Incremental dialogue modeling

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

August 20, 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 modern dialogue management systems, where a human speaker interacts with a on-human entity (for instance the TRAINS project[1]).

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 [Du Bois et al., 2005], 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.

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 [3]. 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 we share a common goal; we want to exchange informations. The participants usually share a certain amount of 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 [4]. Conversely, information can consists of queries using inquisitive sentences [5].

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 [Poesio and Traum, 1997, Eshghi et al., 2015]. This is crucial for the dialogue not to turn into a monologue.

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] as we can see in the first intervention of LYNNE

[4] most of the time, the notion coincides with declarative sentences

[5] most 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 [Frank, 2016]. 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 2nd turn).

Second, there is a large amount of psycholinguistic evidence[6] 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" [Sag and Wasow, 2015]:

"The horse raced past the barn fell." [Bever, 2013]

---

[6]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 [Tanenhaus et al., 1996]. In particular it has also been shown that the processing of verbs introduces bias towards the most probable type of object [Altmann and Kamide, 1999]. [Traxler et al., 1997] showed that interclausal relationships (causation, diagnostic) are processed before the end of the last clause. Using another technique, namely event-related brain potentials (ERP), [Schlesewsky and Bornkessel, 2004] 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 [Sag and Wasow, 2015], for instance:

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

The box is in the pen$_a rea$

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 informations about the most probable meaning (here, the word "box" allows to opt for pen$_t ool$). 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 [Schlangen and Skantze, 2009]. 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 [Hough et al., 2015], after [Milward, 1991]. 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 5, 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 [Frege, 1997], 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
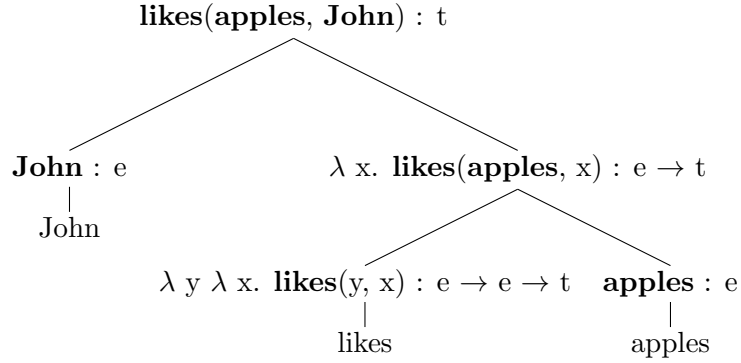

Figure 1: Simple semantic computation (bold text for semantic entities)

types of ambiguity, in particular with regards to quantifier scope [Milward and Cooper, 1994]:

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

As we will see later on (in Section 3.3 in particular), this can be "resolved" – or at least, accounted for – through underspecification

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. The core of the theory is strongly inspired by Head-driven Phrase Structure Grammar (HPSG) [Sag et al., 2003] and Type Theory with Records (TTR) [Betarte and Tasistro, 1998, Cooper, 2005]. For the purpose of *dialogue* modeling, elements of situation semantics [Jon Barwise, 1998] 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) [Betarte and Tasistro, 1998], which allows to use relatively elaborate data structures whose general form is :

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

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 :

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

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:

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

Where:

- $labels(X)$ 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 [Cooper, 2005].

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) [Sag et al., 2003] 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[7].

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);

---

[7]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.

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

Figure 2: Example of an HPSG entry (feature structure) for a verb [Sag et al., 2003]

- 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 (SBCG [Sag and Wasow, 2015]...) 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 90's [Jon Barwise, 1998], 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 [Stojanovic, 2011]. 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 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 modeled as:

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

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

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

8

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 [Jon Barwise, 1998]:

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

2.3 Dialogue GameBoards

The formal features of dialogue are highly inspired from [Ginzburg, 2012, Ginzburg et al., 2017], although some of them are slightly modified. Records and record types will be used to model the various features of a dialogue, pragmatic ones (who speaks, who listens), ontological ones (what do the participants know), semantic ones (what has been said, under which modality), syntactic ones (how things are said, following which grammar rules), lexical ones (which words have been used), and phonetic ones (which sound have been pronounced). This is summarized in the following record type, called a Dialogue Game Board (DGB) Type:

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

There is one DGB per participant, that is why DGB involve "private content" indicated by an asterisk. In the following subsections we will go over the details of the different types introduced in `DGBType` : the `IllocProp` type, the `SemRec` type and the `Clause` type.

2.3.1 The `IllocProp` type

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

$$\texttt{IllocProp} = \begin{bmatrix} \text{Situation} & : & \texttt{Sit} \\ \text{Speaker} & : & \texttt{Ind} \\ \text{Addressee} & : & \texttt{Ind} \\ \text{R} & : & \texttt{SpeechAct} \\ \text{Content} & : & \texttt{SemRec} \\ \text{Situation type} & : & R(\text{Speaker}, \text{Addressee}, \text{Content}) \end{bmatrix}.$$

The field named Situation type binds a certain utterance to a certain situation; this principle is in fact in the centre of austinian semantics. The type called `SpeechAct` is the supertype for illocutionary speech acts as defined by Searle : `Assert` (for "Assertives"), `Direct` (for "Directives"), `Com` (for "Commissives"), `Expr` (for "Expressives"), `Decl` (for "Declarations"). We could imagine many subtypes for each of these illocutionary speech acts.

2.3.2 The `SemRec` type

The type of semantic objects is (`Lambda`)`SemRec`. It is based on Minimal Recursion Semantics (MRS) [Copestake et al., 2005]. MRS is very attractive, because it models expressions of first order logic with generalized quantifiers, and allows for underspecification (scope or reference).

Figure 7 shows how FOL-GQ is translated into MRS decorated trees[8] that capture scope informations. Scope ambiguity can be introduced by simply "disconnecting" the right branch of quantified

---

[8]the interpretation fucntion $[\![.]\!]^h$ turns a FOL expression into a tree and gives the root the label h

nodes. Such a vacuous branch is called a *hole*.

Since each node's position in the tree is encoded in its *handles* (its own handle, also called *label*, and the handles it points to), there is no need to keep the tree representation, and it becomes possible to pack all the nodes together in a record, as represented in Figure **??**. Note that the record of this figure has three main fields :

- Rels sums up the informations available in a MRS tree structure. It is a bag of nodes, each node coming with a label and a predicate / binder that is applied to variables (bound or free) or to other relations (represented by their label).

- Hook is used to combine properly two MRS structures and to deal with reference resolution. The local top of an expression is the predicate that has the widest scope.

- Cons restricts the possible combinations (i.e. how to "plug" back in holes by equating handles)in the case of underspecified MRS. Each constraint is represented by a qeq relation between an "upper" argument handle h and a "lower" label l, connected by a chain of nodes.

2.3.3 The `Clause` type

The field called Pending contains the clause that is about to be produced, and this field will be crucial for incremental modeling. The content of the Pending field is of type `Clause`, which is a rich type record inspired by HPSG's *words* or *phrases* (see [Sag et al., 2003]). It sums up all the linguistic features of the clause at stake, from graphemes to sememes.

$$\text{Clause} \begin{bmatrix} \text{Phon} & : & \text{Str} \\ \text{Syn} & : & \text{SynCat} \\ \text{Sem} & : & \text{LambdaSemRec} \end{bmatrix}$$

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 [Schlangen and Skantze, 2009] for general incremental systems.

3.1 A general model for incremental processing

A very abstract, general framework for incremental processing in dialogue has been developed by [Schlangen and Skantze, 2009]. It is inspired by Young token passing architecture. This model is independent of the formalism used for syntactic/semantic modeling, and can has many applications in dialogue, from speech recognition to semantic interpretation.

The key concept of this model is the concept of incremental unit (or IU). An IU refers to the smallest possible chunk of information that can be processed independently. 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...). In our case, IUs will basically 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?

Furthermore, 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 path being possible). One can also think of other non-linear dependency constraints (modification, specification, θ-roles...);

- vertical relations: these are hierarchical relations, between IUs of different linguistic levels. This includes the *grounded* relation, which links an higher order IUs to the lower order IUs that justify its production.

Note also that the third point concerns modules, which are processing units. A module consumes 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 **??**
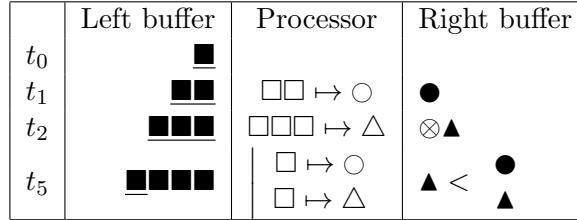

Figure 3: Schematic of a module (incremental processing unit) with delayed, batch computation (t_1), revision (t_2), and production of alternatives (t_3)

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 [Kempson et al., 2001] is a model that incorporates syntactic and semantic parsing. Indeed, DS parsing generates the syntactic tree and decorates it with typed λ-terms, in one pass. In DS-TTR [Hough, 2011, Hough et al., 2015], the λ-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 [Copestake et al., 2005] 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[9]. They denote node addresses in the original tree; and if two EPs have the same handle, they must be conjuncted. 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.

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

Figure 4: Main types in RMRS

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 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 [10] may indeed generate various syntactic trees. This is exemplified in Figure 8. This mechanism is also used for the modeling of Wh-questions, which is known to be a rather complex problem (see for instance [Egg, 1998] for a quite complete account of questions and their interactions with quantifiers).

---

[9]thus this structure is obtained after syntactic parsing

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

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).

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.

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 R is a subtype of P, which means that R contains at least the labels specified in P, and for these labels, their types in R must be subtypes of their types in P. 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 Question

[Ginzburg et al., 2017] question q should not be resolvable with facts

$$
\begin{bmatrix}
Pre & = & \begin{bmatrix} QUD = Q & & : \\ Moves = Ask(spkr, addr, q) :: M & : & IllocProp \end{bmatrix} \\
Post & = & \begin{bmatrix} QUD = q :: Q & : & \end{bmatrix}
\end{bmatrix}
$$

4.3 Assertion

$$
\begin{bmatrix}
Pre & = & \begin{bmatrix} QUD = Q & & : \\ Moves = Assert(spkr, addr, p) :: M & : & IllocProp \end{bmatrix} \\
Post & = & \begin{bmatrix} QUD = p? :: Q & : & \end{bmatrix}
\end{bmatrix}
$$

4.4 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.

ETH:	... But,
	you know,
	... like,
	... she know=s,
	so much about it.
	You know?
	... I said probably that one day the –
	... The teacher and you both just had a bad day,
	and it came off <X on X>.
SHERRY:	... Yeah.

Table 8: SBC031

$$
\begin{bmatrix}
Pre & = & \begin{bmatrix} Facts = F \\ QUD = Q' :: p? :: Q & & : \\ Moves = Ack(spkr, addr, p) :: M & : & IllocProp \end{bmatrix} \\
Post & = & \begin{bmatrix} Facts = p :: F \\ QUD = resolve(Q' :: Q, p) & : & \end{bmatrix}
\end{bmatrix}
$$

4.5 Contradiction

$$
\begin{bmatrix}
Pre & = & \begin{bmatrix} Facts = F \\ QUD = Q' :: p? :: Q & & : \\ Moves = Assert(spkr, addr, not(p)) :: M & : & IllocProp \end{bmatrix} \\
Post & = & \begin{bmatrix} QUD = Q' :: not(p)? :: Q & : & \end{bmatrix}
\end{bmatrix}
$$

4.6 Grounding

$$
\left[
\begin{array}{lll}
Pre & = &
\left[
\begin{array}{lll}
Moves = M & : & List(LocProp) \\
Pending = p :: P & : & List(LocProp)
\end{array}
\right] \\
Post & = &
\left[
\begin{array}{lll}
Moves = p :: M & : & List(LocProp) \\
Pending = P & : & List(LocProp)
\end{array}
\right]
\end{array}
\right]
$$

4.7 Clarification request

$$
\left[
\begin{array}{lll}
Pre & = &
\left[
\begin{array}{lll}
Moves = M & : & List(LocProp) \\
Pending = p :: P & : & List(LocProp)
\end{array}
\right] \\
Post & = &
\left[
\begin{array}{lll}
Moves = Ask(spkr, addr, CQ(p)) :: M & : & List(LocProp) \\
Pending = p :: P & : & List(LocProp)
\end{array}
\right]
\end{array}
\right]
$$

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:

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

Basically, CQ will p abstract over a variable or a predicate. For example:

URT:	... Do you remember the uh,
	... X program on TV,
	.. about hunter virus,
KITTY:	... About what virus?
CURT:	.. Hunter virus,
KITTY:	... Hutter.
	... What's that about.
	I don't re- I don't remember.

Table 9: SBC042

4.8 Self Repair

$$
\left[
\begin{array}{lll}
Pre & = & \left[\ Pending = p :: P & : & List(LocProp)\ \right] \\
Post & = & \left[\ Pending = repair(p) :: P & : & List(LocProp)\ \right]
\end{array}
\right]
$$

ANCE:	For the most part,
	... I mean I was able to k- –
	I think I was able to keep up with that stuff,
RANDY:	With the ARTS.
LANCE:	.. Yeah.

Table 10: SBC022

4.9 Cooperation

$$
\left[
\begin{array}{ll}
Pre & = & \left[
\begin{array}{l}
Speaker = i \\
Addressee = j \\
Pending = p :: P \quad : \quad List(LocProp)
\end{array}
\right] \\
Post & = & \left[
\begin{array}{l}
Speaker = j \\
Addressee = i \\
Pending = predict_j(p) :: P \quad : \quad List(LocProp)
\end{array}
\right]
\end{array}
\right]
$$

4.10 Ellipsis

References

[Altmann and Kamide, 1999] Altmann, G. T. and Kamide, Y. (1999). Incremental interpretation at verbs: restricting the domain of subsequent reference. Cognition, 73(3):247–264.

[Betarte and Tasistro, 1998] Betarte, G. and Tasistro, A. (1998). Extension of martin-löf's type theory with record types and subtyping.

[Bever, 2013] Bever, T. G. (2013). The cognitive basis for linguistic structures1. In Language Down the Garden Path, pages 1–80. Oxford University Press.

[Cooper, 2005] Cooper, R. (2005). Records and record types in semantic theory. Journal of Logic and Computation, 15(2):99–112.

[Copestake et al., 2005] Copestake, A., Flickinger, D., Pollard, C., and Sag, I. A. (2005). Minimal recursion semantics: An introduction. Research on Language and Computation, 3(2):281–332.

[Du Bois et al., 2005] Du Bois, J. W., Chafe, W. L., Meyer, C., Thompson, S. A., Englebretson, R., and Martey, N. (2000-2005). Santa Barbara corpus of spoken American English, Parts 1-4. Philadelphia: Linguistic Data Consortium.

[Egg, 1998] Egg, M. (1998). Wh-questions in underspecified minimal recursion semantics. Journal of Semantics, 15(1):37–82.

[Eshghi et al., 2015] Eshghi, A., Howes, C., Gregoromichelaki, E., Hough, J., and Purver, M. (2015). Feedback in conversation as incremental semantic update. In Proceedings of the 11th International Conference on Computational Semantics, pages 261–271.

[Frank, 2016] Frank, M. C. (2016). Rational speech act models of pragmatic reasoning in reference games.

[Frege, 1997] Frege, G. (1997). Begriffsschrift. a formula language of pure thought modelled on that of arithmetic. In Beaney, M., editor, The Frege Reader. Blackwell.

[Ginzburg, 2012] Ginzburg, J. (2012). The Interactive Stance. Oxford linguistics. OUP Oxford.

[Ginzburg, pear] Ginzburg, J. (To appear). The semantics of dialogue. In Aloni, M. and Dekker, P., editors, Cambridge Handbook of Formal Semantics. Cambridge University Press.

[Ginzburg et al., 2017] Ginzburg, J., Cooper, R., Hough, J., and Schlangen, D. (2017). Incrementality and clarification/sluicing potential. In Proceedings of Sinn und Bedeutung 2016.

[Hough, 2011] Hough, J. (2011). Incremental semantics driven natural language generation with self-repairing capability. In RANLP Student Research Workshop.

[Hough et al., 2015] Hough, J., Kennington, C., Schlangen, D., and Ginzburg, J. (2015). Incremental semantics for dialogue processing: Requirements, and a comparison of two approaches. In Proceedings of the 11th International Conference on Computational Semantics, pages 206–216. Association for Computational Linguistics.

[Jon Barwise, 1998] Jon Barwise, J. P. (1998). Situations and Attitudes. The David Hume Series. CSLI Publ.

[Kempson et al., 2001] Kempson, R., Meyer-Viol, W., and Gabbay, D. (2001). Dynamic syntax : the flow of language understanding. Blackwell Publishers.

[Milward, 1991] Milward, D. (1991). Axiomatic Grammar, Non-Constituent Coordination and Incremental Interpretation. PhD thesis, University of Cambridge.

[Milward and Cooper, 1994] Milward, D. and Cooper, R. (1994). Incremental interpretation: Applications, theory, and relationship to dynamic semantics. In Proceedings of the 15th Conference on Computational Linguistics - Volume 2, COLING '94, pages 748–754, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Poesio and Traum, 1997] Poesio, M. and Traum, D. R. (1997). Conversational actions and discourse situations. Computational Intelligence, 13(3):309–347.

[Sag and Wasow, 2015] Sag, I. A. and Wasow, T. (2015). Flexible processing and the design of grammar. Journal of Psycholinguistic Research, 44:47–63.

[Sag et al., 2003] Sag, I. A., Wasow, T., and Bender, E. M. (2003). Syntactic Theory: A Formal Introduction. CSLI lecture notes. Center for the study of language and information Stanford, California, 2 edition.

[Schlangen and Skantze, 2009] Schlangen, D. and Skantze, G. (2009). A general, abstract model of incremental dialogue processing. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, EACL '09, pages 710–718, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Schlesewsky and Bornkessel, 2004] Schlesewsky, M. and Bornkessel, I. (2004). On incremental interpretation: degrees of meaning accessed during sentence comprehension. Lingua, 114(9-10):1213–1234.

[Stojanovic, 2011] Stojanovic, I. (2011). Situation Semantics. chapter for the volume Introduction to the Philosophy of John Perry, ed. by Raphael van Riel and Albert Newen.

[Tanenhaus et al., 1996] Tanenhaus, M. K., Spivey-Knowlton, M. J., Eberhard, K. M., and Sedivy, J. C. (1996). Using eye movements to study spoken language comprehension: Evidence for visually mediated incremental interpretation. In Inui, T. and McClelland, J. L., editors, Information integration in perception and communication, volume 16 of Attention and performance, pages 457–478. The MIT Press, Cambridge, MA.

[Traxler et al., 1997] Traxler, M. J., Bybee, M. D., and Pickering, M. J. (1997). Influence of connectives on language comprehension: Eye tracking evidence for incremental interpretation. The Quarterly Journal of Experimental Psychology Section A, 50(3):481–497.

Appendix A Figures of Section 1

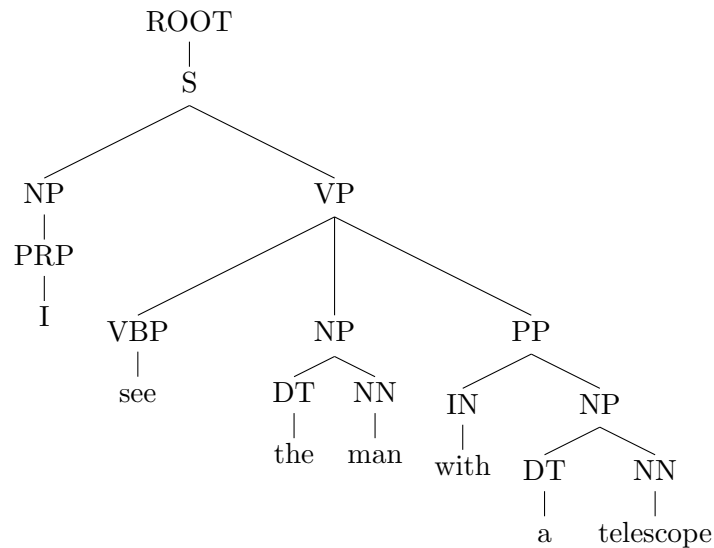## A.1 The attachment problem : a case of structural ambiguity
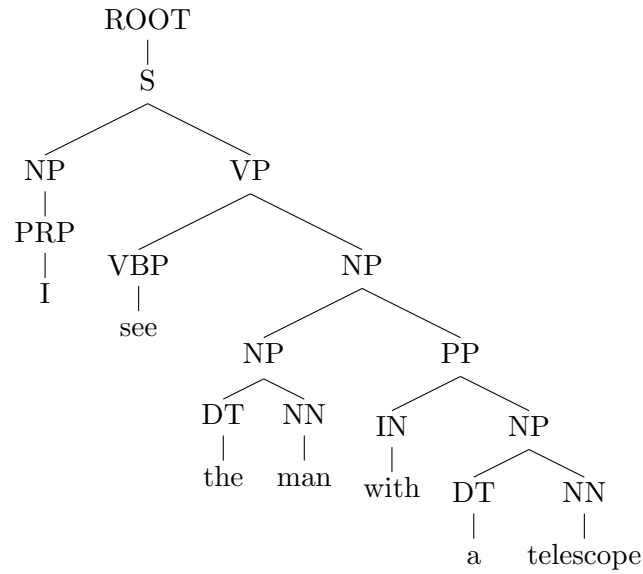

(a) the PP seen as a complement of the NP

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

Figure 5: The PP-attachment problem, generating two ambiguous trees (built using the online version of the Stanford Parser)

$$\text{every}(x)$$
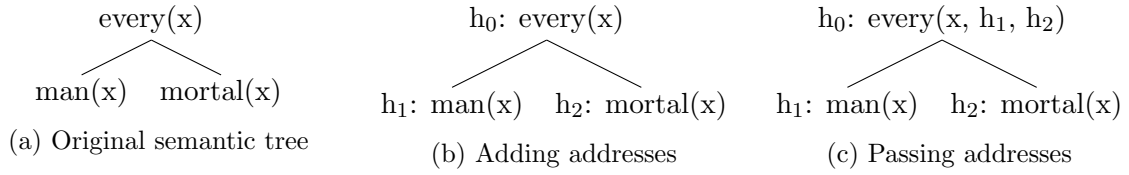$$\text{man}(x) \quad \text{mortal}(x)$$
(a) Original semantic tree

$$h_0: \text{every}(x)$$
$$h_1: \text{man}(x) \quad h_2: \text{mortal}(x)$$
(b) Adding addresses

$$h_0: \text{every}(x, h_1, h_2)$$
$$h_1: \text{man}(x) \quad h_2: \text{mortal}(x)$$
(c) Passing addresses

Figure 6: From tree structure to flat representation $\{h_0 : every(x, h_1, h_2), h_1 : man(x), h_2 : mortal(x)\}$

$\llbracket P(x) \rrbracket^h$, with P atomic
_____

$h : P(x)$

$\llbracket \exists/\forall x : P(x),\ Q(x) \rrbracket^h$
_____

$h : \text{some/every}(x, h_{restr}, h_{body})$
$$\llbracket P(x) \rrbracket^{h_{restr}} \quad \llbracket Q(x) \rrbracket^{h_{body}}$$

$\llbracket \neg P(x) \rrbracket^h$
_____

$h : \text{not}(h')$
$$|$$
$\llbracket P(x) \rrbracket^{h'}$

$\llbracket P(x) \wedge Q(x) \rrbracket^h$
_____

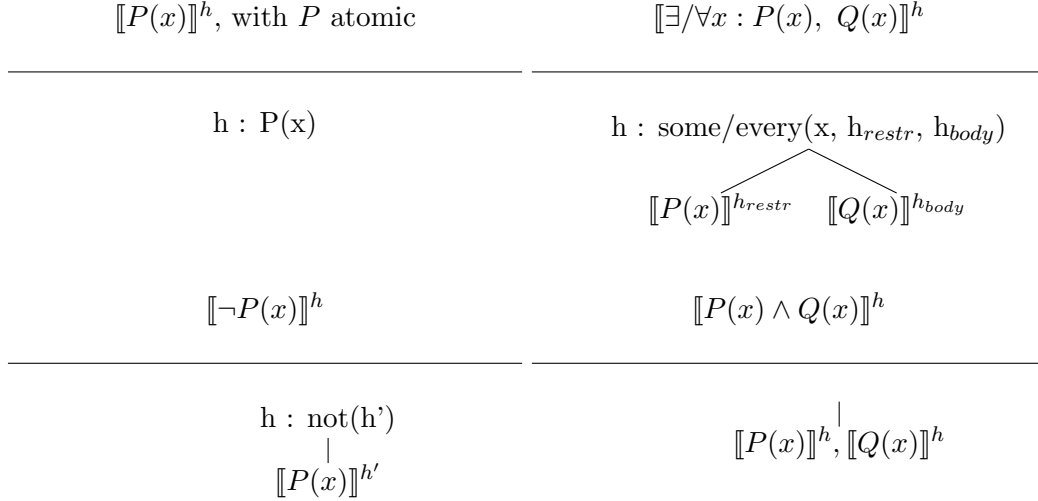$\llbracket P(x) \rrbracket^h, \llbracket Q(x) \rrbracket^h$

Figure 7: Translation of FOL atomic formulae into MRS tree structures with handles

Appendix B Figures of Section 3

B.1 RMRS: from tree to bag

B.2 RMRS : from FOL to RMRS trees

B.3 RMRS: underspecification and scope ambiguity

B.4 RMRS: combination rules

B.4.1 Intersective combination

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

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

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

(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 $h_{0,4} = h_0$)

h_0: every(x, h_1, $h_{2,4}$) h_2: exists(y, h_3, $h_{0,4}$)

h_1: man(x) $h_{2,4}\downarrow$ h_3: woman(y) $h_{0,4}\downarrow$

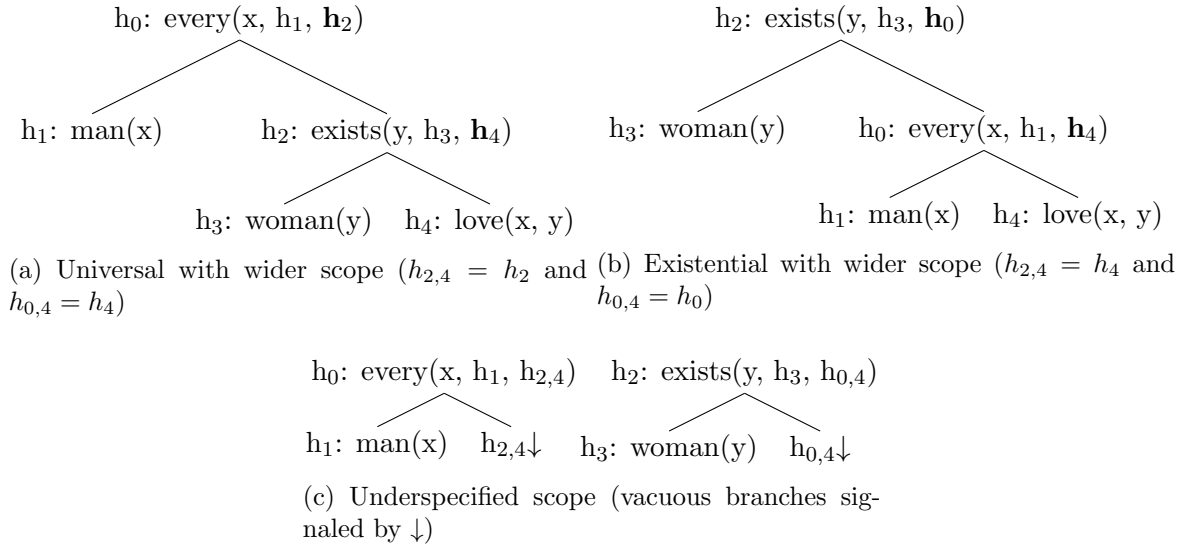(c) Underspecified scope (vacuous branches signaled by $\downarrow$)

Figure 8: How handles allow for scope ambiguity (bold handles = scopal handles that can be underspecified)

B.4.2 Scopal combination

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

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

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

B.4.3 Root condition

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

B.5 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:

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

21

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.