

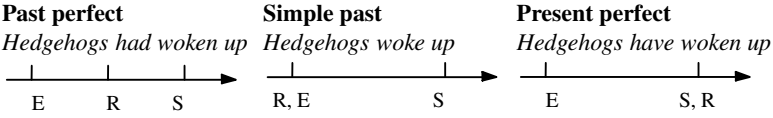


Fig. 14.13. Event, reference, and speech for some English tenses.

Some tenses describe a time stretch of the event, as for the French *imparfait* compared to the *passé composé* (Fig. 14.14), or continuous tenses of English (Fig. 14.15).

14.10 TimeML, an Annotation Scheme for Time and Events

Several schemes have been proposed to annotate temporal information in texts. Many of them were incompatible or incomplete, and in an effort to reconcile and unify the field, Ingria and Pustejovsky (2004) introduced the XML-based Time Markup Language (TimeML). TimeML is a specification language whose goal is to capture most

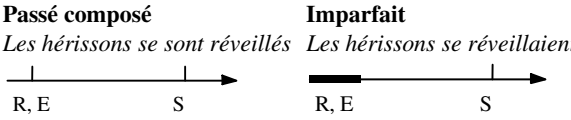


Fig. 14.14. French imparfait and passé composé.

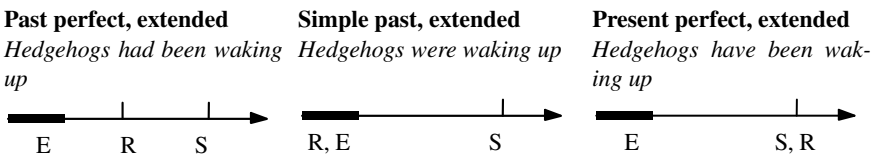


Fig. 14.15. Some English tenses involving a stretch of time.

aspects of temporal relations between events in discourses. It is based on Allen's (1984) relations and inspired by Vendler's (1967) classification of verbs.

TimeML defines the XML elements `TIMEX3` to annotate time expressions (*at four o'clock*), `EVENT` to annotate the events (*he slept*), and "signals". The `SIGNAL` tag marks words or phrases indicating a temporal relation. It includes function words such as *later* and *not* (*he did not sleep*). TimeML also features elements to connect entities using different types of links, most notably temporal links, `TLINK`s, that describe the temporal relation holding between events or between an event and a time.

TimeML elements have attributes. For instance, events have a tense, an aspect, and a class. The seven possible classes denote the type of event, whether it is a `STATE`, an instantaneous event (`OCCURRENCE`), etc.

The sentence

> *All 75 people on board the Aeroflot Airbus died when it ploughed into a Siberian mountain in March 1994*

is marked up as follows (Ingria and Pustejovsky 2004):

```
All 75 people
<EVENT eid="e7" class="STATE">on board</EVENT>
<MAKEINSTANCE eiid="ei7" eventID="e7" tense="NONE"
aspect="NONE"/>
<TLINK eventInstanceID="ei7" relatedToEvent="ei5"
relType="INCLUDES"/>
the Aeroflot Airbus
<EVENT eid="e5" class="OCCURRENCE" >died</EVENT>
<MAKEINSTANCE eiid="ei5" eventID="e5" tense="PAST"
aspect="NONE"/>
<TLINK eventInstanceID="ei5" signalID="s2"
```

```
relatedToEvent="ei6" relType="IAFTER"/>
<SIGNAL sid="s2">when</SIGNAL>
it
<EVENT eid="e6"
class="OCCURRENCE">ploughed</EVENT>
<MAKEINSTANCE eiid="ei6" eventID="e6" tense="PAST"
aspect="NONE"/>
<TLINK eventInstanceID="ei6" signalID="s3"
relatedToTime="t2" relType="IS_INCLUDED"/>
<TLINK eventInstanceID="ei6" relatedToEvent="ei4"
relType="IDENTITY"/>
into a Siberian mountain
<SIGNAL sid="s3">in</SIGNAL>
<TIMEX3 tid="t2" type="DATE" value="1994-04">March
1994</TIMEX3>.
```

In the example, three events `e5`, *died*, `e6`, *ploughed*, and `e7`, *on board*, are annotated and instantiated using the `MAKEINSTANCE` tag. The text contains one time expression, *March 1994*, which is annotated using `TIMEX3`. The events and the time expressions are connected by two temporal links, `TLINK`. The first link specifies that the passengers died after the plane ploughed, using the `relatedToEvent` attribute. The second link specifies that event ploughed is included in March 1994. A third and last `TLINK` refers to an event, `e4`, mentioned in a previous, noncited sentence. The temporal signals *when* and *in* can also be relevant, and they are tagged with a `SIGNAL` tag.

14.11 Further Reading

Schiffrin (1994) and Coulthard (1985) give general introductions to discourse. Ducrot and Schaeffer (1995) and Simone (1998) provide shorter and very readable accounts. Tesnière (1966) is an outstanding description of anaphora (Chap. 42) and anaphors (Chap. 43). Kamp and Reyle (1993) provide a thorough logical model of discourse that they called the discourse representation theory – DRT. Although complex and difficult to implement, it is frequently cited.

The MUCs spurred very pragmatic research on discourse, notably on coreference resolution. They produced a coreference annotation scheme that enabled researchers to evaluate competing algorithms and that became a standard. Research culminated with the design of machine learning strategies. Soon et al. (2001) were first to develop a system offering a performance matching systems with manually written rules. Ng and Cardie (2002) further improved this strategy by extending the parameters from 12 to 38 and produced results better than all other systems.

Corpus Processing for Lexical Acquisition by Boguraev and Pustejovsky (1996) covers many technical aspects of discourse processing. It includes extraction of proper nouns, which has recently developed into an active subject of research.

Introductions to rhetoric include books by Corbett and Connors (1999), Reboul (1994), and Perelman and Olbrechts-Tyteca (1976).

Time processing in texts is still a developing subject. Reichenbach (1947) described a model for all the English tenses. Starting from this foundational work, Gosselin (1996) provided an account for French that he complemented with a process duration. He described rules and an implementation valid for French verbs. Ter Meulen (1995) describes another viewpoint on time modeling in English, while Gagnon and Lapalme (1996) produce an implementation of time processing for French based on the DRT. Johansson et al. (2005) describe how semantic role labeling and time processing can be used to generate animated 3D scenes from written texts.

Exercises

14.1. Choose a newspaper text of about ten lines. Underline the references and link coreferences.

14.2. Write DCG rules to detect noun groups and pronouns of Exercise 14.1 and collect the discourse entities in a Prolog list.

14.3. Write a grammar recognizing names (proper nouns) in English, French, or German. You will consider that a name consists of a title followed by a surname whose first letter is capitalized.

14.4. Choose a newspaper text of about ten lines. Collect noun groups and pronouns in a list with a noun group detector. Write a coreference solver in Prolog to associate each pronoun to all preceding noun groups.

14.5. Using the program written for Exercise 14.4, write a first predicate that retains the first preceding noun group – first potential antecedent – and a second predicate that retains the two first noun groups.

14.6. Implement the Kameyama algorithm of Sect. 14.7.2 in Prolog.

14.7. Select a newspaper article and underline elliptical sentences or phrases.

14.8. Using the result of Exercise 14.7, describe rules that would enable you to resolve ellipses.

14.9. Select one page from a technical text and annotate clauses with rhetorical relations listed in Table 14.7.

14.10. Write rules using the model of Table 14.10 to recognize the EVIDENCE rhetorical relation.

14.11. Describe verb tenses in languages you know in terms of point of the event, of speech, and of reference, as in Sect. 14.9.4.