Chapter 1

A Unification Strategy for Parsing Agreement Errors

JENNIFER FOSTER

ABSTRACT. A parser employing classical unification will fail to find a parse for sentences containing agreement violations, e.g. They comes. This is because classical unification fails when confronted with values which are inconsistent. I propose a new kind of unification called robust unification which takes the two inconsistent values at the root of an agreement violation and relates them using a relation which I will call inconsistent identity. This approach has the advantage that it acknowledges errors yet still produces consistent results. I argue that this “robust unification” is only appropriate for agreement errors and not for constraint violation errors which do not involve an agreement violation.

1 Introduction

Since natural language is rife with error, programs which process natural language must employ strategies to deal with such errors in as sensible a way as possible. The aim of this paper is to provide a treatment for agreement errors, a subset of the class of constraint violation errors. This treatment is based on the theory of feature structures provided by [Carpenter, 1992]. Since this is the underlying theoretical framework for Head-Driven Phrase Structure Grammar (HPSG) [Pollard and Sag, 1994], it is expected that the work presented in this paper will have applications in HPSG-based parsing. However throughout this paper I make no reference to particular theories of grammar and hopefully the feature structures presented will be clear to anyone with a basic knowledge of unification based approaches to grammar.

According to [Douglas, 1995], constraint violation errors “involve, what in most contemporary syntactic theories, are best viewed as the violation of constraints on feature values”. In Section 2 I make a distinction between constraint violation errors which involve an agreement mismatch and those which don’t. Examples of both kinds of error are provided. In Section 3, an overview of Carpenter’s basic feature structure system is provided. The concepts of type hierarchy, feature structure, subsumption and unification are briefly explained and the failure of classical unification to behave robustly
when confronted with inconsistent inputs is outlined. In Section 4, the main section of the paper, the operation of robust unification is explained. This is a new kind of unification specifically designed to treat agreement errors in a robust fashion. An alternative unification strategy called "relaxed unification" put forward by [Vogel and Cooper, 1995] is briefly discussed and then the concept of inconsistent identity which is central to the operation of robust unification is introduced. An outline of the robust unification procedure is provided, and the structures which arise from this operation, namely robust feature structures, are defined. Finally this approach is briefly compared to the approach to robust parsing put forward by [Douglas, 1995]. In Section 5, I argue that robust unification is not an appropriate treatment for constraint violation errors which do not involve agreement.

2 Ill-Formed Data

The data of interest is illustrated in Fig. 1.1. Rather than inventing ill-formed examples I use real-world ones taken from the British National Corpus\(^1\). Some of the errors given in this table are not really errors in that they occur frequently in various dialects of English. However I classify them here as errors since they will pose problems for natural language parsers unless the parser has at its disposal a grammar which is capable of describing all dialectal variations of English.

In the first five sentences, an agreement constraint between two constituents has been violated. These sentences all contain information which is contradictory, e.g. in the sentence And they comes up to the edge, the subject noun phrase is telling us that the agent is plural, whereas the verb is telling us that the agent is singular. A natural language grammar will stipulate that the number information provided by the main verb in a sentence must be consistent with the number information provided by the subject noun phrase and if this is not the case, a number agreement error has occurred. Another form of agreement which occurs in English is person agreement, for example, the sentence So off I goes where a person agreement constraint has been violated since the subject of the sentence is in first person form and the verb is in third person form. We can look to a structurally richer language such as German for more examples of agreement constraints, e.g. gender and case agreement among noun phrase constituents.

The last five sentences do not involve an agreement violation since they do not contain contradictory information, e.g. in the sentence I seen the sergeant, the information provided by the subject noun phrase I and the object noun phrase the sergeant does not contradict the information provided by the verb seen. Where the contradiction does occur, however, is between the grammar and the sentence, because the grammar will expect

\(^1\)Accessed via http://sara.natcorp.ox.ac.uk/lookup.html
<table>
<thead>
<tr>
<th>Agreement Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>And they comes up to the edge</td>
</tr>
<tr>
<td>Singular Verb and Plural Subject</td>
</tr>
<tr>
<td>the gates was left open</td>
</tr>
<tr>
<td>Singular Verb and Plural Subject</td>
</tr>
<tr>
<td>So off I goes</td>
</tr>
<tr>
<td>3rd Person Verb and 1st Person Subject</td>
</tr>
<tr>
<td>Where did this women learn such arrogance?</td>
</tr>
<tr>
<td>Singular Determiner and Plural Noun</td>
</tr>
<tr>
<td>If Paul has had a phone call from one of these woman</td>
</tr>
<tr>
<td>Plural Determiner and Singular Noun</td>
</tr>
<tr>
<td>Non-Agreement Errors</td>
</tr>
<tr>
<td>I seen the sergeant</td>
</tr>
<tr>
<td>Verb Form should be Finite</td>
</tr>
<tr>
<td>Some people would have went mad</td>
</tr>
<tr>
<td>Verb Form should be Past Participle</td>
</tr>
<tr>
<td>No, me Mam'Il have me tea ready</td>
</tr>
<tr>
<td>Possessive Pronoun Required</td>
</tr>
<tr>
<td>they're much more better</td>
</tr>
<tr>
<td>Adjective shouldn't be a Comparative</td>
</tr>
<tr>
<td>What are them tablets?</td>
</tr>
<tr>
<td>Personal Pronoun can't be a Determiner</td>
</tr>
</tbody>
</table>

Figure 1.1: Some Examples of The Data of Interest

a finite verb form, e.g. saw. Consider also the sentence *Some people would have went mad*. The verb *have* is expecting as a complement a verb phrase whose head verb is in past-participle form. What happens in the sentence *Some people would have went mad* is that the complement of the verb *have* has as its head the verb *went* which is a finite rather than a past-participle form. This results in a clash between the expectations of the verb *have* and its actual complement. However, what distinguishes this kind of error from an agreement error is that there is no stipulation that the head and one of its complements should share a value for a particular feature.

So in the last four sentences there is a clash between what the grammar says ought to be the case and what is the case. In the first four sentences, there is also a clash between what the grammar says ought to be the case and what is the case, but this clash arises because the sentences themselves contain two pieces of clashing information.

## 3 Classical Unification

If feature structures are employed as the underlying representation for linguistic objects, feature structure unification, rather than simple terminal-nonterminal symbol matching, becomes a basic operation in natural language parsing. [Carpenter,1992] defines a basic feature structure (over the
set of features, \textit{Feat}, and the set of types, \textit{Type}) to be a tuple consisting of a finite set of nodes, \( Q \), a root node, \( \bar{q} \), a partial function \( \delta \) from a node and a feature to another node, and a total function \( \theta \) which assigns a type to each node. Types are organised into an inheritance hierarchy where a more specific type is subsumed by a more general type positioned beneath it in the hierarchy and where every consistent set of types has a least upper bound, the most general type subsumed by all the types in the set. The most general type in the hierarchy is known as \( \bot \) and \( \top \) is used to refer to the inconsistent type.

[Carpenter,1992] defines a subsumption relation between feature structures: \( F_1 \) subsumes \( F_2 \) if all the nodes in \( F_1 \) can be mapped onto the nodes in \( F_2 \) such that the root node is preserved, the types of all the nodes in \( F_1 \) subsume the corresponding types in \( F_2 \), and all the transitions in \( F_1 \) are present in \( F_2 \). The unification of two feature structures is the least upper bound of the two structures in \( \langle F, \sqsubseteq \rangle \) where \( F \) is the set of feature structures and \( \sqsubseteq \) is subsumption between feature structures. The unification of two feature structures takes the information contained in the input structures and combines this information in the output feature structure. It proceeds by identifying the nodes in the input feature structures into equivalence classes and finding the least upper bound of all the nodes in a class: if there is no least upper bound then unification fails.

[Carpenter,1992] defines a description language for talking about feature structures and provides a set of axioms which capture the logical behaviour of feature structures. The only way for inconsistency to result in Carpenter’s basic system is if an attempt is made to unify two inconsistent types. This behaviour is captured by the following axiom:

\[
\sigma \wedge \tau \Leftrightarrow \top \quad [\text{if } \sigma \sqcup \tau \text{ does not exist }]
\]

The “Top Smashing” axiom demonstrates how inconsistency is propagated:

\[
\pi : \top \Leftrightarrow \top
\]

\( \pi \) indicates a path or sequence of features, and \( \pi : \top \) indicates that the path \( \pi \) has the value \( \top \), or that the node at the end of the path \( \pi \) is labelled with the type \( \top \).

A traditional unification-based parser won’t parse the noun phrase \textit{this women} because unification will fail when an attempt is made to find a least upper bound for the inconsistent types \textit{sing} and \textit{plur}. What we would like a parser to return is that \textit{this women} is a noun phrase, its head is the plural noun \textit{women}, its specifier is the singular determiner \textit{this} and there is a number mismatch between the determiner and the noun. In Section 4, an operation called “robust unification” is outlined which allows just this information to be returned.

\footnote{Extensions to the system are not included in this discussion.}

\footnote{Or the determiner \textit{this}, depending on one’s view of noun phrase syntax.}
4 Agreement Errors and Robust Unification

4.1 Inconsistent Identity

A classical unification response to the ungrammatical phrase *this women* is illustrated in Fig. 1.2. “Top-smashing” ensures that this structure is actually equivalent to $\top$, the inconsistent type. Therefore, a parser employing classical unification will return the unhelpful value $\top$ when confronted with a sentence containing the phrase *this women*. This is clearly an undesirable outcome since we want natural language parsers to recover as much structure as possible from sentences containing errors.

To overcome this problem, [Vogel and Cooper, 1995] propose a relaxed approach to unification which results in “mildly inconsistent” feature structures. An example of a “mildly inconsistent” feature structure is given in Fig. 1.3. In their approach, the inconsistent types $\text{sing}$ and $\text{plur}$ are actually unified and the result is the type $\top(\text{sing}, \text{plur})$. This means that the number value of the whole noun phrase is both singular and plural. So, in their approach, the feature structures contain inconsistent information. The traditional notion of structure-sharing needs to be redefined and as they themselves admit, “this is a precarious position to assume since the logic of the resulting feature structures is not well understood.”

The approach I am proposing is far more conservative since it does not require that two inconsistent nodes actually be identified. I propose an operation called “robust unification” which will produce the structure given in

$$np \left[ \begin{array}{l}
\text{HEAD-DTR:} \quad \text{noun} \left[ \text{NUM: } \square \top \right] \\
\text{SPR-DTR:} \quad \text{det} \left[ \text{NUM: } \square \right]
\end{array} \right]$$

Figure 1.4: The noun phrase *this women*: Robust Unification
The symbol $\leftrightarrow_\tau$ indicates a relation which I will call “inconsistent identity”, and if two nodes in a feature structure are related by “inconsistent identity”, this means that they would be structure-shared were it not for the fact that they are labelled with inconsistent types. The advantage of this is that it allows the information conveyed by *this women* to be represented whilst still informing us that there is a number agreement violation between the noun and the determiner.

### 4.2 Robust Unification in Action

In order to illustrate how exactly robust unification operates, I will work step-by-step through an example. Fig 1.5 shows a feature structure displaying some linguistic properties of the word *this*. This feature structure states that *this* is a singular determiner. The feature structure in Fig. 1.6 provides us with similar information for the plural noun *women*. Fig. 1.7 shows a feature structure which states that the head of a noun phrase is a noun, its specifier is a determiner and both these constituents share a value for their NUMBER feature. Here, the type *num* is the general type subsuming the types *sing* and *plur*.

If we robustly unify the feature structure for the noun *women* (Fig. 1.6) with the value of the `HEAD-DTR` feature in the feature structure for the whole noun phrase (Fig. 1.7) we get the structure shown in Fig. 1.8. Here the type *plur* unifies with the type *num* to yield the more specific type *plur* as a result. Classically unifying these two feature structures would bring the same result, so robust unification behaves in the same way as classical unification when there is no type clash in the feature structures to be unified.
Finally if we robustly unify the value of the SPR-DTR feature in the feature structure in Fig. 1.8 with the feature structure for the determiner this (Fig. 1.5), we get as a result the feature structure shown in Fig. 1.4, repeated for convenience in Fig. 1.9. This is what happens in order to obtain this result:

1. A type clash occurs between the type sing in the feature structure in Fig. 1.5 and the type plur in the feature structure in Fig. 1.8.

2. The structure-sharing constraint between the paths HEAD-DTR: NUM and SPR-DTR: NUM in the feature structure in Fig. 1.8 is relaxed.

3. The two paths which were to share a value now have separate values, the value of the path SPR-DTR: NUM arising from the current unification with the feature structure for the determiner this (see Fig. 1.5), and the value for the path HEAD-DTR: NUM arising from the previous unification with the feature structure for the noun women (see Fig. 1.6).

4. The two nodes labelled with the types sing and plur are related by the “inconsistent identity” relationship, represented by the symbol $\leftrightarrow\tau$.

The order in which the noun or the determiner feature structures are robustly unified with the noun phrase feature structure does not affect the final outcome which will always be the feature structure given in Fig. 1.9. It is always a structure sharing constraint that is relaxed, to be replaced by “inconsistent identity”.

I mentioned in Section 3 that classical unification takes all the information from two input feature structures and combines this information in a third output feature structure. You will have noticed by now that this robust unification operation does not have this property. In Fig. 1.7, for example, the feature structure stipulates that two paths should share a value. However, when the information in this feature structure is combined via robust unification with the information in the feature structures for the determiner this and the noun women (Fig. 1.5 and Fig. 1.6), the resulting feature structure (Fig. 1.9) contains no such stipulation.
4.3 Robust Feature Structures

A definition of the structures that result from “robust unification”, i.e. “robust feature structures”, is provided. This definition is based on Carpenter’s original feature structure definition [Carpenter, 1992, p.26].

Definition 1 A robust feature structure is a quintuple $F = (Q, \bar{q}, \theta, \delta, \leftrightarrow)$, where $F = (Q, \bar{q}, \theta, \delta)$ is a feature structure and where $\leftrightarrow \subseteq Q \times Q$ is an anti-reflexive and symmetric inconsistent identity relation.\(^4\)

The robust feature structure illustrated in Fig. 1.9 is:

$$Q = \{q_1, q_2, q_3, q_4, q_5\}$$

$$\bar{q} = q_1$$

$$\delta(q_1, \text{head} \perp \text{dtr}) = q_2, \ \delta(q_1, \text{spr} \perp \text{dtr}) = q_3, \ \delta(q_2, \text{num}) = q_4, \ \delta(q_3, \text{num}) = q_5$$

$$\theta(q_1) = \text{np}, \ \theta(q_2) = \text{noun}, \ \theta(q_3) = \text{det}, \ \theta(q_4) = \text{plur}, \ \theta(q_5) = \text{sing}$$

$$q_4 \leftrightarrow q_5$$

Carpenter’s description language is extended to include inconsistent path equations: the symbol $\doteq$ is used to equate two paths $\pi_1, \pi_2$ in an “inconsistent equality”. The satisfaction relation between robust feature structures and descriptions containing inconsistent path equations is defined as follows:

$$\langle Q, \bar{q}, \theta, \delta, \leftrightarrow \rangle \models \pi_1 \doteq \pi_2 \text{ if and only if } \delta(\pi_1, \bar{q}) \leftrightarrow \delta(\pi_2, \bar{q})$$

4.4 Parsing Issues

A unification-based chart parser [Popowich and Vogel, 1991] can be adapted so that it employs the robust unification operation described here. Over-generation, a classic problem for robust parsers, won’t pose too large a

---

\(^4\)This definition has much in common with Carpenter’s definition of an inequated feature structure [Carpenter, 1992, p.112], since in both definitions the original feature structure tuple is augmented with an anti-reflexive and symmetric relation. Carpenter’s approach to negation is conservative just as my approach to inconsistency is. According to [Carpenter, 1992, p.112], “an inequated feature structure consists of an ordinary feature structure with an additional inequality relation between the nodes which are known to be distinct.” Similarly, a robust feature structure consists of an ordinary feature structure with an additional inconsistent identity relation between the nodes which are known to be inconsistent.
threat due to the restricted nature of robust unification: the only form of ill-formedness tolerated by robust unification is agreement errors.

[Douglas, 1995] describes a constraint relaxation based approach to parsing in PATR-II. For each rule in the grammar, the constraints associated with each rule are stated to be either obligatory or optional. Only the optional constraints can be relaxed.\footnote{[Douglas, 1995] also describes a variant to this approach which is error-based rather than rule-based. Here the category types associated with particular grammar errors are declared and these are then compiled into a rule-based relaxation schedule. This method is more efficient since the individual processing of each rule in the grammar is avoided.} Douglas's approach is similar to the robust unification approach outlined in this paper since both approaches involve the invocation of constraint relaxation on the discovery of a parse failure. However, there are several differences between the two approaches: the robust unification approach is specifically tailored towards agreement errors whereas the approach put forward by [Douglas, 1995] deals with constraint violation errors in general; in my approach individual grammar constraints do not need to be declared in advance as being either optional or obligatory; finally, in Douglas's approach an error is sourced by looking at the constraints that were relaxed. In the robust unification approach the "inconsistent identity" relationship is used to keep track of errors.

5 Non-Agreemen\textsuperscript{t} Errors

Consider the sentence I seen. When an attempt is made to classically unify the information contained in this sentence with the information contained in the grammar, a type clash will occur since the main verb in the sentence is not in its finite form, a constraint which most grammars of English will enforce. If we related the two inconsistent nodes using "inconsistent identity", we would expect the resulting structure to look something like the structure illustrated in Fig. 1.10. But this structure violates the definition
of a robust feature structure given in Section 4.3 because a node (the one labelled with the type vp) and a feature (the feature VFORM) have two nodes as their value (the node labelled pastp and the one labelled finite), which can’t be the case if δ is defined to be a function. Looking at this structure, there is no way of knowing at which node the error is located. This structure would never be produced by the robust unification operation described in Section 4.2 since this operation only relates two nodes in an inconsistent identity when the two nodes to be related originate from two different constituents which are required by a grammatical constraint to share a value for a feature. This situation does not arise when parsing the sentence I seen.

I propose, instead, that the structure in Fig. 1.11 is produced. That is, in the event of a clash between what the grammar says ought to be the case and what actually is the case, the grammar takes precedence. Thus, for errors like this one where a non-agreement constraint has been violated, the error is automatically corrected by a form of default unification [Briscoe, de Paiva and Copestake, 1993, Lascarides and Copestake, 1999].

The decision to distinguish constraint violation errors which involve agreement and those which don’t, makes sense if we take into account the differences between agreement and non-agreement errors. Agreement errors introduce a form of ambiguity and thus it is necessary that both values involved in the clash be retained until the wider discourse context can provide disambiguating information. Take for example the sentence Where did this women learn such arrogance?, given in Section 2. Without any further contextual information it is impossible to tell whether this sentence means Where did these women learn such arrogance? or Where did this woman learn such arrogance? Non-agreement errors involve no such ambiguity – the speaker/writer has simply used the “wrong” grammatical form of a word.

6 Some disambiguating strategies for agreement errors are provided by [Genthial, Courtin and Ménézo, 1994].

6 Conclusion

The work presented in this paper is work in progress and there are still many issues to be explored. Valuable insights could be obtained, for example, by conducting a thorough corpus-based analysis of constraint violation errors. It would also be interesting to see if any of the research carried out in the domain of robust speech recognition could interact with the ideas presented in this paper. The ramifications of employing two types of unification, “robust unification” based on “inconsistent identity” for agreement errors and a form of default unification for non-agreement errors, have yet to be explored. However the idea of a robust feature structure where inconsistency is retained in a consistent way is worthy of further investigation.

---

Some disambiguating strategies for agreement errors are provided by [Genthial, Courtin and Ménézo, 1994].
Bibliography


