Language Production: the Source of the Dictionary

David D. McDonald
University of Massachusetts at Amherst
April 1980

COINS Technical Report 82-7

Abstract

Ultimately in any natural language production system the largest amount of human effort will go into the construction of the *dictionary*: the data base that associates objects and relations in the program's domain with the words and phrases that could be used to describe them. This paper describes a technique for basing the dictionary directly on the semantic abstraction network used for the domain knowledge itself, taking advantage of the inheritance and specialization machanisms of a network formalism such as KL-ONE. The technique creates considerable economies of scale, and makes possible the automatic description of individual objects according to their position in the semantic net. Furthermore, because the process of deciding what properties to use in an object's description is now given over to a common procedure, we can write general-purpose rules to, for example, avoid redundancy or grammatically awkward constructions.

This paper was published in the Proceedings of the 1981 Annual Meeting of the Association for Computational Linguistics, Stanford University, June 1981.

Language Production: the Source of the Dictionary

David D. McDonald

University of Massachusetts at Amherst

April 1980

COINS Technical Report 82-7

Do not film

Abstract

Ultimately in any natural language production system the largest amount of human effort will go into the construction of the *dictionary*: the data base that associates objects and relations in the program's domain with the words and phrases that could be used to describe them. This paper describes a technique for basing the dictionary directly on the semantic abstraction network used for the domain knowledge itself, taking advantage of the inheritance and specialization machanisms of a network formalism such as KL-ONE. The technique creates considerable economies of scale, and makes possible the automatic description of individual objects according to their position in the semantic net. Furthermore, because the process of deciding what properties to use in an object's description is now given over to a common procedure, we can write general-purpose rules to, for example, avoid redundancy or grammatically awkward constructions.

This paper will appear in the proceedings of the 1981 annual meeting of the Association for Computational Linguistics, Standford University, June 30-July 2.

Regardless of its design, every system for natural language production begins by selecting objects and relations from the speaker's internal model of the world, and proceeds by choosing an English phrase to describe each selected item, combining them according to the properties of the phrases and the constraints of the language's grammar and rhetoric. To do this the system must have a data base of some sort in which the objects it will talk about are somewhow associated with the appropriate word or phrase (or with procedures that will construct them). I will refer to such a data base as a *dictionary.*

Every production system has a dictionary in one form or another, and its compilation is probably the single most tedious job that the human designer must perform. In the past, typically every object and relation has been given its own individual "lex" property with the literal phrase to be used; no attempt was made to share criteria or sub-phrases between properties; and there was a tacit assumtion that the phrase would have the right form and content in any of the contexts that the object will be mentioned. (For a review of this literature, see [5].) However, dictionaries built in this way become increasingly harder to maintain as programs become larger and their discourse more sophisticated. We would like instead some way to tie the extention of the dictionary directly to the extention of the program's knowledge base; then, as the knowledge base expands the dictionary will expand with it with only a minimum of additional effort.

This paper describes a technique for adapting a semantic abstraction hierarchy of the sort provided by KL-ONE [1] to function directly as a dictionary for my production system MUMBLE [4]. Its goal is largely expositional in the sense that while the technique is fully specified and proto-types have been run, many implementation questions remain to be explored and it is thus premature to present it as a polished system for others to use; instead, this paper is intended as a presentation of the issues—potential economies—that the technique is addressing. In particular, given the intimate relationship between the choice of architecture in the network formalism used and the ability of the dictionary to incorporate linguistically useful generalizations and utilities, this presentation may suggest an additional criteria for network design, namely to make it easier to talk about the objects the network

The basic idea of "piggybacking" the dictionary onto the speaker's regular semantic net can be illustrated very simply: Consider the KL-ONE network in figure one, a fragment taken from a conceptual taxonomy for augmented transition nets (given in [1]). The dictionary is to provide the means to describe individual concepts (filled ellipses) on the basis of their links to generic concepts (empty ellipses) and their functional roles (squares), as shown there for the individual concept "C205". The default English description of C205 (i.e. "*the jump arc from S/NP to S/DCL*") is created recursively from descriptions of the three network relations that C205 participates in, i.e. its "superconcept" link to the concept "jump-arc", and its two role-value relations: "source-state(C205)=S/NP" and "next-state(C205)=S/DCL". Intuitively, we want to associate each of the network objects with an English phrase: the concept "arc" with the word "*arc*", the "source-state" role relation with the phrase "*C205 comes from S/NP*" (note the embedded references), and so on.

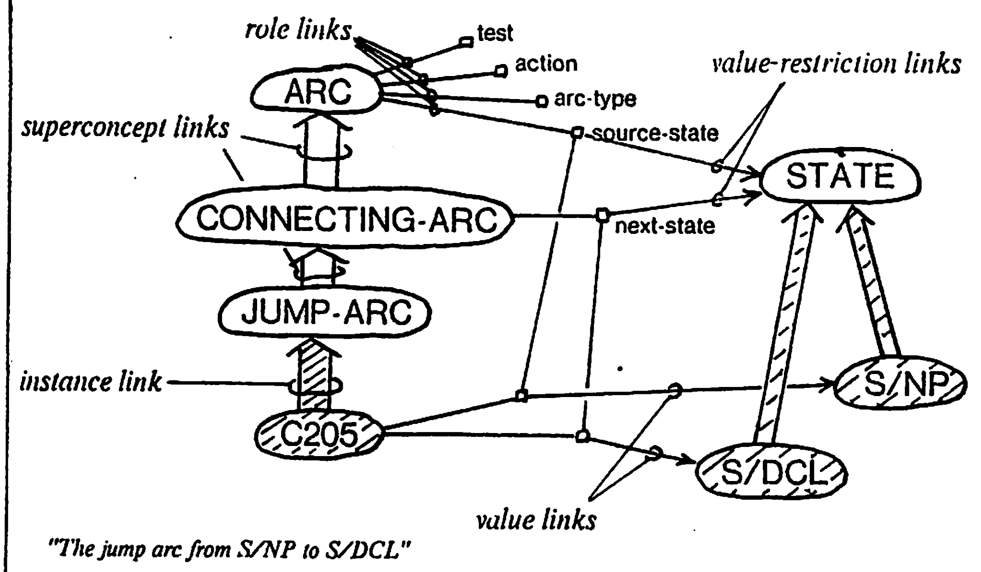

Figure One: the speaker's original network

The machinery that actually brings about this association is, of course, much more elaborate, involving three different meta-level networks describing the whole of the original, "domain" network, as well as an explicit representation of the English grammar (i.e. it is itself expressed in KL-ONE).

What does this rather expensive[1] computational machinery purchase? There are numerous benefits: The most obvious is the economy of scale within the dictionary that is gained by drawing directly on the economies already present in the network: a one-time linguistic annotation of the network's generic concepts and relations can be passed down to describe arbitrary numbers of instantiating individuals by following general rules based on the geography of the network. At the same time, the dictionary "entry" for a object in the network may be specialized and hand-tailored, if desired, in order to take advantage of special words or idiomatic phrases or it may inherit partial default realizations, e.g. just for determiners or adverbial modifiers, while specializing its other parts. More generally, because we have now reified the process of collecting the "raw material" of the production process (i.e. scanning the network), we can impose rules and constraints on it just as though it were another part of the production planning process; we can develop a dictionary *grammar* entirely analogous to our grammar of English. This allows us to filter or transform the collection process under contextual control according to general rules, and thereby, among other things, automatically avoid redundancies or violations of grammatical constraints such as complex-NP.

1. What is expensive to represent in an explicit, declarative structure need not be expensive when translated into procedural form. I do not seriously expect anyone to implement such a dictionary by interpreting the KL-ONE structures themselves; given our present hardware such a tact would be hopelessly inefficient. Instead, a compilation process will in effective "compact" the explicit version of the dictionary into an expeditious, space-expensive (i.e. heavily redundant) version that performs each inheritance only once and then runs as an efficient, self-contained procedure.

In order to adapt a semantic net for use as a dictionary we must determine three points: (1) What type of linguistic annotation to use—just what is to be associated with the nodes of a network? (2) How annotations from individual nodes are to be accumulated—what dictates the pattern in which the network is scanned? (3) How the accumulation process is made sensitive to context. These will be the focus of the rest of the paper.

The three points of the design are, of course, mutually dependent, and are further dependent on the requirements of the dictionary's employers, the planning and linguistic realization components of the production system. In the interests of space I will not go into the details of these components in this paper, especially as this dictionary design appears to be useful for more than just my own particular production system. My assumptions are: (1) that the output of the dictionary (the input to my realization component) is a representation of a natural language phrase as defined by the grammar and has both words and other objects from the domain network as its terminals (the embedded domain objects correspond to the variable parts of the phrase, i.e. the arguments to the original network relation); and (2) that the planning process (the component that decides what to say) will specify that network objects be described either as a composition of a set of other network relations that it has explicitly selected, or else will leave the description to a default given in the dictionary.

Meta-level annotation

The basis of the dictionary is a *meta-level network* constructed so as to shadow the domain network used by the rest of the speaker's cognitive processes. This "dictionary network" describes the domain network from the point of view of the accumulation procedure and the linguistic annotation. It is itself an abstraction hierarchy, and is also expressed in KL-ONE (though see the earlier footnote). Objects in the regular network are connected by meta-links to their corresponding dictionary "entries". These entries are representations of English phrases (either a single phrase or word or a cluster of alternative phrases with some decision-criteria to select among them at run time). When we want to describe an object, we follow out its meta-link into the dictionary network and then realize the word or phrase that we find.

Specializing Generic Phrases

The entry for an object may itself have a hierarchical structure that parallels point for point the hierarchical structure of the object's description in the domain. Figure two shows the section of the dictionary network that annotates the superconcept chain from "jump-arc" to "object"; comparable dictionary networks can be built for hierarchies of roles or other hierarchical network structures. Notice how the use of an inheritance mechanism within the dictionary network (denoted by the vertical links between roles) allows us on the one hand to state the determiner decision (shown here only as a cloud) once and for all at the level of the domain concept "object", while at the same time we can accumulate or supplant lexical material as we move down to more specific levels in the domain network.

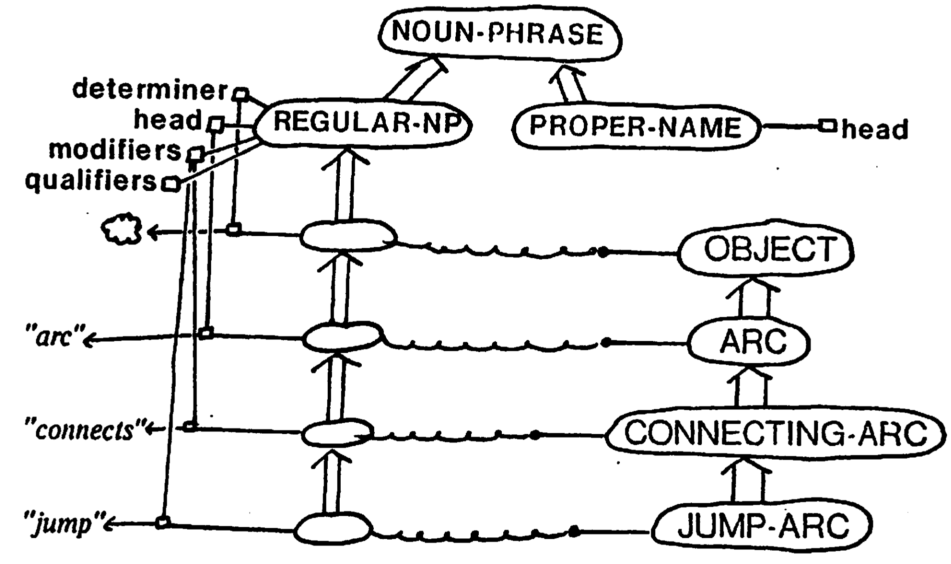

Figure Two: the meta-level dictionary network

After all the inheritance is factored in, the entry for, e.g., the generic concept "jump-arc" will describe a noun phrase (represented by an indiviual concept in KL-ONE) whose head position is filled by the word "*arc*", classifier position by "*jump*", and whose determiner will be calculated (at run time) by the same routine that calculated determiners for objects in general (e.g. it will react to whether the reference is to a generic or an individual, to how many other objects have the same description, to whether any special contrastive effects are intended, etc., see [4]).

Should the planner decide to use this entry by itself, say to produce "*C205 is [a jump arc]*", this description from the dictionary network would be converted to a proper constituent structure and integrated with the rest of the utterance under production. However, the entry will often be used in conjunction with the entries for several other domain objects, in which case it is first manipulated as a description—constraint statement—in order to determine what grammatical construction(s) would realize the objects as a group.

The notion of creating a consolidated English phrase out of the phrases for several different objects is central to the power of this dictionary. The designer is only expected to explicitly designate words for the generic objects in the domain network, while the entries for the individual objects that the generic objects describe and even the entries for a hierarchical chain such as in figure two should typically be constructable by default by following general-purpose linguistic rules and combination heuristics.

Large entries out of small ones

Figure three shows a sketch of the combination process. Here we need a dictionary entry to describe the relationship between the specific jump-arc C205 and the state it leads to, S/DCL, i.e. we want something like the sentence "*<C205> goes to <S/DCL>*", where the references in angle brackets would be ultimately replaced by their own English phrases. When the connecting role relation (in this case "next-state") can be rendered into English by a conventional pattern, we can construct a linguistic relationship to match the domain relationship by using a conventional dictionary entry for the concept-role-value relation as specialized by the specific entry for the role "next-state".

The figure shows diagramatically the relationship between the domain network relation, its meta-level description as an object in the network formalism (i.e. it is an instance of a concept linked to one of its roles linked in turn to the role value), and finally the corresponding conventional linguistic construction. The actual KL-ONE representation of this relation is considerably more elaborate since the links themselves are reified, however this sketch shows the relevant level of detail as regards what kinds of knowledge are needed in order to assemble the entry procedurally. First the domain relation is picked out and categorized; here this was done by a the conventional meta-level description of the relation in terms of the KL-ONE primitives it was built from; below we will see how a comparable categorization can be done on a purely linguistic basis. With the relation categorized, we can associated it with an entry in the dictionary network, in this case an instance of a "basic-clause" (i.e. one without any adjuncts or root-transformations). We now have determined a mapping from the entries for the components of the original domain relation to linguistic roles within a clause and have, in effect, created the dictionary entry for "next-state" which we could proceed compile for efficiency.

There is much more to be said about how the "embedded entries" can be controlled, how, for example, the planner can arrange to say either "*C205 goes to S/DCL*" or "*There is a jump arc going to S/DCL*" by dynamically specializing the description of the clause, however that would be taking us too far afield; the interested reader is referred to [4]. The point to be made here is just that the

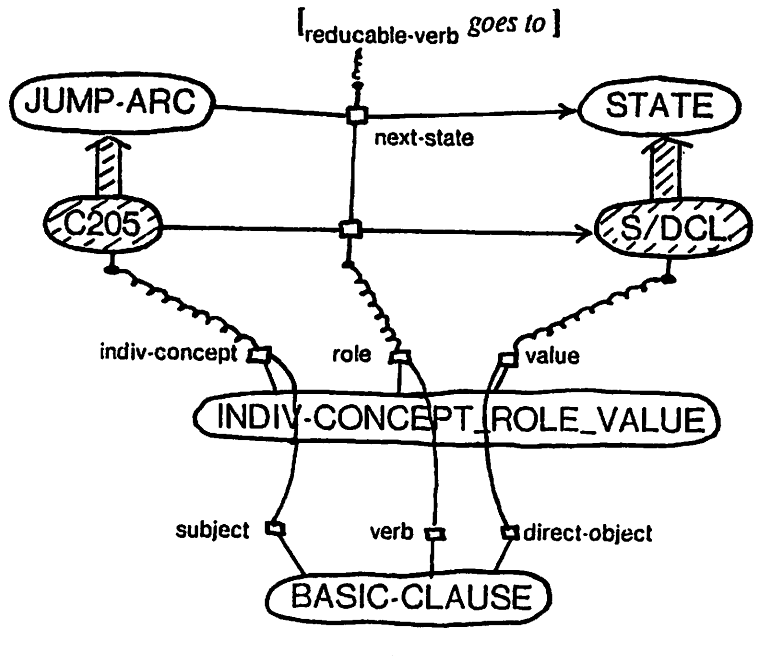

Figure Three: Combining Entries by Network Relations

writer of the dictionary has an option either to write specific dictionary entries for domain relations, or to leave them to general "macro entries" that will build them out of the entries for the objects involved as just sketched. Using the macro entries of course means that less effort will be needed over all, but using specific entries permits one to take advantage of special idioms or variable phrases that are either not productive enough or not easy enough to pick out in a standard meta-level description of the domain network to be worth writing macro entries for. A simple example would be a special entry for when one plans to describe an arc in terms of both its source and its next states: in this case there is a nice compaction available by using the verb "*connect*" in a single clause (instead of one clause for each role). Since the KL-ONE formalism has no transparent means of optionally bundling two roles into one, this compound relation has to be given its own dictionary entry by hand.

Making combinations linguistically

Up to this point, we have been looking at associations between "organic" objects or relations in the domain network and their dictionary entries for production. It is often the case however, that the speech planner will want to talk about combinations of objects or complex relations that have been assembled just for the occasion of one conversation and have no natural counterpart within the regular domain network. In a case like this there would not already be an entry in the dictionary for the new relation; however, in most cases we can still produce an integrated phrase by looking at how the components of the new relation can combine *linguistically.*

These linguistic combinations are not so much the provence of the dictionary as of my linguistic realization component, MUMBLE. MUMBLE has the ability to perform what in the early days of transformational generative grammar were referred to as "generalized transformations": the combining of two or more phrases into a single phrase on the basis of their linguistic descriptions. We have an example of this in the original example of the default description of C205 as "*the jump arc from S/NP to S/DCL*". This phrase was produced by having the default planner construct an expression indicating which network relations to combine (or more precisely, which *phrases* to combine, the phrases being taken from the entries of the relations), and then pass the expression to MUMBLE which produces the "compound" phrase on the basis of the linguistic description of the argument phrases. The expression would look roughly like this:[2]

(describe C205 as (and $[_{np}$ *the jump arc*]
$[_{clause}$ C205 $[_{reducible\text{-}vp}$ *comes from* S/NP]]
$[_{clause}$ C205 $[_{reducible\text{-}vp}$ *goes to* S/DCL]]

2. A "phrase" in a dictionary entry does not consist simply of a string of words. They are actually schemata specifying the grammatical and rhetorical relationships that the words and argument domain objects participate in according to their functional roles. The bracketed expressions shown in the expression are for expository purposes only and are modeled on the usual representation for phrase structure. Embedded objects such as "C205" or "S/NP" will be replaced by their own English phrases incrementally as their containing phrases are realized.

MUMBLE's task is the production of an object description from the raw material of a noun phrase and two clauses. To do this, it will have to match the three phrases against one of its known linguistic combination patterns, just as the individual concept, role, and value were matched by a pattern from the KL-ONE representation formalism. In this case, it characterizes the trio as combinable through the adjunction of the two clauses to the noun phrase as qualifiers. Additionally, the rhetorical label "reducable-vp" in the clauses indicates that their verbs can be omitted without losing significant information, triggering a stylistic transformation to shorten and simplify the phrase. At this point MUMBLE has a linguistic representation of its decision which is turned over to the normal realization process for completion. Exaustive details of these operations may be found in [4].

Contextual Effects

The mechanisms of the dictionary per se perform two functions: (1) the association of the "ground level" linguistic phrases with the objects of the domain network, and (2) the proper patterns for accumulating the linguistic descriptions of other parts of the domain network so as to describe complex generic relations or to describe individual concepts in terms of their specific relations and their generic description (as with C205). On top of these two levels is grafted a third level of contextually-triggered effects; these effects are carried out by MUMBLE (the component that is maintaining the linguistic context that is the source of the triggers), which acts at the point where combinations are submitted to it as just described.

To best illustrate the contextual effects, we should move to a slightly more complex example, one that is initiated by the speaker's planning process rather by than a default. Suppose that the speaker is talking about the ATN state "S/DCL" and wants to say in effect that it is part of the domain relation "next-state(C205)=S/DCL". The default way to express this relation is as a fact about the jump arc C205; but what we are doing now is to use it as fact about S/DCL, which will require the production of a quite different phrase. The planning process expresses this intention to MUMBLE with the following expression:

(<u>say-about</u> C205 <u>that</u> (next-state C205 S/DCL))

The operator "say-about" is responsible for determining, on the basis of the dictionary's description of the "next-state" relation, what English construction to use in order to express the speaker's intented focus. When the dictionary contains several possible realizing phrases for a relation (for example "next-state(C205) *is the next state after* source-state(C205)" or "next-state(C205) *is the target of* C205"), then "say-about" will have to choose between the realizations on the basis either of some stylistic criteria, for example whether one of the contained relations had been mentioned recently or of some default. Let us suppose for present purposes that the only phrase listed in dictionary for the next-state relation is the one from the first example, i.e.

$[_{\text{clause}}$ C205 $[_{\text{reducable-vp}}$ *goes to* S/DCL]]

Now, "say-about"'s goal is a sentence that has S/DCL as its subject. It can tell from the dictionary's annotation and its English grammar that the phrase as it stands will not permit this since the verb "*go to*" does not passivize; however, the phrase is amenable to a kind of clefting transformation that would yield the text: "*S/DCL is where* C205 *goes to*". "Say-about" arranges for this construction by building the structure below as its representation of its decision, passing it on to MUMBLE for realization. Note that this structure is essentially a linguistic constituent structure of the usual sort, describing the (annotated) surface structure of the intended text to the depth that "say-about" has planned it.

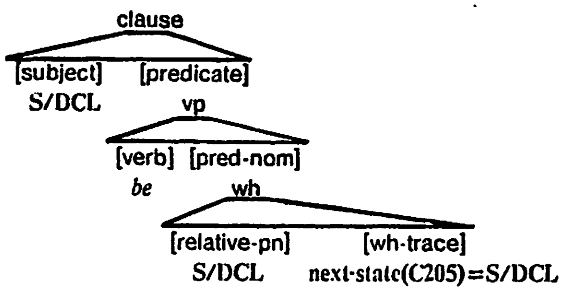

Figure Four: the output of the "say-about" operator

The functional labels marking the constituent positions (i.e. "subject", "verb", etc.) control the options for the realization of the domain-network objects they initially contain. (The objects will be subsequently replaced by the phrases that realize them, processing from left to right.) Thus the first instance of S/DCL, in the subject position, is realized without contextual effects as the name "*S/DCL*"; while the second instance, acting as the relative pronoun for the cleft, is realized as the interrogative pronoun "*where*"; and the final instance, embedded within the "next-state" relation, is supressed entirely even though the rest of the relation is expressed normally. These contextual variations are all entirely transparent to the dictionary mechanisms and demonstrate how we can increase the utility of the phrases by carefully annotating them in the dictionary and using general purpose operations that are triggered by the *descriptions* of the phrases alone, therefore not needing to know anything about their semantic content.

This example was of contextual effects that applied after the domain objects had been embedded in a linguistic structure. Linguistic context can have its effect earlier as well by monitoring the accumulation process and applying its effects at that level. Considering how the phrase for the jump arc C205 would be formed in this same example. Since the planner's original instruction (i.e. "(say-about...)" did not mention C205 specifically, the description of that object will be left to the default process discussed earlier. In the original example, C205 was described in issolation, here it is part of an ongoing discourse context which must be allowed to influence the process.

The default description employed all three of the domain-network relations that C205 is involved in. In this discourse context, however, one of those relations, "next-state(c205)=S/DCL", has already be given in the text; were we to include it in this realization of C205, the result would be garishly redundant and quite unnatural, i.e. "*S/DCL is where the jump arc from S/NP to S/DCL goes to*". To rule out this realization, we can *filter* the original set of three relations, eliminating the redundant relation because we know that it is already mentioned in the text. Doing this entails (1) having some way to recognize when a relation is already given in the text, and (2) a predictable point in the process when the filtering can be done. The second is straight forward: the "describe-as" function is the interface between the planner and the realization components; we simply add a check in that function to scan through the list of relation-entries to be combined and arrange for given relations to be filtered out.

As for the definition of "given", MUMBLE maintains a multi-purpose record of the current discourse context which, like the dictionary, is a meta-level network describing the original speaker's network from yet this other point of view. Meta-links connect relations in the speaker's network with the roles they currently play in the ongoing discourse, as illustrated in figure five. The definition of "given" in terms of properties defined by discourse roles such as these in conjunction with heuristics about how much of the earlier text is likely to still be remembered.

Once able to refer to a rich, linguistically annotated description of the context, the powers of the dictionary can be extended still further to incorporate contextually-triggered transformations to avoid stylistically awkward or ungrammatical linguistic combinations. This part of the dictionary design is still being elaborated, so I will say only what sort of effects are trying to be achieved.

Consider what was done earlier by the "say-about" function: there the planner proposed to say something about one object by saying a relation in which the object was involved, the text choosen for the relation being specially transformed to insure that its thematic subject was the object in question. In these situations, the planner decides to use the relations it does without any particular regard for their potential linguistic structure. This means that there is a certain potential for linguistic disaster. Suppose we wanted to use our earlier trio of relations about C205 as the basis of a question about S/DCL; that is, suppose our planner is a program that is building up an augmented transition net in response to a description fed to it by its human user and that it has reached a point where it knows that there is a sub-network of the ATN that begins with the state S/DCL but it does not yet know how that sub-network is reached. (This would be as if the network of figure one had the "unknown-state" in place of S/NP.) Such a planner would be motivated to ask its user:

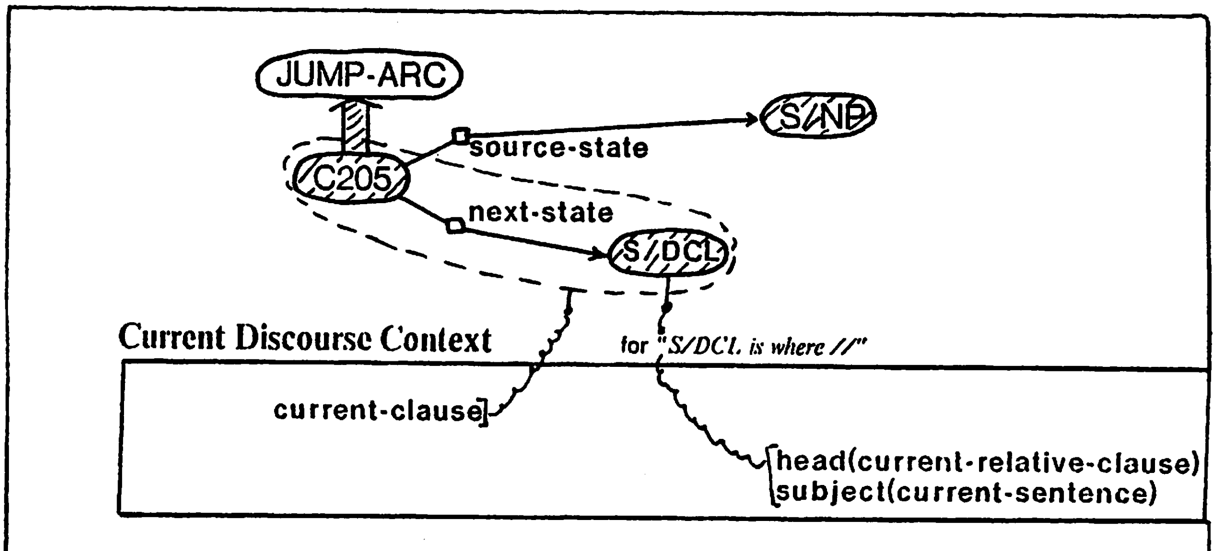

Figure Five: using the discourse-context as a filter (what <state> is-such-that next-state(C205)=<state>)

Realizing this question will mean coming up with a description of C205, that name being one made up by the planner rather than the user. It can of course be described in terms of its properties as already shown; however, if this description were done without appreciating that it occured in the middle of a question, it would be possible to produce the nonsense sentence:

"*where does the jump arc from lead to S/DCL?*"

Here the embedded reference to the "unknown-state" (part of the relation, "source-state(C205)=unknown-state") appeared in the text as a relative clause qualifying the reference to "the jump arc". But, because "unknown-state" was being questioned the English grammar automatically suppressed it. This lead to the nonsense result shown because, as linguists have noted, in English one cannot question a noun phrase out of a relative clause—that would be a violation of an "island constraint"[6].

The problem is, of course, that the critical relation ended up in a relative clause rather than in a different part of the sentence where is suppression would have been normal. It was not inevitable that the nonsense form was chosen; there are equally expressive versions of the same content, e.g. "*where does the jump arc to S/DCL come from?*", the problem is how is a planner who knows nothing about grammatical principles and does not maintain a linguistic description of the current context to know not to choose the nonsense form when confronted with ostensibly synomous alternatives. The answer as I see it is that the selection should not be the planner's problem—that we can leave the job to the linguistic realization component which already maintains the necessary knowledge base. What we do is to make the violation of a grammatical constraint one of the criteria for filtering out realizations when a dictionary entry provides several synonomous choices. In this case, the choice was made by a general transformation already within the realization component and the alternative would be taken from a knowledge of linguistically equivalent ways to ajoin the relations.

A *grammatical dictionary filter* like this one for island-constraints could also be used for the maintaince of discourse focus or for stylistic heuristics such as whether to omit a reducable verb. In general, any decision criteria that is common to all of the dictionary entries should be amenable to being abstracted out into a mechanism such as this at which point it can act transparently to the planner and thereby gain an important modularity of linguistic and conceptual/pragmatic criteria. The potential problems with this technique involve questions of how much information the planner can reasonably be expected to supply the linguistic component. The above filter would be impossible, for example, if the macro-entry where it is applied were not able to notice that the embedded description of C205 could mention the "unknown-state" before it committed itself to the overall structure of the question. The sort of indexing required to do this does not seem unreasonable to me as long as the indexes are passed up with the ground dictionary entries to the macro-entries. Exactly how to do this is one of the pending questions of implementation.

* * *

The dictionaries of other production systems in the literature have typically been either trivial, unconditional object to word mappings [8][7], or else been encoded in unextendable procedures [2]. A notable exception is the decision tree technique of [3] and as refined by researchers at the Yale

Artificial Intelligence Project. The improvements of the present technique over decision trees (which it otherwise resembles) can be found (1) in the sophistication of its representation of the target English phrases, whereby abstract descriptions of the rhetorical and syntactic structure of the phrases may be manipulated by general rules that need not know anything about their pragmatic content; and (2) in its ability to compile decision criteria and candidate phrases dynamically for new objects or relations in terms of the criteria and phrases from their generic descriptions.

The dictionary described in this paper is not critically dependent on the details of the linguistic realization component or planning component it is used in conjunction with. It is designed, however, to make maximum use of whatever constraints may be available from the linguistic context (broadly construed) or from parallel intentional goals. Consequently, components that do not employ MUMBLE's technique of representing the planned and already spoken parts of the utterance explicitly along with its linguistic structure may be unable to use it optimally.

References

[1] Brachman (1979) Research in Natural Language Understanding, Quarterly Technical Progress Report No. 7, Bolt Beranek and Newman Inc.

[2] Davey (1974) Discourse Production Ph.D. Dissertation, Edinburgh University.

[3] Goldman (1974) Computer Generation of Natural Language from a Deep Conceptual Base, memo AIM-247, Stanford Artificial Intelligence Laboratory.

[4] McDonald, D.D. (1980) Language Production as a Process of Decision-making Under Constraints, Ph.D. Dissertation, MIT, to appear as a technical report from the MIT Artificial Intelligence Lab.

[5] ______ (in preparation) "Language Production in A.I. - a review", manuscript being revised for publication.

[6] Ross (1968) Constraints on Variables in Syntax, Ph.D. Dissertation, MIT.

[7] Swartout (1977) A Digitalis Therapy Advisor with Explanations Masters Dissertation, MIT.

[8] Winograd (1973) Understanding Natural Language Academic Press.