3. Efficient Wide-Coverage Realization

3.1 Chart Initialization

Input to the generator consists of an MRS, i.e. a list of elementary predications (EPs). A predication's origin may be a lexeme, as in _athlete_n(x_5), or a syntactic rule, such as a predication expressing apposition. Before syntactic combination can occur, the locations of the lexemes and rules (henceforth *grammar entities* or GEs) which will participate in the procedure must be determined. Wide-coverage grammars offering detailed analyses can support many thousands of elementary predication types (the LinGO ERG has upwards of $10,000$). Therefore, it is necessary to find the minimal required set of GEs in an efficient manner.

Semantically Contentful Grammar Entities We construct a *semantic index* to efficiently locate GEs as follows: As a preprocessing step, a hash table is constructed associating to each predication name the complete list of grammar entities which can originate the corresponding predication. Then at runtime, the system extracts the set of relevant grammar entities by looking up each predication name in the input MRS.

The problem is somewhat complicated by the fact that predications types are organized in a heirarchy; for instance in the LinGO ERG we have predsort ¿ message_m_rel ¿ prpstn_m_rel. If the input semantics specifies message_m_rel, the generation result must be a subtype of message_m_rel, but in fact any supertype of message_m_rel can be specialized into one of its subtypes during unification. Therefore any subtype or supertype of the supplied predication must be considered a candidate during chart initialization.

The basic data structure of the index is a *semantic index predication node* or SPN. An SPN is associated to a particular predication name (e.g. message_m_rel). and contains pointers the SPNs corresponding to every subtype and supertype of the predication for ease of access. This basic network of SPNs is constructed directly from the semantic type heirarchy. A reference to each EP of each GE (lexeme or rule) is then recorded in the SPN corresponding to that EP. The runtime algorithm to quickly instantiate all alignments of the GE's to the input MRSconsists of two phases: *activation* followed by *enumeration*:

- in *activation*, each EP in the input MRSis looked up in the semantic index. The resulting SPN is activated by adding a reference to the input EP to the SPN's *activation list*. Every supertype and every subtype of the predication is also activated in the same fashion, using the list of related predications stored in the SPN. A list of activated SPNs is maintained. At this point, every SPN in the index has an activation list which enumerates the set of input EPs with which its predication is compatible.

- in *enumeration*, each GE (say g) of each activated SPN is considered. Since a GE may introduce multiple predications, we first check that the SPNs corresponding to every predication introduced by g are in fact activated. We then know that g should be instantiated into the generation chart.

Adding a grammar entry to the chart without *specializing* it will result in the generation of many irrelevant output sentences whose semantics are not subsumed by the input (e.g. "The cat chased the dog." instead of "The dog chased the cat."). In general the time spent on irrelevant analyses grows exponentially with the number of GEs added to the chart. Therefore we specialize each GE introduced to correspond to the constraints required by the input MRS. A more detailed discussion of specialization follows in Section 3.2 below.

If more than one EP in the input MRS has the same predication type, the constraints will be different, so separate specialized instances of the corresponding GE must be added to the chart for each EP. We therefore modify the enumeration phase above to instantiate a specialized copy of g into the chart for each alignment of the EPs introduced by g with the input MRS. The set of alignments can be efficiently computed as the cross product over each introduced EP of the union of the activation lists of the SPN for that EP and the activation lists of all the subtype and supertype SPNs.

Semantically Vacuous Grammar Entities

3.2 Relating Chart Edges and Semantic Components

Once lexical lookup is complete and up until a final, post-generation comparison of results to the input MRS, the core phases of our generator exclusively operate on typed feature structures (which are associated to chart edges). For efficiency reasons, our algorithm avoids any complex operations on the original logical-form input MRS. In order to best guide the search from the input semantics, however, we employ two techniques that relate components of the logical form to corresponding sub-structures in the feature structure (FS) universe: (i) Skolemization of variables and (ii) indexing by EP coverage. Of these, only the latter we find commonly discussed in the literature, but we expect some equivalent of making variables ground to be present in most implementations.

As part of the process of looking up lexical items and grammar rules introducing semantics in order to initialize the generator chart, all FS correspondences to logical variables from the input MRS are made 'ground' by specializing the relevant sub-structure with Skolem constants uniquely reflecting the underlying variable, for example adding constraints like [SKOLEM "$x5$"] for all occurrences of x_5 from our example MRS. Skolemization, thus, assumes that distinct variables from the input MRS, where supplied, cannot become co-referential during generation. Enforcing variable identity at the FS level makes sure that composition (by means of FS unification) during rule applications is compatible to the input semantics. In addition, it enables efficient pre-unification filtering (see 'quick-check' below), and is a prerequisite for our index accessibility test described in Section 3.5 below.

In chart *parsing*, edges are stored into and retrieved from the chart data structure on the basis of their string *start* and *end* positions. This ensures that the parser will only retrieve pairs of chart edges that cover compatible segments of the input string (i.e. that are adjacent with respect to string position). In chart *generation*, Kay (1996) proposed indexing the chart on the basis of logical variables, where each variable denotes an individual entity in the input semantics, and making the edge coverage compatibility check a filter. Edge coverage (with respect to the EPs in the input semantics) would be encoded as a bit vector, and for a pair of edges to be combined their corresponding bit vectors would have to be disjoint.

We implement Kay's edge coverage approach, using it not only when combining active and inactive edges, but also for two further tasks in our approach to realization:

- in the second phase of chart generation to determine which intersective modifier(s) can be adjoined into a partially incomplete subtree; and
- as part of the test for whether one edge subsumes another, for local ambiguity factoring (see Section 3.3 below)[4].

[4] We therefore have four operations on bit vectors representing EP coverage ($\mathcal{C}$) in chart edges:

- concatenation of edges e_1 and $e_2 \rightarrow e_3$: $\mathcal{C}(e_3) = \text{OR}(\mathcal{C}(e_1), \mathcal{C}(e_2))$;
- can edges e_1 and e_2 combine? $\text{AND}(\mathcal{C}(e_1), \mathcal{C}(e_2)) = 0$;