Expressing Discourse Dynamics Through Continuations

Thesis Summary
Ekaterina Lebedeva

1 Introduction

This thesis focuses on formal semantics of natural language and presents a framework that facilitates the task of compositionally obtaining the meaning of natural language expressions with mathematically precise methods.

The pioneer work of Richard Montague [13, 12, 14] was limited to single and relatively simple sentences, and could not handle complex phenomena such as, for example, cross-sentential and donkey anaphora. In order to cope with the limitations of Montague’s framework, various theories, aimed specifically on discourse dynamics were developed, including discourse representation theory (DRT) [10, 11], file change semantics (FCS) [8, 9], dynamic predicate logic (DPL) [7], dynamic montague grammar (DMG) [6] and their extensions [2, 1, 15, 5]. Although these frameworks definitely helped in understanding many problems related to discourse dynamics, they are not completely satisfactory. Some of them deviate from Montague’s requirement of compositionality; while others attempt to stay within Montague’s theory but then either use non-standard logics or introduce additional syntactic structures into their logical languages, and thus compromise their simplicity and elegance.

In contrast, the framework presented in this thesis satisfies the principle of compositionality in a simple and elegant way, by being as parsimonious as possible: completely new formalisms or extensions of existing formalisms with ever more complex constructions to fit particular linguistic phenomena have been avoided; instead, the framework handles these linguistic phenomena using only basic and well-established formalisms, such as simply typed $\lambda$-calculus and classical logic. The goal has always been to do more with less. This required a careful investigation of how discourse dynamics could be expressed with standard mathematical tools [4], which later culminated in the development of a dynamic logic based on the notion of continuation.

2 Continuation-based Dynamic Logic

Natural language dynamics poses numerous challenges for formalizing natural language semantics in a truly compositional way. The difficulty lies in the fact that meaning is a function of context, i.e. of some knowledge previously obtained. The existing approaches fail to explicitly assign this context dependence to lexical items. This either undermines the compositionality of the approach (as in the case of the original DRT) or prevents the manipulation of the context (as in the case of DMG), which is necessary, for example, for handling presuppositions and conversational implicatures.

The framework that this dissertation develops not only specially focuses on the explicit representation of context, but also defines the meanings of natural languages constituents, including lexical items, in a way that they are functions of context when necessary. Another advantage of the framework is that the content of expressions is separated from the context. This makes it flexible to potentially handle pragmatic phenomena.

The basic idea of the framework originated in [4], where a simplified framework $G_0$ is used for handling cross-sentential and donkey anaphora. Section 2.1 sketches the main principles $G_0$ as they are necessary for understanding contributions of the thesis summarized subsequently.
2.1 Framework \( G_0 \): Initial Approach

As Montague’s semantics, \( G_0 \) is defined on Church’s simple type theory \([3]\), having two atomic types: \( \mathit{t} \), the type for individuals; and \( \mathit{o} \), the type for propositions. However, it has an additional type parameter \( \gamma \) and context is defined as a term of this type. Natural language sentences are then defined as functions of the context.

The context type \( \gamma \) is a parameter that can define any complex type. Therefore, there is no restriction on the representation of context. On the one hand, one can define it as a simple structure focusing on a particular phenomenon. On the other hand, since the abstract representation of context can become more sophisticated as more complex phenomena (e.g. presuppositions and conversational implicatures) are taken into account, the context can be easily elaborated without affecting the core of the framework.

A sentence can have a potential to change (or update) the context. The updated context has to be passed as an argument to the meaning of the subsequent sentence. This is achieved compositionally by using the notion of continuation \([16]\) from programming languages: the meaning of a sentence is not only a function of context, but also a function of a continuation with respect to the computation of the meaning of the whole discourse. Within the body of the term standing for the meaning of a sentence, the continuation is given the possibly updated context and returns a proposition. Therefore, the continuation has type \( (\gamma \rightarrow \mathit{o}) \).

Thus, a sentence is dynamically interpreted as a function that takes a context \( c \) of type \( \gamma \) and a continuation \( \phi \) of type \( (\gamma \rightarrow \mathit{o}) \) and returns a proposition. Type \( (\gamma \rightarrow (\gamma \rightarrow \mathit{o}) \rightarrow \mathit{o}) \) is, therefore, defined to be the type of a dynamic proposition.

To cope with anaphora, the context has to be accessed. In \( G_0 \) this is accomplished with a special function \( \mathit{sel} \) of type \( (\gamma \rightarrow \mathit{t}) \) that takes a context and returns an individual from this context. The function \( \mathit{sel} \) is assumed to implement an anaphora resolution algorithm and to work as an oracle always retrieving the correct antecedent. Due to this technique, lexical interpretations of pronouns could be expressed in \( G_0 \) and donkey anaphora could be compositionally handled. However, for a framework to be more realistic, \( \mathit{sel} \) should take not only the context as an argument but also a property corresponding to the descriptive content of the required individual. Moreover, interpretations in \( G_0 \) are lengthy, difficult to understand and not systematically constructed. Refining the approach in a modular way by defining a systematic translation from static to dynamic interpretations was necessary.

2.2 Framework \( GL \)

Framework \( GL \), developed in the thesis, elaborates \( G_0 \) by taking into account that context contains some knowledge (such as common knowledge and knowledge learned during the process of computation of the meaning). The translation from static terms to their dynamic equivalents is defined in a modular way by employing a continuation-based logical language. Importantly, \( GL \) defines a systematic translation of non-logical constants as well. This is not trivial, as non-logical constants have various types.

**Definition 1.** [Types] The set \( T \) of types of the continuation-based dynamic logic is defined inductively as follows:

- Atomic types: \( \mathit{t} \in T \) (static individual)
  \( \mathit{o} \in T \) (static proposition)
  \( \gamma \in T \) (context)

- Complex types: \( \alpha, \beta \in T \implies (\alpha \rightarrow \beta) \in T \)

**Definition 2.** [Typing rules] A statement \( t : \delta \) is derivable from the basis \( \Delta \), i.e. \( \Delta \vdash t : \delta \), if \( \Delta \vdash t : \delta \) can be produced using the following rules:
is a function

\[ \Gamma \vdash \exists : (t \to o) \to o \]

of type \((\alpha \to x : \alpha)\) axiom

\[ \Gamma \vdash x : o \to o \]

axiom

\[ \Gamma \vdash \land : o \to o \to o \]

axiom

\[ \Gamma \vdash \exists : (t \to o) \to o \]

axiom

\[ \Gamma \vdash \text{sel} : (t \to o) \to \gamma \to t \]

axiom

\[ \Gamma \vdash \text{upd} : o \to \gamma \to \gamma \]

axiom

\[ \Gamma \vdash c_{iv} : t \to o \]

axiom

\[ \Gamma \vdash c_{iv} : t \to t \to o \]

axiom

\[ \Gamma \vdash c_{n} : t \to o \]

axiom

\[ \Gamma \vdash c_N : (t \to o) \to o \]

axiom

\[ \Gamma \vdash c_{rp} : (((t \to o) \to o) \to o) \to (t \to o) \to (t \to o) \]

axiom

\[ \Gamma \vdash v : \alpha \to \beta \]

axiom

\[ \Gamma \vdash u : \alpha \to \beta \]

application

\[ \Gamma, x : \alpha \vdash v : \beta \]

abstraction

\[ \Gamma, x : \alpha \vdash \lambda x.v : \alpha \to \beta \]

abstraction

where \(c_{iv}, c_{iv}, c_{n}, c_{n}\) and \(c_{rp}\) are constants standing for transitive and intransitive verbs, nouns, noun phrases and relative pronouns respectively. Typing rules for other syntactic categories can be added analogously.

**Definition 3.** [Dynamization of types] Let \(t\) and \(o\) be atomic types, \(\gamma\) be a type parameter, \(\alpha\) and \(\beta\) be any types. Then the types are **dynamized** in the following way:

\[
\begin{align*}
\Gamma & \vdash \tau \equiv \gamma \to t \\
\Gamma & \vdash \sigma \equiv \gamma \to (\gamma \to o) \\
\Gamma & \vdash \alpha \to \beta \overset{\alpha \to \beta}{\equiv} \alpha \to \beta \\
\Gamma & \vdash \psi \overset{\psi}{\equiv} \psi
\end{align*}
\]

Thus, additionally to parametrizing a proposition of type \(o\) with a context of type \(\gamma\) and a continuation of type \((\gamma \to o)\), as in \(G_0\), Definition 3 parametrizes terms of type \(t\) with a context. This is an important novelty of GL, as it makes it possible to give direct (not type-raised) interpretations to, for example, pronouns in a way that these interpretations explicitly encode their anaphoric nature.

Definition 5 below formalizes a systematic translation of every term \(t\) to its dynamic equivalent \(\tau\). The translation is relatively straightforward for all kinds of terms except non-logical constants. Non-logical constants have various types, therefore their dynamization should be defined in a special way. This is accomplished with dynamization \(D_\delta\) and reading \(S_\delta\) functions defined in 4. The dynamization function transforms static terms into equivalent dynamic terms. The reading function transforms dynamic terms (obtained via \(D_\delta\)) into original static terms. Note that the dynamization function and the reading functions are mutually dependent.

**Definition 4.** [Dynamization and reading functions] **Dynamization function** is a function \(D_\delta\) that has type \(((\gamma \to \delta) \to \delta)\) and is inductively defined on the type \(\delta\) as follows:

\[
\begin{align*}
D_1[a] & \overset{1a}{=} a \\
D_o[P] & \overset{1b}{=} \lambda e.\phi.Pe \land \phi (\text{upd}(Pe,e)) \\
D_{\alpha \to \beta}[f] & \overset{1c}{=} \lambda a.D_\beta[\lambda e.\phi.e.S_\alpha[a,e]]
\end{align*}
\]

**Reading function** is a function \(S_\delta\) that has type \((\delta \to \gamma \to \delta)\) and is inductively defined on the type \(\delta\) as follows:

\[
\begin{align*}
S_1[a,e] & \overset{2a}{=} a e \\
S_o[P.e] & \overset{2b}{=} Pe(\lambda e.\top) \\
S_{\alpha \to \beta}[f.e] & \overset{2c}{=} \lambda a.S_\beta[\lambda e.a],e]
\end{align*}
\]

3
Definition 5. [Dynamization of $\lambda$-terms]
A term $t$ of type $\delta$ is dynamized into a term $\bar{t}$ of type $\overline{\delta}$ in the following way:
If $t$ is a non-logical constant $k$, then $\bar{t} = \mathbb{D}_{\delta}[\lambda.ek]$.
If $t$ is a variable $x$, then $\bar{t} = x$.
If $t$ is an application $vu$, then $\bar{t} = \bar{v} \bar{u}$.
If $t$ is an abstraction $\lambda.x.\phi$, then $\bar{t} = \lambda.x.\bar{\phi}$.
If $t$ is a conjunction, then $\overline{\bar{t}} = \overline{\lambda AB.\phi}$.
If $t$ is an existential quantifier, then $\exists \bar{t} = \exists \bar{\phi}$.
If $t$ is a negation, then $\bar{\neg t} = \lambda x.\bar{x}$.

Proposition 6 proves a $\beta$-equivalence that can be useful when manually computing interpretations of certain phrases containing an existentially quantified variable:

Proposition 6. For all terms $P$ of type $(t \rightarrow o)$, $B$ of type $o$ and any variable $x$ of type $(\gamma \rightarrow t)$ such that $x \notin FV(B)$, the following equivalence holds: $\exists (\lambda x.Px) \land B = \beta \exists (\lambda x.Px \land B)$

Proof. See page 123 of the dissertation.

Theorem 7 guarantees that everything that is valid with respect to static semantics is valid with respect to dynamic semantics, and that the opposite direction also holds.

Theorem 7 (Conservation). A proposition $t$ is true in a model $\mathcal{M}$ if and only if its dynamic variant $\mathcal{M}^\text{dyn}$ is true in the same model:

$\mathcal{M} \models t$ iff $\mathcal{M}^\text{dyn} \models \bar{t}$

Proof. See pages 123–141 of the dissertation.

Framework GL offers several linguistically motivated innovations. First of all, contexts are viewed as a conjunction of formulas and $\gamma$ is assumed to be of type $o$: $\gamma \cong o$. This gives an essential advance, namely the possibility of updating the context with a proposition: $\mathcal{P}_1 \ldots t_n = \lambda e \phi.\mathcal{P}_1 \ldots t_n \land \phi$ (upd($\mathcal{P}_1 \ldots t_n, e$)). Context $c$ is said to contain term $x$ (x is stored in $c$) if and only if $x$ is a subterm of $c$.

The selection function is defined in GL as a function of type $((t \rightarrow o) \rightarrow \gamma \rightarrow t)$. Definition 8 gives a possible formalization of sel when context is a conjunction of formulas. It specifies that, given context $c$ and property $P$, sel returns an individual $a$ stored in $c$ satisfying the condition that $c$ contains $Pa$ (i.e. a has property $P$) and a is the only individual in $c$ having property $P$:

Definition 8. Let $P$ be a term of type $(t \rightarrow o)$ and $c$ be a term of type $o$. Then, sel $P c \cong a$ if and only if $c \vdash Pa$ and, for all $x$, if $c \vdash Px$ then $x = a$.

Example 9. Assume non-human, human, male, donkey and farmer are constants of type $(t \rightarrow o)$. Then, the following equations hold:

\[
\text{sel(non-human)(donkeyx} \land \text{non-humanx)} = x
\]
\[
\text{sel(\lambda z.\text{human}z \land \text{male}z)(donkeyx} \land \text{farmer}y \land \text{male}y \land \text{human}y) = y
\]

Context $c$ contains knowledge $c'$ that was recently learned from a preceding dialogue or discourse (e.g. John loves Mary) and common knowledge $\Sigma$ (e.g. There are twenty four hours in a day). Thus, Definition 8 can be stated as follows:

Definition 10. Let $\Sigma$ be some theory formalizing (common) knowledge. Then, sel $P c' \cong a$ if and only if $\Sigma, c' \vdash Pa$ and for all $x$ if $\Sigma, c' \vdash Px$, then $x = a$. 

4
Formalization of $\Sigma$ is a difficult task. In the scope of this dissertation $\Sigma$ is represented by a finite set of formulas and, therefore, it can be expressed by a formula $\bigwedge_{\sigma \in \Sigma} \sigma$ of type $\sigma$. Thus, by equating $c$ with $\bigwedge_{\sigma \in \Sigma} \sigma \land c'$ here, the context $c$ is considered to contain the common knowledge and, therefore, it is sufficient to use Definition 8.

Function $\text{sel}$ is only partially defined in 8. First, there may be no individual variable $a$ in the context $c$ such that $c \vdash Pa$. This means that the context $c$ fails with respect to familiarity presupposition and accommodation should take place. A possible way to handle presupposition accommodation is presented in Section 3.\footnote{See Chapter 5 of the thesis for more details.} Second, the uniqueness requirement may not hold. This corresponds to the apparent ambiguity that can be resolved by a salience property of individuals in the context; or by rhetorical relations that hold between segments of discourse. It can even be a real ambiguity, as is often the case in natural language. The framework then has to cope with this ambiguity by “asking” for a clarification (as humans do).

The static type of individuals $i$ is transformed by Definition 3 into type $\gamma(t)$ of dynamic individuals. The interpretation of the pronoun $he$ in $G_0$ is $\lambda P. \lambda\phi. P(\text{sel}(\eta)e)e\phi$. Since $\text{sel}$ is redefined as having type $((t \rightarrow o) \rightarrow \gamma \rightarrow i)$, the interpretation of $he$ in GL could be $\lambda P. \lambda\phi. P(\text{sel}(\lambda x. \text{malex} \land \text{human}x)e)e\phi$. This term, of type $((I \rightarrow \sigma) \rightarrow \sigma)$, is the dynamic counterpart of the type-raised static interpretation. However, if there is a need to interpret a non-quantified noun phrase without type-raising, i.e. as simply having the type $i$, the dynamic term has to be of type $\iota$. Then, considering that part of the meaning of a pronoun is the search for the referent based on the descriptive content of the pronoun, the desired interpretation would be $\text{sel}(\lambda x. \text{malex} \land \text{human}x)$, which is $\eta$-equivalent to $\lambda e. \text{sel}(\lambda x. \text{malex} \land \text{human}x)e$. The latter two terms are of type $\gamma(t)$, since $\text{sel}$ is of type $((t \rightarrow o) \rightarrow \gamma \rightarrow i)$ and $(\lambda x. \text{malex} \land \text{human}x)$ is of type $((t \rightarrow o))$. Hence, type $\gamma(t)$ is taken to be the dynamic version $\iota$ of type $t$.

A systematic modular translation of static terms into dynamic terms is provided by Definitions 4 and 5. A special non-trivial case is the translation of non-logical constants, since they have various types and manually assigning dynamic interpretations for all of them would be problematic, if possible at all. A special systematic treatment is proposed in Definition 4.

**Example 11.** Consider the non-logical constant $\text{try}$ of type $((t \rightarrow (t \rightarrow o)) \rightarrow o)$. Its dynamic equivalent $\overline{\text{try}}$ of type $((I \rightarrow (I \rightarrow \sigma)) \rightarrow \sigma)$ is computed as follows:

$$
\overline{\text{try}}
= \mathcal{D}_{t \rightarrow (t \rightarrow o) \rightarrow o}[\lambda e. \text{try}]
= \lambda x. \mathcal{D}_{(t \rightarrow o) \rightarrow o}[\lambda e. \text{try}](x, e)] (by \ (1c))
= \lambda x P. \mathcal{D}_{o}[\lambda e. \text{try}](x, e)/[(x, e)[P, e]] (by \ (1c))
= \lambda x P. \mathcal{D}_{o}[\lambda e. \text{try}](x e)/[(x, e)[P, e]] (by \ (2a))
= \lambda x P. \mathcal{D}_{o}[\lambda e. \text{try}](x e)(\lambda y. (S_{o}P)\mathcal{D}_{i}[\lambda e, y, c]) (by \ (2c))
= \lambda x P. \mathcal{D}_{o}[\lambda e. \text{try}](x e)(\lambda y. (S_{i}P)\mathcal{D}_{i}[\lambda e, y, e]) (by \ (1a))
= \lambda x P. \mathcal{D}_{i}[\lambda e. \text{try}](x e)(\lambda y. P(\lambda e. e)(\lambda e. T)) (by \ (1b))
= \lambda x P. \lambda \phi. (\lambda e. \text{try}(x e)(\lambda y. P(\lambda e. e)(\lambda e. T)) e) \land (\lambda P. \lambda \phi. \text{try}(x e)(\lambda y. P(\lambda e. e)(\lambda e. T)) e)) (by \ (1b))
= \lambda P. \lambda \phi. \text{try}(x e)(\lambda y. P(\lambda e. e)(\lambda e. T)) e \land (\lambda \phi. \text{try}(x e)(\lambda y. P(\lambda e. e)(\lambda e. T)) e))
3 From Static to Dynamic Meaning: Interpretation of Lexical Items and Sentences

All static lexical interpretations can be translated into equivalent dynamic interpretations simply by applying Definitions 4 and 5. This translation is sufficient for lexical items that do not have any additional linguistic meaning apart from their direct content. However, some expressions have non-trivial meaning and can cause so-called linguistic side-effects. This dissertation proposes to use an exception raising and handling mechanism in the style of ML to account for dynamic linguistic side-effects.

The ability of exceptions to carry information is very useful for formalization of the failure of familiarity presupposition. Function sel is only partially defined in 8 because the context c may not contain an individual satisfying the property P, i.e., there may be no a that \( c \vdash Pa \). This corresponds to the failure of familiarity presupposition and exception raising can be used to express this failure. Thus, the definition of sel can be made total in the following way:

**Definition 12.** Let \( P \) be a term of type \((ι \rightarrow o)\) and \( c \) be a term of type \( o \). Then sel \( P \) \( c \) is defined as follows:

\[
\text{sel} P \ c = \begin{cases} 
\text{choose}\{ a \mid c \vdash Pa \} & \text{if } \{ c \vdash Pa \} \neq \emptyset \\
\text{raise \text{AbsentIndividualExc} } P & \text{otherwise}
\end{cases}
\]

According to this definition, when an individual with the required property \( P \) is absent in \( c \), the function sel raises exception \text{AbsentIndividualExc} that carries the property \( P \). The operator \( \text{choose} \) is used in 12 to articulate the fact that an appropriate (e.g., more salient) individual having property \( P \) has to be chosen if the number of candidates is greater than one.

**Example 13.** Assume non\_human, human, male, donkey, farmer and unicorn are constants of type \((ι \rightarrow o)\). Then the following equations hold:

\[
\text{sel} (\text{human}) (\text{donkey} \ x \land \text{non\_human} \ x) = \text{AbsentIndividualExc} \text{human} \\
\text{sel} (\lambda x. \text{unicorn} \ x) (\text{donkey} \ x \land \text{farmer} \ y \land \text{male} \ y \land \text{human} \ y) = \text{AbsentIndividualExc} (\lambda x. \text{unicorn} \ x)
\]

The exception (\text{AbsentIndividualExc} \( P \)) is subsequently handled by creating and accommodating a new individual having the property \( P \). Exception raising and handling happens when a sentence is provided some context, which takes place on the level of interpretation of the discourse. The approach follows van der Sandt [17] in regard to the relationship between anaphora and presupposition. Formal details of accommodation as exception handling are sketched in the following section.

While selection function sel is a function of type \(((ι \rightarrow o) \rightarrow γ \rightarrow ι)\), i.e., a function that takes a static property, it is convenient to define a term of type \(((ι \rightarrow o) \rightarrow γ \rightarrow ι)\) working on a dynamic property. This term is abbreviated as sel and is defined in as \( \lambda P \cdot \text{sel} (\lambda x. P(\lambda e.x)e(\lambda e. T)) \). Note that sel is defined via sel. In the body of sel the dynamic property \( P \) is transformed into the static property \( (\lambda x. P(\lambda e.x)e(\lambda e. T)) \) that is given to static sel as an argument.

Table 1 of Appendix A shows examples of resulting dynamic interpretations of lexical items causing linguistic side-effects. For the sake of comparison, Table 2 shows static interpretations for the same lexical items. Note that for every lexical item, its dynamic type is analogous to its static type with the only difference that each atomic type is dynamized.

---

2See subsections 5.1.1-5.1.5 of the thesis.

3For details on how these interpretations are obtained see Subsections 5.2.2-5.2.5 of the thesis. For examples of how dynamic meanings of sentences are computed from dynamic meanings of their lexical items see Section 5.4 of the thesis.
4 Continuation-based Dynamic Logic with Exceptions: Interpretation of Discourse

The principle of compositionality is a fundamental requirement necessary to finitely express an infinite number of possible meanings. This section defines a function that takes an interpretation of a discourse and an interpretation of a sentence and returns a compositional interpretation of an updated discourse. This function uses exception handling mechanism for accommodating presuppositions triggered by referring expressions when the context of the discourse does not contain appropriate referents. The use of exception raising and handling mechanism motivates the extension of the calculus of framework GL, resulting in framework GL\(\chi\).

4.1 Discourse Update: Accommodation as Exception Handling

Recall that interpretations of sentences are terms of type \((\gamma \to (\gamma \to o) \to o)\), the type of a dynamic proposition. Informally, one can think of the meaning of an individual sentence as something that has to be evaluated with respect to a context to be fully understood. A discourse, on the other hand, can be seen as something that already has a concrete context. Therefore, interpretations of discourses can be defined as the terms of type \(((\gamma \to o) \to o)\). Thus, the current discourse is represented by a term that takes a continuation as its only argument, which is necessary for the computation of the meaning of the remainder of the discourse. Every new sentence updates the discourse and its context. Then a discourse update function \(\text{dupd}_g\), which takes the meaning of the current discourse and the meaning of the new sentence as arguments, can be defined as follows:

Definition 14. [Discourse Update] Let \(D\) be an interpretation of a discourse and \(S\) be an interpretation of a sentence. Then the function \(\text{dupd}_g\) is defined in (3a), where \(gacc\) is a function of type of type \(((\gamma \to o) \to o) \to (\gamma \to (\gamma \to o) \to o)\) that is responsible for global accommodation and is defined in (3b):

\[
\text{dupd}_g\ D\ S = \lambda\phi.\ D(\lambda e.gacc\ S\ e\ \phi) \quad (3a)
\]

\[
gacc\ S\ e\ \phi \triangleq S\ e\ \phi\ \text{handle} (\text{AbsentIndividualExc} P) \quad (3b)
\]

with \(\exists\lambda x.(Px)\land gacc\ S\ (\text{upd}(Px,e))\ \phi\)

A raised exception has to be handled. By having a handler of \(\text{AbsentIndividualExc}\ P\), \(\text{dupd}_g\) deals with presuppositions triggered by referring expressions with descriptive content \(P\) and absent in the context. The exception handler performs global accommodation of the presuppositional content of presuppositions. Thus, the handler of \(\text{AbsentIndividualExc}\ P\) introduces an individual (variable) \(x\) into the logical form of the discourse, updates the context with \((Px)\), and makes a new recursive call of \(gacc\) with \(S\) applied to the amended context \((\text{upd}(Px,e))\) and to the continuation \(\phi\) (note that continuation \(\phi\) is the variable over which the resulting discourse is abstracted). The function \(gacc\) is a recursive function because the interpretation \(S\) evaluated on the context \(e\) can raise more than one exception.\(^4\)

It is important to see why the exception handling happens on the level of discourse. Each sentence has a dynamic interpretation independent of the discourse it appears in. However, due to the fact that the interpretation of the sentence has an argument standing for the context, it “receives” the context of the discourse in which it is appended. Whether an exception is raised or not depends on the discourse’s context and the interpretation of the sentence. That is why the exception handling is performed when the interpretation of an updated discourse is computed.

\(^4\)See Section 6.1 of the thesis for more details and examples of discourse update.


4.2 Framework GLχ

The addition of exception raising and handling requires the extension of the \( \lambda \)-calculus defined in 2.2.

**Definition 15.** [\( \lambda \)-terms] The set of \( \lambda \)-terms \( T \) is defined as follows:

\[
T ::= x \mid k \mid (ET) \mid \lambda x.T \mid (TT) \mid (\text{raise } T) \mid \text{handle } (Ex) \text{ with } T
\]

where \( x \) is a variable, \( k \) is a constant, \( E \) is an exception constructor.

In the following definition an additional type \( \chi \) for exceptions is added:

**Definition 16.** [Types] The set \( T \) of types of framework GLχ is defined inductively as follows:

- Atomic types:
  - \( t \in T \) (individual)
  - \( o \in T \) (proposition)
  - \( \chi \in T \) (exception)

- Complex types:
  - \( \alpha, \beta \in T \implies (\alpha \rightarrow \beta) \in T \)

The typing rules are typing rules given in Definition 2 plus typing rules defined in 17:

**Definition 17.** [Typing rules] Let \( \alpha, \beta \) be arbitrary types and \( E \) be an exception constructor of type \( (\beta \rightarrow \chi) \). A statement \( t : \delta \) is derivable from the basis \( \Delta \), i.e. \( \Delta \vdash t : \delta \), if \( t : \delta \) can be produced using rules defined in 2 and rules given below:

\[
\Gamma \vdash e : \chi \\
\Gamma \vdash \text{raise } e : \alpha \\
\Gamma \vdash t_1 : \alpha \\
\Gamma \vdash E : \beta \rightarrow \chi \\
\Gamma, t_2 : \beta \vdash t_3 : \alpha
\]

**Definition 18.** [Strong and weak normal forms] Let \( x \) and \( y \) be variables, \( k \) be a constant, \( E \) be an exception constructor. Terms in strong (S) normal form and in weak (W) normal form are defined as follows:

- \( Q ::= x \mid k \mid (ES) \mid (QS) \)
- \( S ::= Q \mid \lambda y.\text{raise } S[y] \mid \lambda y.S[y] \)
- \( W ::= Q \mid \lambda x.T \)

**Definition 19.** [Uncaught exception] Uncaught exceptions are defined as follows: \( U ::= \text{raise } S \)

**Definition 20.** [Weak and strong evaluation rules] Let \( x \) be a variable, \( k \) be a constant, \( E, E_1 \) and \( E_2 \) be exception constructors, \( t, t_1, t_2 \) and \( t_3 \) be terms, \( w, w_1 \) and \( w_2 \) range over weak normal forms, \( v \) range over strong normal forms, \( u \) range over strong normal forms and uncaught exceptions. Terms in framework GLχ are evaluated according to the following rules:

**Weak evaluation rules:**

\[
\begin{align*}
x & \rightarrow_w x \\
k & \rightarrow_w k \\
\lambda x.t & \rightarrow_w \lambda x.t \\
t & \rightarrow_s v \\
E & \rightarrow_w Ev \\
\text{raise } E & \rightarrow_w \text{raise } v
\end{align*}
\]

\[
\begin{align*}
t_1 \rightarrow_w \lambda x.t_3 & \quad t_2 \rightarrow_w w_1 & \quad t_3[x := w_1] \rightarrow_w w_2 \\
& \quad t_1 t_2 \rightarrow_w w_2 \\
t_1 \rightarrow_w w & \quad t_2 \rightarrow_s v & \quad \text{w is not a } \lambda \text{-abstraction} \\
& \quad t_1 t_2 \rightarrow_w \text{raise } v \\
t_1 \rightarrow_s \text{raise } v & \quad t_1 t_2 \rightarrow_w \text{raise } v
\end{align*}
\]

\[
\begin{align*}
t & \rightarrow_s v \\
\text{raise } t & \rightarrow_w \text{raise } v \\
t & \rightarrow_s \text{raise } v & \quad \text{raise } t \rightarrow_w \text{raise } v \\
t & \rightarrow_s \text{raise } (Ev) & \quad t_2 \rightarrow_w w_1 & \quad w_1[x := v] \rightarrow_w w_2 \\
& \quad t_1 \rightarrow_s \text{handle } (Ex) \text{ with } t_2 \rightarrow_w w_2
\end{align*}
\]
Strong evaluation rules:

\[
\frac{t \rightarrow_w \lambda x.t_1 \quad t_1 \rightarrow_s v}{t \rightarrow_s \lambda x.v} \quad \frac{t \rightarrow_w u \quad t_1 \rightarrow_s v}{t \rightarrow_s \lambda x.v} \quad \text{provided } x \notin FV(v) \quad \frac{t \rightarrow_w \lambda x.t_1 \quad t_1 \rightarrow_s v}{t \rightarrow_s \lambda x.raise v} \quad \text{provided } x \in FV(v)
\]

The idea of the evaluation system is as follows. The normal form of a term should be in \( S \cup U \). The evaluation of a term starts by weakly evaluating it. Weak evaluation directly results in a term in \( S \cup U \) (due to the last strong evaluation rule) except when the evaluated term is an abstraction (in this case the evaluation proceeds according to one of the first three strong evaluation rules).

The next proposition proves that the rules in Definition 20 always yield either a normal form or an uncaught exception. Moreover, for each term to be evaluated, Definition 20 provides a single rule that can be applied. This guarantees the uniqueness of term evaluation in the framework.

**Proposition 21.** If \( t \rightarrow_s t' \) then \( t' \in S \cup U \). If \( t \rightarrow_w t' \) then \( t' \in W \cup U \).

**Proof.** See pages 189–193 of the thesis.

**Proposition 22** (Subject Reduction). If \( t \rightarrow_w t' \) and \( \Gamma \vdash t : \delta \) then \( \Gamma \vdash t' : \delta \). If \( t \rightarrow_s t' \) and \( \Gamma \vdash t : \delta \) then \( \Gamma \vdash t' : \delta \).

**Proof.** The proof is by induction on the derivation. See pages 193–201 of the thesis.

This section briefly presented the framework GL\( \chi \). On the example of referring expressions it is shown in Section 6.3 of the thesis that exception raising can be used to implement the emergence of presuppositions: when the interpretation of a sentence containing a referring expression is evaluated with respect to a context that does not contain a referent satisfying the descriptive content \( P \) of the referring expression, the exception \((\text{AbsentIndividualExc } P)\) is raised. Handling of the exception corresponds to accommodation of the presupposition. Moreover, possible ways of implementing global and intermediate accommodation are suggested in the thesis and solutions for presupposition projection and binding problems are proposed in Sections 6.4 and 6.5 of the thesis.

## 5 Directions for Further Development of the Framework

Due to its simplicity and flexibility, framework GL\( \chi \) has potential to be extended for handling even more complex phenomena in natural language. Chapter 7 of the thesis discusses some directions for possible extensions of the framework. Section 7.1 illustrates how presuppositions triggered by the verb \textit{know} can be handled in GL\( \chi \). Section 7.2 outlines an approach to deal with some conversational implicatures. Both cases required extension of the discourse update function with additional exception handlers. Interestingly, the terminology of GL\( \chi \) allowed to give a clear and concise comparison of presuppositions and conversational implicatures, summarized in Table 3 of Appendix A. Section 7.3 concludes with such a comparison. Thereby, chapter 7 illustrates the possibility to handle semantic and (some) pragmatic phenomena in a unified framework.

On the one hand, the chapter illustrated that the framework has potential to be adapted for even more complex natural language phenomena and presented general analyses of some of these phenomena in GL\( \chi \). On the other hand, the chapter assumed that solutions to many issues needed for a complete formalization of the discussed phenomena are provided. Among them are the formalization of the abduction
function has been formalized in the makesat function which chooses which facts to be removed from an unsatisfiable context to make it satisfiable. Moreover, a suitable theorem prover should stand behind functions checkvalid and checkprovable. These issues are possible directions for future research.

6 Conclusions

The goal of showing that semantics of discourse dynamics can be compositionally expressed with standard well-established mathematical methods was successfully achieved. In the proposed framework GLχ, dynamic meanings are expressed using only simply-typed lambda calculus and classical logic, using a systematically defined translation from their corresponding static lexical interpretations.

Advantages of the Developed Framework

- The approach is truly compositional: the meaning of a complex expression is computed by β-reducing the composition, obtained by functional application, of the meanings of its lexical items.
- Continuation-based dynamic logic serves as an “intermediate” logical language for an “indirect” model-theoretical interpretation of sentences of a natural language. Therefore, there is no need to be concerned with model-theoretical issues. The resulting logical formula can be interpreted with classical well-studied model theory, if desired.
- The approach is independent of the “intermediate” language used to express meanings of the expressions and, hence, is general. This generality allows the use of well-established mathematical and logical theories, possibly developed outside computation linguistics, to explain natural language phenomena. Moreover, standard automated deduction tools can be used to reason about natural language sentences once they have been translated to a standard intermediate logical language.
- Variables do not have any special status and are variables in the usual mathematical sense. Therefore, the notions of free and bound variables are standard.
- There are no imperative dynamic notions, such as assignment functions. Therefore, the destructive assignment problem does not occur. Meanings assigned to expressions are closed λ-terms.
- There is no need for rules that artificially extend the scope of quantifiers.
- Context and content are distinct, but do interact during the computation of the discourse’s meaning. The approach does not depend on any specific structure given to the context. In contrast, context is defined as a term of type parameter γ and, therefore, its structure can be altered when necessary. The context is represented by a term, and, hence, it is explicit. Therefore, the troubles related to implicit states are avoided.
- The approach treats semantic and pragmatic phenomena in a unified way.
- The framework allows the integration of any desired anaphora resolution algorithm (into the selection function sel). Since sel is part of the interpretations of anaphoric lexical items and receives the current context as one of its arguments, anaphora is always resolved compositionally.
- Triggers of various linguistic side-effects can be expressed by modifying the standard dynamic interpretations of lexical items of the same syntactic category in a way that potentially raises exceptions. Then appropriate exception handlers take these additional linguistic side-effects into consideration.

Taken together, the advantages above lead to greater flexibility. The framework is extensible for handling more complex phenomena of natural language and, therefore, opens opportunities for future development.
A Appendix

<table>
<thead>
<tr>
<th>Lexical item</th>
<th>Syntactic category</th>
<th>Dynamic type</th>
<th>Dynamic interpretation in GL</th>
</tr>
</thead>
<tbody>
<tr>
<td>story</td>
<td>n</td>
<td>$\overline{t} \rightarrow \overline{\sigma}$</td>
<td>story $\lambda YX. X(\lambda X. Y(\lambda Y. lovexy))$</td>
</tr>
<tr>
<td>loves</td>
<td>np $\rightarrow$ np $\rightarrow$ s</td>
<td>$((t \rightarrow o) \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o) \rightarrow o$</td>
<td>loves $\lambda YX. X(\lambda X. Y(\lambda Y. lovexy))$</td>
</tr>
<tr>
<td>a</td>
<td>n $\rightarrow$ np</td>
<td>$((t \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o) \rightarrow o)$</td>
<td>a $\lambda PQ. \exists(\lambda X. P(x) \land Qx)$</td>
</tr>
<tr>
<td>every</td>
<td>n $\rightarrow$ np</td>
<td>$((t \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o) \rightarrow o)$</td>
<td>every $\lambda PQ. \exists(\lambda X. P(x) \land Qx)$</td>
</tr>
<tr>
<td>who</td>
<td>(np $\rightarrow$ s) $\rightarrow$ n $\rightarrow$ n</td>
<td>$((t \rightarrow o) \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o)$</td>
<td>who $\lambda YXP. P(\exists(\lambda X. ((X(x) \land Y(\lambda X. Y(\lambda Y. lovexy))))))$</td>
</tr>
<tr>
<td>John</td>
<td>np</td>
<td>$(t \rightarrow o) \rightarrow o$</td>
<td>John $\lambda P. Pj$, where j is a constant</td>
</tr>
<tr>
<td>he</td>
<td>np</td>
<td>$(t \rightarrow o) \rightarrow o$</td>
<td>he $\lambda P. Pj$, where j is a constant</td>
</tr>
<tr>
<td>the</td>
<td>n $\rightarrow$ np</td>
<td>$(t \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o)$</td>
<td>the $\lambda NP. P$</td>
</tr>
<tr>
<td>’s</td>
<td>np $\rightarrow$ n $\rightarrow$ np</td>
<td>$((t \rightarrow o) \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o)$</td>
<td>’s $\lambda YXP. P$</td>
</tr>
</tbody>
</table>

Table 1: Dynamic lexical interpretations in framework GL.

<table>
<thead>
<tr>
<th>Lexical item</th>
<th>Syntactic category</th>
<th>Static type</th>
<th>Static interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>story</td>
<td>n</td>
<td>$t \rightarrow o$</td>
<td>story $\lambda P. Pj$, where j is a constant</td>
</tr>
<tr>
<td>loves</td>
<td>np $\rightarrow$ np $\rightarrow$ s</td>
<td>$((t \rightarrow o) \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o) \rightarrow o$</td>
<td>loves $\lambda YX. X(\lambda X. Y(\lambda Y. lovexy))$</td>
</tr>
<tr>
<td>a</td>
<td>n $\rightarrow$ np</td>
<td>$((t \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o) \rightarrow o)$</td>
<td>a $\lambda PQ. \exists(\lambda X. P(x) \land Qx)$</td>
</tr>
<tr>
<td>every</td>
<td>n $\rightarrow$ np</td>
<td>$((t \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o) \rightarrow o)$</td>
<td>every $\lambda PQ. \exists(\lambda X. P(x) \land Qx)$</td>
</tr>
<tr>
<td>who</td>
<td>(np $\rightarrow$ s) $\rightarrow$ n $\rightarrow$ n</td>
<td>$((t \rightarrow o) \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o)$</td>
<td>who $\lambda YXP. P(\exists(\lambda X. ((X(x) \land Y(\lambda X. Y(\lambda Y. lovexy))))))$</td>
</tr>
<tr>
<td>John</td>
<td>np</td>
<td>$(t \rightarrow o) \rightarrow o$</td>
<td>John $\lambda P. Pj$, where j is a constant</td>
</tr>
<tr>
<td>he</td>
<td>np</td>
<td>$(t \rightarrow o) \rightarrow o$</td>
<td>he $\lambda P. Pj$, where j is a constant</td>
</tr>
<tr>
<td>the</td>
<td>n $\rightarrow$ np</td>
<td>$(t \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o)$</td>
<td>the $\lambda NP. P$</td>
</tr>
<tr>
<td>’s</td>
<td>np $\rightarrow$ n $\rightarrow$ np</td>
<td>$((t \rightarrow o) \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o) \rightarrow ((t \rightarrow o) \rightarrow o)$</td>
<td>’s $\lambda YXP. P$</td>
</tr>
</tbody>
</table>

Table 2: Static lexical interpretations.

<table>
<thead>
<tr>
<th>Source</th>
<th>Lexical item (trigger)</th>
<th>Presuppositions</th>
<th>Implicatures$_1$</th>
<th>Implicatures$_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Assertion w.r.t. a context</td>
<td>Assertion w.r.t. a context or presupposition accommodated in a context or another implicature accommodated in a context.</td>
<td>Unsatisfiable context.</td>
</tr>
<tr>
<td>Exception</td>
<td>Failure in finding the proof of a proposition from the axioms contained in the context.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exception handling</td>
<td>Abducing what is missing in the proofs. If the abduced formula is exactly the proposition that was searched in the context and now is carried by the exception handler, it is a presupposition. All other abduced formulas are implicatures.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Remving those formulas from the context that are necessary for its refutation. Negating the removed formulas (these negated formulas are conversational implicatures). Adding these implicatures to the context.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Comparison of presuppositions and two types of conversational implicatures.
References


