Notes toward a Semantic Simulation of a Fragment of Child Language

Andrew Todd and William Todd
University of Oregon and University of Cincinnati

Scenario

A boy of three, out with his mother, sees a strange dog some thirty yards away. He likes dogs and wishes to approach and pet it. He is also afraid that it will bite him, and, to a lesser degree, that it will jump up and lap his face. At this point, his mother says to him, ´That dog is old.´ Since the sentence is a simple one, it can easily be parsed, and there are many parsing programs that will handle it quickly. The problem we wish to address is a semantic one. Once the child has resolved the sentence into its components, how will he interpret them? That is, how will he process them, and what difference will that processing make to his beliefs, intentions, and behavior?

While these questions are extremely difficult, we will suggest some ways in which a computer simulation of this aspect of the boy´s functioning might be approached. We will then engage in some speculations about the reality to be simulated. Before proceeding to the semantics, there are some important phonetic assumptions that must be made. The mother´s utterance will make no difference unless it is uttered within a certain range of tones of voice. Moreover, there may be some tones that would effectively forbid the boy to approach the dog, or which give him permission to do so, regardless of the words that are uttered. In these cases there will be no semantic processing. We hope to interest Ilse Lehiste, who is far more competent in this area than ourselves, in answering questions of this sort.

Let us here assume that the sentence is uttered in such a way that the child listens to it and takes it seriously, but still feels free to decide how to deal with the dog. It must now be recognized that the boy already has a great many beliefs about the world in general, and about dogs in particular. The instant he sees the dog, he will begin to apply as many of these beliefs as possible to the present case. Our simulation will therefore assume an existing database and a method of generating predictions about the dog. The importance of "That dog is old", as received and parsed, is that it will alter these previously existing beliefs in ways to be discussed. If one felt compelled to ask what the sentence means (in a philosophical way), or what its semantic content is, one would be asking for a generalization about the ways in which it affects the existing beliefs of individuals. Such questions are not particularly useful.

A simulation of the child must contain a parser which is capable of isolating the subject, no great problem in the case of such a simple sentence. Once "that dog" is returned from the subject search, the general problem would be to find out what it refers to on this occasion. We here hypothesize that the child´s problem is much simpler than this might seem. He cares only about the question he already has in mind, whether to approach the dog. He is not interested, at such a moment, in storing general information which may, or may not, be useful later on. He thus assumes that "that dog" refers to the object of his current interest, the dog, and will make only a minimal check. In order to do this he must have a database in which "dog" is associated with some of the observable features of a dog. If a certain proportion of these features are not observed, the whole sentence is thrown out as being of no current interest.

The most important of the boy's beliefs about the dog probably do not concern such things as its color and size. They are expectations concerning the behavior of the dog when approached in various ways. One way of putting it is that there is a procedure which the child expects the dog to follow. It would seem that very young children can have rather elaborate expectations about the behavior of persons and animals. Most important from our point of view, these expectations can be modified by verbal input.

There are, at this point, two ways of looking at the situation. One can think of the child as expecting the dog to follow a program with many sub-routines, each of which concerns the behavior of the dog in some hypothetical situation. On the other hand, one can think of each sub-routine merely as representing a dispositional property on the part of the dog. For example, "bad-tempered" means, more or less, that the dog will bite in a certain range of circumstances, growl in others, and so on. In one sense, in makes little difference whether we speak of a dispositional property or a program. On another level, however, it makes a great deal of difference. If we stick to properties, the program that the child follows can simply chain them together, allowing that the links in the chain are only statistical, and much less than foolproof. When verbal input, such as "That dog is old" comes along, it can be allowed to affect the chains, that is, the data.

Alternatively, if we have sub-routines instead of dispositional properties, we are likely to have fewer of them. One sub-routine is altered in certain ways to make it represent a new and different dispositional property. For example, an extremely bad-tempered dog follows the same basic program as a bad-tempered one, except that it takes less provocation to make him growl and bite. It might seem, then, that it is more economical to choose a few sub-routines which, with seemingly minor modifications, will represent a large number of dispositional properties. If, on the other hand, each dispositional property is taken as independent, the master program that the child follows will not "know" about the connections between those properties (and the programs corresponding to them). There is, however, one great difficulty in the program approach. It is extremely difficult to set up a general program to modify sub-programs. It may be virtually impossible to get the degree of generality to handle economically the changes the child would have to bring about in the sub-routines when he gets verbal input, as in our example. It is much easier to effect alterations in chains. It will be more economical, in the long run, to ignore or "lose" a certain body of information (the relative degree of similarity or overlap between dispositional properties), but, at the same time, avoid the pitfalls of writing programs to alter other programs.

Let us take the following set of items as an example of a fragment of our database.

```
[OLD][~YOUNG]
[YOUNG][ACTIVE]
[ACTIVE][MAKE NOISE]
/DOG\[ACTIVE][JUMP UP]
```

/DOG\[ACTIVE][BITE]
/DOG\[BARK][MAKE NOISE]
/DOG\[JUMP UP][LICK]
[FRIENDLY][~BITE]

The input from the mother (root item) will be in subject-predicate form, and the subject, here DOG, may well refer to a particular dog. However, the words appearing in database items refer only to general properties, and the item itself is merely the record of one or more observed co-occurrence of those properties. The order of the words in a database item (but not a root item) will thus make no difference. We also assume that the child makes no distinction between the general and particular uses of DOG. Nothing in the procedures to come will depend on it, and we are suggesting that the most rudimentary and fastest-acting system best fits the needs of the child at a certain stage.

One could certainly hypothesize that there is another (perhaps later) database containing information in subject-predicate form, but we will look first to the minimal model. Even this database does contain a feature which does some of the work of predication. Anything enclosed in /\´s is a non-exchangeable matching word which must appear in the string under consideration if this particular item is to be used. The chaining algorithm uses these items to generate transformations of the original input. It works along the following lines (entries from the database are enclosed in {}´s):

```
[DOG][OLD] (root)
    {[OLD][~YOUNG]}
[DOG]     [~YOUNG]
          {[YOUNG][ACTIVE]}
[DOG]             [~ACTIVE]
                  {/DOG\[ACTIVE][JUMP UP]}
[DOG]                           [~JUMP UP]
                                {/DOG\[JUMP UP][LICK]}
[DOG]                                           [~LICK]
```

We have, in effect, allowed the inference from {[YOUNG][ACTIVE]} to {[~YOUNG][~ACTIVE]}. While this sort of inference can cause problems, we have here in mind a context so limited that allowing it will do more good than harm in terms of efficiency. Since there are many transformations which can be made, we have to specify an objective. Let "%" be defined to be the logical equivalent of "plus or minus". Then define the objective as being of the form [%A][%B]... or [A][%B][%C]... For example, [DOG][JUMP UP] or [DOG][~JUMP UP], the two answers that the child is interested in, are of the form [DOG][%JUMP UP]. We will later suggest an algorithm capable of selecting an appropriate path to the end result.

The child is likely to receive information that conflicts with his previous beliefs. His mother´s input will create the root [DOG][OLD], but he may have {[DOG][YOUNG]} or {[DOG][~OLD]} in his database, thus believing, in effect, that all dogs in his environment are young. He would therefore have to choose between the new information and the old. If we build our model in that way, the child being represented must be either excessively susceptible to suggestion or immune to it altogether. In fact, when the mother says that the dog is old, that should induce a slight increase in the child´s acceptance of the dog. It should not produce a response as if the child had been on intimate terms with the dog since birth. What we want is an increment which, when repeated, produces a belief of increasing strength. The simplest way to achieve this is to give the proposition, not a truth value, but something like a probability, which, being continuous, can have an infinitely fine variation of values. Let us therefore introduce a statistical measure of association, "&", which has a range of -1 to 1 inclusive. The co-efficient, -1, when attached to a word, represents the situation where the property is believed (with practical but not absolute certainty) not to be present, and 1 that where the property is similarly believed to be present. The value 0 implies no belief either way. If we use this "&" in place of the "%" , it will have certain useful properties. Double negations will cancel, and, when we multiply co-efficients, a chain of reasoning built on a series of dubious assumptions will reflect the cumulative uncertainty of the whole. The calculated value of "&" will have a sign which is, in a sense, a result. It will also have a magnitude, which is the reliability of that result. Our new data base look like this:

```
[(1)OLD][(-1)YOUNG]
[(1)YOUNG][(.9)ACTIVE]
[(1)ACTIVE][(.9)MAKE NOISE]
[(1)ACTIVE][(.9)JUMP UP]
[(1)ACTIVE][(.1)BITE]
/(.2)DOG\[(.9)BARK][(.9)MAKE NOISE]
/(.2)DOG\[(.9)JUMP UP][(.8)LICK]
[(1)FRIENDLY][(-.95)BITE]
```

The non-substitutable matching word (in /\'s) now has an associated factor which must be used in computing "&" if the item is used under conditions where the matching word does not appear. e.g using {/(x)A\[(y)B][(z)C]}, [(j)A][(m)D][(n)B] yields [(j)A][(m)D][(n*y*z)C], but [(m)D][(n)B] becomes [(m)D][(n*x*y*z)C].

Note: For purposes of computation we can add to an item any substituteable word with a co-efficient of 0 or any non-substituteable word with a co-efficient of 1.

We now have a derivation like this:

```
[(1)DOG][(.9)OLD]                      (root)
        {[(1)OLD][(-1)YOUNG]}
[(1)DOG]        [(-.9)YOUNG]
                 {[(1)YOUNG][(.9)ACTIVE]}
[(1)DOG]                  [(-.81)ACTIVE]
                            {[(1)ACTIVE][(.9)JUMP UP]}
[(1)DOG]                             [(-.729)JUMP UP]
```

When we use the database, coefficients are always multiplied together, and, within that application, have no separate importance. However, when the mother (or anyone) speaks to the child, the coefficients in the root item have a different significance. In [(1)DOG][(.9)OLD] we assign 1 to DOG since the child assumes its presence and has his attention centered on it. The other coefficient is a measure of confidence the child has in this particular speaker <u>before</u> he consults his own database. The result of the derivation, [(1)DOG][(-.729)JUMP UP], does not, in itself, imply an approach to the dog, but would be one component in a larger model that might represent desires as well as beliefs. Having reconciled them, it would produce output which represents intentional actions. It is worth noting, however, that the model which produces the best output may not be one which preserves the ordinary distinction between desire and belief.

Let us now turn to the database itself and ask how it might be formed. There must, in the beginning, be categories. A child is more likely to recognize and remember a cat than an object which comprises, say, the lower 60% of the cat and three square feet of the surface on which it is standing. Philosophically, there is nothing wrong with the latter sort of object, but it is unsuited for our model because, if it were a category, it would give rise to a less useful database than the sort the child seems to have.

There will be a word associated with each category, and the general principle is that, whenever the child has a sufficiently striking experience, a new item is created. If he notices only an active squirrel, SQUIRREL and ACTIVE will both have positive coefficients. If he notices a young man with a hat, and notices that he has no coat, YOUNG, MAN, and HAT will have positive co effiencts and COAT a negative one. The magnitude of the coefficients will depend on the extent to which the child is "struck" by each feature, or by combinations of them. This allows for the representation of non-rational factors. The child may, for example, be intensely affected by an object or aspect of an object because he fears it, and this may predispose him to expect its re-occurrence. Another possibility is that the child may not be impressed by an experienced combination at a given conscious or unconscious level, yet repetition may still heve its effect. Thus, on the tenth occurrence of the combination, he may "feel" that the two factors which are then co-present will always co-occur. In that case both coefficients will be higher than they would otherwise be. A completely developed model would have to have some mechanism for measuring these factors and deciding what sort of environment and prior condition of the child would give rise to an input which is striking to one degree or another. It may ultimately be found that it is better to simulate a whole environment with a number of persons in it, as opposed to constructing a model for the child alone. For the present, we would envision a series of models representing a single individual, beginning with extremely simplistic ones, but which would gradually grow more sophisticated. The algorithms used to set coefficients would mirror that development.

This process of database development will, in the course of time, produce items which have the same words but different coefficients. In reconciling those differences we must remember that it is not a matter of averaging them. If we have both {[(1) DOG][(.75) OLD]} and {[(1) DOG][(.65) OLD]}, we must remember that the second item provides additional confirmation for the first, and vice versa, so that the reconciled coefficient for OLD ought to be higher than in either previous instance. We will therefore need an algorithm for so handling items in the data base, and for reconciling them with new information, as, for example, that which comes from the mother.

We can think of this process as one of "churning the database", and it is stimulated, not only by new input, but by many other occurrences. Since each new item must be "bounced off" and reconciled with each relevant old item, there is a natural conservatism which favors a considerable body of old information (subject to qualifications to be made later) over new information. Churning is suspended each time there is a need for action, and thus for definite coefficients. When that happens, the relevant database item most easily reached is used, thus importing a random element into the resulting beliefs and actions. As a churning algorithm, we suggest the following:

```
     Let a && b = a+b+c(a,b)
       where c(a,b) = -a*abs(b) if a*b > 0, else c(a,b) = 0

     Then, taking the item from the database to be {/(x)A\[(y)B][(z)C..]},
and linking from B to C,
       [(j)A][(m)D][(n)B][(p)C..] becomes [(j)A][(m)D][(n)B][(r)C..]
           where r = (p && ((x && abs(j))*n*y*z)),
but, if A does not appear in the derived root item (j is 0),
       [(m)D][(n)B][(p)C..] becomes [(m)D][(n)B][(r)C..]
           where now, r = (p && (n*x*y*z))
In either case, the coefficient z in the database is replaced by w : w =
z+e*((y*r/n)-z) where e=abs(p&&(-r)).
```

The fact that some of the algorithms required for these tasks in the model may be complex does not imply a claim that the child does complex calculations in his head. These and other algorithms are arrived at by setting forth plausible cases, plotting them, and then finding a formula that fits the curve. The result might be taken to describe a neural electro-chemical process within the brain. In all models of this sort there are many algorithms used in the computation which can be progressively modified and adjusted to produce results more nearly corresponding to the observed reality being modelled. The battle is largely won if the model is flexible in enough ways so that the results can be skewed in the desired direction by changes of algorithm.

A critical question in this sort of model construction is: How long should items be held in the database? We have argued elsewhere (Todd, Thompson and Todd 1984: Part 6) that human reasoning is more likely to suffer from too much information than from too little. The child needs a system that works fast. It is better to supply the need for action with conclusions, even if a significant percentage are false, than to have action delayed or stultified by too much processing. We also suggested there that certain phenomena of aphasia can be understood best on the assumption of a periodic wipe-out of most of the database while, at the same time, new conclusions are constantly being generated. It is often better to generate a conclusion anew than to store it indefinitely, particularly since the coefficients need periodic revision in any case. This kind of periodic wipe-out will lose connections which would have been "confirmed" if the timing of the wipe-out cycle had been different. But, again, minimal information loss is to be tolerated in the interests of speed and simplicity. At least, that is the hypothesis about the child embodied in our model. We will again leave open the exact procedure for deleting items from the data base on this ground.

In scientific investigation, some concepts, such as that of density, have turned out to be inordinately productive. At the opposite extreme are concepts such as Nelson Goodman's "emeruby", denoting an object that abruptly changes from an emerald to a ruby at time t (Goodman 1965: 102-3). If t is taken as the present, any evidence which confirms the belief that an object is an emerald equally confirms the belief that it is an emeruby (and hence will change color, etc. immediately). An emeruby is, of course, an extreme case.

There are indefinitely many other concepts which are, to one degree or another, unsuited for scientific or everyday reasoning. Goodman has shown that there is no logical or inductively justified way of ruling such properties out of court. We may not like them or use them, but science itself gives us no reason for rejecting them. A consequence of Goodman's point is that the child, "looking over his concepts", has no way of knowing which may be, to some degree, like that of an emeruby. His only real guide will be the input he gets from others. Thus, a tally must be kept of the frequency with which each word denoting a category in his database is spoken to him by others. Thus, in addition to the UP-Dating Algorithm and Churning Algorithm, there will have to be a Lack-of-Frequency Algorithm which systematically lowers the coefficients of such words wherever they appear in the database. If we now, at the periodic wipe-out phase, eliminatate, roughly speaking, all items the products of whose coefficients are distanced from 0 by less than a given threshold, the database will be skewed in favor of the concepts used by the larger society.

We have seen that working with the database changes the database. We must therefore have a means of restoring the database to the state that it would have been in if we had not done the last x transformations. The simplest way to do this is to treat a change or changes as a series of wholesale insertions and deletions of items, the series being stored in a stack which exists for that purpose. These are all reversible so, to go back up the tree structure of possible transformations towards the starting point, we merely take entries from the stack, insert the deletions, and delete the insertions.

Suppose now that we want to use two or more external roots. We will have a separate external root database in which these are put and it will be temporarily appended to the main database. We will then start transforming one item with the use of the others. If all the items in the external root database are used then the derived result can be said to have been derived from them. It is, of course, possible that one of the items in the external root database will be totally unrelated to the subject at hand, and, in that case, it cannot be incorporated in a chain leading to the desired result.

Let us consider each possible state of the database and derived root item as a node in a branching structure with the branches being different possible transformations of the database and derived root item in the state associated with the node from which the branch issues. The branching structure would look rather like this:

```
               {1}
              /   \
             a     b
            /       \
          {2}       {3}
          / \       / \
         a   b     a   b
        /     \   /     \
      {4}     {5} {6}    {7}
```

where the nodes {1}, {2}, {3}, {4}, {5}, {6}, and {7} are possible states of the database and derived root item, and the branches <u>a</u> and <u>b</u> are possible transformations of the same.

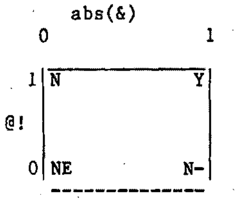
Let us next consider searching all possible combinations of items or, rather, some reasonable subset of them. This is where "&" comes into its own. Consider a quantity called "&*" and let "&*" be the product of all the "&"s of the derived root item. At this point, &* obviously pertains to the whole derived root item, rather than to a part of it. If "&*" falls below a certain threshold, then we branch back and try a different branch from the previous node. If all the branches from that node are untenable, then we branch back still farther, and so on. To ensure that the first items, comprising the external root database, are used, we have the rule that possible branches are considered in the order that they appear in the database.

We now have a scheme which searches for what, speaking somewhat loosely, amounts to the set of statistically significant implications of the original state of the database, together with the external items, with special stress being placed on the implications of the latter. But this is not yet what we want. We want to know, not only whether the derivation is reasonable, but whether it is relevent. As stated before, we have a target item of a form similar to the items in the database except that it does not have "&" coefficiants. It may however have weights, which we shall call "@!", taking 0 to mean that the word does not appear in the target and 1 to mean that it is fully present. The object is to determine which of its words should get preference in being matched with words in the derived root item. Further, we have some statistic which we shall call "&#" for measuring closeness of fit between the target and the derived root item. One possible formula would be the following:

&# = sum of &#(j) for all possible words (where &#(j) is a measure of fit between the occurrence of a word in the derived root item and the occurance of that same word in the target.)

```
 &#(j) is computed as follows:
   if @! >= abs(&) then
    &#(j) = 2 * abs(&) - @!
   otherwise
    &#(j) = 1.5 * @! - .5 * abs(&)
```

This formula was obtained by taking four cases of abs(&) and @!, intuitively selecting appropriate values of &#(j) for them and then contriving a function to fit them. Here are the four cases plotted on a graph. It should be noted that the linking together of word computations is effected by addition. Therefore the identity element is 0. Wth symbolic values, the graph is:



where N is no, Y is yes, NE is no effect, and N- is no, only less emphatic than N. With numbers, bearing in mind that NE must be 0,

		abs(&)	
		0	1
@!	1	-1	1
	0	0	-.5

The result is a system that finds what are, in effect, statistical inferences about the relevance of the root item, based on new information. It should be noted that these are not definitive, as the number of items which can be derived is not finite, and therefore we search only a small subset of the possible range of combinations.

We would like to treat briefly the means whereby the algorithm described above would be implemented in hardware in what might be called a realistic case, by which we mean a case involving much larger amounts of data. This may serve to give some insight into the sort of processes going on inside a child´s brain. Let us consider that the child is at a node called {A} in the algorithm above and let us consider that {A} has daughter nodes {B}, reached by branch b, {C}, reached by branch c, {D}, reached by branch d, and so on. It should be understood that the limits of speed in going through the algorithm are not posed by the total amount of computation to be done, but by the number of things which must be done in sequence. If many different parts of the job do not depend on each other for inputs, they may be done at the same time by different equipment. That said, let us assume that there are processors available for each of the branches b, c, and d. First, each of them must receive a copy of the information making up node {A}, that is a complete copy of the database, a complete copy of the change stack, and the derived root item. While this may seem an impossible amount of material to transfer, it can all be sent at the same time if the data path is broad enough. There is no reason why this should not be the case, as it only means that the data path or what would be called the bus in a computer must be on about the same scale, the same order of complexity, as the storage medium.

Let the processors execute the branches on their copies of the node {A} and generate "&#" for the daughter nodes. The results determine whether the search will continue through that node or not. If that node is not a good candidate for continued development, its processor will then be released to a common pool of unemployed processors. If, on the other hand it is worthy of development, the paths leading to its own daughter nodes will be allocated processors from the pool, if they are available. If not enough are available, the available processor or processors will work through the paths in sequence as required. This approach is standard practice and is different only in scale and not in kind from the facilities available on most large mainframe and supermini computers. It will be noted that we use an underlying mechanism which is very simple of itself, in that there is no attempt to predict which branches are worthy of development.

In conclusion, it should be remarked that the suggested model would occupy a position in two different series of models. While it intentionally ignores many distinctions to be found in natural language, the result is a high degree of simplicity and speed of operation. There are, of course, many kinds of simplicity, some of which conflict with others, but we have chosen the kinds we think appropriate at this stage of language acquisition. One series of models then represents different stages of acquisition, terminating with full adult competence. Our larger speculation is that, starting with a model such as that outlined here, subsequent ones can be fitted with additional features without there ever being a need for a radical re-design.

The other series of models, starting from our outline, represents improved attempts to simulate a given level of linguistic competence. We have suggested that a great deal can be done by improving the algorithms. However, the important thing is to work toward an actual computer model whose input and output can be compared with that of the child. Then, even if the results do not tally, we would be in a position to work toward a radically improved model.

References

Goodman, Nelson. (1965). Fact, Fiction, and Forecast, 2nd Ed. New York: Bobbs Merrill.

William Todd, George Thompson, and Andrew Todd. (1984) A Logic for Computers. Cincinnati: Ehling Clifton Books.