# Incremental dialogue modeling – internship report

Author : Adèle Mortier
Supervisor : Jonathan Ginzburg
Lab : LLF Paris 7

December 15, 2018

## General Introduction

Dialogue modeling is a subfield of computational linguistics that intends to tackle natural language using a wider perspective. While natural language semantics usually focuses on single sentences, dialogue semantics addresses the problem of long run interactions between several speakers. This crucially requires to define adequate and more sophisticated structures that could keep track of the many aspects of the ongoing dialogue: what has been asserted so far, what questions remain unanswered, what kind of contextual elements are involved etc. Such a fine grained analysis should of course be based on previous work in semantic modeling, but should also adopt a more holistic viewpoint. That is where the use of structures from the field of computer science appears to be beneficial.

Another challenge beyond information storage and modeling relates to the cognitive processes that underlie real-world, multi-speaker interactions. Indeed, a realistic conversation is seldom totally fluid: people hesitate, do (self-)repair, produce disfluencies or tend to digress. All these phenomena have only been recently taken into account, and seem difficult to model in a systematic, computer-oriented way. We have tried to consider a model for incremental speech processing that could both emulate effective cognitive processes and compute information in a more flexible, reversible way. Applications of this models could be found in dialogue management systems, where a human speaker interacts with a non-human entity (one of the first attempts was the TRAINS project[1], and more recently of course Amazon Alexa, the Google Assistant etc.).

## 1 The nature of dialogue

We begin by a brief review of the notion of dialogue. What is a dialogue? The notion seems to be rather straightforward in the sense that it is deeply rooted in our use of language. We learn a language by listening to dialogues, and dialogue is our principal means of expression, leaving aside the literary language that we use in written communication, for instance in this report. However, the knowledge that we have about dialogue *instances* is not a knowledge about dialogue in itself. Here we will try to clarify what we mean by dialogue; we will introduce its main features through corpus examples[2]; and we will give a sketch of the underlying linguistic background (syntax, semantics).

[1]https://www.cs.rochester.edu/research/cisd/resources/trains.html

[2]we have used for this purpose the Santa Barbara corpus of spoken American English [**?**], which is based on a large body of recordings of naturally occurring spoken interaction from all over the United States. The Santa Barbara Corpus represents a wide variety of people of different regional origins, ages, occupations, genders, and ethnic and social backgrounds. The predominant form of language use represented is face-to-face conversation (source: http://www.linguistics.ucsb.edu/research/santa-barbara-corpus). Each excerpt we picked goes with the name of the file it comes from within the corpus. A more precise description of the context is given for each file in the webpage previously mentioned.

## 1.1 What is dialogue? First explanations and examples

The word dialogue is opposed to monologue; indeed, a dialogue most evidently involves two or more participants that are supposed to speak in turn[3].

In Table 1 for instance, the dialogue involves three speakers, who perform 2 speech acts each. But of course, the boundaries between speech acts are not always clear, when people tend to interrupt each other, or simply when the turn-taking is triggered in the middle of an utterance [4]. Thus, a dialogue is not necessarily sequential, and does not necessarily include well formed sentences. It involves a kind of concurrency between the participants.

| | |
|---|---|
| LENORE: | [Is !Amanda] her kid, |
| | or, |
| LYNNE: | .. Yeah. |
| LENORE: | [Oh]. |
| DORIS: | [she had] a little three-year-old daughter. |
| LYNNE: | ... It's just too bad. |
| | ... I just hope !Orville and !Genetta get that little girl. |
| DORIS: | Oh they won't. |

Table 1: dialogue as a turn-taking process (SBC001.trn)

Besides its superficial structure, a dialogue is also a series of motivated acts: if we talk to each other, it is not usually for the sake of talking. We speak, because most of the time we want to exchange information. The participants usually share a certain amount of that information, which is considered as the *common ground* (for instance, that the earth is flat, that human beings live about a hundred years...). But they also have personal ideas they can share using informative sentences [5]. Conversely, information can consists of queries using inquisitive sentences [6].

All these actions, all these speech acts, have a direct influence on the common ground, that is continuously updated during the interaction. Conversation is hence a continuous effort of coordination between the participants that try to align themselves to the common ground in real time [**?**, **?**]. This is essential, because this ensures that the conversation remains coherent and useful for both participants.

An example of such a coordination mechanism is given in Table 2. First, the fact that "he" is a nice guy should be added to the common ground (we assume implicit acknowledgement). After that, the fact that Ron wonders whether "he" teaches in schools should be tracked in some way. Then, the fact that "he" is not married and gives private lessons and orchestras should be grounded as well (here the acknowledgement is made explicit).

Finally, information can be conveyed by non-linguistic elements, like perceptions of the environment, and gestures. The processing of these informations, that include multimodal channels, is particularly challenging (cf. Table 3 and Table 4).

Therefore, dialogue has a specific structure that relies on turn-taking, and involves complex exchange of informations. In our study, we will thus assume some simplifications :

- we will focus on *oral dialogue* (as opposed to the written dialogues that can be found on forums etc.);
- we will only consider *two speakers*;

[3]the notion of turn-taking is not very problematic when the number of participants is limited (typically 2); however, with multi-party conversation it becomes a crucial issue. So the partition between *mono* and *dia* is maybe a bit reductive.

[4]as we can see in the first intervention of LYNNE

[5]some of the time, the notion coincides with declarative sentences

[6]some of the time, the notion coincides with interrogative sentences

| | |
|---|---|
| FRANK: | ... (TSK) He seems like a prety nice guy though. |
| RON: | Does he teach in the school system too? |
| | Or – |
| FRANK: | M-m. |
| | ... (TSK) He's strictly, |
| | ... pri- – |
| | he's not married or anything, |
| | and I think strictly private lessons, |
| | a=nd orchestra. |
| RON: | Hm. |

Table 2: dialogue and the duality between informativeness and inquisitiveness (SBC019.trn)

| | |
|---|---|
| BEN: | ... Anybody needs the elevator it's available. |
| >ENV: | ((CROWD_AND_MACHINE_NOISE)) |
| BEN: | Anybody that needs the elevator it's ¡X is X> available. |
| >ENV: | ((CROWD_AND_MACHINE_NOISE)) |
| BEN: | ... Come on. |
| >ENV: | ((CROWD_AND_MACHINE_NOISE)) |

Table 3: dialogue and the environment (SBC038.trn)

- we will not deal with simultaneous utterances, only *sequential* ones.

## 1.2 How is dialogue processed?

In this section we will argue for the inherent incrementality of dialogue, and justify why it should be processed in that way too.

### 1.2.1 Cognitive realism

First, it is obvious that dialogue is produced incrementally: when people speak together, they cannot have a precise idea of what will be said in the long run. What people say depend on what they think of course, but also on what they think the addressee will think... and so forth [?]. We have also seen that conversations can involve asymmetries of information when the two participants are not "synchronized" with respect to the common ground. They thus produce utterances – if not even words – on the fly. This incremental production mechanism is evidenced by so-called *disfluencies* or self-corrections (cf. Table 2, FRANK 2$^{nd}$ turn).

Second, there is a large amount of psycholinguistic evidence[7] showing that people understand sentences incrementally. In other words, people begin to form and compose semantic objects that correspond to the word they read or hear in one shot; and they are even able to anticipate the form/meaning of the subsequent entities. This of course can lead to syntactic misunderstanding and backtrackings, as showed by the so-called "garden-path sentences" [?]:

"The horse raced past the barn fell." [?]

[7]Experimentally, eye-tracking techniques have shown that reference resolution begins right after the first hint is given and long before all the ambiguity is cleared up [?]. In particular it has also been shown that the processing of verbs introduces bias towards the most probable type of object [?]. [?] showed that interclausal relationships (causation, diagnostic) are processed before the end of the last clause. Using another technique, namely event-related brain potentials (ERP), [?] advocate for the existence of two different interpretation pathways (a syntactic one and a thematic one) that however do not achieve the same degree of incrementality. But for reasons of space we cannot dive deeply into these topics.

| | |
|---|---|
| DEBRA: | (H) he is twenty-four years old, |
| | and stands a very impressive (H) seventeen point one hands tall. |
| | ... Ladies and gentlemen, |
| | the great handicap horse, |
| | ... Forego. |
| MANY: | ... ((APPLAUSE)) |

Table 4: dialogue and the environment (SBC040.trn)

Intuitive first reading: [The horse]$_{NP}$ [[raced]$_V$ [past the barn]]$_{VP}$ [fell]$_?$.

More consistent second reading: [[The horse]$_{NP}$ [raced past the barn]]$_{NP}$ [fell]$_{VP}$.

Other examples involving lexical ambiguity can be found in [?], for instance:

The pen$_{area?,tool?}$ is in the box
The box is in the pen$_{area}$

In the first sentence, the ambiguity regarding the meaning of "pen" remains until the end of the sentence. Yet we are able to understand this sentence quite rapidly, most often without backtracking. This kind of phenomenon advocates for the fact that we do not actually process all the possible alternatives in parallel and "prune the tree" afterwards. We rather keep in mind and underspecified version of the lexeme (here, "pen"), and enrich only when we get additional information about the most probable meaning (here, the word "box" allows to opt for pen$_tool$). Thus as we will see, underspecification is a strong desideratum of incrementality.

An automatic dialogue system that could achieve incrementality would thus be in the position to achieve cognitive realism as well as speech naturalness, as pointed out by [?]. Indeed, incrementality goes with a following valuable features, which is sub-sentential intervention, that is closely related to efficiency as well.

### 1.2.2 Computational efficiency

On the other hand, incremental processing may prove more effective than usual holistic parsing methods. First, the system would certainly gain in time efficiency, since incremental processing can begin as soon as the first word (or more generally, the first *incremental unit*, cf. 3.1) is produced. Thus, even if the incremental computation takes a longer (but reasonable) time, it can still be advantageous.

Second, an incremental system should be able to converse with the user more efficiently, *i.e.* avoid useless explanations or conversational dead-ends. A perfect incremental system should indeed process the maximal amount of information at each step (in a word-by-word basis), including for instance anaphora resolution. This feature, called *strong incremental interpretation* is notably pointed out by [?], after [?]. An even more desirable system should be in a position to predict with a relatively high confidence what will come *next*. Such an advanced ability would allow for ellipsis, sentence completion and "just-in-time" correction; more precisely, this could be used to:

- show early agreement, and thus possibly avoid further unnecessary explanations (cf. Table 5);
- underline an inconsistency to trigger justifications or self-correction, thus avoiding a propagation of the mistake (cf. Table 6);
- complete a sentence when its meaning appears sufficiently predictable, and thus save the speaker's time and effort (cf. Table 5 and Table 7).

In the following section we will introduce very briefly the current challenges of formal syntax and semantics and how this relates to theoretical computer science. We will set aside lower levels of linguistics (phonology, morphology), because they do not directly concern our main problem.

| | |
|---|---|
| JUDY: | Well Dad always [4picks4], |
| TIM: | [4I4] .. get a little picky o=n ... my shirts (Hx). |
| JUDY: | .. @(Hx) ... Yeah, |
| | I know. |

Table 5: sentence completion and early agreement (SBC048.trn)

| | |
|---|---|
| GILBERT: | .. I don't think [2there's as2] much [3trust3], |
| MONTOYA: | [2Oh2], |
| | [3Perot3]. |
| GILBERT: | .. XX, |
| MONTOYA: | Trust. |
| GILBERT: | Trust in the system. |
| MONTOYA: | .. [Why]. |

Table 6: echoing and sentence correction (SBC012.trn)

## 1.3 A bird's eye view on sentence-level linguistics

Here we give a very brief account of the formal, computer-science oriented treatment of linguistics. The ideas that follow are from the most mainstream theories and representations of the field; that is why we will not necessarily stick to them in the following sections.

### 1.3.1 Syntax

The formal syntax of natural language relies on the notion of tree. The syntactic tree can be understood in two ways that are equivalent:

- the top-down approach: the whole sentence is divided into sub-constituents called phrases (verb phrase, noun phrase...), and each of these constituents are divided again until a so-called head is produced. The head (verb, noun, determiner, preposition) has a unique child, which is a lexical item (a word). The top-down approach is often related to language generation;
- the bottom-up approach: the words of the sentence are mapped to part-of speech tag (the heads), which are in turn merged to form phrases. The merge operation stops when the root (the whole sentence) is produced. The bottom-up approach is often related to syntactic parsing;

However, it is important to note that the merge strategy (or equivalently the branching strategy) is not fully deterministic; sometimes several rules can be applied and thus the same sentence can lead to different parse trees. This is exemplified in Figure 9, where the constituent "with a telescope" can be seen as a complement of the verb or as a complement of the NP "a man". This syntactic (structural) ambiguity generates a semantic ambiguity. Therefore, the parsing strategy is considered as crucial in an incremental dialogue system, that is expected to process the words in one pass.

### 1.3.2 Semantics

It is generally assumed that the semantics of a sentence computed by applying a kind of morphism to its syntactic structure. Indeed, this idea complies with the so-called principle of compositionality [?], according to which the semantics of a sentence depends on the semantics of its smallest constituents, and on the way they combine.

Thus, the computation of the semantics of a sentence amounts to decorating the syntactic tree with semantic expressions. Semantic expressions are basically:

| | |
|---|---|
| JUDY: | Oh, |
| | this is great. |
| | ... Stainless steel? |
| | ... Hmm? |
| TIM: | [Yep]. |
| JUDY: | [Stai-] – |
| LEA: | [Stain]less. |
| | .. Unhunh. |

Table 7: sentence completion (SBC048.trn)

- truth values of type `t` [base case];
- functions from entities (type `e`) to semantic expressions [inductive case].

The "merge" operation that is used to create the syntactic tree is also mapped to a particular operation in the semantic domain, which is, in the most simple framework, functional application. Note that the last merge operation must generate a term of type `t`, because – in truth-conditional semantics at least – sentences must have a certain truth value. Again, the semantic layer can be subject to certain

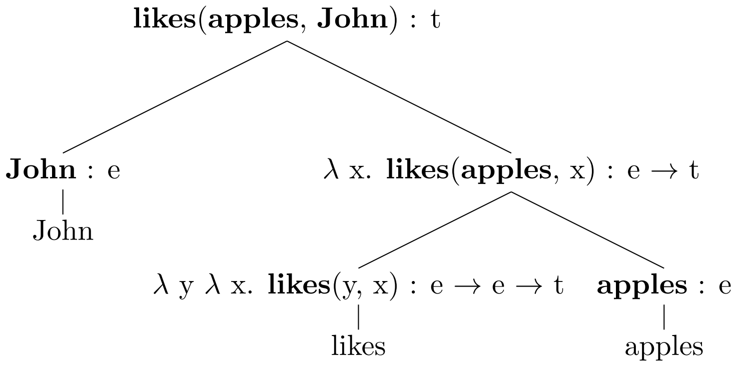

**likes**(**apples**, **John**) : t
**John** : e
John
λ x. **likes**(**apples**, x) : e → t
λ y λ x. **likes**(y, x) : e → e → t
likes
**apples** : e
apples

Figure 1: Simple semantic computation (bold text for semantic entities)

types of ambiguity, in particular with regards to quantifier scope [**?**]:

"Every man loves a woman"

$$\text{Universal reading : } [\forall x : \mathbf{man}(x)]([\exists y : \mathbf{woman}(y)](\mathbf{love}(x, y)))$$

$$\text{Existential reading : } [\exists y : \mathbf{woman}(y)]([\forall x : \mathbf{man}(x)](\mathbf{love}(x, y)))$$

As we will see later on (in Section 3.3 in particular), this can be "resolved" – or at least, accounted for – through underspecification. More generally, we will see that this traditional semantic approach is not sufficient to deal with dialogue because it does not integrate all the information available (for instance, visual information, event-based information).

## 2 A model for dialogue

We present here a framework for dialogue modeling, that unifies and brings together ideas from various schools of thought in formal semantics. It should be considered as an additional layer with respect to usual linguistic categories. The core of the theory is strongly inspired by Head-driven Phrase Structure Grammar (HPSG) [**?**] and Type Theory with Records (TTR) [**?**, **?**]. For the purpose of *dialogue* modeling, elements of situation semantics [**?**] are also taken into account.

## 2.1 Type theory with records and HPSG

### 2.1.1 Type theory with record (TTR)

Our framework for dialogue modeling is based on Type Theory with Records (TTR) [?, ?, ?], which allows to use relatively elaborate data structures whose general form is :

$$R = \begin{bmatrix} l_1 & = & k_1 \\ l_2 & = & k_2(l_1) \\ & \dots & \\ l_i & = & k_i(l_1, \dots l_{i-1}) \\ & \dots & \\ l_n & = & k_n(l_1, \dots l_{n-1}) \end{bmatrix} \quad \text{where} \quad R : \begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_2(T_1) \\ & \dots & \\ l_i & : & T_i(T_1, \dots T_{i-1}) \\ & \dots & \\ l_n & : & T_n(T_1, \dots T_{n-1}) \end{bmatrix} \tag{1}$$

These are basically ordered tuples with labels $(l_i)_{i\in[1,n]}$, values $(k_i)_{i\in[1,n]}$, dependent types $(T_i)_{i\in[1,n]}$, and a simple rule for typing :

$$\forall i \in [1, n],\ k_i : T_i(T_1, \dots T_{i-1})$$

Note however that the relation : is a kind of maximal typing relation. Any subset of fields extracted from the maximal record type can also be considered as a type for R, provided that the pairwise typing relation (restricted to the remaining fields) is still verified. More precisely:

$$R : T \iff \begin{cases} \forall l \in \mathit{labels}(T),\ l \in \mathit{labels}(R) \\ \forall l :_T T' \in \mathit{labels}(T),\ R.l : T' \end{cases}$$

Where:

- $\mathit{labels}(X)$ is the set of labels that are present ix $X$, $X$ being a record or a record type;
- $R.l$ is the value of the field labeled by $l$ in the record $R$. This is a classic "path" notation.

This more permissive typing allows in fact for subtyping and polymorphism [?].

TTR also allows for manifest types, which are of the form:

$$l_{=k} : T$$

This means that the label $l$ is typed by a type $T$ that has only one inhabitant. This unique inhabitant is written in the subscript of the label (here, it is named $k$). This may be seen as a shorter way to type and declare a variable at the same time.

### 2.1.2 Head-driven Phrase Structure Grammar (HPSG)

Head-driven Phrase Structure Grammar (HPSG) [?] is a descendant of Generalized Phrase Structure Grammar. It is an integrated formalism for linguistic modeling. HPSG entities are highly lexicalized *feature structures* inspired from the Saussurian concept of *sign*. A sign is – in this sense – the union of a *signifier* (phonic component) and a *signified* (syntactic category, semantic content). Feature structures indeed pack all these elements together, in the form of what could be seen as records. A very simplified example of an HPSG entry can be found in Figure 2[8].

HPSG is a constraint-based approach, which thus involve unification processes. In brief, three kinds of syntactic rules apply to HPSG feature structures:

- phrase-structural schemata (inheritance relationships);

[8]Note that in this original presentation, types do not appear as a central element, in the sense that they are not systematically specified (they appear as "headers" of the feature structures). However the theory actually takes these types into account: types feature structures are indeed constrained by multiple-inheritance trees.

[9]the symbol "/" is used in HPSG to indicate that a certain specification is "defeasible" in the sens that it can be overridden by a conflicting specification [?].

$$\begin{bmatrix} verb-lxm & \\ SYN & \begin{bmatrix} HEAD & \begin{bmatrix} verb & \\ PRED & - \\ INF & /^9 \\ AUX & /- \\ POL & - \end{bmatrix} \end{bmatrix} \\ SEM & \begin{bmatrix} MODE & prop \end{bmatrix} \\ ARG-ST & \left\langle \begin{bmatrix} HEAD & nominal \\ VAL & \begin{bmatrix} SPR & \langle\,\rangle \\ COMPS & \langle\,\rangle \end{bmatrix} \end{bmatrix}, \ldots \right\rangle \end{bmatrix}$$

Figure 2: Example of an HPSG entry (feature structure) for a verb [?]

- feature percolation principles (related to headedness, syntactic merge and unification);
- principles accounting for major syntactic phenomena (binding theory...).

Like HPSG entries, HPSG rules are modeled as records with two fields (one for the input, one for the output). This formulation will be used later to model conversational rules in Section 4.

HPSG *per se* is not totally well suited for incremental processing, however we will see that existing meta-languages (especially MRS, see Section 3.3) or variants of this formalism (for instance SBCG [?]...) allow for more flexibility, in particular with respect to underspecification.

## 2.2 Situation semantics

Situation semantics is a framework that has been developed by Jon Barwise and John Perry in the 1980s [?, ?], and which takes ideas from the Austinean theory of truth and speech acts. For Austin, "A statement is made and its making is an historic event, the utterance by a certain speaker or writer of certain words (a sentence) to an audience with reference to an historic situation, event or what not.". For a statement to be true, the *descriptive conventions* (that assign situation types to words) must be correlated with the *demonstrative conventions* (that assign to each statement some "historic" situations in the world).

This idea that truth is a relation between situations and situation types has been picked up by Barwise and Perry in more formal terms. A good introductory account of this can be found in [?]. Situation semantics, unlike the various derivatives of fregean semantics, focuses on the notion of *situation* (and not truth values, or "Sinn"...). Truth and falsity are just treated as side-effects of the theory.

So, in situation semantics, the meaning of an utterance is not equivalent to its truth conditions, it is rather seen as a relation between situations: the situation related to the speech act, and the situation related to what is said. A situation can then be seen as an underspecified world (in the sense of possible world semantics), where some constraints are met, but some other properties remain unknown. Situations can easily be modeled as records. For instance, a situation where John loves Mary could be described by the record type:

$$\begin{bmatrix} j & : & \mathtt{Ind} \\ m & : & \mathtt{Ind} \\ s & : & \mathtt{love}(j,m) \end{bmatrix}$$

More precisely, the meaning of an utterance u (i.e. a string of words) about an event e is modeled as:

$$s_u [\![u]\!] s_e$$

Where $[\![u]\!]$ specifies a certain number of constraints linking the fields of $s_u$ and $s_e$. In TTR, these constraints can easily be expressed as typing constraints, for instance [?]:

$$s_u[\![\text{ I am sitting }]\!]s_e \iff s_u : \begin{bmatrix} i & : & \texttt{Ind} \\ l & : & \texttt{Loc} \\ s & : & \texttt{speak}(i,l) \end{bmatrix} \bigwedge s_e : \begin{bmatrix} i & : & \texttt{Ind} \\ l & : & \texttt{Loc} \\ s & : & \texttt{sit}(i,l) \end{bmatrix} \bigwedge \begin{cases} s_u.i = s_e.i \\ s_u.l = s_e.l \end{cases}$$

## 2.3 Dialogue GameBoards

The formal features of dialogue are highly inspired from [?, ?], although some of them are slightly modified. Records and record types will be used to model the various (sometimes heterogeneous) features of a dialogue:

- turn (who speaks and who listens);
- "doxastic" states (facts people believe);
- environment (in particular visual informations);
- questions under discussion (issues raised by the dialogue so far);
- moves (dialogue "history");
- pending utterance (what is being said but not grounded).

This is summarized in the following record type, called a Dialogue Game Board (DGB) Type:

$$\texttt{DGBType} = \begin{bmatrix} \text{Speaker} & : & \texttt{Ind} \\ \text{Addressee} & : & \texttt{Ind} \\ \text{Facts} & : & \langle \texttt{Set}(\texttt{Prop}), \texttt{Set}(\texttt{Prop}) \rangle \\ \text{Visual} & : & \texttt{Situation} \\ \text{QUD} & : & \texttt{Poset}(\texttt{Questions}) \\ \text{Moves} & : & \texttt{List}(\texttt{LocProp}) \\ \text{Pending} & : & \texttt{List}(\texttt{LocProp}) \end{bmatrix}$$

In the following subsections we will go over the details of the different fields and types that ahve just been introduced in `DGBType`.

### 2.3.1 Turns: speaker and addressee

These fields are the most straightforward; indeed they just keep track of the turn ownership (who speaks and who listens at a given time). "Speaker" and "Addressee" fields are of type `Ind`, which can be seen as a subtype of `Entity`[10]. A simple way to implement that would be to give each participant a unique ID.

### 2.3.2 Facts

"Facts" are assumptions held by the participants. These assumptions allow them to answer questions, to justify themselves, and possibly to start a conversation about a given subject. This structure is presented as a pair of partially ordered sets, a view that differs from [?, ?], where "Facts" are seen as "shared assumptions". Indeed, conversational events like contradiction come from an asymmetry of belief between participants, and that is why we thought it would be beneficial to store two separate copies of the participants' respective beliefs.

Each set of facts, one per participant, is thus a poset of "propositions" (type `Prop`). These propositions can be written in First-Order Logic, but also in other systems (like for instance MRS,

[10]type `e` of Montague grammars

cf. Section 3.3, or TTR cf. Section 3.2)[11]. The partial order would be the implication relation (this choice will be justified in Section 4.6).

### 2.3.3 Questions

The field "Questions" is also a partially ordered set of element of type `Question`. The order used in that case is a bit more unclear; however it has to account for a type of *precedence* [?]. This means that at a given point of the conversation, a question should be more topical than the others. We could say that it is basically the last question that has been set, but yet not always. This notion of ordering between questions is crucial to establish a notion of relevance in the Gricean terms [?], as we will see in Section 4.

### 2.3.4 Moves

The field Moves is a history of the speech acts that have been grounded. This field has been introduced by Ginzburg (see for instance [?, ?]) and its content relates to situation semantics. Indeed, the objects of type `LocProp` that are listed in this field are of the form:

$$\texttt{LocProp} = \begin{bmatrix} \text{Situation} & : & \texttt{Sit} \\ \text{Speaker} & : & \texttt{Ind} \\ \text{Addressee} & : & \texttt{Ind} \\ \text{R} & : & \texttt{SpeechAct} \\ \text{Content} & : & \texttt{Prop} \\ \text{Situation type} & : & R(\text{Speaker}, \text{Addressee}, \text{Content}) \end{bmatrix} \simeq R(\text{Speaker}, \text{Addressee}, \text{Content})^{12}$$

The field named "Situation type" binds a certain utterance to a certain situation; as we have seen this is the core idea of Austinean semantics. The type called `SpeechAct` can be seen as the supertype for illocutionary speech acts as defined in [?] : `Assert` (for "Assertives"), `Direct` (for "Directives"), `Com` (for "Commissives"), `Expr` (for "Expressives"), `Decl` (for "Declarations"). But of course we could imagine many more precise subtypes for each of these illocutionary speech acts. We will use some of them in Section 4 too.

### 2.3.5 Pending

The field "Pending" is quite similar in its structure to the field "Moves". Indeed, we will see that utterances tend to be pushed first in the "Pending" field, before being moved to the "Move" field. The field "Pending" is indeed used to store the ungrounded utterances, which includes the forthcoming utterance. This last utterance is a bit particular because, for the sake of incrementality, it must be updated in a word-by-word basis. This also means that this last utterance is most of the time underspecified. This justifies the need for incremental representation of meaning and incremental parsing methods. These topics will be studied more in depth in the next section.

## 3 Dealing with incrementality

In this section, we present the current approaches that tackle the problem of incremental semantic interpretation : Dynamic Syntax with TTR and Incremental Processing (DS-TTR IP) and Incremental Robust Minimal Recursion Semantics (iRMRS). Both tend to comply with a more abstract, high-level framework designed by [?] for general incremental systems.

[11]we will finally opt for one of these last alternatives, because the use of type `Prop` in the field "Pending" forces us to have a richer representation that must include underspecification. So this choice is not driven by the nature of "Facts", but rather by our will to unify the types for propositions across the different fields, some of them – like "Pending" – being incremental.

[12]we will use this although not completely correct "wrapped" notation in the following, for reasons of space and clarity.

### 3.1 A general model for incremental processing

A very abstract, general framework for incremental processing in dialogue has been developed by [?]. It is inspired by [?] token passing architecture. This model is independent of the formalism used for syntactic/semantic modeling, and has many applications in dialogue, from speech recognition to semantic interpretation. The key concepts of this model are *incremental units* (or IUs) and *modules.* Modules basically process incremental units.

#### 3.1.1 Incremental Units

An IU refers to the smallest possible chunk of information that can be processed independently by the system. IUs are of different types, from raw input (audio signal, character string...) to higher order structures (records as we will see later, or elements of first order logic...). The segmentation of IUs can be temporal (fixed time window) or categorical (word, phoneme etc.) In our case, IUs will basically have the granularity of a word, so they will represent words or their syntactic/semantic counterparts. More generally, IUs carry three types of information:

- a confidence score: how sure is the system about the production of this IU?
- a commitment flag: is it possible to revoke the IU (correction)?
- a history: which module has processed the IU so far?

Besides this metadata, IUs are defined by the relations they may have with other IUs:

- horizontal relations: between IUs of the same linguistic level (semantic, syntactic...), *i.e.* produced by the same module. This includes the *successor* relation that keeps track of linear order in which the IUs have been produced (alternative paths being possible). One can also think of other non-linear dependency constraints (modification, specification, $\theta$-roles...);
- vertical relations: these are hierarchical relations, between IUs of different linguistic levels. This includes most importantly the *grounded* relation, which links an higher order IUs to the lower order IUs that justify its production.

#### 3.1.2 Modules

Modules are processing units. A module consumes IUs and produces IUs; it consists of three elements:

- a left (input) buffer: where IU arrive before being processed;
- a right (output) buffer: where IU exit after being processed;
- a processor, that turns a bunch of input IUs to a bunch of output IUs

Since the system is not necessarily instantaneous, a processor may sometimes "wait" for more than one IU in the left buffer to produce a single output IU. It is thus a function from sets of IUs to sets of IUs, and not just a function from IUs to IUs. This is depicted in Figure 3

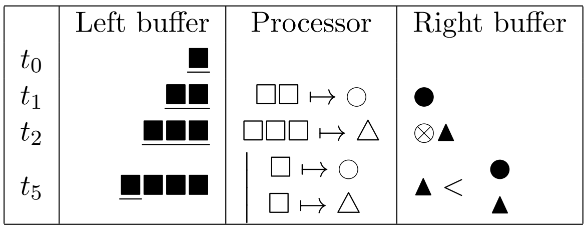


Figure 3: Schematic of a module (incremental processing unit) with delayed, batch computation ($t_1$), revision ($t_2$), and production of alternatives ($t_3$)

When IUs flow through the system, they go from one module to another. Modules thus form a particular kind of graph. The edges of this graph allow IUs to circulate through modules, with several potential restrictions:

- directions: edges are oriented; they may or may not be bidirectional;
- multimodality: edges can allows for multimodal input (IUs of different types) or output;
- filtering: edges may filter certain IUs depending on their syntactic/semantic characteristics

The two approaches we will mention in the following section have been implemented within the Incremental Unit framework. Moreover, both of them rely on record types.

## 3.2 DS-TTR with incremental processing

Dynamic syntax [?] is a model that incorporates syntactic and semantic parsing. Indeed, DS parsing generates the syntactic tree and decorates it with typed $\lambda$-terms, in one pass. In DS-TTR [?, ?], the $\lambda$-terms have enriched types:

- a record $r$ is a term in DS-TTR;
- a function from a record to a term $t$ of DS-TTR ($\lambda r.t$) is also a term of DS-TTR.

Besides, DS relies on a lexicon that is composed of actions. These actions are formulated as IF-THEN-ELSE statements that enrich monotonically the content of the semantic tree (completion of an existing node or creation of a new node). Actions are of two kinds:

- lexical actions (triggered when a new word is uttered);
- computational actions (that can be used to enrich the semantic tree depending on the words that are already present).

A DS parse is then a series of lexical actions with possibly intermediate computational actions:

$$(LEX\ COMP^*)^*$$

Each action is performed according to a pointer that signals the node that is currently under construction. This node is the one that is evaluated when checking the preconditions in the IF clause of an action. The progression of the pointer is bottom-up.

## 3.3 Incremental RMRS

Minimal Recursion Semantics [?] is a model that is characterized by a flat semantics, whose central element is the elementary prediction (EP). As we will see, an utterance (type `Utt`) is modeled as a bag of predications, a top and a local top handle, and a list of constraints (cf. Figure 4).

### 3.3.1 Predications

Predications are handle-predicate pairs. Handles (in that situation they are also called *labels*) are indices that allow to keep track of the original tree structure on which the MRS has been built[13]. They denote node addresses in the original tree; and if two EPs have the same handle, they must be conjoined. Predicates can have two kinds of possible arguments:

- ordinary variables like x, y, z... modeled as a list of elements of primary type `Var`;
- handles that refer to the EP's daughters in the underlying tree. Handles seen as arguments (also called *holes*) thus reify the branches of the tree and flatten the structure (predicates no longer apply to other predicates...). Holes are naturally modeled as a list of handles (of type `Handle`).

In so doing, predicates can be represented as elements of type `Pred`, see Figure 4.

In that setting, generalized quantifiers have for instance one variable (the one that is bound by this quantifier) and two holes (one that refer to the restriction predicates, and one that refers to the body predicates). Modifiers like adverbs usually have only one hole (that refers to the predicate they

[13]thus this structure is obtained after syntactic parsing

$$\texttt{Utt} = \begin{bmatrix} \text{Hook} & : & \begin{bmatrix} \text{GTOP} & : & \texttt{Handle} \\ \text{LTOP} & : & \texttt{Handle} \end{bmatrix} \\ \text{Rels} & : & \texttt{Bag(Pred)} \\ \text{Cons} & : & \texttt{List(QEQ)} \end{bmatrix} \quad \texttt{Pred} = \begin{bmatrix} \text{Rel} & : & \texttt{Prop} \\ \text{Label} & : & \texttt{Handle} \\ \text{Args} & : & \texttt{List(Var)} \\ \text{Holes} & : & \texttt{List(Handle)} \end{bmatrix} \quad \texttt{QEQ} = \begin{bmatrix} \text{HARG} & : & \texttt{Handle} \\ \text{LARG} & : & \texttt{Handle} \end{bmatrix}$$

Figure 4: Main types in RMRS

modify) and no variables. And "regular" predicates (like verbs) usually have only variables.

Note that handles also allow for underspecification and ambiguity, which makes RMRS more attractive for our purpose than simple predicate logic. An underspecified handle [14] may indeed generate various syntactic trees. This is exemplified in Figure 12. This mechanism is also used for the modeling of Wh-questions, which is known to be a rather complex problem (see for instance [?] for a quite complete account of questions and their interactions with quantifiers).

### 3.3.2 Constraints, (local) top handle

Conversely, underspecification can be resolved by adding constraints to the structure. Constraints are separate equations between handles; basically, these equations allow:

- to "reconnect" the underlying tree, when a vacuous hole is equated with a label;
- to conjunct daughters, when two labels are equated;
- equating the top handle with a label.

Constraints are added incrementally while parsing a sentence (and thus merging constituents). However in that case, a special, more relaxed kind of equality is used: equality modulo quantifier ($=_{qeq}$). A hole is qeq to a label ($h =_{qeq} l$) iff:

- they are equal ($h = l$);
- or the hole refers to the label of a quantifier and that quantifier's body contains another hole $h'$ such that $h' =_{qeq} l$.

In other words, a right traversal of the tree from the hole to the label shall only cross quantifier nodes. The relaxed equality allows to keep scopal ambiguity when necessary.
Parsing involves constraints, but also involves headedness, like in traditional syntactic theories including HPSG. In RMRS, the notion of headedness is embodied by the local top feature which is the topmost label that is not a quantifier label. When two constituents are merged, the local top of the head is preserved (cf. the three syntactic rules of Figure 13).

## 4 Updating DGBs

Models such as Dynamic Syntax and Minimal Recursion Semantics build semantic content monotonically and incrementally. The question is then how to use this content to update the current dialogue state in a relevant way. In this section we devise higher-level operations that model sub-sentential phenomena that are typical of dialogue: acknowledgement, clarification request, repair, cooperation, ellipsis.

[14] *i.e.* a handle that is not addressed and that is passed to a predication

## 4.1 Notations

We will define updates as programs from DGBs to DGBs. An update occurs when the input DGB satisfies some preconditions. The output DGB is then produced according to some postconditions (as we will see, the operation that is performed is always of the same kind). Preconditions and postconditions are typing conditions, that are represented using record types.

A record R satisfies the preconditions P iff P is a subtype of R, which means that all the labels in R are specified in P, and for these labels, their types in P must be subtypes of their types in R. The postconditions Q are automatically verified, because the operation that performed on the input record type is:

$$\lambda R.R \boxed{\wedge} Q$$

Where $\boxed{\wedge}$ is the asymmetric merge operation that produces a record type by performing the following updates:

- copy the labels that are only in R;
- copy the labels that are only in Q;
- for all label that is both in R and Q:
  - if the label is typed with a simple type, keep the type specified in Q;
  - if the label is typed with a record type, take the asymmetric merge of R's version and Q's version.

This ensures that all the labels in Q are present in the output with the right type (the one specified in Q), and thus that the output record type is a subtype of Q (see Appendix **??** for details). This also means that the update operation creates monotonic refinements of the initial DGB.

## 4.2 Move-generated rules

Move-generated rules deal with Moves, that is, complete speech acts that have been grounded. The last move that have been performed will basically trigger a certain number of operations on the other fields of the DGB (in particular the QUD and the Facts fields). Most of the update rules depicted in this section have already been studied in [**?**]. We add some constraints on the doxastic sets of the participants (field Facts) to better comply with Grice's maxim (in particular with respect to quality).

## 4.3 Question

The question-update is one of the most simple update rules. It has already been presented in [**?**, **?**]. The idea is that, if the last Move has been identified as a "Ask" speech act whose content is a question *q* which is not resolvable by the speaker (on the basis of his personal assumptions), then the question *q* becomes the topical question of the conversation. The only distinction we make with respect to the previous models is that we assume the speaker *alone* is not able to resolve the question. Conversely, the addressee may know a Fact that could solve it. In this rule, we used a dependent type *resolve*.

$$\begin{bmatrix} \text{Pre} & = & \begin{bmatrix} \text{QUD} = Q & : & \texttt{Poset(Question)} \\ \text{Moves} = Ask(spkr, addr, q) :: M & : & \texttt{List(LocProp)} \\ \text{Relevance-cond} & : & \neg resolve(Facts.spkr, q) \end{bmatrix} \\ \text{Post} & = & \begin{bmatrix} \text{QUD} = q :: Q & : & \texttt{Poset(Question)} \end{bmatrix} \end{bmatrix}$$

Figure 5: Question update

### 4.4 Assertion

The assertion-update rule is almost as simple as the question-update, and also very similar. Indeed, asserting something is – from a more pragmatic point of view – equivalent to raise the issue of whether this thing is true or not. The assertion rule thus says that, if the last Move has been identified as an "Assert" speech act whose content is $p$, where $p$ is a proposition that can be deduced from a fact $p'$ that belongs to the speaker's beliefs, then the issue of whether $p$ becomes the topical question under discussion.

$$\begin{bmatrix} \text{Pre} & = & \begin{bmatrix} \text{QUD} = Q & : & \texttt{Poset(Question)} \\ \text{Moves} = \mathit{Assert}(\mathit{spkr}, \mathit{addr}, p) :: M & : & \texttt{List(LocProp)} \\ \text{H} & : & \texttt{Set(Prop)} \\ \text{Justification-cond} & : & \mathit{subset}(H, Facts.\mathit{spkr}) \\ \text{Quality-cond} & : & \mathit{deduced}(p, Facts.\mathit{spkr}, H) \end{bmatrix} \\ \text{Post} & = & \begin{bmatrix} \text{QUD} = ?p :: Q & : & \texttt{Poset(Question)} \end{bmatrix} \end{bmatrix}$$

Figure 6: Assertion update

Here, we used three external operators: *deduced*, *subset* and ?. The fact $p$ is deduced from a set of Facts $F$ with evidence $H$ ($\mathit{deduced}(p, F, H)$) iff $H$ a the minimal subset of $F$ that allows to derive $p$[15]. $H$ being a subset of $F$ complies with Grice's maxim of quality, while $H$ being minimal complies with Grice's maxim of manner. The notation *subset* is quite evident.

### 4.5 Acknowledgement

Acknowledgement may be triggered when one speaker approves what the other speaker has just said or was about to say. In the last case, the approver has been able to mentally complete what has been said orally by the other speaker. Moreover, the fact that is acknowledged must be consistent with the other facts the approver believes in. It may or may not be part of the approver's initial beliefs.

$$\begin{bmatrix} \mathit{Pre} & = & \begin{bmatrix} Facts.spkr = F_s & : & \texttt{Poset(Prop)} \\ Facts.addr = F_a & : & \texttt{Poset(Prop)} \\ QUD = p? :: Q & : & \texttt{Poset(Question)} \\ Moves = \mathit{Ack}(\mathit{spkr}, \mathit{addr}, p) :: M & : & \texttt{List(LocProp)} \end{bmatrix} \\ \mathit{Post} & = & \begin{bmatrix} Facts.spkr = p :: F_s & : & \texttt{Poset(Prop)} \\ Facts.addr = p :: F_a & : & \texttt{Poset(Prop)} \\ QUD = \mathit{resolve}(Q, p) & : & \texttt{Poset(Question)} \end{bmatrix} \end{bmatrix}$$

Figure 7: Acknowledgement update

### 4.6 Contradiction

A contradiction arises when a proposition is currently discussed ($p?$ is topical in QUD). More precisely, this can happen when a speaker achieves to deduce $\neg p$ from a set of hypotheses ($H$) present in his own doxastic set, and when the last Move consists in this speaker asserting $\neg p$. In that case, the question of whether the assumptions made by the speaker are true becomes topical ($H?$ is pushed on the top of QUD). The question of whether $p$ is changed into the question of whether $\neg p$, but is not prioritary[16].

[15]note that if $p$ is already in $F$, then $H = p$ is an obvious witness for the *deduced* relation

[16]otherwise the process would be certainly stuck in an infinite loop: $p$ contradicted by $\neg p$ contradicted by $p$ etc. But of course this kind of phenomenon can appear to a certain extent in real world interactions...

$$\begin{bmatrix} \text{Pre} & = & \begin{bmatrix} \text{Facts} = F & : & \texttt{List(Prop)} \\ \text{QUD} = p? :: Q & : & \texttt{Poset(Question)} \\ \text{H} & : & \texttt{Set(Prop)} \\ \text{Justification-cond} & : & \mathit{subset}(H, \mathit{Facts.spkr}) \\ \text{Quality-cond} & : & \mathit{deduced}(\mathit{not}(p), \mathit{Facts.spkr}, H) \\ \text{Moves} = \mathit{Assert}(\mathit{spkr}, \mathit{addr}, \mathit{not}(p)) :: M & : & \texttt{List(LocProp)} \end{bmatrix} \\ \text{Post} & = & \begin{bmatrix} \text{QUD} = H? :: \mathit{not}(p)? :: Q & : & \texttt{Poset(Question)} \end{bmatrix} \end{bmatrix}$$

Figure 8: Contradiction update

## 4.7 Move-generating rules

Move-generating rules involve the field "Pending" which stores utterances that have not been grounded. The process of creating a Move consists in taking the last pending utterance and pushing it in the list of Moves with or without certain modifications. These modifications are basically equivalent to sub-sentential interventions: clarification request, self-correction etc.

### 4.7.1 Grounding

The grounding rule is a very simple rule that moves a locutionary proposition $p$ from Pending to the list of Moves, which means that the content of $p$ cannot be revoked. This happens when the semantic content of $p$ is fully specified, or when it is enough predictable. In particular, the intention of the speaker (the type of the speech act involved, assertion, question etc.) must be determined.

$$\begin{bmatrix} \mathit{Pre} & = & \begin{bmatrix} \mathit{Moves} = M & : & \mathit{List}(\mathit{LocProp}) \\ \mathit{Pending} = p :: P & : & \mathit{List}(\mathit{LocProp}) \end{bmatrix} \\ \mathit{Post} & = & \begin{bmatrix} \mathit{Moves} = \mathit{predict}(p) :: M & : & \mathit{List}(\mathit{LocProp}) \\ \mathit{Pending} = P & : & \mathit{List}(\mathit{LocProp}) \end{bmatrix} \end{bmatrix}$$

The mechanisms behind the function *predict* remain a bit unclear to us[17].

## 4.8 Clarification request

$$\begin{bmatrix} \text{Pre} & = & \begin{bmatrix} \text{Moves} = M & : & \texttt{List(LocProp)} \\ \text{Pending} = p :: P & : & \texttt{List(LocProp)} \end{bmatrix} \\ \text{Post} & = & \begin{bmatrix} \text{Moves} = \mathit{Ask}(\mathit{spkr}, \mathit{addr}, \mathit{CQ}(p)) :: M & : & \texttt{List(LocProp)} \\ \text{Pending} = p :: P & : & \texttt{List(LocProp)} \end{bmatrix} \end{bmatrix}$$

$CQ$ is a function from assertive propositions to questions. It focuses on one element of the proposition (an entity, or a predicate) and asks about this element. Therefore, it complies with the so-called Reprise Content Hypothesis (RCH) [?]:

A fragment reprise question queries exactly the standard semantic content of the fragment being reprised.

## 4.9 Self Repair

Self-repair occurs when the speaker realizes that what he is saying (or about to say) is not consistent anymore. This is described by the violation condition. This triggers a repair of the utterance being

[17] we could think of a simple Markov chain that would predict the next words pronounced by the speaker. These words would then be passed to one of the incremental pipelines we have seen in section 3. A confidence score for the prediction could be easily retrieved using the probabilities of the Markov chain.

produced. This repair operation should backtrack to a previous state of the locutionary proposition $p$ where there is no conflict with the facts.

$$\left[\begin{array}{lcl} Pre & = & \left[\begin{array}{lcl} Pending = p :: P & : & \texttt{List}(\texttt{LocProp}) \\ \text{Violation-cond} & : & \neg deduced(predict(p), Facts(p.spk)) \end{array}\right] \\ Post & = & \left[\begin{array}{lcl} Pending = repair(p) :: P & : & \texttt{List}(\texttt{LocProp}) \end{array}\right] \end{array}\right]$$

## 5 Conclusion

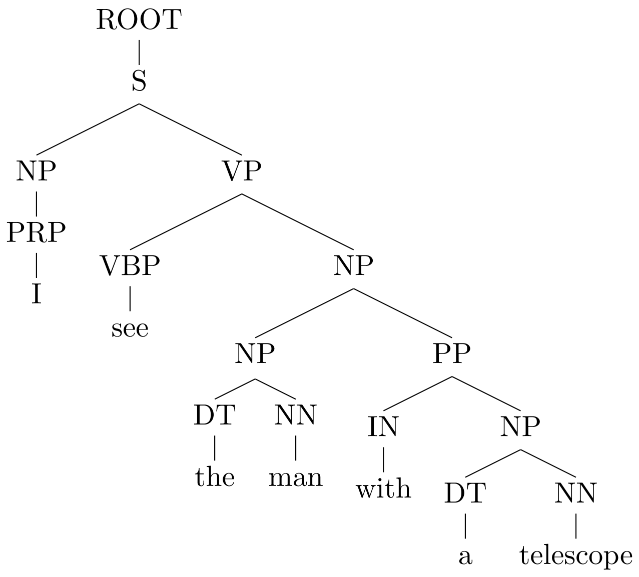
Formal semantics, and maybe even more the semantics of dialogue was not primarily our main field of study, even though the course on formal linguistics given in the MPRI was a really good introduction. Becoming familiar with all the concepts used in linguistics (and especially the non-mainstream theories like situation semantics and HPSG) took quite a long time. That is why the content of this report is mainly a review of the existing work in the field. But still we are sorry for that.
On the basis of the existing research work in parsing and dialogue modeling, we sought to improve the dynamics of dialogue by devising conversational rules that comply with empirical data and theories in pragmatics. However some aspects of the model remain a bit unclear, namely the use of a statistical layer for utterance projection. A implementation and test results would also be very valuable.

## 6 Acknowledgements

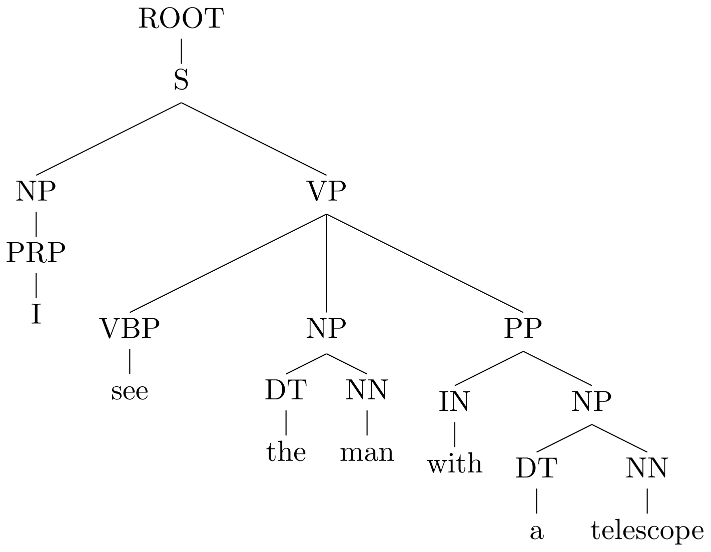
Thanks to my internship supervisor Jonathan Ginzburg who guided me in a field that I did not knew so well while allowing me considerable freedom. He gave me very relevant material to read and valuable advise for everything. Thanks also to the many researchers of Institut Jean Nicod (Philippe Schlenker, Benjamin Spector, Paul Egré, Salvador Mascarenhas) who gave me supplementary readings in formal semantics and pragmatics, and encouraged me to take part in a summer school on that subject[18]. Finally, thanks to the lecturers of the MPRI language modeling course (Philippe de Groote, Sylvain Schmitz), who gave me a great preview of what I have done in my internship.

[18]namely the Crete Summer School of Linguistics

## Appendix A



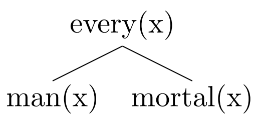
(a) the PP seen as a complement of the NP



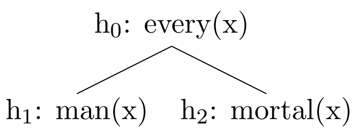
(b) the PP seen as a complement of the VP

Figure 9: The PP-attachment problem, generating two ambiguous trees (built using the online version of the Stanford Parser)

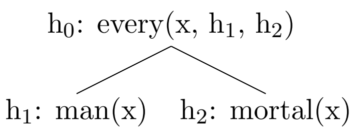
## Appendix B



(a) Original semantic tree

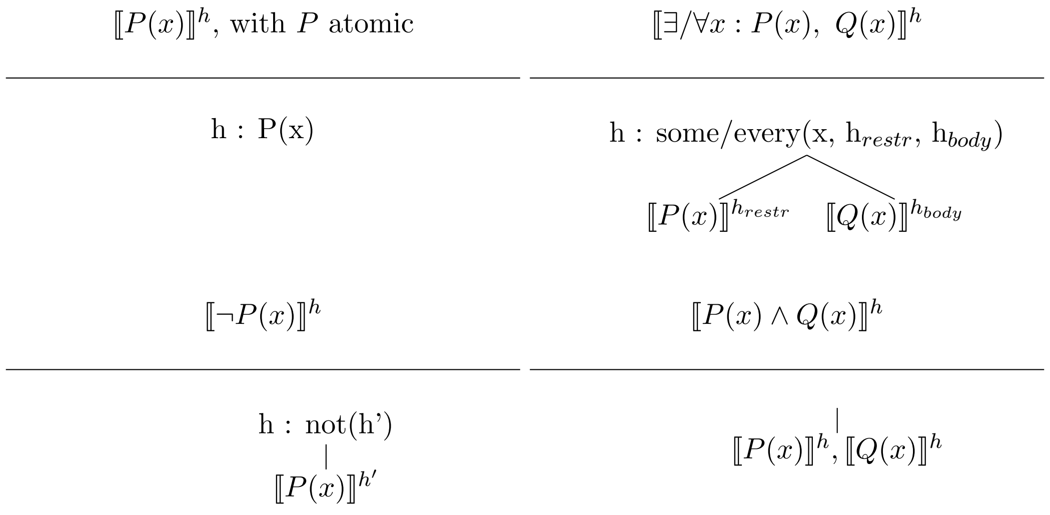


(b) Adding addresses



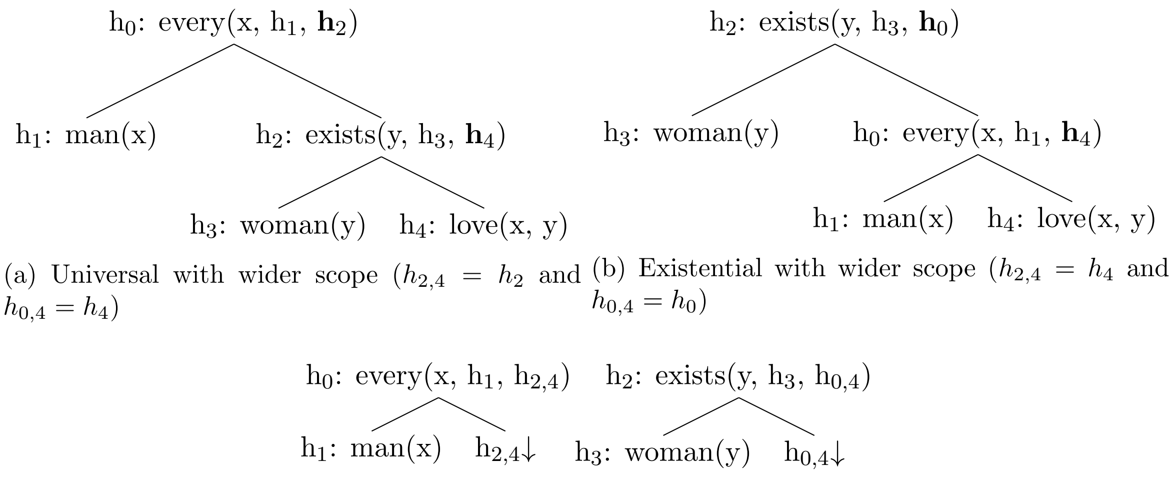
(c) Passing addresses

Figure 10: From tree structure to flat (bag) representation $\{h_0 : every(x, h_1, h_2), h_1 : man(x), h_2 : mortal(x)\}$



$[\![P(x)]\!]^h$, with $P$ atomic
h : P(x)
$[\![\exists/\forall x : P(x),\ Q(x)]\!]^h$
h : some/every(x, $h_{restr}$, $h_{body}$)
$[\![P(x)]\!]^{h_{restr}}$
$[\![Q(x)]\!]^{h_{body}}$
$[\![\neg P(x)]\!]^h$
h : not(h')
$[\![P(x)]\!]^{h'}$
$[\![P(x) \wedge Q(x)]\!]^h$
$[\![P(x)]\!]^h$, $[\![Q(x)]\!]^h$

Figure 11: Translation of FOL atomic formulae into MRS tree structures with handles



$h_0$: every(x, $h_1$, $\mathbf{h}_2$)
$h_1$: man(x)
$h_2$: exists(y, $h_3$, $\mathbf{h}_4$)
$h_3$: woman(y)
$h_4$: love(x, y)
$h_2$: exists(y, $h_3$, $\mathbf{h}_0$)
$h_3$: woman(y)
$h_0$: every(x, $h_1$, $\mathbf{h}_4$)
$h_1$: man(x)
$h_4$: love(x, y)
$h_0$: every(x, $h_1$, $h_{2,4}$)
$h_1$: man(x)
$h_{2,4}\downarrow$
$h_2$: exists(y, $h_3$, $h_{0,4}$)
$h_3$: woman(y)
$h_{0,4}\downarrow$

(a) Universal with wider scope ($h_{2,4} = h_2$ and $h_{0,4} = h_4$)

(b) Existential with wider scope ($h_{2,4} = h_4$ and $h_{0,4} = h_0$)

(c) Underspecified scope (vacuous branches signaled by $\downarrow$)

Figure 12: How handles allow for scope ambiguity (bold handles = scopal handles that can be underspecified)

## RMRS: combination rules

### Intersective combination

$$\text{LEFT DAUGHTER} = \begin{bmatrix} \mathit{Hook} & = & H \\ \mathit{Rels} & = & R_1 \\ \mathit{Cons} & = & C_1 \end{bmatrix}$$

$$\text{RIGHT DAUGHTER} = \begin{bmatrix} \mathit{Hook} & = & H \\ \mathit{Rels} & = & R_2 \\ \mathit{Cons} & = & C_2 \end{bmatrix}$$

$$\text{MOTHER} = \begin{bmatrix} \mathit{Hook} & = & H \\ \mathit{Rels} & = & R_1 + R_2 \\ \mathit{Cons} & = & C_1 + C_2 \end{bmatrix}$$

### Scopal combination

$$\text{SCOPAL DAUGHTER} = \begin{bmatrix} \mathit{Hook} & = & \begin{bmatrix} \mathit{GTop} & = & G \\ \mathit{LTop} & = & L_s \end{bmatrix} \\ \mathit{Rels} & = & R_s = R_{s'} + \begin{bmatrix} \mathit{Label} & = & L_s \\ \ldots & & \\ \mathit{Hole} & = & h \\ \ldots & & \end{bmatrix} \\ \mathit{Cons} & = & C_s \end{bmatrix}$$

$$\text{NON-SCOPAL DAUGHTER} = \begin{bmatrix} \mathit{Hook} & = & \begin{bmatrix} \mathit{GTop} & = & G \\ \mathit{LTop} & = & L_{ns} \end{bmatrix} \\ \mathit{Rels} & = & R_{ns} \\ \mathit{Cons} & = & C_{ns} \end{bmatrix}$$

$$\text{MOTHER} = \begin{bmatrix} \mathit{Hook} & = & \begin{bmatrix} \mathit{GTop} & = & G \\ \mathit{LTop} & = & L_s \end{bmatrix} \\ \mathit{Rels} & = & R_s + R_{ns} \\ \mathit{Cons} & = & C_s + C_{ns} + \{h =_{qeq} L_{ns}\} \end{bmatrix}$$

### Root condition

$$\mathit{ROOT} = \begin{bmatrix} \mathit{Hook} = \begin{bmatrix} \mathit{GTop} = G_0 \\ \mathit{LTop} = L_0 \end{bmatrix} \\ \mathit{Rels} = R \\ \mathit{Cons} = C_0 :: G_O =_{qeq} L_0 :: C_0' \end{bmatrix}$$

Figure 13: Merge rules for RMRS with TTR

## **Proof:** $\lambda R.R \boxed{\wedge} Q$ **produces a subtype of Q**

In this section we will prove that if a record type R is such that $\exists P,\ R = P\boxed{\wedge}Q$, then $R \leq: Q$. The subtyping relation $R \leq: Q$ means that $labels(Q) \subseteq labels(R)$ and:

$$\forall l \in labels(Q) \implies \begin{cases} l :_Q T_s & \implies \quad l :_R T_s \\ l :_Q T_r \wedge l :_R T_r' & \implies \quad T_r' \leq: T_r \end{cases}$$

Where $T_s$ is a simple type, and $T_r, T_r'$ are record types.
We prove this result by strong induction on the maximal level of record type embedding in R.
<u>Base case:</u> no embedding (i.e. all labels have simple types)

$$R = P \boxed{\wedge} Q = \begin{bmatrix} l_1 & : & T_1 \\ \dots & & \\ l_n & : & T_n \end{bmatrix}$$

First, $labels(Q) \subseteq labels(R)$. Indeed, let $l \in labels(Q)$

- if $l \notin labels(P)$, then rule $\boxed{2}$ ensures that $l$ is copied in R
- in $l \in labels(P)$ then rule $\boxed{3.1}$ ensures that $l$ is copied in R

Now, let $l \in labels(Q)$ and suppose that $l :_Q T_s$. Let us show that $l :_R T_s$.

- if $l \notin labels(P)$, then rule $\boxed{2}$ ensures that $l$ is copied in R, so $l :_R T_s$
- if $l \in labels(P)$, then rule $\boxed{3.1}$ ensures that Q's version of $l$ in copied in R, so $l :_R T_s$.

That proves our property for the base case.
<u>Inductive case:</u> suppose the property is verified for all level of embedding in $[0; n]$. Let us take R with a maximal level of embedding $n+1$:

$$R = P \boxed{\wedge} Q = \begin{bmatrix} l_1 & : & T_1 \\ \dots & & \\ l_n & : & T_n \end{bmatrix}$$

For all $i \in [1; n]$, the maximal level of embedding of $T_i$ is $n$, so $\exists P',\ T_i = P' \boxed{\wedge} Q' \implies T_i \leq: Q'$ (induction hypothesis).
Let $l \in labels(Q)$:

- if $l \notin labels(P)$, then rule $\boxed{2}$ ensures that $l$ is copied in R, so $l \in labels(R)$, and the copy operation ensures that $l :_Q T \iff l :_R T$
- if $l \in labels(P)$ and $l :_Q T_s$, then rule $\boxed{3.1}$ ensures that Q's version of $l$ is copied in R, so $l \in labels(R)$ and $l :_Q T \iff l :_R T$
- if $l \in labels(P)$, $l :_Q T_r$ and $l :_P T_r''$, then rule $\boxed{3.2}$ ensures that $l$ is copied in R, so $l \in labels(R)$, and that $l :_R T_r'' \boxed{\wedge} T_r$. Since $T_r'' \boxed{\wedge} T_r$ has a level of embedding that lies in $[0; n]$, we can apply the induction hypothesis and conclude that $T_r'' \boxed{\wedge} T_r \triangleq T_r' \leq: T_r$

That concludes the proof.