Babble: Simple Conversations With a Computer

Karen Blaedow
Custom Technology Ltd.
Trace Center, University of Wisconsin

Abstract

This paper introduces a novel approach to natural language understanding called tridbit™ technology. This approach is being explored as the basis for conversational user interfaces on devices, such as hand held computers, with limited processing power. Tridbit technology processes natural language directly into tridbits, which are the underlying knowledge structure for retrieving, storing and reasoning with information. The efficiency of the system derives from exploiting the inherent constraints of the tridbit structure and harmonizing various aspects of natural language processing with it. The system requires very little pre-defined knowledge, instead building grammatical and world knowledge from natural language interaction. This paper provides an introduction to these structures and processes.

1   Introduction

How it is that humans are able to use natural language to convey knowledge is a topic addressed by philosophers, psychologists and most recently computer scientists. This paper discusses a computer simulation of human language behavior. The computer simulation, which is called Babble, is able to converse with a user in English, extract meaning from what it is told, respond to questions and commands, store the knowledge it acquires, reason and initiate behavior based on its knowledge, and learn words and syntax on the fly.

Babble uses a system of semantic representation called tridbit technology. While the approach is quite novel, it has produced promising results. An example dialog showing Babble's basic abilities is shown below. Input to Babble in shown in blue and Babble's response in red

> June likes computers.
> They are smarter than typewriters.
> She drove the sporty gold car to the store that sells computers.
> Who likes computers?
> > June
>
> What color was the car?
> > gold
>
> Robyn is the mother of Tyler.
> Who is the child of Robyn?
> > Tyler
>
> Spiders have eight legs.
> Spiders have how many legs?
> > 8
>
> The television under the table is broken.
> The television is under what?
> > table
>
> What is broken?
> > television
>
> The car that Jeff drove to the store that sold electronics crashed.
> What crashed?
> > car

What sold electronics?
store

What is smarter than typewriters?
computer

This technology is currently being applied in providing a natural language interface for household device control. Recently Babble has been programmed to read TV program listings to handle requests concerning TV programs a user might want to watch, record, or just get information about. An example of such a dialog is show below:

When is NYPD Blue?
Tuesday, February 22 2005 at 9:00 pm

When is American Chopper on?
There are 2:
Thursday, February 24 2005 at 1:00 am and
Wednesday, February 23 2005 at 6:00 pm

Record it Thursday.
Recording American Chopper Thursday, February 24 2005 at 1:00 am on The Discovery Channel

Record As the World Turns tomorrow.
Recording As the World Turns Thursday, February 24 2005 at 1:00 pm on WISC

Who broadcasts Star Trek?
Sci-Fi Channel

What time is Crossfire?
3:30 pm

Record it.
Recording Crossfire Thursday, February 24 2005 at 3:30 pm on Cable News Network

This paper will highlight the knowledge representation and methodologies on which the tridbit model is based. The first section describes *tridbits*, the structure for representing meaning. The claim here is this extraordinarily simple monadic unit of information places additional structure and constraints on information that facilitates disambiguation of natural language. Comparisons will be made to other representational schemes that employ triples and predicate logic, especially Boris Katz's ternary expressions, and John Sowa's conceptual graphs.

Next, the method for going from the surface structure of a natural language utterance to its representation in tridbits is outlined. This process uses pattern detection, contextual learning and a functional method for classifying word use instead of traditional transformational grammar and ontologies. This process is further helped by the more natural correspondence between the surface structure and the knowledge representation.

The last section briefly describes Babble's reasoning, observation and management of its dynamically grown knowledgebase.

On occasion metaphysics is interjected into the discussion, as the success of simulating speech behavior is an achievement both of computer algorithms and structures and of a workable metaphysical model.

2   What is a Tridbit?

Babble represents knowledge as tridbits. Like many other knowledge representations, such as Boris Katz's ternary expressions or John Sowa's conceptual graphs, tridbits are triples. However, tridbits have an internal structure that prescribes how each of its elements must relate to each other, thereby providing a method to disambiguate language. The ternary expression shown below, which Katz uses as an example [Katz 1997] could not be translated to a valid tridbit.

<Bill surprise Hillary>

The expression must be split in two as shown below before it can be translated to tridbits.

<Bill subject surprise>
<Hillary object surprise>

Most systems that use predicate logic as their underlying knowledge representation allow verbs to be mapped to predicates. Tridbits more closely resemble conceptual graphs in that a verb such as surprise is not mapped to a predicate, but to the concept of the verb. This concept is then related to its subject and object. In conceptual graph terminology surprise is mapped to a conceptual node. The connections to Bill and Hillary would be represented as two separate conceptual relations agentive and theme as follows:

[Bill]←(Agnt)←[Surprise]→(Thme)→[Hillary]

Proponents of conceptual graphs [Fargues et al. 1986] cite several very good reasons for not simply mapping verbs to predicates. These include not having to fix whether the predicate for the verb is unary, binary, ternary, etc. In the above example the predicate for surprise takes two arguments. But other uses of surprise, such as "Bill surprised Hillary with his answer", take additional arguments despite representing the same event. Another cited advantage is the ability to associate semantic constraints using canonical graphs.

Tridbits impose constraints on its triples that preclude a verb from being used as an attribute. The constraints require certain relationships between the elements of assert tridbits, which will be described shortly. The structure of an assert tridbit, which is similar to a binary predicate, is illustrated below:
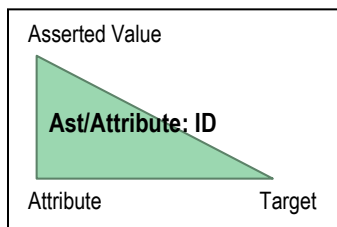

Figure 1: Structure of an assert attribute tridbit

In the center of the tridbit is its identification, which gives its type and a unique ID. Tridbits come in several types, the main two being assert tridbits discussed here and referent tridbits discussed next. An assert tridbit asserts a relationship between its three elements. There are two types of assert tridbits: assert attribute tridbits, diagrammed at left and assert compare tridbits, which will only be mentioned briefly.

In the tridbit structure diagram, the three elements of the tridbit are located at each point of the triangle. The target and value elements of the tridbit can refer to anything Babble is capable of understanding, usually other tridbits. However these elements can also refer to names, which are stored in the dictionary or rules stored in a syntax rule table. Attributes are constrained by their relationship with the value element. The attribute must be a category of which the value element is a member. Aside from this constraint, an attribute is no different than any other general referent defined by category.

In addition to the three interrelated elements, attribute, target and value, a small amount of additional information is stored. Of significance is a pair of values whose interpretation is different for assert vs. referent tridbits. In assert tridbits it represents truth and certainty. In referent tridbits it may be used to represent scope and count. More will be said about this later.

A tridbit is more than a transformation from natural language to a logical representation or a node within a semantic net, though its ability to function in those ways is important. A tridbit is a monadic unit of information that can be used to build a consistent metaphysical understanding of the world.

Asserted Value

Bill
Hillary
Austin
blue

Category
The asserted value is a member of the attribute category.

Property
The target has the asserted value.

subject
object
capital
color

Attribute

Attribute
The target has the specified attribute.
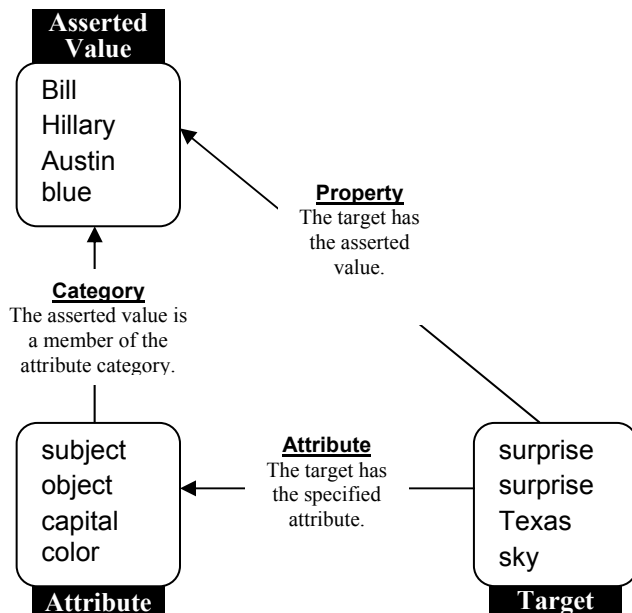
surprise
surprise
Texas
sky

Target

Figure 2: Constraints within an assert attribute tridbit

The diagram at left illustrates the constraints within an assert attribute tridbit. In addition to "Bill surprised Hillary," constraints for the following two sentences are also depicted:

The capital of Texas is Austin.
The color of the sky is blue.

As the tridbits are generated by the grammar, they are first tested to see if they make "structural sense." If not the rule is rejected. By understanding the constraints of an assert attribute tridbit, it is possible to disambiguate attribute information, even though the grammar may not prescribe a fixed position for each element, or leave one out.

Consider the following sentences:

1) The color of the sky is blue.
2) The sky is blue. (or Blue is the sky.)
3) The color is blue.
4) The color of the sky (fragment)

Sentence 1 will be represented by a complete tridbit. Sentences 2, 3 and 4 have one of the elements missing. The behavior of the expression depends on what element is missing. Omitting the attribute, as in sentence 2, is common when the value (blue) belongs to only one category (color) that could be an attribute of the target (sky). It results in an ambiguity if there are multiple categories, such as:

John is blue. (color or mood?)
Texas is big. (size or population?)

If the target is omitted, as in sentence 3, the result provides very little information except that blue is a color. Given the sentence "The color is blue", Babble will reply "The color of what is blue?" Occasionally the context of the sentence might allow the target to be filled in. For example, the target becomes clear if sentence 2 were proceeded by "Guess what color Jenny's hair is."

Finally if the value is omitted the tridbit becomes a fragment that can be used to reference the value, as in "His eyes were the color of the sky." A referent tridbit defined in this way is called an inferval.

Thus the structure of an assert tridbit sheds light on the meaning of these constructions and how the grammar can disambiguate them. By recognizing the constraints of the tridbit, each element looks like a puzzle piece that can only go in one place.

While assert attribute tridbits are the most important type of assert tridbits, a second type of assert tridbit exists for comparative information. An assert compare tridbit does not share the constraints of an assert attribute tridbit, but instead has its own unique constraints, while retaining essentially the same structure.

The referents of a sentence are also represented by tridbits. A tridbit used to represent a referent is called a referent tridbit and has the structure shown below:

Figure 3: Structure of a referent tridbit
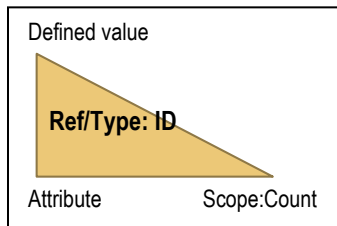
The elements of a referent tridbit shown at left are:
Type: <thing, event or property>
ID: unique value for identifying the referent
Attribute: How the referent is defined, typically by name or category
Value: Name, category or other attribute value assigned to the referent.
Scope: <general, specific or qualified>
Count: numeric value representing how many

A referent tridbit might be considered a degenerate case of an assert tridbit where the target is always the referent itself and thus need not be specified. In the current implementation of Babble, this field is used to provide the referents scope and count information. The attribute in a referent tridbit performs essentially the same role as in an assert attribute tridbit. The assertion states the specified attribute of me is the specified value. Because it is a referent tridbit, this assertion defines the referent. There are two attributes that are typically used to define a referent, category and name. In the sentence:

The dog barked.

The referent tridbit for the dog is defined by category. In the sentence:

Fido barked.

The referent tridbit for Fido is defined by name. In the case of a pronoun or inferval there is not a defining attribute per se and the referent is defined by inference. However the referent is defined, the goal is to have only one referent tridbit for each thing, event or property being referred to. Determining when two references are referring to the same thing turns out to be one of the harder discriminations Babble must do. It gets at the metaphysical question of what makes two things identical.

Moreover, the desire not to have multiple tridbits represent the same thing in the world implies a correspondence between referents and things that exist in the world. On the other hand, this one to one correspondence is only desired for referents that are specific in scope, or single terms in metaphysical terminology. Referents of general scope, i.e. universals, share a single referent tridbit. Thus there is only one referent tridbit that represents the category of dogs, one for cats, one for animals, etc. In this case it would be difficult to say what that referent corresponds to in the world. Since there is no answer that improves the understanding or implementation of general referents, it doesn't seem useful to claim there is any correspondence to something in the world, but rather an internal organization that facilitates language and thought.

The next section describes how scope and related issues in tridbits are implemented. It is useful at this point to introduce a notation for defining a tridbit using text instead of a tridbit diagram. The general form of the notation for any tridbit is given below.

ID < Tridbit type/Subtype, Attribute, Target, Value >

where:

ID is a number uniquely identifying the tridbit
Tridbit type is "ref" for referent or "ast" for assert
Subtype for referent tridbits is "thing", "event" or "property"
Subtype for assert tridbits is "attribute" or "compare"

The format to specify the Attribute, Target or Value elements is given below.

Element type . ID . scope:count . defined category or "name"

While up to 4 fields separated by periods can be specified, in most cases only one or two will be chosen to best indicate to what the element refers. Unspecified fields will have default values as discussed below.

Element type is a letter indicating if the element points to a tridbit (t), word (w) or rule (r). Tridbit is the default so "t" can be omitted.

ID specifies the unique ID of the element. There is no default ID per se, but can be omitted in cases where an ambiguity does not result. General concepts, i.e. referent tridbits with scope G:-100, typically omit their ID. Only one such tridbit is allowed to exist for each concept so it is easier and more readable to specify the concept. The ID, if specified, it is sufficient to uniquely identify the element so any further fields are provided solely for the sake of readability.

scope:count can be specified if the element is a referent tridbit. It defaults to G:-100 if no ID is specified.

defined category or "name" can be specified if the element is a referent tridbit. If the value is quoted is indicates the referent is defined by name otherwise by category. There is no default.

When listing tridbits, a two column listing is preferred, with referent tridbits in the first column and assert tridbits in the second column. The example below shows a tridbit representation for the sentence:

The color of the dress is blue.

1<ref/thing, category, G:-100, color> 2<ref/ thing, category, S:1, dress> 3<ref/property, category, G:-100, blue>	4<ast/attribute, 1.color, 2.dress, 3.blue>

It is acceptable to omit the tridbits representing general concepts, which in the above example would leave the two tridbits listed below.

1<ref/ thing, category, S:1, dress>	2<ast/attribute, color, 1, blue>

2.1 Scope – single terms vs. universals

A referent of specific scope corresponds roughly to a single term in metaphysics. It indicates that the referent is not arbitrary, but refers to a specific individual or group. Any assertion in which a referent of specific scope is involved, applies only to that specific referent. The existential quantifier in first order predicate logic could be used to indicate specific scope, but it is not an exact fit.

A referent of general scope corresponds roughly to universals in metaphysics. It indicates that the identity of the referent is arbitrary, it can be any or all referents (depending on the count) that satisfy the criteria by which the referent is defined. General referents are typically defined by category, although they can also be infervals where the criteria is given by an assertion with a missing value element such as "a student of the University". If the scope of a referent tridbit is general, the attribute that defines it can not be name.

Scope only addresses whether the referent is picking out a specific individual or group or whether the referent is arbitrary, subject to certain qualifications. Count indicates how many of the defined type are included in the referent. Count can be absolute, which is indicated by a positive number or proportional indicated by a negative number. Thus the universal quantifier in first order predicate logic would correspond to a scope:count of G:-100.

In order to understand natural language, the listener must be able to identify the referents within the utterance. Scope and count capture the information conveyed by natural language that facilitates this process. Suppose the listener is told that there is an apple on the desk, followed by one of three sentences which are shown below in English, first order predicate logic and tridbit notation.

John ate the apple. (∃x:Apple)(∃y:Ate) (subject(y,John) ∧ object(y,x))

1<ref/thing, category, S:1, apple> (on the desk) 2<ref/thing, name, S:1, John > 3<ref/event, category, S:1, eat > 4<ref/thing, category, S:1, apple> (reduced to 1)	 5<ast/attribute, subject, 3.eat, 2."John"> 6<ast/attribute, object, 3.eat, 1.apple>

John ate an apple. ((∃x:Apple)(∃y:Ate) (subject(y,John) ∧ object(y,x)))

1<ref/thing, category, S:1, apple> (on the desk) 2<ref/thing, name, S:1, John > 3<ref/event, category, S:1, eat > 4<ref/thing, category, S:1, apple>	 5<ast/attribute, subject, 3.eat, 2."John"> 6<ast/attribute, object, 3.eat, 4.apple>

John might eat an apple. ◊((∃x:Apple)(∃y:Ate) (subject(y,John) ∧ object(y,x)))

1<ref/thing, category, S:1, apple> (on the desk) 2<ref/thing, name, S:1, John > 3<ref/event, category, S:1, eat > 4<ref/thing, category, G:1, apple> 7<ref/event, category, S:1, occur >	 5<ast/attribute, subject, 3.eat, 2."John"> 6<ast/attribute, object, 3.eat, 4.apple> 8<ast/attribute, subject, 7.occur, 3.eat> (certainty = might)

A human listener is likely to identify the apple differently in each of these sentences. Thus the representation for the meaning of these sentences should capture that difference. Tridbits does this through its encoding of scope and count, as well as its algorithm for referent reduction.

In the first sentence "John ate the apple," the apple refers to a specific individual apple, because the eat event has taken place. The statement expresses an event involving only one specific apple, which therefore is given specific scope and a count of 1. The use of the definite article indicates to the listener that its is important to resolve the identity of that specific apple. Thus referent reduction will take place causing the apple John ate to be reduced to the apple on the desk.

In the second sentence "John ate an apple," the apple also refers to a specific individual apple, again because the eat event has taken place. As in the previous example, the apple is given specific scope and a count of 1. The use of the indefinite article in this case indicates to the listener that its is not important to resolve the identity of that specific apple. Therefore the apple referent will not be reduced, so it will not refer to the same apple as the apple on the desk.

In the third sentence "John might eat an apple," the apple does not refer to any specific apple. This apple is an arbitrary member of the set of apples and therefore is given general scope with a count of 1. The statement expresses an eat event in which any apple could participate and so it is stored as information about apples in general. The natural language clues that indicate this are the use of the indefinite article, together with a tense that indicates the event has not actually occurred. Hypothetical tenses have not been implemented at this point, so tridbits 7 and 8 are speculation how it might be encoded.

The interaction between the verb used, its tense and referent identification is quite complex, with exceptions to many of the rules of thumb Babble currently employs. In the future, methods to allow Babble to learn the nuances of this interaction will be explored. For now, the pragmatic approach used in Babble produces reasonable results in most common situations.

2.2 Referent types

A referent is either a thing, event or property as indicated by its referent type. Descriptions of each of these basic metaphysical categories is given below:

A **thing** exists, such as a person, an idea, a chair or chairs in general
An **event** occurs, such as specific episodes of going, seeing, breaking, meeting or meetings in general
A **property** describes another tridbit, such as loud, red or big

The most controversial of these categories is event [Casati et al 2002]. Interestingly events do not have a long history of metaphysical acceptance. It is only in the twentieth century that Whitehead, Davidson and others have argued for their existence and even then as a somewhat questionable category. A successful computer simulation of natural language behavior does seem to require some notion of events as separate types from things or properties, although this distinction seems less fundamental than that of properties vs. things and events.

Babble's most basic understanding of a word is to determine whether or not it represents a referent and then what type of referent it represents. A traditional English grammar assigns words into one of eight parts of speech. There is a general correspondence between this traditional classification system and referent types in that nouns indicate things, verbs indicate events, adjectives and adverbs indicate properties. However, this correspondence is not quite sufficient for Babble to map a word to its role in the meaning of the sentence.

For this purpose, tridbits uses its own system of word use with three main categories that correspond roughly with the traditional noun, verb and a combined category of adjective/adverb. Words that do not represent referents, such as articles, conjunctions, some propositions, etc., are categorized as tag words in the tridbit system. Tag words provide information about how the referents relate to each other.

There is occasional disagreement between traditional and tridbit classification, for example a tridbit verb is never considered an adjective, even when used descriptively as in "dancing bear" or "proven theory". More important is the subcategorization of tridbit nouns, verbs and adjectives. These subcategories, along with their designations, are described below:

Noun subcategories include:

Thing category N1	A *thing category* is the most common type of noun, representing a category of things. **Examples are rocks, cities, water, people and ideas.**
Event category N5	An *event category* is a noun that represents a category of events. **Examples are dances, conversations, concerts, repairs, singing and thinking**.
Property category N4	A *property category* is a noun that represents a category of properties. **Examples are color, size, temperature and weight.**
Assertion category N7	An *assertion category* is a noun that represents a category of assertions. **Examples are causes, reasons, results and beliefs.**
Proper noun N2	*Proper nouns* are names used to refer to specific things or events. **Examples are the Mississippi River, Patrick Henry, Woodstock and the Rose Bowl.**
Pronoun (various types) N3, N6	*Pronouns* are used to refer to something, without providing a name or category of the referent. **Examples are you, me, they it, who and what.**

Verb subcategories include:

Event V1	Most verbs are used as *events*, which refer to a specific event. **Examples are give, laugh, drive, think, like** and all forms of person and tense, such as give, gives, gave, given and giving.
Other function V2	The verb type *other function*, is given to those words that act like verbs, but don't refer to a specific event, such as **is, do and has**. In the future these verbs may be reclassified as tag words rather than a special type of verb.
Subject property V3	A *subject property* is the form of a verb that is used as an adjective to indicate the referent being described is the subject of the event. **Examples are dancing, throwing, chewing, painting and knowing**.
Object property	An *object property* is the form of a verb that is used as an adjective to

V4	indicate the referent being described is the object of the event. **Examples are thrown, chewed, painted and known**.

Adjective/Adverb subcategories include:

Relative property P1	*Relative properties* name a relative position within a property category such as brightness where the category is scalar in nature. **Examples are bright, red, big, silly and heavy, brightest, reddest, biggest, etc.**
Compare operator P2	*Compare operators* indicate a ranking within a property category. **Examples are more, better, brighter, redder, bigger, hotter and heavier.**
Quantity P3	*Quantities* names an amount, independent of any units. **Examples include five, a third, ninety-nine and 5,467,892.45.**
Date/Time P5	*Date/Time* names a point in time. **Examples include July 4, 1776, 6:00, tomorrow, noon or 11/2/2004 23:59.**
Names/Enumerated	This category is still in development, but one or more additional categories may be required for properties that do not lend themselves to comparisons and other manipulations of scalar property categories.

Babble's dictionary has an entry for each use of a word, which can actually be a multiple word chunk, but for this discussion we'll just refer to words. Thus the dictionary would have three entries for a word like "bat"; an N1 representing a flying mammal, another N1 representing a baseball bat and a V1 representing a bat event. If the entry is not a tag word, i.e. it represents a referent, there will be a link to the tridbit that represents the general category of referents represented by that word use. By categorizing the words in Babble's dictionary using tridbit word use, the mapping from word to referent tridbit is made easier. This subject is discussed in the next section.

Note the first four noun subcategories listed above represent categories of referents or assertions, not the referents or assertions themselves. For example, the word "reading" could be used in two ways as shown below:

Word use	Example sentences	Scope/count of the read event
N5 - category of events	Reading is fun. The reading of the poem was inspirational. The reading is over. Dennis will have a reading.	G:-100 S:1 S:1 G:1
V1 - event	John is reading a book. I will be reading tomorrow.	S:1 S:1

Both N5s and V1s have syntax rules that generate event referents. The event referent generated from a V1 is always specific in scope. The event referent generated from an N5 is general in scope, but may be modified by the use of "the" and "a" as illustrated in the example. Thus, it is necessary to use a tridbit N5 noun to refer to a category of events, such as reading in the first example sentence. This is similar for properties, where words that refer to a category of properties, such as color, size, quantity, etc. are categorized as a noun (N4), but the properties themselves are adjective/adverbs.

The relevant point is that a referent has two important dimensions, referent type and scope, that are made readily accessible through word use and syntax. By simply knowing a referent's type and scope, and utilizing the constraints imposed by tridbits, a significant amount of disambiguation can be done prior to a full blown semantic analysis. Specifically, structural nonsense such as "The color of the dress is Texas" can be easily eliminated.

In addition to its role in grammar, referent type is the basis of other functional differences in how referents behave. Each referent type has its own set of attributes. For example, only events can have participants and time attributes. In addition, only things and events can have specific scope, properties are always general in scope. This latter claim is equivalent to saying that properties are always universals, a claim sometimes disputed in metaphysics. From the perspective of implementing a tridbit knowledge representation, allowing properties to be instantiated adds significant complexity to representing the situation of a target having an attribute that belongs to a category of properties, with no seeming benefit. Thus the simpler model of all properties being universals was adapted. This is

also appealing on the basis of properties such as red, big or loud not, on their own, having referents in the world as the instantiations of things or events would.

Categorizing referents into things, events and properties is the only ontological information built into the tridbit model. Instead the model limits built in primitive knowledge to just that required for language. It would be hard to imagine how someone unable to discriminate things, events and properties could participate meaningfully in language. On the other hand, not knowing that a dog is an animal does not preclude one from having language skills.

In addition to the primitive referent types, other primitive concepts are defined in tridbits. Among these are category, attribute and property which are the constraining relationships in an assert tridbit. Also primitive are name, equivalence, order, if, result, group and member. The above mentioned concepts are primitive in that the simulation is built to recognize and treat them in specific ways. This is not intended to be an exhaustive list of primitives as Babble is a work in progress.

Certain aspects of properties are also primitive. Some types of properties are preconfigured such as quantity, time, color, size, etc. Not all properties are built in to the degree that the previously mentioned primitives are, nor does language ability require all of them, or even a majority. What is does seem to require is that the listener knows how to apply certain categories of properties appropriately. For example, temperature, loudness, brightness, redness, shyness and air pressure all have values that can be ordered and compared in a linear fashion. What is built in is a property type that behaves in this way. For some properties, such as brightness, the property type may be preconfigured and for others such as air pressure, it may be learned. Beyond these primitives all other knowledge is acquired though natural language interaction and represented as tridbits.

Thus knowledge is constructed tridbit by tridbit from natural language input. From the sentence "Karen drinks tea" Babble learns that Karen can be the subject of a drink event and tea can be the object and that a general subset of drink events exist where Karen is the subject and tea is the object. Telling it "Tea is a beverage" would provide what is typically considered a piece of ontological information. But telling it "Beverages are drunk" provides better information about beverages and allows it to conclude that tea is a beverage. All these statements may be part of Babble's knowledge of beverages, drinking, tea and Karen. More is said about managing this messy accumulation of knowledge later.

Moreover, the effectiveness of massive, predefined ontologies has not been proven. The Cyc project [Lenat, D.B. 1995] was one of three competitors in the first phase of the Halo project [Friedland et al. 2004]. Cyc has been working toward implementing the common sense knowledge of an average adult since 1984, with over a person-century of effort going into the task. Cyc includes an ontology of over 200,000 general terms and a knowledge base of over two million facts and rules about those terms expressed in predicate calculus. Despite having such a well-developed ontology to apply to the Halo project, it faired no better than its competitor who developed their knowledgebase from scratch.

The goal of phase 1 of the Halo project was to determine how well the three systems could master the knowledge presented in seventy pages of a chemistry textbook. Mastery was assessed by the system's ability to answer novel, previously unseen questions about the material. The knowledge in the chemistry textbook was hand coded into each system's knowledge representation by knowledge engineers over a four-month period at a cost of about $10,000 per page. The test questions were also translated from natural language to each system's respective knowledge representation language. The average test score was an impressive 3 on a scale of 1 to 5, compared to an average student score of 2.8. However, the cost and effort to achieve this result in such a narrow area of knowledge leaves unanswered the practicality of the approach.

3   Going from words to tridbits

Babble uses a context aware pattern-matching algorithm to generate tridbits directly from the surface structure of a sentence. The generated tridbits are pushed onto a processing stack, which also serves as a stream of consciousness while Babble converses with the user. Information that is to be remembered beyond the current conversation must be transferred from this processing stack to Babble's knowledgebase.

Syntax rules, such as the ones below, define the triggering pattern and define how to map the elements of the pattern into elements of the tridbits. The rules are fundamentally an association between meaning structures (tridbits) and linguistic patterns. Babble is able to learn these associations during conversation.

RuleID	Pattern	AssociateList	Consume	Clausal Type	Clausal Pos	Description
22	>N1	<782>	Y	-	0	Produces: <Ref/Thing Category G-100 P1> Example: dog
28	>N2	<886>	Y	-	0	Produces: <Ref/Thing Name S1 P1> Example: Fred
70	>N2	<3065>	Y	-	0	Produces: <Ref/Property Category G-100 P1> Example: Fred
90	>N5	<5232>	Y	-	0	Produces: < Ref/Event Category G-100 P1> Example: running
6	>V	<611>	Y	-	0	Produces: <Ref/Event Category S1 P1> Example: drove
47	>T>E1>T	<5060><5061>	Y	-	0	Produces: <Ast/Attribute Subject P2 P1> <Ast/Attribute Object P2 P3> Example: I drink tea.
51	=to>T	<5217>	YY	E	65552	Produces: <Ast/Attribute Destination P0 P2> Example: to the store

A syntax rule's pattern is simply a string of tokens that must be matched either literally (proceeded by =) or by category (proceeded by >). Categories for matching include tridbit word use categories (N, V and A) and referent tridbit types (T, E and P).

When a sentence is read, each word is looked up in the dictionary. A word use tree structure is created that facilitates choosing a word use when multiple possibilities are listed in the dictionary. This choice is initially made based on past associations between the word in question and the words occurring before and after this word. Theoretically, if the chosen word uses do not produce an interpretation of the sentence that makes sense, other choices can be tried until a sensible interpretation is found.

Thus a sentence's initial pattern consists of just word use categories (N, V and A), plus literals for any tag words. As the pattern is processed, the word use tokens are consumed and replaced by referent tridbit tokens, which are then also processed and replaced by assert tridbits. These are the basic steps in the main processing of a sentence. In the flowchart in figure 5, this processing is represented by boxes 1 – 4. Some of the more interesting aspects of the main processing are described next.

The generation of tridbits is directed by template tridbits listed in the associate list. Each template tridbit in the list generates one tridbit by specifying how elements within the pattern are mapped to elements of the generated tridbit. In addition to using elements from the pattern, a clause references a tridbit outside of the pattern, whose type and relative position are given by ClauseType and ClausePos.

Before a rule is accepted, the tridbits it generates must all make sense. The current test for making sense is simply the structural tests that result from the constraints imposed by the tridbit structure. These tests alone provide a significant amount of discrimination between applicable and non-applicable rules. In addition, Babble learns from experience the contexts in which each rule has been successfully applied.

The application of rules to specific contexts has proven to be a useful practice in resolving certain types of elipsis. For example in the question "When is NYPD Blue?" in the TV dialog, the thing referent for NYPD Blue triggers generation of the implied broadcast event, since only events can have a date/time attribute. This rule will only be triggered when the thing referent it matched is a TV program. Thus far, selecting the correct rule has been a manageable issue, though admittedly the grammar and vocabulary is still limited. At this writing there are 84 syntax rules and that number should grow as Babble learns more specialized syntax use.

Babble is taught new syntax rules in the following manner. For every sentence Babble processes, it can draw a meaning map, such as the one at right, which represents the sentence "Karen drinks tea." The meaning map displays the sentence's surface structure on the left side of the map and its tridbit meaning representation on the right side, with rule lines indicating the rules applied to move from the surface structure to the meaning. The user can edit this map to teach Babble what rules to apply in the given context or to teach it new rules.

For example, if Babble does not properly interpret "Karen drinks tea from China" the map of its best attempt can be displayed and edited. Perhaps it understands the first part of the sentence as displayed at right, but is unfamiliar with how to process "from China." The user can highlight the words "from China" and manually construct the tridbit meaning representation that corresponds to "from China" in this context.

It seems a reasonable conjecture that humans learn syntax by repeated association of syntax patterns with meaning representations. Unlike Babble, humans have built-in mechanisms to construct meaning representations from the stimulus they receive. This skill is prerequisite to the development of language, since language seeks to transfer meaning to others. Since Babble does not have built in mechanisms to construct meaning representations from external stimulus (other than speech), a trainer must

Figure 4: Meaning map of the sentence "Karen drinks tea."

provide the tridbit meaning representation that is to be associated with the syntax pattern. Future work will involve making this a more automated learning process, perhaps by giving Babble methods by which it can "guess" what something is likely to mean and ask for user verification.

A flow chart of the entire process of going from words to tridbits is given below. Generating tridbits in response to syntax patterns is at the heart of the procedure. Backtracking and evaluations are performed until the sentence is understood or all the possibilities are exhausted, in which case the best interpretation is chosen.

After a sentence is successfully processed in phase 1, a post-processing phase is begun. During post processing the tridbits generated in phase 1 undergo five separate processing passes to accomplish the following:

1) Fill in clausal assertions
2) Fill in and reduce pronouns
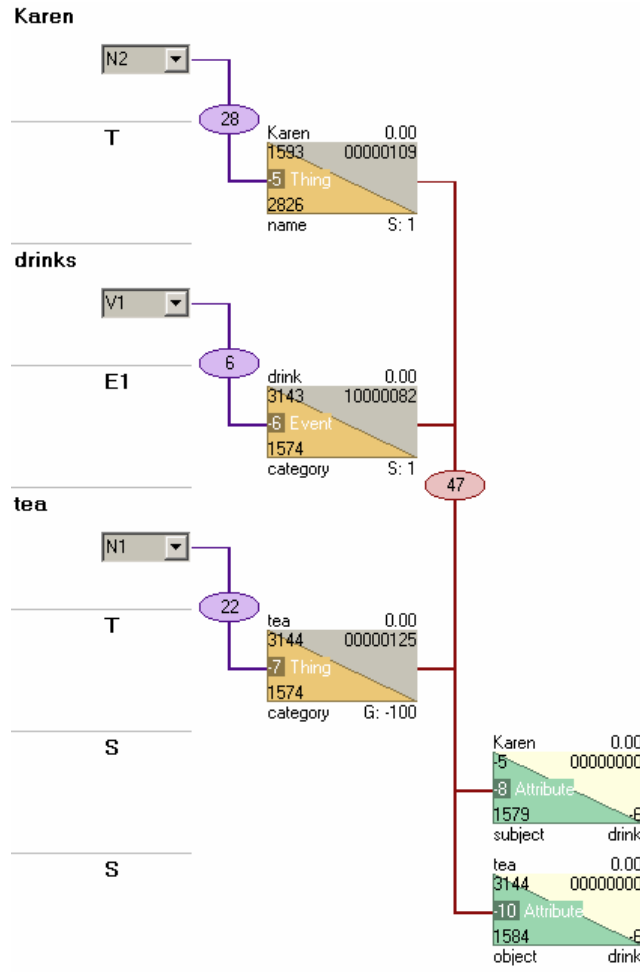3) Reduce other referents
4) Process questions
5) Process commands

```
┌────────────────────────────────────────┐
│ 1) Read in the text input, creating a   │
│    word use tree structure that         │
│    includes a row for each chunk in     │
│    the sentence.                        │
└────────────────────────────────────────┘
                   │
                   ▼
┌────────────────────────────────────────┐
│ 2) Make an entry for each chunk and     │
│    push it on the processing stack.     │
└────────────────────────────────────────┘
                   │
                   ▼
┌────────────────────────────────────────┐
│ 3) Chose word use for each chunk based  │
│    on prior experience within similar   │
│    context.                             │
└────────────────────────────────────────┘
                   │
                   ▼
┌────────────────────────────────────────┐      ┌───────────────────────────────────┐
│ 4) Process the processing stack by      │ ◄─── │ 11) Select rule to undo and undo  │
│    applying rules triggered by patterns │      │     it.                           │
│    found within the processing stack    │      └───────────────────────────────────┘
│    until no more patterns are found.    │
└────────────────────────────────────────┘
```

Main Processing (bracket over boxes 1–4)

```
                   │
                   ▼
              ◇─────────────◇
             5) Have all the entries that       No ──►
             were expected to be
             consumed been used?
              ◇─────────────◇
                   │
                   ▼
┌────────────────────────────────────────┐  Fail ──►
│ 6) Fill in causal assertions.          │
└────────────────────────────────────────┘
                   │
                   ▼
┌────────────────────────────────────────┐  Fail ──►
│ 7) Fill in and reduce pronouns.        │
└────────────────────────────────────────┘
                   │
                   ▼
┌────────────────────────────────────────┐
│ 8) Reduce other referents.             │
└────────────────────────────────────────┘
                   │
                   ▼
┌────────────────────────────────────────┐
│ 9) Process questions.                  │
└────────────────────────────────────────┘
                   │
                   ▼
┌────────────────────────────────────────┐
│ 10) Process commands.                  │
└────────────────────────────────────────┘
```

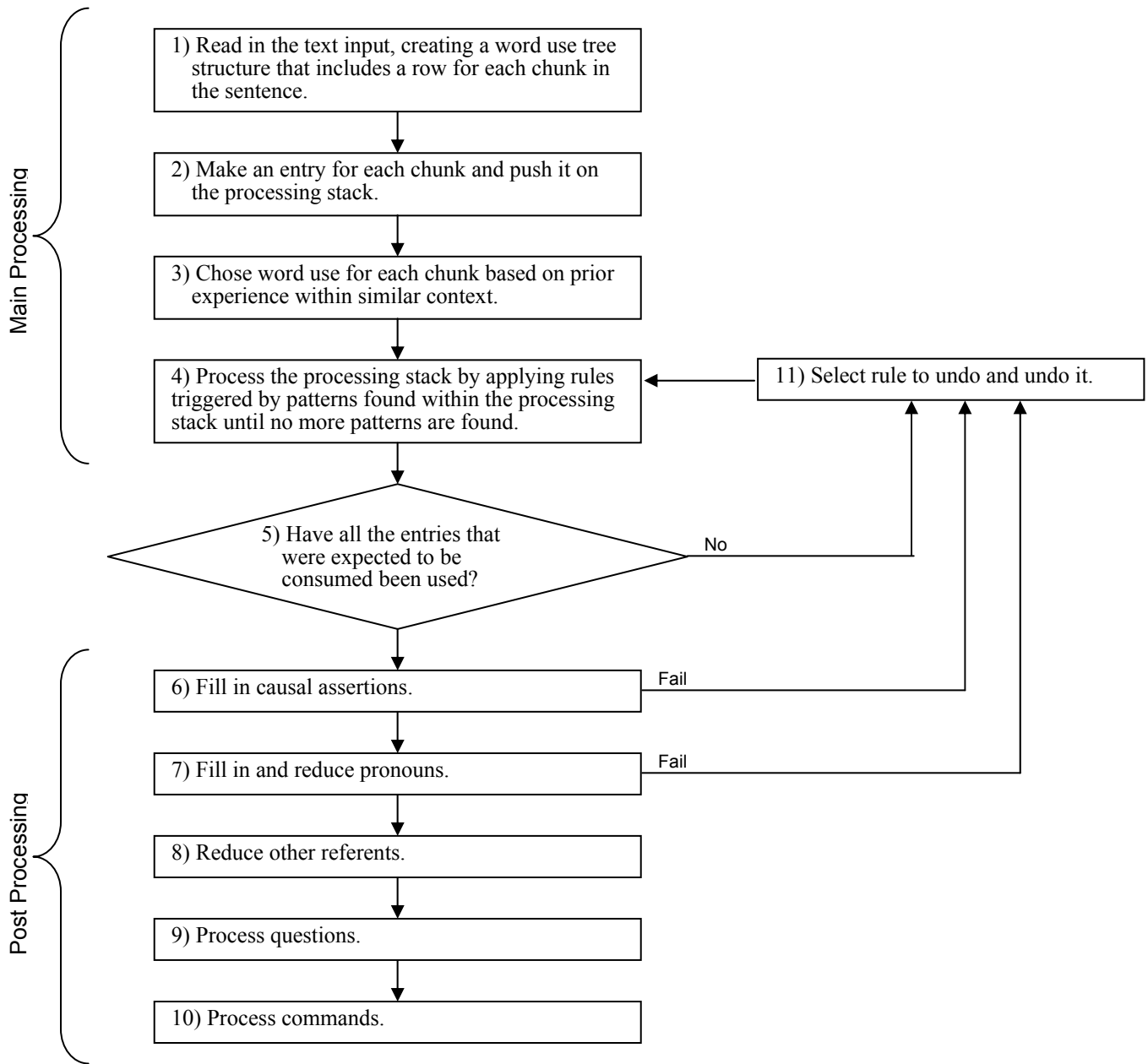**Post Processing** (bracket over boxes 6–10)

Figure 5: Flowchart of sentence processing

The first post processing function, depicted in box 6 of figure 5, fills in clausal referents. A clausal referent is a tridbit that a syntax rule refers to, but is outside of the pattern that triggered it. Many times this referent can be identified while the rule is being processed. An example is "Meg dislikes computers that crash," where "computers" is the clausal referent of "that crash".

In other cases, such as "To the finish line John raced," the clausal referent of "to the finish line" has not even been encountered when the rule is processed. The post processing that takes place in box 6, finds these unresolved clausal referents and fills them in, in this case with the race event.

The second post processing function, depicted in box 7 of figure 5, fills in and reduces pronouns. Pronouns generate referents that act as placeholders. The actual or originating referent is inferred by context. The type of pronoun determines how the originating referent is resolved.

First and second person pronouns, such as me and you, are filled in based on Babble's knowledge of who it is and who it is speaking to.

Resolving third person pronouns is more complex. To determine the originating referent for a third person pronoun, a qualification list is made from all the assertions that were generated about the pronoun referent. So in the sentence "He wears leather shoes," there would be one qualification on the list. The originating referent of "he" needs to be the subject of a wear event for which leather shoes is the object. The referents in the processing stack are examined, starting from the most recent and searching backwards until a referent that could fulfill the qualifications is found. If we expand the sentence to "He wears leather shoes even though he is a vegetarian," we would add the additional qualification that the originating referent of "he" belongs to the category of vegetarians. If the pronoun is the target or value of an equivalence assertion, an example of which is generated by "He is John Doe," the equivalence supplies the identity of the originating referent.

If a satisfactory originating referent is found, the value of the pronoun referent is filled in with the originating referent's ID. In addition the pronoun is reduced, a process that replaces any references to the pronoun's referent tridbit with the originating referent. This prevents two referent tridbits from referring to same thing.

The third post processing function, depicted in box 8 of figure 5, reduces other referents. Referent reduction was explained above in the context of reducing pronouns. In fact, all referents must be examined to see if they might be referring to a referent for which a referent tridbit has already been generated. This is a similar process to reducing pronouns, but entails significantly more judgment and subtlety. Consider the first sentence in the example below and the question that refers back to it:

June likes computers.
Who likes computers?

The verb "like" in each sentence will generate a referent tridbit, so the processing stack will contain two different like event referents. The like event must be examined after the question is processed to determine that it actually refers to the same like event as the first sentence and be reduced. Without doing this, the question can not be answered because Babble will think the question is asking about a different like event than the sentence.

One of the things that makes pronouns easier to reduce than other referents is that there is no question that they need to be reduced. A non-pronoun referent might be the originating referent or it may be a repeat. To make the determination, the qualifications of the referent being evaluated must be compared to its potential originator to judge the feasibility that they have the same referent.

In the example above, the "like" referent tridbit being evaluated has an unknown subject and an object of computers, while its potential originator has a subject of June and an object of computers. The objects match (although they may themselves need to be reduced first to determine this) and the subjects do not, but are compatible, so the judgment is that both referent tridbits refer to the same like event.

But the judgment gets quite subtle as we add additional attributes to referents. Consider the two sentences below:

She drove the sporty gold car to the store that sells computers.
The persnickety car crashed.

Asserting that the car in the second sentence is persnickety does not seem to prevent the judgment that the two cars are the same. However if the second sentence was:

The green car crashed.

The cars would be assumed to be different, because being green is incompatible with being gold.

It is not uncommon for an entirely different word to refer back to the originating referent, as would be the case in:

The vehicle crashed.

Another thing that makes pronouns easier to reduce is that their originating referent is expected to exist in the processing stack. Non-pronoun referent tridbits need to check whether an originating tridbit exists in the knowledgebase as well as the processing stack. For example, in order to answer:

What is the capital of Wisconsin?

The tridbit that represents Wisconsin needs to be identified with the tridbit in the knowledgebase that represents Wisconsin, because only the tridbit in the knowledgebase has a capital attribute of Madison.

The remainder of the processing performs any functions implied by sentence. If the sentence is a statement, nothing more needs to be done. If the sentence is a question or a command however, it needs to be answered or the requested action performed.

The fourth post processing function, depicted in box 9 of figure 5, processes questions. Questions are processed the same as any other sentence, resulting in a tridbit representation. In detecting if a sentence is a question, Babble does not rely on the question mark at the end of a sentence but rather looks for a tridbit that has either an inquiry attribute or inquiry truth value.

When an inquiry attribute is detected, the sentence is processed as an information inquiry. A referent tridbit with an inquiry attribute is typically generated by a pronoun such as who or what. The resolution of an inquiry referent proceeds much like the resolution of a third person pronoun. Consider the first question in the example:

Who likes computers?

If this were a statement with a third person pronoun it would be:

She likes computers.

Babble resolves the pronouns Who and She in exactly the same way to determine the originating referent. The difference is that the question needs to be answered, so Babble formulates an answer by coming up with a word or phrase for the originating referent.

Yes-no type questions ask for confirmation of an assertion rather than supplying information, for example:

Is five more than three?

"Is five more" is processed by a rule that generates an assert compare tridbit whose truth value indicates an inquiry. When Babble detects an inquiry truth value, rather than look for a missing referent given a list of qualifications, it sets out to confirm or refute the given assertion.

If Babble does not come up with the information it needs to answer either type of question, it has two additional methods it can invoke to generate new knowledge: reasoning and observation. These will be discussed in the next section.

The fifth post processing function, depicted in box 10 of figure 5, processes commands. Babble can be given a command such as:

Drive the red car to the store.

The tridbit representation of this command has three assert attribute tridbits. The first asserts the object of the drive event is the car. The second asserts the property of the car is red. The third asserts the destination of the drive event is the store. The first assert tridbit was generated by a rule that specifies it is used to convey the main theme of a command. The event within its pattern, in this case the drive event, is what the command is requesting to have done.

If Babble knows how to initiate the type of event being requested, it will do so, using any other information it has about the event, such as its object, destination, etc. For things it cannot do, such as "Drive the red car to the store" Babble will reply "I can not drive."

This methodology for invoking an action from a natural language request allows new abilities to be easily added. The ability to control household appliances and similar devices fits naturally within this framework, as well as most any action one could conceive of asking a computer to do.

4   Reasoning, observation and other uses of knowledge

The previous sections went through how meaning is represented by tridbits and the process of generating a tridbit meaning representation from a sentence. This section discusses methods for generating new knowledge from the tridbit knowledgebase and managing large amounts of related and sometimes contradictory information using tridbits.

The following example illustrates how Babble uses reasoning to generate new knowledge. Consider the sentences:

Jumbo is an elephant.
What size is Jumbo?

Babble has not been told directly the size of Jumbo, but it does know that elephants are big. It also has an if-then rule that says if the category of X is C and if C has property P then X has property P. Once Babble realizes it does not have any specific information about the size property of Jumbo, it will check for if-then rules that will result in the type of information it is looking for. It will find the if-then rule that transfers the properties of a category to its members. By applying the rule to Jumbo being an elephant, Babble generates a new assertion that Jumbo has the property of big, and is thereby able to answer the question. If-then rules are represented by tridbits stored in the knowledgebase.

Babble can also observe information in order to answer a question. Consider:

What time is it?

This generates a tridbit asserting that the time attribute of here-now is an inquiry referent. Babble finds no current tridbits that provide the answer, so it checks to see if it can generate useful information through observation. An observation is a sort of black box that Babble can invoke to attempt to fill in the value element of a specific attribute-target pair. It is the equivalent to a human attending to a specific attribute of a sensory experience and coming up with a value. If you are asked the color of your shoes and you've forgotten, you look down to retrieve the color attribute of your shoes. In the time example, Babble observes the time attribute of here-now by invoking the appropriate observation function, which reads the computer's internal clock. The information is placed in Babble's steam of consciousness, i.e. the processing stack, allowing Babble to answer the question.

These are two examples of deduction that Babble currently performs. Babble uses if-then rules to tell it what the consequence of a set of conditions or an action will be. Currently these rules are hand coded tridbits, but it should not be long before Babble can generate such rules from natural language, for example by being told:

If an appliance is unplugged it will not turn on.

Of course, the above statement is not always true. A battery-operated appliance will turn on even if it's unplugged. Since a tridbit knowledgebase simply adds tridbits as they are experienced, it can easily store both the rule and examples of contradictory events. This provides data that could be used to detect potential exceptions when applying a rule.

Babble does not strive for any global consistency between the tridbits stored in the knowledgebase. It is the cumulative effect of hundreds or thousands of tridbits stored in a knowledgebase that allows complex and even contradictory knowledge to be understood.

Consider this example. Babble is told by Alice that milk is healthy because it prevents osteoporosis. Babble stores this as one tridbit assigning a property of healthy to milk, two more to define milk as the subject of a prevent event and osteoporosis as its object and a last one to assign milk being healthy as the consequence of the prevent event. It would also be a good idea for Babble to add an assertion assigning Alice as the source of these tridbits, since the current information about milk and health is very complex and contradictory.

Later Babble reads an article by Walter stating that milk does not prevent osteoporosis. This generates a tridbit representing a completely contradictory prevent event. But the subject and object of both prevent events will be the same, the general concepts of milk and osteoporosis. So when Babble goes to search for information about milk or osteoporosis, both prevent events will be retrieved. Babble will need to evaluate in some fashion what it should conclude, if anything, from such contradictory tridbits. It might make this judgment using other attributes of the tridbits, such as their certainty values, how strongly reinforced each is or the source of the tridbits.

The situation gets more complex as new information is added. It could take thousands of tridbits to represent opinions from other sources, related studies, additional information about milk, osteoporosis, etc. But each thread of information would in some way be linked back to the concept of milk, represented by just a single tridbit.

The result is similar to what Sowa refers to as knowledge soup [Sowa, J. F. 1990]. A critical component in managing this knowledge soup is to ensure that each piece of information placed into the soup is related appropriately to the rest of the soup. With tridbits this requires that each unique thing the knowledgebase refers to is consistently represented by a single referent tridbit. Referent reduction is the process that enforces this by determining when two referent tridbits refer to the same thing and eliminating the redundant one. Previous examples illustrated some of the subtle issues involved. Adding a sentence to the knowledgebase, i.e. remembering it, adds a few more wrinkles to an already complex issue.

When a sentence is remembered, the assert tridbits generated by the sentence and any tridbits linked to those assertions are stored. Before being stored, the referents of the assertion must be reduced to referents in the knowledgebase if they exist, or established if not. In most cases, the new information is added to the existing information. But if identical information already exists in the knowledgebase, the existing information is reinforced instead.

Consider what happens if Marge tells Babble that she agrees with Alice, milk is healthy. The representation of Marge's statement should use the same referent tridbit for milk as Alice's statement. In fact, Babble might choose not to create another tridbit assigning a property of healthy to milk, but to reinforce the one created when Alice made her statement. That would make it impossible for Babble to differentiate the two statements in the future, but that might be a fair trade-off for reduced complexity. But what if Marge only drinks nonfat milk, which is actually the only type she finds healthy. Nonfat milk is a general concept like milk, but it does not represent all milk, just a subset, so it can not be represented by the same referent tridbit as milk. Nor can we just reinforce Alice's statement without compromising the meaning of what Marge has told us. Nonfat milk is a general qualified concept in tridbit terminology, a subset created from a general scope referent by adding criteria.

Consider how the addition of information affects the ability to answer simple questions such as "Is milk healthy?" When Babble has one data point that milk is healthy, answering the question is simple. When Babble has the three data points discussed above, it needs some way to evaluate and summarize the information before it answers. Developing optimal methods for summarizing overlapping tridbits into a manageable set from which speech responses can be produced is an area of future work.

The simple structure and referential consistency of a tridbit knowledgebase make it easy to access information in meaningful ways. These same characteristics will facilitate the types of manipulation that Sowa envisions, such as reasoning by induction or analogy. Future work will focus on building real world applications using this new paradigm, specifically in the area of device control. We hope to achieve a level of fluency and intelligent interaction to make such applications feasible. Babble's current abilities are noteworthy, and there is every indication that this framework is robust enough to go much further.

5   Acknowledgement

References

[Casati, Roberto, Varzi, Achille 2002] "Events", *The Stanford Encyclopedia of Philosophy (Fall 2002 Edition)*, Edward N. Zalta (ed.), URL = <http://plato.stanford.edu/archives/fall2002/entries/events/>.

[Fargues, J., Landau, M.C., Dugourd, A., and Catach, L. 1986] Conceptual Graphs for Semantics and Knowledge Processing. IBM J. Res. Develop., 30(1), pp.570-79.

[Friedland, Allen, Matthews, Witbrock, Baxter, Curtis, Shepard, Miraglia, Angele, Staab, Moench, Oppermann, Wenke, Israel, Chaudhri, Porter, Barker, Fan, Shaw Yi Chaw, Yeh, Tecuci, Clark 2004] Project Halo: Towards a Digital Aristotle. AI Magazine 25(4): 29-47.

[Katz, B. 1997] Annotating the World Wide Web using Natural Language. In *Proceedings of RIAO '97*.

[Lenat, D.B. 1995] Cyc: A Large-Scale Investment in Knowledge Infrastructure [Electronic Version]. Communications of the ACM, 38(11), 32-38.

[Sowa, J. F. 1984] Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Reading, MA.

[Sowa, J. F. 1990] "Crystallizing theories out of knowledge soup," in Intelligent Systems: State of the Art and Future Directions}, edited by Zbigniew W. Ras and Maria Zemankova, Ellis Horwood, New York, pp. 456-487.