Features and Values

Lauri Karttunen
University of Texas at Austin
Artificial Intelligence Center
SRI International
and
Center for the Study of Language and Information
Stanford University

Abstract

The paper discusses the linguistic aspects of a new general purpose facility for computing with features. The program was developed in connection with the course I taught at the University of Texas in the fall of 1983. It is a generalized and expanded version of a system that Stuart Shieber originally designed for the PATR-II project at SRI in the spring of 1983 with later modifications by Fernando Pereira and me. Like its predecessors, the new Texas version of the "DG (directed graph)" package is primarily intended for representing morphological and syntactic information but it may turn out to be very useful for semantic representations too.

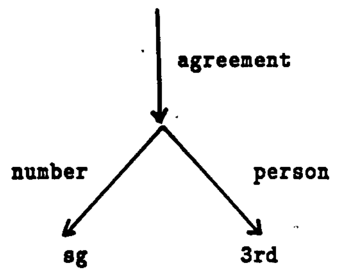
1. Introduction

Most schools of linguistics use some type of feature notation in their phonological, morphological, syntactic, and semantic descriptions. Although the objects that appear in rules and conditions may have atomic names, such as "k," "NP," "Subject," and the like, such high-level terms typically stand for collections of features. Features, in this sense of the word, are usually thought of as attribute-value pairs: [person: 1st], [number: sg], although singleton features are also admitted in some theories. The values of phonological and morphological features are traditionally atomic; e.g. 1st, 2nd, 3rd; they are often binary: +, -. Most current theories also allow features that have complex values. A complex value is a collection of features, for example:

$$\left[\text{agreement:}\ \begin{bmatrix}\text{person: 3rd}\\ \text{number: sg}\end{bmatrix}\right]$$

Lexical Functional Grammar (LFG) [Kaplan and Bresnan, 83], Unification Grammar (UG) [Kay, 79], Generalized Phrase Structure Grammar (GPSG) [Gazdar and Pullum, 82], among others, use complex features.

Another way to represent feature matrices is to think of them as directed graphs where values correspond to nodes and attributes to vectors:



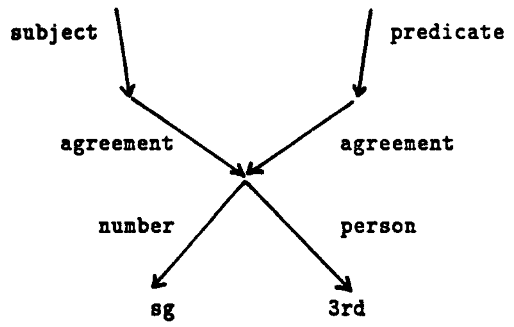
In graphs of this sort, values are reached by traversing paths of attribute names. We use angle brackets to mark expressions that designate paths. With that convention, the above graph can also be represented as a set of equations:

```
<agreement number> = sg
<agreement person> = 3rd
```

Such equations also provide a convenient way to express conditions on features. This idea lies at the heart of UG, LFG, and the PATR-II grammar for English [Shieber, *et al.*, 83] constructed at SRI. For example, the equation

```
<subject agreement> = <predicate agreement>
```

states that subject and predicate have the same value for agreement. In graph terms, this corresponds to a lattice where two vectors point to the same node:



In a case like this, the values of the two paths have been "unified." To represent unification in terms of feature matrices we need to introduce some new convention to distinguish between identity and mere likeness. Even that would not quite suffice because the graph formalism also allows unification of values that have not yet been assigned.

A third way to view these structures is to think of them as partial functions that assign values to attributes [Sag *et.al.*, 84].

2. Unification and Generalization

Several related grammar formalisms (UG, LFG, PATR-II, and GPSG) now exist that are based on a very similar conception of features and use unification as their basic operation. Because feature matrices (lattice nodes) are sets of attribute-value pairs, unification is closely related to the operation of forming a union of two sets. However, while the latter always yields something–at least the null set, unification is an operation that may fail or succeed. When it fails, no result is produced and the operands remain unchanged; when it succeeds, the operands are permanently altered in the process. They become the same object. This is an important characteristic. The result of unifying three or more graphs in pairs with one another does not depend on the order in which the operations are performed. They all become the same graph at the end.

If graphs A and B contain the same attribute but have incompatible values for it, they cannot be unified. If A and B are compatible, then (Unify A B) contains every attribute that appears only in A or only in B with the value it has there. If some attribute appears both in A and B, then the value of that attribute in (Unify A B) is the unification of the two values. For example,

$$A = \begin{bmatrix} \text{agreement: } \begin{bmatrix} \text{number: pl} \end{bmatrix} \end{bmatrix}$$

$$B = \begin{bmatrix} \text{agreement: } \begin{bmatrix} \text{person: 3rd} \end{bmatrix} \\ \text{case: nominative} \end{bmatrix}$$

$$\text{(Unify A B)} = \begin{bmatrix} \text{agreement: } \begin{bmatrix} \text{number: pl} \\ \text{person: 3rd} \end{bmatrix} \\ \text{case: nominative} \end{bmatrix}$$

Simple cases of grammatical concord, such as number, case and gender agreement between determiners and nouns in many languages, can be expressed straight-forwardly by stating that the values of these features must unify.

Another useful operation on feature matrices is generalization. It is closely related to set intersection. The generalization of two simple matrices A and B consists of the attribute-value pairs that A and B have in common. If the values themselves are complex, we take the generalization of those values.

For example,

$$A = \begin{bmatrix} \text{agreement: } \begin{bmatrix} \text{number: sg} \\ \text{person: 2nd} \end{bmatrix} \\ \text{case: nominative} \end{bmatrix}$$

$$B = \begin{bmatrix} \text{agreement: } \begin{bmatrix} \text{number: sg} \\ \text{person: 3rd} \\ \text{gender: masc} \end{bmatrix} \\ \text{case: genitive} \end{bmatrix}$$

$$\text{(Generalize A B)} = \begin{bmatrix} \text{agreement: } \begin{bmatrix} \text{number: sg} \end{bmatrix} \end{bmatrix}$$

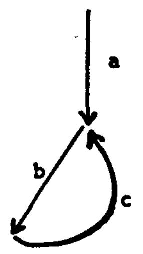
Generalization seems to be a very useful notion for expressing how number and gender agreement works in coordinate noun phrases. One curious fact about coordination is that conjunction of "I" with "you" or "he" in the subject position typically produces first person verb agreement. In sentences like "he and I agree" the verb has the same form as in "we agree. " The morphological equivalence of "he and I," "you and I," and "we" is partially obscured in English but very clear in many other languages. The problem is discussed in Section V below.

3. Limitations of Some Current Formalisms

Most current grammar formalisms for features have certain built-in limitations. Three are relevant here:

- no cyclic structures
- no negation
- no disjunction.

The prohibition against cyclicity rules out structures that contain circular paths, as in the following example.



Here the path <a b c> folds back onto itself, that is, <a> = <a b c>. It is not clear whether such descriptions should be ruled out on theoretical grounds. Whatever the case might be, current implementations of LFG, UG, or GPSG with which I am familiar do not support them.

The prohibition against negation makes it impossible to characterize a feature by saying that it does NOT have such and such a value. None of the above theories allows specifications such as the following. We use the symbol "-" to mean 'not.'

$$\begin{bmatrix} \text{case: } \begin{bmatrix} \text{-dat} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{agreement: } \neg\begin{bmatrix}\text{person: 3rd}\\ \text{number: sg}\end{bmatrix}\end{bmatrix}$$

The first statement says that case is "not dative," the second says that the value of agreement is "anything but 3rd person singular."

Not allowing disjunctive specifications rules out matrices of the following sort. We indicate disjunction by enclosing the alternative values in {}.

$$\begin{bmatrix} \text{agreement: } \left\{\begin{matrix}\begin{bmatrix}\text{gender: fem}\\ \text{number: sg}\end{bmatrix}\\ \begin{bmatrix}\text{number: pl}\end{bmatrix}\end{matrix}\right\} \\ \text{case: \{nom acc\}}\end{bmatrix}$$

The first line describes the value of case as being "either nominative or accusative." The value for agreement is given as "either feminine singular or plural." Among the theories mentioned above, only Kay's UG allows disjunctive feature specifications in its formalism. (In LFG, disjunctions are allowed in control equations but not in the specification of values.)

Of the three limitations, the first one may be theoretically justified since it has not been shown that there are phenomena in natural languages that involve circular structures (cf. [Kaplan and Bresnan, 83], p. 281). PATR-II at SRI and its expanded version at the University of Texas allow such structures for practical reasons because they tend to arise, mostly inadvertently, in the course of grammar construction and testing. An implementation that does not handle unification correctly in such cases is too fragile to use.

The other two restrictions are linguistically unmotivated. There are many cases, especially in morphology, in which the most natural feature specifications are negative or disjunctive. In fact, the examples given above all represent such cases.

The first example, [case: -dat], arises in the plural paradigm of words like "Kind" *child* in German. Such words have two forms in the plural: "Kinder" and "Kindern." The latter is used only in the plural dative, the former in the other three cases (nominative, genitive, accusative). If we accept the view that there should be just one rather than three entries for the plural suffix "-er", we have the choice between

$$\text{-er} = \begin{bmatrix}\text{number: pl}\\ \text{case: \{nom gen acc\}}\end{bmatrix}$$

$$\text{-er} = \begin{bmatrix}\text{number: pl}\\ \text{case: } [\text{-dat}]\end{bmatrix}$$

The second alternative seems preferrable given the fact that there is, in this particular declension, a clear two-way contrast. The marked dative is in opposition with an unmarked form representing all the other cases.

The second example is from English. Although the features "number" and "person" are both clearly needed in English verb morphology, most verbs are very incompletely specified for them. In fact, the present tense paradigm of all regular verbs just has two forms of which one represents the 3rd person singular ("walks") and the other ("walk") is used for all other persons. Thus the most natural characterization for "walk" is that it is not 3rd person singular. The alternative is to say, in effect, that "walk" in the present tense has five different interpretations.

The system of articles in German provides many examples that call for disjunctive feature specifications. The article "die," for example, is used in the nominative and accusative cases of singular feminine nouns and all plural nouns. The entry given above succinctly encodes exactly this fact.

There are many cases where disjunctive specifications seem necessary for reasons other than just descriptive elegance. Agreement conditions on conjunctions, for example, typically fail to exclude pairs where differences in case and number are not overtly marked. For example, in German [Eisenberg, 73] noun phrases like:

des Dozenten (gen sg) *the docent's*
der Dozenten (gen pl) *the docents'.*

can blend as in

der Antrag des oder der Dozenten
the petition of the docent or docents.

This is not possible when the noun is overtly marked for number, as in the case of "des Professors" (gen sg) and "der Professoren" (gen pl):

*der Antrag des oder der Professors
*der Antrag des oder der Professoren
the petition of the professor or professors

In the light of such cases, it seems reasonable to assume that there is a single form, "Dozenten," which has a disjunctive feature specification, instead of postulating several fully specified, homonymous lexical entries. It is obvious that the grammaticality of the example crucially depends on the fact that "Dozenten" is not definitely singular or definitely plural but can be either.

4. Unification with Disjunctive and Negative Feature Specifications

I sketch here briefly how the basic unification procedure can be modified to admit negative and disjunctive values. These ideas have been implemented in the new Texas version of the PATR-II system for features. (I am much indebted to Fernando Pereira for his advice on this topic.)

Negative values are created by the following operation. If A and B are distinct, i.e. contain a different value for some feature, then (Negate A B) does nothing to them. Otherwise both nodes acquire a "negative constraint." In effect, A is marked with -B and B with -A. These constraints prevent the two nodes from ever becoming alike.

When A is unified with C, unification succeeds only if the result is distinct from B. The result of (Unify A C) has to satisfy all the negative constraints of both A and C and it inherits all that could fail in some later unification.

Disjunction is more complicated. Suppose A, B and C are all simple atomic values. In this situation C unifies with {A B} just in case it is identical to one or the other of the disjuncts. The result is C. Now suppose that A, B, and C are all complex. Furthermore, let us suppose that A and B are distinct but C is compatible with both of them as in the following:

$$A = \begin{bmatrix} \text{gender: fem} \\ \text{number: sg} \end{bmatrix}$$

$$B = \begin{bmatrix} \text{number: pl} \end{bmatrix}$$

$$C = \begin{bmatrix} \text{case: acc} \end{bmatrix}$$

What should be the result of (Unify {A B} C)? Because A and B are incompatible, we cannot actually unify C with both of them. That operation would fail. Because there is no basis for choosing one, both alternatives have to be left open. Nevertheless, we need to take note of the fact that either A or B is to be unified with C. We can do this by making the result a complex disjunction.

$$C' = \{(A\ C)\ (B\ C)\} \quad .$$

The new value of C, C', is a disjunction of tuples which can be, but have not yet been unified. Thus (A C) and (B C) are sets that consist of compatible structures. Furthermore, at least one of the tuples in the complex disjunction must remain consistent regardless of what happens to A and B. After the first unification we can still unify A with any structure that it is compatible with, such as:

$$D = \begin{bmatrix} \text{case: nom} \end{bmatrix}$$

If this happens, then the tuple (A C) is no longer consistent. A side effect of A becoming

$$A' = \begin{bmatrix} \text{gender: fem} \\ \text{number: sg} \\ \text{case: nom} \end{bmatrix}$$

is that C' simultaniously reduces to {(B C)}. Since there is now only one viable alternative left, B and C can at this point be unified. The original result from (Unify {A B} C) now reduces to the same as (Unify B C).

$$C'' = \{(B\ C)\} = \begin{bmatrix} \text{number: pl} \\ \text{case: acc} \end{bmatrix}$$

As the example shows, once C is unified with {A B}, A and B acquire a "positive constraint." All later unifications involving them must keep at least one of the two pairs (A C), (B C) unifieable. If at some later point one of the two tuples becomes inconsistent, the members of the sole remaining tuple finally can and should be unified. When that has happened, the positive constraint on A and B can also be discarded. A more elaborate example of this sort is given in the Appendix.

Essentially the same procedure also works for more complicated cases. For example, unification of {A B} with {C D} yields {(A C) (A D) (B C) (B D)} assuming that the two values in each tuple are compatible. Any pairs that could not be unified are left out. The complex disjunction is added as a positive constraint to all of the values that appear in it. The result of unifying {(A C) (B C)} with {(D F) (E F)} is {(A C D F) (A C E F) (B C D F) (B C E F)}, again assuming that no alternative can initially be ruled out.

As for generalization, things are considerably simpler. The result of (Generalize A B) inherits both negative and positive constraints of A and B. This follows from the fact that the generalization of A and B is the maximal subgraph of A and B that will unify with either one them. Consequently, it is subject to any constraint that affects A or B. This is analogous to the fact that, in set theory,

$$(A - C) \cap (B - D) = (A \cap B) - (C \cup D) \quad .$$

In our current implementation, negative constraints are dropped as soon as they become redundant as far as unification is concerned. For example, when [case: acc] is unified with with [case: -dat], the resulting matrix is simply [case: acc]. The negative constraint is eliminated since there is no possibility that it could ever be violated later. This may be a wrong policy. It has to be modified to make generalization work as proposed in Section V for structures with negative constraints. If generalization is defined as we have suggested above, negative constraints must always be kept because they never become redundant for generalization.

When negative or positive constraints are involved, unification obviously takes more time. Nevertheless, the basic algorithm remains pretty much the same. Allowing for constraints does not significantly reduce the speed at which values that do not have any get unified in the Texas implementation.

In the course of working on the project, I gained one insight that perhaps should have been obvious from the very beginning: the problems that arise in this connection are very similar to those that come up in logic programming. One can actually use the feature system for certain kind of inferencing. For example, let Mary, Jane, and John have the following values:

$$\text{Mary} = \begin{bmatrix} \text{hair: blond} \end{bmatrix}$$

$$\text{Jane} = \begin{bmatrix} \text{hair: dark} \end{bmatrix}$$

$$\text{John} = \begin{bmatrix} \text{sister: \{Jane Mary\}} \end{bmatrix}$$

If we now unify John with

```
[sister: [eyes: blue]],
```

both Jane and Mary get marked with the positive constraint that at least one of them has blue eyes. Suppose that we now learn that Mary has green eyes. This immediately gives us more information about John and Jane as well. Now we know that Jane's eyes are blue and that she definitely is John's sister. The role of positive constraints is to keep track of partial information in such a way that no inconsistencies are allowed and proper updating is done when more things become known.

5. Future prospects: Agreement in Coordinate Structures

One problem of long standing for which the present system may provide a simple solution is person agreement in coordinate noun phrases. The conjunction of a 1st person pronoun with either 2nd or 3rd person pronoun invariably yields 1st person agreement. "I and you" is equivalent to "we," as far as agreement is concerned. When a second person pronoun is conjoined with a third person NP, the resulting conjunction has the agreement properties of a second person pronoun. Schematically:

```
1st + 2nd = 1st
1st + 3rd = 1st
2nd + 3rd = 2nd.
```

Sag, Gazdar, Wasow, and Weisler [84] propose a solution which is based on the idea of deriving the person feature for a coordinate noun phrase by generalization (intersection) from the person features of its heads. It is obvious that the desired effect can be obtained in any feature system that uses the fewest features to mark 1st person, some additional feature for 2nd person, and yet another for 3rd person. Because generalization of 1st and 2nd, for example, yields only the features that two have in common, the one with fewest features wins.

Any such solution can probably be implemented easily in the framework outlined above. However, this proposal has one very counterintuitive aspect: markedness hierarchy is the reverse of what traditionally has been assumed. Designating something as 3rd person requires the greatest number of feature specifications. In the Sag *et al.* system, 3rd person is the most highly marked member and 1st person the least marked member of the trio. Traditionally, 3rd person has been regarded as the unmarked case.

In our system, there is a rather simple solution under which the value of person feature in coordinate NPs is derived by generalization, just as Sag it et al. propose, which nevertheless preserves the traditional view of markedness. The desired result can be obtained by using negative constraints rather than additional features for establishing a markedness hierarchy. For example, the following feature specifications have the effect that we seek.

$$\text{1st} = \begin{bmatrix}\text{conversant: } + \\ \text{speaker: } +\end{bmatrix}$$

$$\text{2nd} = \begin{bmatrix}\text{conversant: } + \\ \text{speaker: } -\end{bmatrix}$$

$$\text{3rd} = \begin{bmatrix}\text{conversant: } - \\ \text{speaker: } -\end{bmatrix}$$

The corresponding negative constraints are:

$$\text{1st} = \left[-\begin{bmatrix}\text{conversant: } - \\ \text{speaker: } -\end{bmatrix}\right]$$

$$\text{2nd} = \left[-[\text{conversant: } -]\right]$$

$$\text{3rd} = \text{(no constraints)}$$

Assuming that generalization with negative constraints works as indicated above, i.e. negative constraints are always inherited, it immediately follows that the generalization of 1st person with any other person is compatible with only 1st person and that 2nd person wins over 3rd when they are combined. The results are as follows.

$$\text{1st} + \text{2nd} = \begin{bmatrix}\text{conversant: } + \\ -\begin{bmatrix}\text{conversant: } - \\ \text{speaker: } -\end{bmatrix}\end{bmatrix}$$

$$\text{1st} + \text{3rd} = \left[-\begin{bmatrix}\text{conversant: } - \\ \text{speaker: } -\end{bmatrix}\right]$$

$$\text{2nd} + \text{3rd} = \begin{bmatrix}\text{speaker: } - \\ -[\text{conversant: } -]\end{bmatrix}$$

Note that the proper part of 1st+2nd excludes 3rd person. It is compatible with both 1st and 2nd person but the negative constraint rules out the latter one. In the case of 1st+3rd, the negative constraint is compatible with 1st person but incompatible with 2nd and 3rd. In the last case, the specification [speaker: -] rules out 1st person and the negative constraint -[conversant: -] eliminates 3rd person.

When negative constraints are counted in, 1st person is the most and 3rd person the least marked member of the three. In that respect, the proposed analysis is in line with traditional views on markedness. Another relevant observation is that the negative constraints on which the result crucially depends are themselves not too unnatural. In effect, they say of 1st person that it is "neither 2nd nor 3rd" and that 2nd person is "not 3rd."

It will be interesting to see whether other cases of markedness can be analyzed in the same way.

6. Acknowledgements

I am indebted to Martin Kay for introducing me to unification and to Fernando Pereira, Stuart Shieber, Remo Pareschi, and Annie Zaenen for many insightful suggestions on the project.

References

Eisenberg, Peter, "A Note on Identity of Constituents," *Linguistic Inquiry 4:3*, 417-20 (1973).

Gazdar, Gerald and G. Pullum. "Generalized Phrase Structure Grammar: A Theoretical Synopsis." Indiana University Linguistics Club, Bloomington, Indiana (1982).

Kaplan, Ronald M. and Joan Bresnan, 1983: "Lexical-Functional Grammar: A Formal System for Grammatical Representation," Ch.4 in J. Bresnan, *The Mental Representation of Grammatical Relations* (ed.), Cambridge, MIT Press.

Kay, Martin, 1979: "Functional Grammar." *Proceedings of the Fifth Annual Meeting of the Berkeley Linguistic Society*, Berkeley Linguistic Society, Berkeley, California (February 17-19, 1979), pp. 142-158.

Pereira, Fernando and Stuart Shieber, 1984: "The semantics of Grammar Formalism Seen as Computer Languages." *Proceedings of the Tenth International Conference on Computational Linguistics*, Stanford University, Stanford California (4-7 July, 1984).

Sag, Ivan, Gerald Gazdar, Thomas Wasow, and Steven Weisler, 1984: "Coordination and How to Distinguish Categories." CLSI Report No. 3. Center for the Study of Language and Information, Stanford, Ca., (March 1984).

Shieber, S., H. Uszkoreit, F. Pereira, J. Robinson, and M. Tyson, 1983: "The Formalism and Implementation of PATR-II," in B. Grosz and M. Stickel, *Research on Interactive Acquisition and Use of Knowledge*, SRI Final Report 1894, SRI International, Menlo Park, California (November 1983).

A. Appendix: Some Examples of Unification

(These examples were produced using the Texas version of the DG package.)

$$\texttt{die} = \begin{bmatrix} \text{infl:} & \begin{bmatrix} \text{case:} & \{\text{nom acc}\} \\ \text{agr:} & \left\{ \begin{array}{l} \begin{bmatrix} \text{gender:} & \text{fem} \\ \text{number:} & \text{sg} \end{bmatrix} \\ \begin{bmatrix} \text{number:} & \text{pl} \end{bmatrix} \end{array} \right\} \end{bmatrix} \end{bmatrix}$$

$$\texttt{Kinder} = \begin{bmatrix} \text{infl:} & \begin{bmatrix} \text{case:} & [\text{-dat}] \\ \text{agr:} & \begin{bmatrix} \text{gender:} & \text{neut} \\ \text{number:} & \text{pl} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\texttt{die Kinder} = \begin{bmatrix} \text{infl:} & \begin{bmatrix} \text{case:} & \{\text{nom acc}\} \\ \text{agr:} & \begin{bmatrix} \text{gender:} & \text{neut} \\ \text{number:} & \text{pl} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\texttt{den} = \begin{bmatrix} \text{infl:} & \left\{ \begin{array}{l} \begin{bmatrix} \text{case:} & \text{acc} \\ \text{agr:} & \begin{bmatrix} \text{gender:} & \text{masc} \\ \text{number:} & \text{sg} \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \text{case:} & \text{dat} \\ \text{agr:} & \begin{bmatrix} \text{number:} & \text{pl} \end{bmatrix} \end{bmatrix} \end{array} \right\} \end{bmatrix}$$

den Kinder = *FAILS*

$$\texttt{den Kindern} = \begin{bmatrix} \text{infl:} & \begin{bmatrix} \text{case:} & \text{dat} \\ \text{agr:} & \begin{bmatrix} \text{gender:} & \text{neut} \\ \text{number:} & \text{pl} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\texttt{I} = \begin{bmatrix} \text{infl:} & \begin{bmatrix} \text{case:} & \text{nom} \\ \text{agr:} & \begin{bmatrix} \text{number:} & \text{sg} \\ \text{person:} & \text{1st} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\texttt{he} = \begin{bmatrix} \text{infl:} & \begin{bmatrix} \text{case:} & \text{nom} \\ \text{agr:} & \begin{bmatrix} \text{gender:} & \text{masc} \\ \text{number:} & \text{sg} \\ \text{person:} & \text{3rd} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\texttt{do} = \begin{bmatrix} \text{infl:} & \begin{bmatrix} \text{tense:} & \text{present} \\ \text{agr:} & \begin{bmatrix} - \begin{bmatrix} \text{number:} & \text{sg} \\ \text{person:} & \text{3rd} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\texttt{I do} = \begin{bmatrix} \text{infl:} & \begin{bmatrix} \text{tense:} & \text{present} \\ \text{case:} & \text{nom} \\ \text{agr:} & \begin{bmatrix} \text{number:} & \text{sg} \\ \text{person:} & \text{1st} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

he do = *FAILS*

$$\texttt{x} = \left\{ \begin{array}{l} \begin{bmatrix} \text{a:} & - \\ \text{b:} & + \end{bmatrix} \\ \begin{bmatrix} \text{b:} & - \\ \text{c:} & + \end{bmatrix} \\ \begin{bmatrix} \text{a:} & + \\ \text{c:} & - \end{bmatrix} \end{array} \right\} \qquad \texttt{y} = \left\{ \begin{array}{l} \begin{bmatrix} \text{a:} & + \\ \text{b:} & - \end{bmatrix} \\ \begin{bmatrix} \text{b:} & + \\ \text{c:} & + \end{bmatrix} \end{array} \right\} \qquad \texttt{z} = \begin{bmatrix} \text{a:} & + \\ \text{c:} & + \end{bmatrix}$$

(Unify x y)

$$= \left\{ \begin{array}{l} \left(\begin{array}{l} \begin{bmatrix} \text{b:} & - \\ \text{c:} & + \end{bmatrix} \\ {}^{1}\begin{bmatrix} \text{a:} & + \\ \text{b:} & - \end{bmatrix} \end{array} \right) \\ \left(\begin{array}{l} \begin{bmatrix} \text{a:} & + \\ \text{c:} & - \end{bmatrix} \\ \langle 1 \rangle \end{array} \right) \\ \left(\begin{array}{l} \begin{bmatrix} \text{a:} & - \\ \text{b:} & + \end{bmatrix} \\ \begin{bmatrix} \text{b:} & + \\ \text{c:} & + \end{bmatrix} \end{array} \right) \end{array} \right\}$$

(Unify (Unify x y) z)

$$= \begin{bmatrix} \text{a:} & + \\ \text{b:} & - \\ \text{c:} & + \end{bmatrix}$$