

# Conjunctive Vector Representations for Set Valued Feature Descriptions

Michael Carl

IAI, Martin-Luther-Straße 14, 66111 Saarbrücken, Germany

Set valued feature descriptions can be coded into bit vectors to be efficiently treated by unification. However, the coding must be designed to allow cooccurrence restrictions on possible combinations of values that attributes may take.

$$r1 : \{\text{head} = (\{\text{gen} = \text{fem}, \text{case} = (\text{gen}; \text{dat})\}; \{\text{gen} = \text{masc}, \text{case} = \text{nom}\})\} \quad (1)$$

In  $r1$  (e.g. the description of the German article “der”), either the value of the attribute “head/gen” is “fem” and the attribute “head/case” has one of the values “gen” or “dat”, or the value of “head/gen” is “masc” if and only if the value of “head/case” is “nom”.

A coding technique has been proposed ([Pul94]) that codes the rules into a DNF-like vector representation where each bit in the vector codes a unique set of values. The length  $N_a$  of the vector required to store the values of an attribute  $a$  according to the DNF-like vector coding can be determined by equation (2):

$$N_a = |a| + \prod_{s \in S} N_s \quad (2)$$

where  $|a|$  is the number of possible atomic values of an attribute  $a$  and  $S$  is the set of attributes  $s$  in the scope of  $a$ , if  $a$  allows complex values.

For big NLP-systems this approach requires huge data structures. In practice, most parts of the feature structure are not concerned with cooccurrence restrictions. Thus, in an implementation of the CAT2 MT system (cf. [Str96]), although the attribute “sem” allows more than  $6 * 10^9$  different value combinations, only around 90 combinations appear in the lexicon. In terms of DNF-like coding this means that by far most of the bits never are switched on.

To avoid such (linguistically motivated but computational unnecessary) redundancies, we propose a CNF-like coding technique that requires much shorter vectors. These vectors contain two sections: a feature section that codes the atomic values and a constraint section that codes the cooccurrence restrictions as described in the set of rules. Both sections consist of conjunctions of bloks. A blok in a feature section codes the values of an attribute; the length of the feature section is thus equivalent to the amount of atomic values (first part of equation (3)).

$$N_a = |a| + \sum_{s \in S} N_s + \left| \bigcup_{r \in R} \text{cons}_r \right| \quad (3)$$

The length of the constraint section (last term of equation (3)) corresponds to the amount of the different constraints *cons* that appear in the set of rules  $R$ .

in Proceedings of  
ASIAN'96,  
Springer, LNCS 1179

To calculate the vector length of the attribute “sem” by means of this equation 159 bits are required; 69 bits for the feature section and 90 bits for the constraint section. In the remainder of the article, we briefly describe the coding technique and the unification procedure.

Each rule is coded into a vector of length  $N_{rule}$  according to equation (3). Each position in the vector may take one of three characters (we therefore refer to them as positions in place of bits, since bits take only two values). The position in the vector that codes the value in question is switched “on” (1). Positions of values, that are incompatible with the represented value are switched “off” (0). All others are marked “don’t care”.

A position in the feature section is switched “on” (1) if the coded value appears in the rule. It is switched “off” (0) if the coded value is incompatible with the rule, otherwise the position is marked “don’t care”.

A position in the constraint section is switched “off” if the rule is incompatible with the coded constraint. It is switched “on” if the rule fills the constraint i.e. if the constraint is a subset of one of the rule’s constraints. Otherwise, if the rule is merely compatible with the coded constraint the position is marked “don’t care”.

The unification of two vectors yields a target vector (according to 1. - 3.) and compatibility of both vectors is checked (4. - 5.).

1. If two “don’t care” positions are unified, a “don’t care” character is introduced into the target position.
2. If a “don’t care” position is unified with a “non don’t care” position, the “non don’t care” character is introduced into the target position.
3. If two “non don’t care” positions are unified, the target position is set to the logical AND.
4. A unification clash in the feature section occurs, if all positions in a target vectors blok are set off i.e. both unifying vectors share no common values for the attribute in question.
5. A unification clash in the constraint section occurs if all positions in a constraint blok of both unifying vectors are either switched on (1) or off (0) and if XOR of all positions yields “0”, i.e. both vectors have different cooccurrence requirements.

We believe that the presented coding technique is superior to a DNF-like vector coding, since the vector length remains in treatable limits.

## References

- [Pul94] Steven Pulman. Expressivity of lean formalisms. In Stella Markantonatou and Louisa Sadler, editors, *Grammatical Formalisms: Issues in Migration and Expressivity*, Studies in Machine Translation and Natural Language Processing, Volume 4. Commission of the European Communities, Brussels, Luxembourg, 1994.
- [Str96] Oliver Streiter. *Linguistic Modeling for Multilingual Machine Translation*. Shaker Verlag, Aachen, BRD, 1997.