TR-IT-0075

Generic Software for Language Understanding: a design based on layered abduction

John R. Josephson †*
Christopher Bailey-Kellogg *
Kevin Lenzo †

September 1994

Abstract

This report describes the design of LLU(Layered Language Understander). LLU is application-building software (a "shell") specialized for natural language processing, but designed to be independent of any particular language or level of processing (syllables, words, sentences etc.). It is a specialization of a more general generic abduction mechanism with support for layered abduction and for handling temporally-bounded hypotheses. It is possible that it is general enough for vision or other sensory modalities.

† ATR Interpreting Telecommunications Research Laboratories, Kyoto, Japan
* Laboratory for Artificial Intelligence Research, Department of Computer and Information Science, The Ohio State University, Columbus, Ohio

Table of Contents

Background 1
LLU 1
Design strategy 2
Some basic concepts 2
Layers, agoras, strata, and information pathways 3
Strategy for processing at each layer - summary 4
Top-down information flow 7
Summary of the control strategy 8
Hypotheses and Hypothesis types 9
Evoking elementary hypotheses 9
Instantiating elementary hypotheses 10
Determining what a hypothesis explains 16
Hypothesis relationships 16
Control of hypothesis assembly 19
Simulating concurrent processing - the Agenda mechanism 20
Processing expectations 21
Control of hypothesis refinement 22
Continuing hypothesis assembly 26
Retracting acceptance and rejection decisions 27
Recovering from mistakes 28
Learning 29
Acknowledgments 33
Bibliography 33
Appendix A: Specifying Hyptypes 34
Appendix B: Test Case 35

Background

Abduction is a pattern of reasoning that goes approximately as follows:

D is a collection of data (facts, observations, givens).
H explains *D* (would, if true, explain *D*).
No other hypothesis can explain *D* as well as *H* does.

Therefore, *H* is probably true.

Abductive inferences appear to be widespread in scientific reasoning, in diagnosis, and in ordinary life. Several researchers have proposed that abductive inferences underlie language understanding, e.g., (Hobbs, 1993), (Charniak, 1986), (Dasigi, 1991), (Josephson & Josephson, 1994). In some sense this is obviously true – we form what we take to be the best interpretation of an utterance or passage, and we do not accept it as correct unless we are confident that it is indeed the best interpretation. Nevertheless it is possible that language understanding is logically abductive without being procedurally so. It is an open question whether the information processing that actually occurs during language understanding is described usefully and accurately as abduction. Perhaps abduction provides only an idealized competence account of the inputs and outputs of the language understanding processes without providing any insight into the details of the processing.

It is a working hypotheses of our research effort that language understanding can be usefully described and computationally modeled (even emulated) by treating it as abductive inference. Specifically, we hypothesize that language understanding occurs in discrete layers, with abduction occurring at each layer, the conclusions formed at one layer being the data to be explained at layers above.

It is plausible that besides language understanding vision and other sensory modalities, human as well as robotic, can also be usefully computationally modeled as abduction in layers. If the information processing that occurs in the various layers and senses is functionally similar, then perhaps their mechanisms are similar too at a certain level of description. Thus one of us (Josephson) has proposed the "layered-abduction model of perception" according to which the information-processing mechanisms that occur in vision, in hearing, in understanding spoken language, and in interpreting information from other senses (natural and robotic) are all variations, incomplete realizations, or compilations (domain-specific optimizations) of one basic computational mechanism. (see Josephson & Josephson, 1994).

LLU

This document describes the design of LLU (Layered Language Understander). LLU is application-building software (a "shell") specialized for natural language processing, but designed to be independent of any particular language or level of processing (syllables, words, sentences, etc.). It is a specialization of a more general generic abduction machine with special support for layered abduction and for handling temporally-bounded hypotheses. It is possible that it is general enough for vision or other sensory modalities.

The computational strategy used in LLU is a direct descendent of computational strategies devised for diagnostic problems. These strategies can be described in domain-independent terms, and they have demonstrated a significant degree of domain independence by their successful use in various systems (Tanner, 1991).

LLU is implemented in C++. One difference from previous implementations of layered abduction (see Josephson & Josephson, ch. 9, 10) is that it is designed to support supervised learning of new elements and of the strengths and boundaries of parametric relationships.

Design strategy

Using the experience gained from constructing previous generations of generic abduction mechanisms, we have designed a new implementation to meet the demands of a specific project - that of building a system for speech recognition based on the C/D model of Osamu Fujimura (see for example Fujimura, 1994). In so doing we tried to engineer into LLU those aspects of problem solving that appear to be generic to language understanding, and engineer other aspects of the concrete problems to be solved either as domain knowledge proper, or as pieces of domain-specific problem-solving strategy. Our goal has been to minimize the need for implementing domain-specific problem solving strategy, making as much use as possible of generic mechanisms.

Some basic concepts

The set of findings that a hypothesis can explain is not in general the same as the set of findings that a hypothesis "expects to be true," which is not in general the same as the set of findings that potentially cue the hypotheses as being worth considering. While all of these findings tend to be causal consequences of the hypothesis, only the set of findings that a hypothesis can explain must consist only of such consequences. For example a hypothesis may expect preconditions to be satisfied, and any good correlates of a hypothesis may serve as cues, not just its causal consequences. The findings that a hypothesis "expects to be true" are typically the basis for setting the confidence score of the hypothesis and a basis for updating the confidence of the expected hypothesis if the expecting hypothesis is accepted.

Explanatory power might be the ability to explain more items, or the ability to explain better (more precisely, or more accurately). So far we have not used explaining-better in any of the machines described in the abduction book (Josephson & Josephson, 1994). (Though we have used hypothesis refinement, i.e., movement to a more specific hypothesis in a type-subtype hierarchy.) It is possible that for some domain it will be necessary to use explaining better, though we do not now know how to control hypothesis formation taking this into consideration.

By an "**explanatory hypothesis**" we mean a volatile entity created during problem solving activity that offers to explain (or potentially offers to explain) some particular finding or findings. Explanatory hypotheses may be contrasted with "predictive hypotheses," which are also volatile entities, but that have expectations towards findings, rather than explanations of them. Predictive hypotheses may simultaneously be explanatory hypotheses, and vice versa. Hypotheses may be abbreviated as "hyps." Conjunctions of hyps are hyps, often called "composite hypotheses." A single hypothesis which is not a conjunction (other than trivially as a conjunction of itself) is called an "elementary hypothesis."

A hyp will typically have a numerical confidence score, which may change during the lifetime of the hyp.

Hyps have **belief status** ("doxastic status" if we want a more exotic word). Initially the belief status of an elementary explanatory hyp is HYPOTHESIZED; this may subsequently be changed to one of: ACCEPTED, REJECTED, GUESSED, GUESSED_NOT or EXPLANATORILY_USELESS. A composite hyp consisting of ACCEPTED hyps is itself ACCEPTED.

"Findings" to be explained are hyps with belief status ACCEPTED. An "expectation" of some hypothesis is a hypothesis that is expected to be or to become true. If the expected hyp is accepted, then the expectation is said to "succeed;" if the hyp is rejected, then the expectation is said to "fail." An expectation may be negative - an expectation that some negative statement is true succeeds if the corresponding positive hyp is rejected, and fails if the corresponding positive hyp is accepted.

Hypotheses are to be contrasted with non-volatile entities called "hypothesis types" (abbreviated "hyp types" or simply "hyptypes") that are used for forming elementary hypotheses. "Frog" is a hyp type; "this frog" is presumably a hypothesis.

Layers, agoras, strata, and information pathways

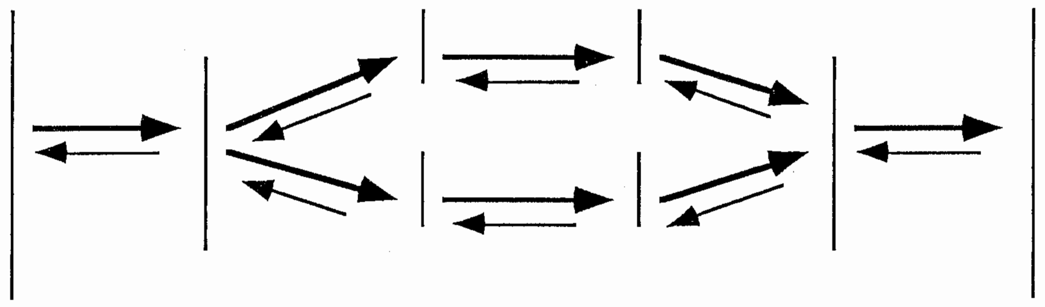
Computational models of information processing for spoken language understanding have commonly supposed an orderly progression of layers, beginning near the auditory periphery, where hypotheses are formed about low-level features, such as bursts or sonorance, and proceeding by stages to higher level hypotheses. Models intended to be comprehensive often suppose three or more major layers, often with sublayers, and sometimes with parallel channels that separate and combine to support higher level hypotheses. Audition, phonetics, grammar, and semantics have sometimes been proposed as distinct layers of interpretation for speech understanding. Within the phonetic layer various distinct sublayers have been proposed, such as phonemes, syllables, and words.

We call each locus of hypothesis formation an **agora** after the marketplace where the ancient Greeks gathered for dialog and debate. The idea is that an agora is a place where hypotheses of a certain type gather and contend and where, under good conditions, a consensus hypothesis emerges. In typical cases the emerging hypothesis will be a composite, coherent in itself, and with different subhypotheses accounting for different portions of the data. For example the syllable agora is the presumed location where syllable hypotheses are formed and accepted; each specific syllable hypothesis accounting for certain specific data from lower level agoras.

The agoras are organized in an information-flow network with a clear sense of direction defining the difference between bottom-up and top-down flow. Bottom-up processing goes from data to interpretation. The main output from an agora is "upward," the data to be interpreted by "higher" agoras. Another possible output is "downward," for example an expectation might influence the consideration and evaluation of hypotheses at lower agoras. Thus the relationship between "data" and "explanation" is a relative one, and an explanation accepted at one agora becomes data to be explained at the next. Although we sometimes use the word "layer" to describe an agora or a contiguous cluster of agoras, we do not suppose that the agoras are all neatly lined up. The paths may branch and join (but no cycles are permitted).

Major groupings of layers are called "**strata**" (for example the "articulatory stratum").

Generically



Potentially branching and joining.

Strategy for processing at each layer - summary

We suppose that the information processing at each agora is decomposed into three functionally distinct types of activity:

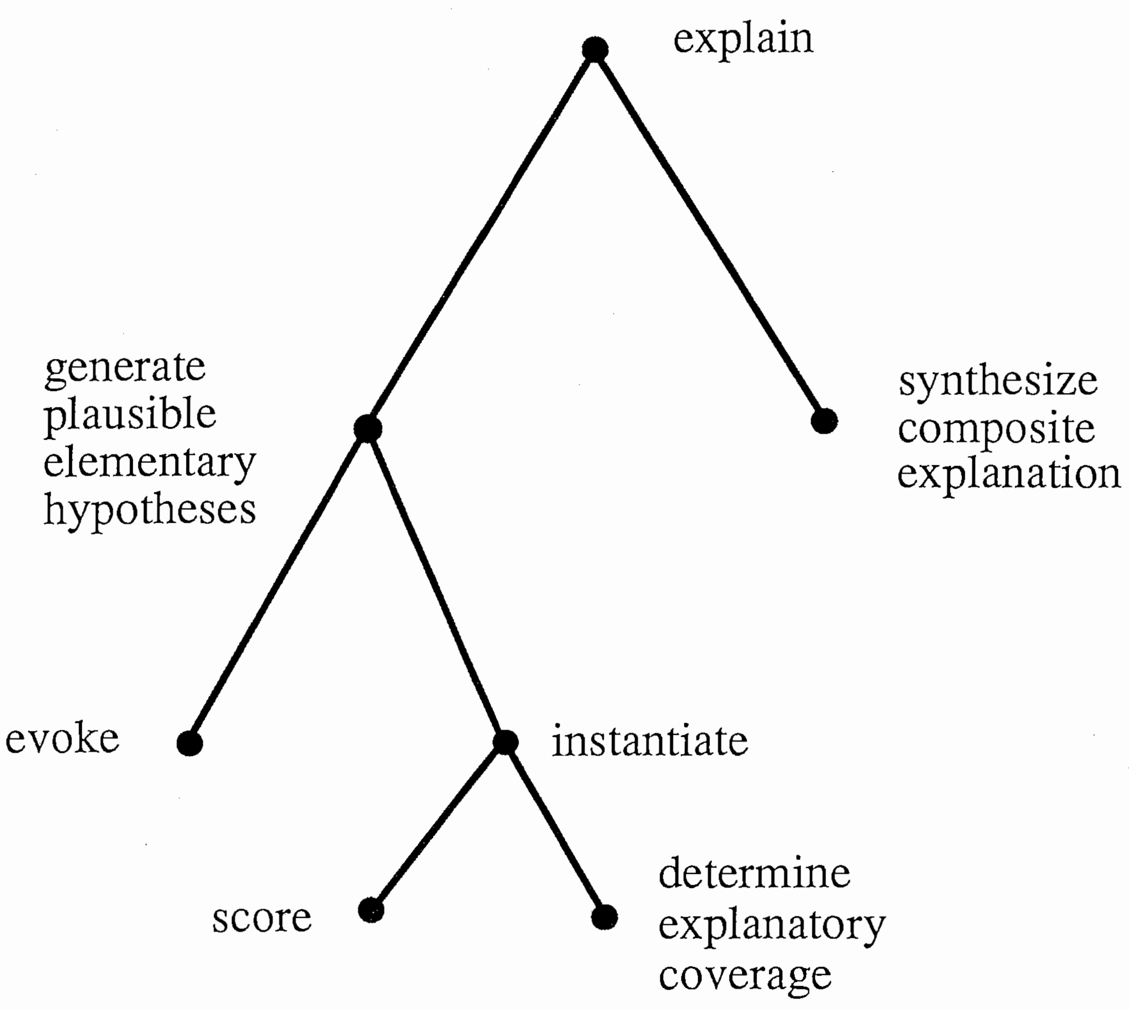
evocation of hypotheses,

instantiation of hypotheses,

composition of hypotheses.

Generating explanations by instantiation and composition

task-subtask breakdown


explain
generate plausible elementary hypotheses
synthesize composite explanation
evoke
instantiate
score
determine explanatory coverage

Evocation typically occurs bottom up, a hypothesis being stimulated for consideration - cued or triggered - by the data presented at a layer below. More than one hypothesis may be stimulated by a given datum. Evocation may also occur top down, either as the result of priming (an expectation from a level above) or as a consequence of data-seeking activity from above, which can arise from the need for evaluation of hypotheses in isolation, or from the need to discriminate between remaining alternative explanations. Evocations can generally be performed in parallel and need not be synchronized. Besides these pointed evocations of hypotheses from below or above, if hyp types are organized hierarchically according to specificity, exhaustive search can be performed efficiently to generate all hypotheses applicable to explaining some finding.

Instantiation occurs when each stimulated hypothesis is independently scored for confidence (evaluation) and a determination is made of what part or aspect of the data the hypothesis can account for (determination of explanatory scope). Instantiation is in general top down. (Under certain circumstances it is good enough just to score on the basis of voting by the stimulating data from below, and then no top-down processing need occur, at least for scoring. Although this strategy is less than logically ideal, it is computationally less expensive, at least in the short run; accuracy is traded for quickness.) During instantiation, data may be sought that were not part of the original stimulus for evoking a hypothesis. Each hypothesis is given a confidence value as a number on the interval [0, 1], which can be taken to be a "local-match" or prima facie likelihood, that is, a likelihood of being true based only on consideration of the match between the hypothesis and the data, with no consideration of interactions between potentially rival or otherwise related hypotheses. Typically, many evoked hypotheses receive low scores and can be tentatively eliminated from further consideration. The data that a hypothesis accounts for may or may not be identical to the data used to score the hypothesis or the data that evoked it.

During instantiation, the hypothesis set may be expanded by including subtypes and supertypes of high-confidence hypotheses, if the space of hyp types is organized hierarchically by level of specificity. Instantiation is most efficient when it is done by matching against prestored patterns of features, but slower processes of instantiation are also possible whereby the features to match are generated at run time.

The result of a wave of instantiation activity is a set of hypotheses, each with a measure of confidence, and each offering to account for a portion of the data. Within a particular wave of instantiation, hypotheses are considered independently of one another, so this can go on in parallel.

Composition occurs when interactions among the instantiated hypotheses are taken into account and (under good conditions) a coherent best interpretation emerges. At first, many hypotheses will probably have intermediate scores, representing hypotheses that can be taken neither as practically certain nor as of such low confidence as to be ignorable. So knowledge of interactions between the hypotheses is brought to bear to reduce the degree of uncertainty, increasing confidence in some of them, and decreasing confidence in others. The basic strategy is to try to solve the overall abduction problem at a given agora by solving a sufficient number of smaller and easier abductive subproblems at that agora. The strategy used is basically that of Machine 5, described in chapter 9 of Josephson & Josephson (1994).

This strategy begins the composition process by tentatively accepting the highest confidence, small abductive conclusions that can be identified, and then propagating the logical consequences of that acceptance along known lines of hypothesis interaction. If the hypothesis composition process stalls after accepting the highest confidence conclusions and propagating the consequences, and if some data remain ambiguous, then the process is continued by accepting somewhat less confident conclusions, and again propagating the consequences. The process halts when all data are interpreted or when the remaining unexplained data are too ambiguous to be discriminated with sufficient confidence (see chapter 9 for a fuller description). In the layered-abduction model, it is also possible that remaining ambiguities will be resolved later, either as a result of additional processing stimulated by downward-flowing expectations from layers above, or by reverberations of downward processing appearing later as revisions flowing upward from below.

Regions of data are ambiguous if, for each datum, more than one viable hypothesis exists, and no distinctly best interpretation can be chosen given the current evidence. Regions of ambiguous data (or equivalently, the corresponding sets of alternative hypotheses with no distinctly best combination) can be the targets of further data gathering, guided by the need to discriminate between the leading hypotheses. In medical diagnosis we might order more tests to try to resolve the differential diagnosis problem; in active vision we might take another look in the region of the ambiguous perception; in empirical science we might design and perform a crucial experiment; in interactive language interpretation we might ask for clarification. Further data gathering is governed by pragmatics - whether resolving the ambiguity is important for currently active goals.

In our implementation an "**abducer**" is a software agent that is in charge of the problem solving at a single agora. The agora is also represented by a software entity whose function is basically that of a bookkeeper. The abducer relies on an agora to keep track of problem state information and the abducer makes certain decisions that directly affect problem state. Different abducers in the system may have somewhat differing problem solving strategies, for example in the priority given to hypothesis refinement. The abducers in a system are managed by a global "abducer manager."

Top-down information flow

Downward-flowing information processing between layers can occur in at least four ways. One is that the data-seeking needs of hypothesis evaluation or discrimination can provoke instantiation (top-down evocation and evaluation of a hypothesis). Another is that expectations based on ACCEPTED hypotheses at one layer can prime certain data items (i.e., evoke consideration of them and bias their scores upwards). A third way is that hypotheses that are uninterpretable as data at the higher level (no explanation can be found, other than Noise) can be "doubted," and reconsideration of them provoked (also reconsideration of any higher level hypotheses whose confidence depended on the questionable datum can be provoked). Finally, data pairs that are jointly uninterpretable (e.g., two words, the co-occurrence of which cannot be reconciled syntactically or semantically) can be considered to be incompatible (to some degree of strength), and reconsideration can be provoked from above. In these ways higher level interpretations can exert a strong influence on the formation of hypotheses at lower levels. Layer-layer harmony is a two-sided negotiation.

Summary of the control strategy

We may summarize the LLU control strategy for hypothesis composition by saying that it employs multilevel and multiple intralevel island-driven processing. Islands of high confidence are seeded by local abductions and propagate laterally (incompatibilities, positive associations), downward (expectations), and upward (accepted hypotheses become data to be accounted for). Processing occurs concurrently and in a distributed fashion. Higher levels provide *soft* constraints through the impact of expectations on hypothesis evocation and scoring, but this does not strictly limit the hypotheses that may be accepted at lower levels.

This control strategy is somewhat similar to that of the HEARSAY-II speech-understanding system (Erman & Lesser, 1980), which had a layered blackboard upon which hypotheses were suggested, tested, and composed, and where they triggered the formation of further hypotheses. One important difference is that LLU replaces the general-purpose blackboard with a control structure designed specifically for layered, interpretive, inferential tasks. Although our control structure could in principle be implemented on a blackboard machine, doing so would impose an unnecessary and undesirable centralization of control. A second important difference is that HEARSAY-II's "islands of certainty" were based simply on the highest scoring hypotheses rather than on hypotheses whose initial scores were significantly higher than alternatives for explaining some datum. That is, HEARSAY-II's initial acceptance was "recognition driven," rather than "abduction seeded," thus producing initial hypotheses that were not very secure. As a result, HEARSAY-II encountered a significant computational expense from backtracking, and did not scale up well.

LLU uses the "EFLI" (Essentials-First Leveraging-Incompatibility) strategy for control of hypothesis assembly (Josephson & Josephson, 1994, ch. 9). EFLI may be briefly described as:

- Find data of lowest ambiguity (either a datum with only one possible explanation, an "essential hypothesis, "or a datum with one explanation being much better than all others)

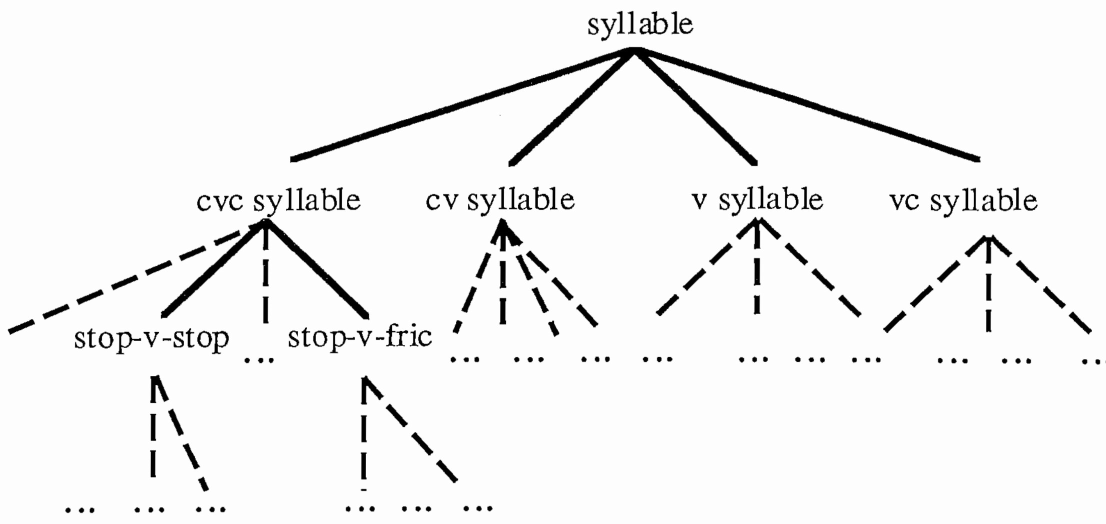
- Accept the best explanation for each low-ambiguity data point. That is, make a local, confident abduction.

- Propegate the consequences of hypothesis acceptance using knowledge of hypothesis relationships. This includes rejecting hypotheses which are (hard) incompatible with accepted hypotheses, and rescoring hypotheses that have expectations towards the accepted hypothesis.

- If necessary, lower standards for hypothesis acceptance and continue. (Lowering the standard for acceptance means to accept hypotheses that are best explanations, but are not best by such a large difference.)

Hypotheses and Hypothesis types

Type hierarchies for expressing hypothesis specificity



Hyp types may be arranged in hierarchies of specificity obeying subset semantics; that is, anything that is an example of the subtype is an example of the parent type. The basic data structure is that of a directed acyclic graph, rather than a tree, allowing for hyp types with multiple parents. (Commonly, however, multiple parenthood will not be needed and tree-structured hierarchies will be sufficient to adequately represent a domain.) The subset semantics is supported by arranging that if a hypothesis is accepted, all parents are accepted on up the hyp type hierarchy to the top. Conversely, if a hyp is rejected, all children are rejected, down the type hierarchy to the tip nodes; (implementationally, this does not require that instances are created only for the purposes of rejection).

Hierarchies of hyp types help with hypothesis evocation (triggering), hyp scoring, hyp refinement, and top-down resolution of ambiguity.

Evoking elementary hypotheses

So far in the present implementation, evocation is by cueing (as opposed to working top down through the hyp type hierarchy, for example). The responsible cues may be in the bottom-up data, may come from above, or may be at the same level (e.g., hypothesis cued by a confident predecessor hypothesis at the same level). In some cases evoked hypotheses may cue subtypes and supertypes. Cues are sometimes called "triggers."

Triggers are not the same as the expectations associated with a hyptype. An expectation is used for giving evidence for a hyp based on the acceptance or rejection of some other hyp. It has the side effect of stimulating consideration of the expected hypothesis if it has not already come under consideration. On the other hand the primary function of triggers is to stimulate consideration. Expectations have graded strengths, but triggers do not.

Instantiating elementary hypotheses

Generating an elementary hypothesis (at run time, not at learn time) has two distinct subtasks: determining an initial confidence score for the hypothesis and determining the initial explanatory coverage of the hypothesis.

Generating an elementary hyp occurs in two phases (which are not the same as the subtasks just mentioned) "attempted cheap rule out" and, if a cheap rule out is not possible, "hyp instantiation proper."

If a hypothesis is ruled out (cheaply or otherwise) there is no need to determine explanatory coverage. "Ruled out" is understood to be synonymous with having a confidence score that is below some threshold, which is set globally.

Cheap rule out is based on the status of related hyps already accepted or rejected. If a hyp has been accepted that is incompatible with the new one, the new one will be immediately rejected. When a hyp is first spawned, its expectations are checked to determine a score based on already accepted and rejected hyps (and without spawning any new hyps based on those expectations; we can call this "cheap matching"); if this initial score is below some preset threshold, the hyp will be ruled out. Rule out will not automatically remove the hyp from further consideration; it may be revived given sufficient provocation, but rule out will put the hyp into a state of dormancy where it will normally consume no more computational resources.

Hypothesis instantiation proper may fail (that is, result in ruling out after all). Alternatively, instantiation will succeed, setting a confidence score that is above the rule-out threshold and determining explanatory coverage (hopefully non-nil).

Confidence scores

The confidence score of a hyp can be thought of as a probability, and will usually be based on kept frequency counts. Scores are in the [0,1] interval.

The recognition (confidence-score setting) knowledge of hypotheses is encapsulated, allowing for the engineering of specific scoring methods for specific hyp types.

However LLU provides a universal scoring function based on how well expectations are met. Typically a hypothesis type is associated with a set of positive and negative "prototype features" ("expectations"), and positive and negative relationships among the prototype features (relationships among features is not yet implemented). The features and relations determine the confidence score of a hypothesis in the current state. Each feature that matches a positive prototype finding increases the confidence of the hypothesis, while each input that matches a negative prototype finding decreases the confidence. No evidence incoming (an expectation not having belief status ACCEPTED, REJECTED, GUESSED, or GUESSED_NOT) has no effect on confidence score. Finding relationships contribute (or will contribute) to the score in a similar manner.

Some prototype findings simply specify that an event must occur within a certain time interval relative to a key event, while others specify possible values for data. Similarly, relationships may include parameters specifying how findings are related.

The scoring function does not directly take account of the confidence scores of expected items but rather just their belief status. Thus the scoring function will typically look like:

confidence(h) = f(threshold-parameters, weight(exp-1), weight(exp-2), etc.),

where weight(exp-i) is a function of:

whether exp-i is T, F, or U,

the forward likelihood of exp-i given h,

the forward likelihood of ~exp-i given h,

the backward likelihood of h given exp-i, and

the backward likelihood of ~h given exp-i.

T means that the hyp has belief status ACCEPTED or GUESSED; F means that the hyp has status REJECTED or GUESSED_NOT; U means that the belief status of the hyp is none of these. By taking account only of the belief status of the expected items, the expected item needs to report back to the h scorer only when acceptance or rejection is determined, not at every update of its own score.

If exp-i is U, then weight(exp-i) is neutral; otherwise weight(exp-i) is a function of the likelihoods.

The likelihoods are estimated based on frequency counts.

The threshold-parameters have the function of compensating for the unknown statistical dependencies among the expectations.

Let H be a hyp type and E be an expectation of H. In the context of cases where H hyps are considered, we count only cases where both H and E are determined. (We may require H to be confirmed by the user, while E may be confirmed by the abducer; that is a separate issue.) We keep 4 counts:

h_and_e,

h_and_not_e,

not_h_and_e,

not_h_and_not_e

Note that we are counting occurrences in 4 disjoint classes of events whose union is the set of events where both E and H are determined. Let n be the total count of events. Then

n = h_and_e + h_and_not_e + not_h_and_e + not_h_and_not_e .

Let e be the count of times that E is T. Then

e = h_and_e + not_h_and_e.

Similarly, if ~e is the count of times that E is F,

~e = h_and_not_e + not_h_and_not_e.

Now suppose we have a new case where we want to score an H hyp, and suppose that E has been determined to be T. What we want is an estimate of the conditional probability p(H | E).

In classical probability theory,

p(H | E) = p(H & E) / p(E)

which we can estimate from our counts as

(h_and_e / n) / (e / n)

which simplifies to

h_and_e / e .

Substituting for e we get

estimate of p(H | E) = h_and_e / (h_and_e + not_h_and_e) .

Similarly, if we want to score an H, but E has been determined to be F, we can derive

estimate of p(H | ~E) = h_and_not_e / (h_and_e + not_h_and_not_e).

Note that we can use these counts to decide whether E is giving useful information about H. When these estimators are near to 1/2, not much evidence is being given.

Note also that these counts are not enough to pick up information about whether E is often determinable when H is being considered. It is possible that checking for E is not worth the effort because it is so rarely determinable; if we keep counts only for when both H and E are determined, we cannot pick up this fact. The solution appears to be to keep one or more further counts; perhaps the best scheme is simply to keep a count of h_determined_and_e_undetermined.

This leaves at least two remaining questions about scoring. One is how to combine the evidence from separate expectations. Our intention is to use sigmoid functions that are continually adjusted based on feedback about correctness so that h's overall confidence tends to mirror h's actual likelihood of being true. The other question is how it works if H has been accepted and E has not. How does H give evidence for E? The simplest answer seems to be that expectations are symmetrical in the sense that, if H expects E (with some concomitant statistics) then E expects H (based on those same statistics); thus, when H is accepted, E gets rescored because one of its expectations has now been determined.

Frequency counts are accumulated when the system is in **learning mode**. Besides the frequencies for expectations, frequencies are accumulated for how often hyps turn out to be true and false given that they come under consideration. These frequencies are useful for determining a case-independent initial score (a prior probability) for circumstances in which there are no determinate expectations for determining the confidence score.

Hypothesis scoring occurs iteratively. That is, the score may be updated several times as the truths of its expectations are determined. Scoring stops once the hyp is accepted or rejected. (Though some forms of retraction are available, which might cause scoring to restart.)

If a hyp is accepted or rejected, evidential force moves from the hyp to its expected items; accepting a hyp causes its expected items to be rescored. This rescoring of the expected item may not use the same weighted threshold scoring function described previously, since hyp scoring is encapsulated and this scoring function is provided by the system without being mandatory.

Evidential force only flows from hyps determined to be T or F to hyps whose {T, F} status has not been determined. "Expectation" is a two-way link (annotated with statistics) over which evidence can flow in either direction, depending on which side is known and which is under consideration. The association between h and e will be marked as useless only if it is useless from both sides.

Each time the score is updated, a hypothesis informs the abducer of its current score and the maximum score still possible.

The supers of an expected hyp provoke a "weaker response." That is, a hyp expecting a particular other hyp for scoring, gets a weaker response from supers of that feature. For example, if the hyp is looking for an "i" but the vocalic channel is showing the presence only of an unrefined vowel (that is, a vowel hyp is accepted, but it has not been refined), then the scoring of the hyp acts like a weak "i."

This raises two questions almost immediately: (1) how much weaker should the response be? and (2) what should happen if the super has been accepted and then refined, with something other than the expected feature being accepted?

Question (2) is easy. What should happen in this case depends on whether what has been accepted is incompatible with the expected feature. If so, the expected feature has been rejected (explicitly or implicitly). If not, then it is the same as if the super had not been refined. It is true that some additional information is present from the refinement that has been accepted, but trying to harvest that additional information is probably not worth the effort.

Returning to question (1), how much weakening of response is appropriate? Quite possibly it does not matter much; it is the qualitative relationships that count. But having said that, we still need to choose a method. So let us assume a hierarchy in which a super S of the desired feature F has been accepted. This need not be an immediate super; but if any hyp is accepted, all its supers are accepted, so let us assume that S is the most refined super of F that has been accepted. If S is not unique (which can happen if the hierarchy is not a tree), then we will deal with that separately. So for now let us consider that S is the unique most refined accepted super of F. Some hypothesis inquires whether F has been established so it can set confidence. Let us assume that a "yes" would bring a response of 1; "no," meaning F has been rejected, brings a response of 0. Total ignorance brings a response corresponding to the prior probability of F. Total ignorance corresponds to S being the root node (assumed to be always T). Then the ideal response is $p(F \mid S)$, the conditional probability of F in S.

That is the ideal. Obviously we do not have any true probabilities (even assuming that "true" probabilities are a meaningful notion in this context). What we have are some observed frequencies. A good, compact way to keep track of the frequencies that allows for a computationally inexpensive estimate of $P(F \mid S)$, is that we keep track of the frequencies of occurrence of each subtype as a proportion of occurrences of its immediate supertype. Then we simply multiply the proportions on the path from F to S.

A hanging question is what to do if there is more than one minimal established super of F. We just take the maximum of the relevant conditional probabilities. This should be good enough.

Possible confidence trajectory of a single hypothesis

case-independent confidence score (prior probability)

↓

case-specific initial score from instantiation (prima facie posterior probability)

↓

score updated as a result of propagating consequences of accepted hypothesis

↓ ↑

more updating from propagation

↓

hypothesis accepted (or rejected)

↓

retraction of acceptance (or rejection, etc.)

Determining what a hypothesis explains

A hypothesis can explain data only at "lower" agora, never laterally or above. Determining what a hypothesis offers to explain may require an ad hoc method for each hyp type. That is, it may be hyp-type specific knowledge, with little systematicity. It may be engineered as a system is built, or specified as a result of a learning process. A typical data structure for determining what a hyp type can account for is a list of features which an instance hyp will offer to explain, should they be present. For explaining a scalar feature, the entry in the can-explain list may specify a maximum value that hyps of the type may account for.

When a finding occurs, it asks for potential explainers. All hyps (accepted or not) that can account for it respond. If a responder is already accepted, the finding can immediately be marked EXPLAINED. Otherwise it tries to pick the best explainer, etc.

When an abducer notices a new hyp, it asks the hyp which of the unexplained findings it can account for. When an abducer notices a new finding, it asks each of the previously-noticed hyps whether or not it can account for the finding. Thus nothing is missed. A hyp determines that it can account for a finding using a kind of prototype (as with expectations) of hyptype, time, and params that must match. When a hyp can account for a finding, bi-directional links are set up so that an abducer deciding on a finding can consider all possible explainers, and an abducer accepting a hyp can mark all the findings that it explains.

Hypothesis relationships

Enumeration of relationship types

Hypothesis relationships can be considered to be of two general types, each with its own kind of significance for the problem solving:

1. *explanatory relationships* (e.g., that one hypothesis is capable of explaining another, or that hypotheses overlap in what they can account for)

2. *interactions of mutual support or incompatibility* (e.g., resulting from causal or logical relations)

For example, two hypotheses might offer to explain the same findings without being especially compatible or incompatible causally, logically, or definitionally. On the other hand, hypotheses might be mutually exclusive (e.g., because they represent definitionally distinct subtypes) or mutually supportive (e.g., because they are causally associated). In general the elements of an explanatory differential need to be *exhaustive* of the possibilities so that at least one must be correct, but they need not be mutually exclusive. If they are exhaustive, then evidence against one of them is transformed into evidence in favor of the others.

Following are some specific types of possible hypothesis relationships:

1. A can account for (explain) B.

2. A and B are mutually compatible and represent explanatory alternatives where their explanatory capabilities overlap.

3. Hypothesis A is a subhypothesis of B (i.e., a more detailed refinement).

4. A and B are mutually incompatible. (See next subsection.)

5. A implies B. (See next subsection.)

6. A and B cooperate additively where they overlap in what they can account for.

   If such interactions occur, this knowledge needs to be incorporated into the methods for computing what a composite hypothesis can explain.

7. Using A as part of an explanation influences the proper estimation of confidence in B either positively or negatively . (See next subsection.)

   Such an interaction occurs presumably because there is some statistical association arising because of some underlying causal relationship.

8. A, if it is accepted, raises explanatory questions of its own that can be resolved by appeal to B.

   This is the kind of interaction that the layered abduction mechanism is designed particularly to support.

9. If A is accepted B should be considered.

Hard and Soft hypothesis relationships

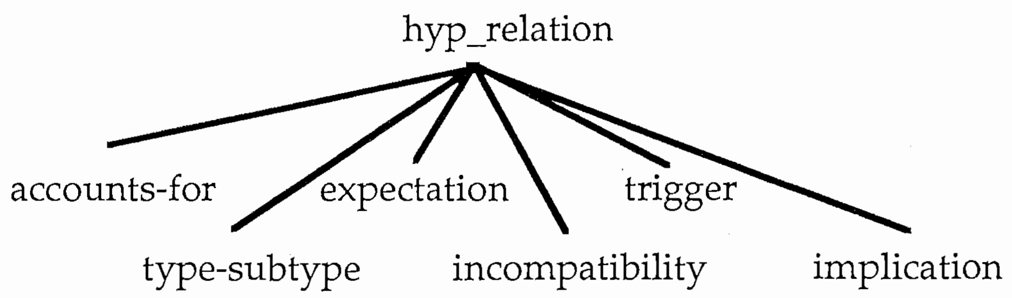
We treat relationships of "hard incompatibility" and "hard implication" differently from "soft" relationships of "sympathy and antipathy" among hypotheses.

Hard implications and hard incompatibilities come from "definitional" sorts of knowledge, rather than empirical-association knowledge. Besides engineering them in by hand, the system would know about hard incompatibilities when it learned to make a new distinction; when a category gets split into two (or more) mutually exclusive subtypes, the relationship is one of hard incompatibility (this hyptype splitting is not worked out yet). The main difference in processing between hard and soft incompatibilities is that accepting a hyp leads the abducer directly to reject hard incompatibles, while it leads directly only to lowering of the score of soft incompatibles. Soft incompatibility comes in degrees, hard incompatibility does not.

Hard implications also do not come in degrees, and act similarly: when a hyp is accepted, anything hard implied by it (a consequent) is accepted too; if the consequent cannot be accepted (e.g., it has already been rejected) the implying hyp must be immediately retracted. Logically, a hyp hard implies its super-hyps (in the refinement hierarchy), though it is not necessary that the hard-implication mechanism be used to process acceptance and rejection over sub-super-hype relationships, which is a distinct special case.

Hard incompatibilities are symmetric.

Hypothesis Relationship Classes



Besides the relationships shown in the figure, there may be a need for two kinds of consequences of hypothesis rejection: exhaustiveness (rejection of A automatically induces acceptance of B) and sink-together (rejection of A automatically induces rejection of B). Whether these additional relationships will be needed should be determined by the domain and what appears to be the most natural way of encoding the knowledge. No such need has yet appeared.

It appears to be most straightforward to implement hyp relations as classes where an instance of the relation is set up for each relation to be maintained between hyptypes. Any statistics to be kept are kept with this instance. Each of the hyps participating in a relation hold a pointer to "an instance of the instance" of the relation, so either hyp can access the other through the relation.

Constituency

Constituent structure seems important and deserving of LLU support in its own right. For example, a phrase is made up of words, a phonetic word is made up of syllables, a syllable is made up of onset, coda and affixes, etc. It is the part-whole relationship that is important. This seems to occur in nonlinguistic perceptual domains too, as for example a perceived object may have perceived parts, or a visual region may have an upper part, etc.

In linguistic domains the entities we hypothesize are "functional objects" and so (without going into a long rationale) the whole helps to explain the parts. Thus the "parts" associated with a linguistic hyp are presumably included as a subset of what the hyp explains. Yet the relationship seems to have more to it than that. We have parsed a phrase when we know what kind of phrase it is, and when we know how each role in the phrase type is filled. Thus it seems that sometimes a hyp should be thought of as a kind of slot-and-filler structure where the slots represent roles to be filled by constituents. Instantiating the hyp initiates an attempt to locate appropriate constituents, and the score of the hyp properly depends on how well it does at finding those constituents.

In our present implementation we can handle all this by treating the constituents as expectations and also things that can be accounted for (but there is no automatic coordination between these). But it seems that constituents are more than simply expectations and things that can be accounted for - they are part of the detailed identity of the hyp. That is, "behind the house" is not just an instance of a prepositional phrase, it is the particular prepositional phrase that has these specific constituents. Furthermore, it is impractical to prestore "behind the house" as a hyptype awaiting instantiation; the instance needs to get its particular identity at run time. Yet expectations and accounts-for specifications, as we have set them up, can expect and account for typed entities. This is functionally the same as constraining slots to have fillers of specified type. Thus it appears that we have all the representational power already for slot-and-filler structures, thought it would be nice to be able to compactly specify that a hyptype is constituted of a certain structured set of constituents, and have the result, among other things, that the constituents are constrained to be within the temporal scope of the hyp.

Besides composing hypotheses by "hypothesis assembly," it seems important to be able to compose hypotheses by filling roles in structured hyps.

Control of hypothesis assembly

The system includes agents called "abducers" that are responsible for hyp acceptance and rejection, and thus for hypothesis assembly. There is one abducer for each agora. Agora are more passive agents, primarily responsible for bookkeeping functions.

When an abducer gets stuck, it reports that to a global manager; when all abducers are stuck, the manager relaxes the confidence required for acceptance. (In LLU, for the first time, this confidence is based on the sum-of-scores-of-alternatives. In the preceding abduction machines described in Josephson & Josephson it was based on the difference between the best and next-best hypotheses).

There is no reason to tightening the confidence requirements again for interpreting the same data, since the overall confidence in the interpretation has already been reduced and the damage done, so to speak. The confidence requirements rise and fall across the whole machine, not locally for each abducer, so that the subproblems that can be solved most confidently will be solved next, no matter where they appear.

However, as new data comes in, the required confidence should head upwards again in some fashion. Exactly what fashion is not clear. The main alternatives seem to be: 1 - to start again with the highest standards (or course carrying over the influence of the expectations generated from interpreting the previous data, but that seems ok at the moment. It represents a kind of informative influence); 2 - drift upwards a click (this makes general sense in that the circumstances of noise, ambiguity, or whatever, that caused the necessity of lowering the confidence requirements will in general not change rapidly. It probably does not matter much. Alternative 1 is more logically strict, in the sense that it begins looking for the strongest possible islands of confidence on which to base subsequent processing; but probably alternative 2 is slightly more efficient (less likely to waste time being overly strict and getting nowhere). We use alternative 1 to begin with. We can change it later if it appears to incur too high a processing cost for the added strictness.

How are hyps rejected?

There are essentially two kinds of reasons for hypothesis rejection: poor match with data, and incompatibility with something that matches uniquely well with the data. [The terms "data" and "matching" used broadly here.] Essentially, the case of mismatching with data does not lead immediately to rejection - most immediately it leads to low confidence score. Rejection for low confidence score is under control of the relevant abducer, and depends on a preset threshold. the abducer rejects these low-scoring hypotheses at its discretion. This puts the threshold of ruled-out-ness under abducer control, where it belongs, so it can change dynamically if necessary.

Simulating concurrent processing - the Agenda mechanism

In order to simulate in an uncomplicated way the intrinsically non-serial problem solving architecture of LLU, we incorporate an agenda mechanism hiding the underlying seriality of the implementation from the higher-level mechanism, which is then written in concurrent-processing style code. We can thus tamper with the agenda mechanism as desired in order to achieve (or control for) various effects of concurrency. This design will also smooth the path for porting the mechanism to parallel architectures.

The agenda mechanism maintains a task set from which it picks tasks for execution. (The tasks correspond roughly to executions of the Invoke function in the IGTT toolset; see Josephson, et al., [1989]). At some future time we will probably want to gain greater control of the simulated parallelism by tagging tasks with their originating abducers, or with the originating subtasks, or with priorities that will be meaningful for scheduling; for now, however, agenda tasks are all assumed to be all the same type and priority.

Initially three alternative scheduling algorithms are provided: RANDOM, FIFO, and LIFO. RANDOM is the basic parallelism simulator in effect representing the hypothesis that execution order is relatively insignificant. Presumably FIFO or LIFO will work significantly better if there are certain kinds of dependencies, and these can be used experimentally as alternatives.

Processing expectations

Hypotheses talk to each other in that expectations report to their expectors, but it is always the abducer that chooses to accept or reject a hypothesis. After that point, the hyp can report to its expectors. In principle, sometimes, the abducer may decide not to reject, e.g., it might judge that there were masking conditions so that the expectation not being detected as fulfilled might not imply that it was false.

Expectation recursion

HIGH importance is assigned to first-order expectations and LOW importance to their expectations, with LOW importance sometimes being raised to HIGH if an expectation is very important for something else or is somewhat important for a lot of something elses. LOW importance expectations do not create their own expectations but do look around to see if any of them are already out there. Thus a LOW importance hyp will get a score based on the threshold and based on those terms of the scoring function that already have non-neutral values. This promises to be very efficient. If a LOW importance hyp gets an initially low confidence score and there are better hyps available to the abducer for explaining something of interest, it stays LOW in importance indefinitely. On the other hand the abducer may bump it up to HIGH importance based on explanatory need. A third possibility is that a LOW importance hyp gets a high confidence score and if it has no explanatory rivals with high scores, the abducer may decide to accept it with no further scoring. So HIGH verses LOW importance marks which hyps stimulate new evidence gathering. This scheme might be more complicated than necessary, but simpler might not work well enough.

Delta expectations

The basic idea of delta expectations is that if H expects an important E which is not met, this will count as evidence against H unless the divergence from expectation can be accounted for (i.e., explained). Logically, it seems that the divergence finding, "E-prime rather than E," needs to be posted to some agora, and an explanation of it pursued. If no explanation is found, handle appropriately as negative evidence, else handle as not being negative evidence. The problem with this logical solution is that it may have stimulated much unnecessary processing, and determining a score for H cannot be held ransom to some open-ended recursive abduction. So it seems best to give some update to the score of H based on the failed expectation, while awaiting further info about whether the delta can be explained. So, two cases:

1 - it is known that a divergence of this sort can never be accounted for;

2 - either it is known that a divergence of this sort can sometimes be accounted for, or it is not known whether such a divergence can be accounted for. (Presumably this knowledge is associated with the "finding concept" of "E-prime when E was expected").

Case 1 is easy to handle - the negative evidence has its appropriate impact (if H is still under consideration, its score is reduced.)

Case 2 - it would be good to set a score initially, then update later if/when it is determined whether the divergence can be accounted for. It seems clear that the initial scoring should take the evidence at face value as negative evidence. Then, how vigorously an explanation for the divergence is pursued should depend on the importance of scoring H, and the significance for the scoring of H of handling the failed expectation; that sets the priority for explaining the finding of divergence.

It is not clear what the priority is for cleaning up and implementing delta expectations. We have not done it initially.

Control of hypothesis refinement

At the problem-solving level there will be a need for hypothesis refinement. That is, commonly, hypotheses will be instantiated in general form and refined later. It appears that a common pattern for impetus for refinement, possibly the dominant pattern, is that we need to refine a hypothesis because it turns out at higher levels to matter. For example, something like this: our initial wave of processing leaves a word ambiguity between, say, /bit/ and /bet/ because the vowel was still ambiguous, so a stimulus from above, driven by the need to disambiguate, causes a stimulus to refine the categorization of the vowel (which may in principle provoke lower-level hypotheses to refine).

In principle, refinements may be of at least two kinds. One is where there is a shift to a preestablished subtype, and the other where there are parametric changes, typically increases of precision. These two might be mixed.

We handled hyp refinement in abduction machine 3 (Josephson & Josephson, 1994, ch. 4) but only the kind involving moving to a subtype.

Composite hypotheses may have type-subtype relationships between the members. The parent hypothesis would normally defer to the child to do the explaining, though there may be cases where they should share the duties (where multiple levels of description are involved, e.g., that it is a stressed syllable explains why it was longer and louder than average, but that it's a stressed /ri/ explains why it is such and so degree loud). In general we hope to avoid this complication, and simply have parent hypotheses defer explanatory power to any of their children that are also accepted into the composite hypothesis.

To perform abductions we need to reason by exclusion; we need to make acceptance judgments like "Accept H-1 because it's the only way to explain f" (or the most plausible way, etc.). So to accept H-1, we have to be reasonably sure that we have considered all of the plausible alternatives ways to explain f. This can be done effectively if in general we can determine what a hypothesis might be able to explain *at most*. If H-2 *might* explain f, we cannot go ahead and confidently accept some other hypothesis to explain f. So H-2 needs to tell us quite plainly if it cannot explain f. It can do that by telling us what it could explain at most (if it lives, if its refinements turn out that way).

If a hypothesis can be counted on to offer to explain anything that any of its refinements might offer to explain, then having accepted a hypothesis (or if it has high promise of solving unsolved explanatory problems), we can go ahead and consider refinements of it, working down through the refinement hierarchy, confident that previous acceptances will not be overridden (on that account anyway). If, on the other hand, refinements of hyps can explain more than their parents, we can no longer work top-down through the refinement tree.

Thus we adopt the principle: what a hypothesis "offers to explain" is understood to include everything that it might explain, including what any refinements of the hypothesis might explain.

Processing refinements: If the hyp to be refined has been accepted, then everything that it is explaining should be temporarily retracted (marked as unexplained), and an attempt made to instantiate hyps for the subs, and any subs for which instantiations are successfully made, should be included in the set of candidate explainers. If the hyp to be refined has not been accepted, and is only being refined in an exploratory fashion, then everything is the same except the need to retract EXPLAINED marks.

For refining a hyp that has been previously accepted, if all goes smoothly, the abducer will chose one of the subs for explaining what the super was explaining. I could be that something other than one of the subs will be accepted as better, in which case so be it, the original hyp should then be retracted as being explanatorily superfluous.

Refinement by parameter narrowing can sometimes be made to work through subtypes. Rather than iteratively trying to narrow the parameters and ensure that the hyp is still accepted, which could be costly, subtypes could be defined with larger steps in the narrowing of parameters. One main difficulty with refinement of parameterized hyps is that, as much as possible, we want to do automatic learning. How is a hyp to learn that it is supposed to have params, how many it should have, what ranges they should cover, etc.? Thus if good dividing lines can be drawn, hyps with different params could just belong to qualitatively distinct hyp types, and it seems clearer how to learn when different hyp types apply. Almost certainly we will need parameterized hyps, and refinements of them that come from making the parameter estimates more precise by subsequent processing. We will have to see what looks like the best way to do this in our domain. We may have to engineer parametric refinement of parameterized hyps, even if we do not know how to use learning to set them up. On the other hand it may work out just to use discrete subtypes (but that may not be a very natural way to treat them). We will have to wait and see.

Typically the sub-hyp will take what parameters it wants to from the super, and will shadow or override what it wants to. Parameters are inherited in order of the superclass entries in the subclass definition.

A sub-hyp can have multiple supers - we assume the semantics of AND, that is, the sub-hyp is construed as belonging to the classes defined by *both* super-hyps.

Hyp h1 should not be considered to be a refinement of h2 if it does too much overriding of the parametric description of h2. It should be adding detail, not rejecting the description given by the super. Hyps that refine other hyps never substantially overrule a parent's parametric description but only add detail or precision or minor correction. If this is followed, then a sub is a sub is a sub, independently of parameterization.

A hyptype includes a list of param (parameter) types. A hypothesis includes a list of corresponding params. A trigger or expectation includes a list of param initializers. The param type is in charge of creating params, given param initializers. It also checks matches between params. Initializers can be NULL, indicating that a default initialization should be performed.

Supertypes initialize their subtypes with the same parameters. Any additional params in a subtype can either be defaulted or specified by an initializer in the refinement process.

Matches between params are checked to see if two hyps match (to see if an expectation is satisfied); we need to see that they are of the same hyp type, with the same params, at the same time, etc. The param-type is in charge of checking for sameness of params (similar to the way that there are separate checks for equality among integers vs. among strings). There could be a default tolerance on what counts as close enough to support conflating hypotheses that is associated with the param-type, and the hyp type could control it if it cared to.

One challenge is to design the system so that constraints from a narrowed "diagnostic differential" (that is, a presumably exhaustive set of alternative explanations for something) can be used to help resolve the differential. That is, suppose we have scored alternative explanations for something, have ruled out many, and are left with only a few good alternatives, but they are too close in confidence scores to distinguish (discriminate). It would be good to be able to use the information that there are only few alternatives to constrain lower-level conclusions, which would possibly then rebound upwards with enough information to do the needed discrimination. For example (we will use handwriting recognition for simplicity, but the issue is general), suppose we can recognize enough of a word to narrow the possibilities to just "news" and "views" both of which could fit equally well into the context. So we would like to take another look at the scribble at the beginning of the word, USING THE INFORMATION THAT IT

IS PROBABLY AN N OR A VI, and see if we can distinguish which. To do so we look to the expected features of n and vi and discover that, e.g., the vi has a dot, but the n does not, so we look for the dot.

Some of this kind of inferencing may be captured in our design, but it appears that some is not. There should be some way (in this example) for the word level to inform the letter level that the possibilities have been narrowed to n and vi, and then for the letter level to use that information to interrogate the strokemarks level with a focused question. Quite likely some of the required functionality will emerge as a byproduct of what we are already building in. The remainder might not be all that hard to add once we have enough machinery in place, and in any case it might not be useful anytime soon for the current project. Almost certainly we will want it eventually; it should be very powerful at squeezing more conclusion out of less data.

So far several of the places we were thinking we would need parametric refinement are turning out to be expressed more naturally as climbing up levels of interpretation. For example, rather than estimating syllable strength by measuring acoustic energy in a certain way in a certain frequency band and then re-estimating it more accurately later, it appears more natural to treat the first measurement as estimating "Source Signal Magnitude" (a kind of oompf of articulation) which is explained partly by syllable strength. So there may be less need for parametric refinement that we previously thought.

Stuck abducers, control of refinement, and hyp retraction

Stuckness criterion: when nothing more can be done by the abducer at the given confidence level. Under these circumstances nothing is going to change unless other levels help, or more data comes, or the acceptance threshold is lowered. There is a tradeoff between lowering confidence, and just waiting for more data.

Recall the idea that unrefined hyp-type supers of an expectation should weakly contribute to the scoring of any hyp that has the expectation. Thus, possibly, the leading potential explainers for an ambiguous finding have their scores based on such unrefined expectations. If so, then the abducer is not stuck until it has tried to resolve the ambiguity by asking for refinements for these critical expectations.

If an abducer gets a request to refine an accepted hypothesis, it must suspend the explanatory coverage for that hyp, generate subs, and add them to the pool of potential explainers at that agora. Hopefully a best explanation for the newly uncovered findings can be accepted by the abducer, and hopefully it will be from among the new subs. This can fail to happen, however, if the subs score poorly or will not explain enough. If all subs are ruled out, then the then the parent must be retracted. Also, if some other hyp can be accepted instead, making the subs explanatorily superfluous, then the parent must be retracted. If there are still viable subs but no sub can be accepted, and if nothing else can be accepted either to explain what the parent had been explaining, then the parent can be reestablished as the explainer of what it had been explaining - refinement has failed for now.

Thus an abducer is not stuck until it has tried to resolve ambiguities by provoking any significant expectation refinements.

Besides provoking refinements of significant expectations, must the abducer also attempt refinements on its own before it is officially stuck? That is, suppose f is ambiguous because H-1 and h-2 offer to explain it but are close in score. The abducer requests refinements for any unrefined expectations upon which the scores of H-1 and H-2 are based. Should the abducer also try to refine H-1 and H-2, in hopes that will clarify the potential explanation of f? In some sense the abduce is not really stuck if this has not been tried. Nevertheless, we propose that the default should be that it declares itself stuck without trying this move. Why? Because it might not be needed. It might be OK for f to be left ambiguous. If some abducer, at some agora, cares about H-1 or H-2 it can say so (e.g., by having H-1 or H-2 as an interesting expectation, or one of their subs as an interesting expectation). If nothing declares any interest in H-1 or H-2, directly or indirectly, then there seems to be insufficient reason to try to resolve the ambiguity of f.

Continuing hypothesis assembly

A perceptual abducer might not always be active, but it is never done; it may have finished interpreting all of the data before it, but it must be prepared to act on new data even if it has not yet satisfactorily resolved the old data.

Continuing abductive processes, (e.g. speech recognition), differ from static abductive processes (e.g. medical diagnosis) in that they continually receive findings to explain. There is no point at which a continuing abducer can consider its findings complete and determine the hypotheses that best explain them. Especially if we ever hope to build an abduction system for real-time speech recognition, we must be concerned with how the abducer will keep up with the input data, proceeding forward through time.

LLU must perform continuing abductive assembly of a composite hypothesis that changes over time. For example we need to form a composite hypothesis at the word level that is consistent, etc. But we get to flush the buffer, at least the older parts, from time to time. The hypotheses accepted at each level flow into a buffer to oblivion. Presumably this goes on at all levels of speech and vision - each level has something like a FIFO buffer of conclusions that are data for higher levels. Conclusions form mostly at the recent end, and expire at the other. Presumably the higher-level buffers hold data for covering longer times (but not necessarily more data).

Automatic expiration for hypothesis that are considered but never accepted or rejected is not the same as rejecting them, but more like garbage collection. Potential explainers should not expire before the expiration of what they are offering to explain.

It should be possible to implement a procedure with knowledge content something like: a hypothesis is presumed false if it cannot be established as true within such-and-so an amount of time, or before such-and-so an event. If we do something like this, it ought to be under the control of the hyp entities themselves; that is, at instantiation they need to hand this information to the abducer so it can handle the hyp rejections based on events or the clock. The hyps should initiate, since there is where the knowledge should be, if there is any. It may not be worth the processing complexity, however. We will try to do without, and see if something like this is really needed.

To the degree that we need to do it, we can isolate episodes of problem solving. For example we can isolate the processing of a single "utterance." Some forms of learning will take place in the aftermath, and "off line" so to speak, e.g. some of the parametric adjustments that take place when the system gets confirmation that it was right. This learning depends on keeping around a lot of information about the processing that took place during that episode of problem solving. This is still consistent with the idea that we can build the underlying perceptual abduction machine so that it could in principle operate continuously if it did not have any learning to do.

The kind of learning that occurs when the system wants the user to confirm that it needs a new hyp type may occur in the thick of the problem solving.

Standards for acceptance of hyps. - The standard for hypothesis acceptance (for now based on a threshold for the maximum sum of scores of other hyps) gets relaxed during processing when abducers get stuck. The standard is a global property, established for all abducers. It gets reset as soon as new data come in from outside the system (typically new data to be accounted for at the "front end," but also perhaps new input from the user or external problem solvers.)

How far to raise the standards? should it go all the way back to the highest standards or drift upwards by degrees? For now we adopt the strategy of going back to the highest standards, because that will potentially produce the most secure and confident next steps. There may be some waste of processing time, since it is likely that the highest standards cannot be achieved on the new data either, since ambiguity is likely to persist. On the other hand, there very well may come an unambiguous datum, and we might as well treat it as such.

Retracting acceptance and rejection decisions

The system must be prepared for (occasional) retraction of hypothesis acceptance based on:

1 - a hard implication of an accepted hypothesis not coming about (and there being no compensating explanation for why not);

2 - incompatible high confidence best explanations.

This non-monotonicity is of a very specialized sort, and a general purpose non-monotonic logic mechanism does not seem to be called for. Retracting acceptance per se is not hard, what may be hard is retracting all the subsequent decisions based on the acceptance. But it does not have to be done perfectly.

Retracting acceptance (and maybe rejection) decisions may be fairly common. Apparently we humans commonly engage in bits of hypothetical reasoning in the midst of abductions and the simplest way to do that is to believe tentatively, i.e., be prepared for retraction. The phenomenon appears to be universal. I think it's raining because of the light noise outside, and take my umbrella, but it turns out that the noise was due to a lawn sprinkler, so I go put the umbrella back.

Of course "correct" "careful" retraction of acceptance, after long chains of subsequent reasoning has occurred, will probably be computationally too expensive. It seems that humans do not commonly do it; and it appears not to be needed for practical abductive reasoning. Either the reasoning chains must be short, or the "clean up" after the retraction will be incomplete but good enough for practical purposes. If a hypothesis is accepted, and subsequently overthrown by the weight of distributed evidence teaming up against it, then simple retraction with little cleanup will probably be good enough, since the distributed evidence will assert itself in enough places to keep everything reasonably honest on the whole. What this means for our abduction machine is that we can probably do well enough with simple retraction plus cleanup, where the cleanup algorithm is something that never has pretensions of perfection.

A hypothesis keeps track of the findings it can explain, so in the case of retracted acceptance, the abducer can simply reset those findings back to unexplained and reconsider other hyps offering to explain them. It should also tell higher levels not to bother explaining the accepted hyp. It will also be useful to locate hypotheses whose scores were impacted as a result of the acceptance, and rescore them. Retracted rejection entails reevaluating the findings the hyp can explain and perhaps retracting acceptance of other hyps that thought they explained those findings.

Hypothesis are not rescored while they have a belief state of accepted or rejected (or guessed or guessed-not). Retractions are initiated by events related to the hyp by the hard relationships (incompatibility and implications). For example if a hyp is accepted and incompatible one is accepted too, then this is noticed and handled appropriately.

Recovering from mistakes

Abductive inferences are fallible. The mechanism described here can minimize the occurrence of certain mistakes and can recover from certain errors that do occur. We discuss two types of errors: mistakes in initial hypothesization and scoring, and mistakes in choice of initial islands of confidence.

Mistakes in initial hypothesization and scoring.

1. Hypothesis triggers come from above and laterally as well as below, thus hypotheses that would be missed on bottom-up processing can still be considered.

2. Hypothesis evaluation is augmented by encouragement and discouragement from positive and negative associations with other hypotheses in the same agora. This occurs as part of the hypothesis-composition processing. Thus the initial local-

occurs as part of the hypothesis-composition processing. Thus the initial local-match confidence score is improved by contextual information as this information develops.

3. Hypothesis evaluation is also augmented by encouragement and discouragement based on expectations from accepted higher level hypotheses. This constitutes another kind of context-based improvement and check on the confidence score.

4. A hypothesis is accepted according to how well it surpasses explanatory alternatives. Thus, after recognition-based scoring, a significant additional uncertainty-reducing operation (based on explanatory relationships) is performed before acceptance.

5. Strength of confidence is supported by "the consilience of inductions," whereby converging lines of inference all support the same hypothesis. Thus hypothesis scoring should not be excessively sensitive to single factors, and overall system performance should be robust.

6. Acceptance, when it finally occurs, is still tentative and liable to be overthrown by relationships to the mass of other confident hypotheses or because of an unresolved anomaly.

Mistakes in choice of initial islands of confidence.

1. Actually the islands are strong. They are never based only on a hypothesis having high initial confidence; a hypothesis must also be a best explanation for some datum.

2. Inconsistencies lead to detected anomalies, which lead to special strategies that weigh alternative courses of action. Originally accepted hypotheses can collide with others and subsequently be called into question.

3. Inconsistency collisions can occur laterally or from above or below. In effect there is broad cross-checking of accepted hypotheses.

4. An inexplicable datum can be doubted and called into question. If after reevaluation (provoked refinement) the datum remains strong despite the doubt, then the system can detect that it has encountered the limits of its knowledge, and it is positioned to learn a new hypothesis category.

5. Sometimes two parts of a compound hypothesis are inconsistent in context; that is, a consistent hypothesis cannot be formed at the next highest level. Under these circumstances special handling takes over, similar to that for other kinds of detected inconsistencies.

There are three main points at which learning is invoked: to develop a new hypothesis type, to positively reinforce a correct answer, and to adjust appropriately for an incorrect answer.

Learning new hypothesis types and accounts-for knowledge

Learning new hypothesis types uses a form of supervised learning where the human user confirms the machine's hypothesis that a new category applies to the situation. The machine determines that something is very likely to be NEW (a generic hypothesis type) because that is the best explanation in the context; then the system captures a description of the new item as best as can be done from the information of the occasion. To avoid spurious learning and the associated need for recovery from learning mistakes, we begin by supposing that the human user will confirm that a NEW hypothesis is needed before the learning takes place. Once this is confirmed, the machine then watches for the same thing to happen again, and improves its description of the category.

It is in the context of continual abduction that we need to understand how new things are learned. Information about NEW categories must be captured before it is lost to the relevant agoras.

When the user confirms that a new concept is needed to account for some findings, it must be indicated that a single concept will suffice to account for the unexplained findings. Alternatively the user must guide the system to properly partition the unexplained findings among several new categories. The new categories or categories will set their can-account-for knowledge to reflect the unexplained material in the initiating episode. The coverage of this can-account-for knowledge may be extended in subsequent episodes as user feedback confirms that the concept needs to be extended to account for new material.

A new hypothesis type can be generated by simply assuming that it explains all the unexplained data, and that it expects to see exactly the same situation whenever it is the correct hypothesis. Thus the positive prototype findings for the new hypothesis type are the unexplained findings, together with other "unusual characteristic" that obtain "in the vicinity," which may include accepted hypotheses in the same or adjacent agoras. Initially, negative prototypes are rejected hypotheses "in the vicinity," and the finding relationships are those that hold among the unexplained findings. This implies that there is some list of possible relationships that the new hypothesis generator checks for each pair of findings. This may be specific to each agora, so that, for example, only before/after relationships are considered for findings from certain channels, while before/after relationships parameterized on time are considered for findings from another channel.

Recall that processing for elementary hypotheses occurs in two stages: stimulus, and instantiation, which typically begin with attempted cheap rule out and results in determination of explanatory coverage and setting of initial confidence. Learning an elementary hypothesis requires noticing that there is something unexplained, being given confirmation by the supervisor that a new hypothesis-former is to be learned (usually a symbolic name will be given then too), and acquiring from the instigating case (perhaps with help) some initial knowledge for each phase of generation. After that, the knowledge for each phase should be improved on the basis of information about successes and failures, turning to the supervisor in case of significant difficulty.

Learning triggers

Features for triggering are adapted independently of features used for rule-out, scoring, and explanatory coverage determination.

Evocation is all based on positive evidence, that is, a set of cues needs to be learned for a given hypothesis type. The job of learning cues for hyptypes is that of compiling enough cues to virtually guarantee that a correct hypothesis will always be among those cued, while minimizing overcueing to minimize extra work from having to get rid of the wrong ones.

Triggers are determined initially by a relaxation of the descriptions of features that occur in the initial example. Relaxation is by expanding range, generalizing over the type hierarchy, and weighting all features together into a big disjunction for triggering. The intention is to initially relax enough to produce overgeneralization, and then to gradually tightened on the basis of subsequent experience to minimize overcuing, but without undercuing. This can be done gradually and separately for each dimension of generalization. The basic strategy for learning triggers is one of overgeneralization, followed by gradual extinction of the over-general response and tightening towards what amounts to the convex hull of the observed examples (that is, the tightening goes no farther than a known case in any dimension.). The goal is to never (or almost never) fail to trigger formation of a hyp of the hype type if such a hyp is true, while minimizing the amount of over-triggering.

We set up a range by generalization even before we see negative examples. If we have max and min values for a parameter as a whole we can just let the confidence trapezoid hit 0 height at those values. (In general the confidence trapezoid is such that the sides taper from highest to lowest confidence as the parameter goes from known positive to known negative values).

The hyp type hierarchies are used to assist in learning triggers. The features are generalized up their type hierarchies (e.g., one step), that is, for example, the new syllable 'guj' occurred right after the known syllable 'mor' (the words was "mortgage") so not only will the occurrence of the syllable 'mor' tend to trigger the new one, but also so will the super or supers of 'more' in the syllable class hierarchy, e.g., maybe the type 'nasalized_onset-rhoticized _coda.' Later this triggering feature, if it is over-general, can get cut back to the more specific class, because the over-general category is never used significantly.

Triggers are set over some "sphere" of potential relevance whose diameter extends to the layer below (the accounted-for), the layer above, and into the past and future on these and the present layer.

Learning expectations

There is a need to distinguish negative evidence from absence of evidence. Predictions (expectations) IF CHECKED OUT and fail, damage confidence proportionate to the strength of the prediction. The strength of a prediction is based on frequency count.

Predictions are learned in two stages: 1 - conjecture relationship, 2 - keep track of strength (low strength, i.e. unreliable predictions die off). Conjecturing includes features the concept can account for and other features on the same and adjacent levels that are noticed to co-occur, starting with the first example.

Predictions cannot be checked out if they predict events too far in the future. Other cases predict into observable range. 3 cases: can check immediately, range is noisy (forget it), range is ambiguous (wait to see if the ambiguity reduces from further processing).

When the abduction correctly matches a hypothesis to the data, it reinforces that hypothesis type. All prototypes that correctly match the data are rated to be more valuable in rating the hypothesis. Prototypes that do not match the data are either adjusted parametrically to match the data, or else rated to be less valuable in scoring the hypothesis. A similar process is used to strengthen and weaken the importance of finding relationships as appropriate. (Finding relationships have not been implemented at the time of this report).

When the abduction incorrectly matches a hypothesis to the data (incorrectly chooses it as the best explanation), it must adjust both the hypothesis that incorrectly matched and the hypothesis that incorrectly failed to match. The incorrectly matching hypothesis is modified by reducing the importance of prototypes and relationships that match, by introducing new negative prototypes, and by strengthening the importance of negative prototypes that match. The incorrectly non-matching hypothesis is modified by reinforcing prototypes as described above, and by weakening the importance of negative prototypes that match.

Since different kinds of prototypes match the data in different ways (e.g. simple event or event with parameters), different methods are required to adjust prototypes to match data. For example, a prototype that expects a specific value can be adjusted to accept a range of values as the hypothesis encounters positive examples with different values for that parameter. A smart hypothesis could also notice when positive prototypes overlap negative prototypes and discard both or reduce the acceptable range of values.

A specific system must decide how to strengthen and weaken the importance of hypothesis prototypes. One way to do this is to maintain statistics recording how often a positive example correctly matches and how often it incorrectly matches. This is the method we have chosen to implement initially.

Each hypothesis type is characterized by a set of positive and negative prototype findings, and relationships among the positive prototype findings. For example, the set of positive prototype findings could be { "stop", "voicing on", "voicing off" }, positive prototype findings could be { "stop", "voicing on", "voicing off" }, the set of negative prototype findings could be { "lip closure" }, and the set of relationships could be { ' "stop" is before "voicing off" ', ' "voicing on" is 10 or 20 time units before "voicing off" ' }. The positive prototype findings and prototype relationships tend to confirm the suspicion that a hypothesis explains some set of data, while the negative prototype findings tend to rule out that hypothesis as a possible explainer.

Since different kinds of prototypes match the data in different ways (e.g. simple event or event with parameters), different methods are required to adjust prototypes to match data. For example, a prototype that expects a specific value can be adjusted to accept a range of values as the hypothesis encounters positive examples with different values for that parameter. A smart hypothesis could also notice when positive prototypes overlap negative prototypes and reduce the match confidence for occurrences in the range of the overlap. If the overlap is sever, as for example when negative values occur in the midst of the positive range, then stranger strategies of adjustment are called for, e.g., dividing the hyptype into two subtypes, one corresponding to each range of positive examples.

Acknowledgments

This work was supported mostly by ATR Interpreting Telecommunications Research Laboratories.

Bibliography

Charniak, E. (1986). A Neat Theory of Marker Passing. In *Proceedings of AAAI-86*, 1 (pp. 584-588). Los Altos: Morgan Kaufmann.

Dasigi, V. (1991). Abductive Modeling in Sub-Domains of Language Processing. In *Notes from the AAAI Workshop on Domain-Independent Strategies for Abduction*, (pp. 25-31). Anaheim. (Josephson & Dasigi, 1991).

Erman, L. D., & Lesser, V. R. (1980). The Hearsay-II Speech Understanding System: A Tutorial. In W. Lea (Ed.), *Trends in Speech Recognition*. Englewood Cliffs, NJ: Prentice Hall. Reprinted in Readings in Speech Recognition, Ed. Alex Waibel and Kai-Fu Lee, Morgan Kaufmann, 1990.

Fujimura, O. (1994). Syllable Timing Computation in the C/D Model. In *Proceedings of the 3rd International Conference on Spoken Language Processing*, Yokohama Japan. forthcoming.

Hobbs, J. R., Stickel, M., Appelt, D., & Martin, P. (1993). Interpretation as Abduction. *Artificial Intelligence Journal*, 63(1-2), 69-142.

Josephson, J. R., & Josephson, S. G. (Eds.). (1994). *Abductive Inference: Computation, Philosophy, Technology*. New York: Cambridge University press.

Josephson, J. R., Smetters, D., Fox, R., Oblinger, D., Welch, A., & Northrup, G. (1989). *The Integrated Generic Task Toolset, Fafner Release 1.0.* Technical Report, The Ohio State University, Laboratory for Artificial Intelligence Research.

Tanner, M. C., Fox, R., Josephson, J. R., & Goel, A. K. (1991). On a Strategy for Abductive Assembly. In *Notes from the AAAI Workshop on Domain-Independent Strategies for Abduction,* (pp. 80-83). Anaheim. (Josephson & Dasigi, 1991).

Appendix A: Specifying Hyptypes

Hyptype specifications are put in a single file with extension ".ht" . Following is a specification for those specifications.

```
<.ht file entry> ::=
 <hyptype name> [ (<param defs>) ]
 <agora name>
 <total considered>, <total true>, <total false>
 <triggers, etc., in any order (order only matters for inheritance>
 -


<param def> ::=
 <intparam def>
 -- others can be added as the need is determined

<intparam def> ::=
 d:<default initial value (an integer)>

<refined from> ::=
 r:<super hyptype>
<trigger> ::=
 t:<triggered hyptype> [ (<derived params>) ], <temporal offset>
<expectation> ::=
 e:<expected hyptype> [ (<derived params>) ], <temporal offset>,
   h&e, ~h&e, h&~e, ~h&~e
<implication> ::=
 i:<implied hyptype> [ (<derived params>) ], <temporal offset>,
   <expectation decision (y or n)>,
   <resulting hypothesis decision (y or n)>
<accounts for> ::=
 a:<accounts for hyptype> [ (<derived params>) ], <temporal offset>

<derived param> ::=
 d:<intparam init>
 _  (use default value)

<intparaminit> ::=
 <use this param # in the original> /
 <add this offset to it to yield derived param>

<temporal offset> ::=
 '[' '[' <min-start> <max-start> ']' '[' <min-end> <max-end> ']' ']'
```

Appendix B: Test Case

Following is an example of a run of LLU. At the time of this run several features are not yet implemented. There are probably bugs remaining also.

Three agora are defined: "feature," "syllable," which takes it data from feature, and "word," which takes its data from syllable. Here is the contents of the file setting up the agora and their relationships:

```
feature
syllable:feature
word:syllable
*
```

Here is a portion of the contents of the file setting up the hypothesis types.

```
fricative
feature
100,50,50
t:dh_ae_ts,[[0/0.0 1/0.0] [2/100.0 3/650.0]]
t:dh_ae_ts,[[0/-650.0 1/-300.0] [2/0.0 3/0.0]]
t:fUl,[[0/0.0 1/0.0] [2/60.0 3/300.0]]
-
spirantized
feature
100,50,50
-
glide
feature
100,50,50
t:wUn,[[0/0.0 1/0.0] [2/0.0 3/300.0]]
-
fUl
syllable
100,50,50
e:fricative,[[0/0.0 1/0.0] [0/150.0 1/210.0]],10,5,2,2
e:labial,[[0/0.0 1/0.0] [0/150.0 1/210.0]],10,5,2,2
e:lateral,[[2/-200.0 3/-50.0] [2/0.0 3/0.0]],10,5,2,2
a:fricative,[[0/0.0 1/0.0] [0/150.0 1/210.0]]
a:labial,[[0/0.0 1/0.0] [0/150.0 1/210.0]]
a:lateral,[[2/-200.0 3/-50.0] [2/0.0 3/0.0]]
t:wonderful,[[0/-600.0 1/-400.0] [2/0.0 3/0.0]]
-
wonderful
word
100,50,50
e:wUn,[[0/0.0 1/0.0] [0/160.0 1/300.0]],10,5,2,2
e:dR,[[0/200.0 1/400.0] [2/-10.0 3/-30.0]],10,5,2,2
e:fUl,[[2/-300.0 3/-200.0] [2/0.0 3/0.0]],10,5,2,2
a:wUn,[[0/0.0 1/0.0] [0/160.0 1/300.0]]
a:dR,[[0/200.0 1/400.0] [2/-10.0 3/-30.0]]
a:fUl,[[2/-300.0 3/-200.0] [2/0.0 3/0.0]]
```

Here is the contents of the file setting up the system input:

```
voiced_cons,[[20 30] [70 90]]
interdental,[[20 30] [70 90]]
fricative,[[20 30] [70 90]]
low,[[90 100] [240 250]]
stop,[[250 260] [300 310]]
apical,[[250 260] [300 310]]
fricative,[[305 315] [440 455]]
apical,[[305 315] [440 455]]
rhoticized,[[470 490] [530 540]]
high,[[510 520] [560 570]]
front,[[510 520] [560 570]]
reduced,[[560 570][625 635]]
lateral,[[630 640] [705 715]]
high,[[700 720] [840 850]]
front,[[700 720] [840 850]]
glide,[[850 860] [940 950]]
labiovelar,[[850 860] [940 950]]
reduced,[[940 960] [990 1010]]
nasal,[[1000 1015] [1100 1110]]
apical,[[1000 1015] [1100 1105]]
stop,[[1110 1115] [1130 1140]]
apical,[[1110 1115] [1130 1140]]
voiced_cons,[[1110 1115] [1130 1140]]
rhoticized,[[1130 1150] [1220 1230]]
fricative,[[1220 1230] [1340 1350]]
labial,[[1220 1230] [1340 1350]]
lateral,[[1370 1380] [1490 1500]]
```

Here is an abbreviated version of the trace of the run. Obviously too much processing is being done for an easy problem. Much work remains to be done on the system

```
1. Run
2. Read abducer-agora file
3. Read hyp-type file
4. Write hyp-type file
5. Toggle learn mode
0. Quit
> 1

Filename [llu-2.in] >
----------
<AbducerMgr>:
     resetting acceptance constraint
<Hyp #11739670 voiced_cons @[[20 30] [70 90]] *?* $[0 0 0]>:
     started
*******************************
<Hyp #11739670 voiced_cons @[[20 30] [70 90]] *y* $[1 1 1]>:
     accepted

[some output deleted...]

<Hyp #11750088 fricative @[[20 30] [70 90]] *y* $[1 1 1]>:
     accepted
<Hyp #11740520 dh_ae_ts @[[-630 -270] [70 90]] *?* $[0.8 0.9 1]>:
     started
<Hyp #11745366 fUl @[[20 30] [130 390]] *?* $[0.8 0.9 1]>:
     started
<Hyp #11750088 fricative @[[20 30] [70 90]] *y* $[1 1 1]>:
     added expector:
     <Hyp #11740520 dh_ae_ts @[[-630 -270] [70 90]] *?* $[0.8 0.9 1]>
<Hyp #11740520 dh_ae_ts @[[-630 -270] [70 90]] *?* $[0.8 0.9 1]>:
     expectation updated:
     <Hyp #11750088 fricative @[[20 30] [70 90]] *y* $[1 1 1]>

[some output deleted...]

<Hyp #11754102 voiced_cons @[[-630 -270] [70 190]] *?* $[0 0 0]>:
     added expector:
     <Hyp #11740520 dh_ae_ts @[[-630 -270] [70 90]] *?* $[0.8 0.9 1]>
<Hyp #11754422 fricative @[[-630 -270] [70 190]] *?* $[0 0 0]>:
     added expector:
     <Hyp #11740520 dh_ae_ts @[[-630 -270] [70 90]] *?* $[0.8 0.9 1]>
<Hyp #11754742 interdental @[[-630 -270] [70 190]] *?* $[0 0 0]>:
     added expector:
     <Hyp #11740520 dh_ae_ts @[[-630 -270] [70 90]] *?* $[0.8 0.9 1]>
<Hyp #11755062 low @[[-580 -150] [-230 90]] *?* $[0 0 0]>:
     added expector:
     <Hyp #11740520 dh_ae_ts @[[-630 -270] [70 90]] *?* $[0.8 0.9 1]>

[much output deleted...]

<Hyp #11747952 low @[[70 150] [20 680]] *y* $[1 1 1]>:
     accepted
<Hyp #11741638 dh_ae_ts @[[20 30] [320 680]] *?* $[0.8 0.9 0.9]>:
     expectation updated:
     <Hyp #11747952 low @[[70 150] [20 680]] *y* $[1 1 1]>
<Abducer feature>:
     refining:
      <Hyp #11747952 low @[[70 150] [20 680]] *y* $[1 1 1]>
<Abducer feature>:
      refining:
      <Hyp #11750088 fricative @[[20 30] [70 90]] *y* $[1 1 1]>
<Abducer syllable>:
      updating:
      <Hyp #11741638 dh_ae_ts @[[20 30] [320 680]] *?* $[0.8 0.9 0.9]>

[much output deleted...]

<Hyp #11809662 really @[[460 500] [850 1090]] *y* $[0.6 0.8 1]>:
      accepted
<AbducerMgr>:
      status finished for abducer:
      <Abducer word>
<AbducerMgr>:
      1 active, 0 stuck, 2 finished
<Hyp #11809662 really @[[460 500] [850 1090]] *y* $[0.7 0.9 1]>:
      expectation updated:
      <Hyp #11802384 ri @[[470 490] [530 680]] *y* $[0.8 0.9 1]>
<Hyp #11805978 lone_vowel_syll @[[530 610] [570 700]] *?* $[0.8 0.9 1]>:
      started

[much output deleted...]

-*-*-*-*-*-*-*-*-*-
<Abducer feature>:
      report

Data to be explained:
none

Accepted hypotheses:
1 <Hyp #11739670 voiced_cons @[[20 30] [70 90]] *y* $[1 1 1]>
2 <Hyp #11741108 interdental @[[20 30] [70 90]] *y* $[1 1 1]>
3 <Hyp #11750088 fricative @[[20 30] [70 90]] *y* $[1 1 1]>
4 <Hyp #11747952 low @[[70 150] [20 680]] *y* $[1 1 1]>
5 <Hyp #11763278 stop @[[250 260] [300 310]] *y* $[1 1 1]>
6 <Hyp #11756588 apical @[[250 260] [300 310]] *y* $[1 1 1]>
7 <Hyp #11748942 fricative @[[120 670] [320 680]] *y* $[1 1 1]>
8 <Hyp #11770824 apical @[[305 315] [440 455]] *y* $[1 1 1]>
9 <Hyp #11783218 rhoticized @[[470 490] [530 540]] *y* $[1 1 1]>
10 <Hyp #11807854 high @[[480 670] [530 680]] *y* $[1 1 1]>
11 <Hyp #11808184 front @[[480 670] [530 680]] *y* $[1 1 1]>
12 <Hyp #11817514 reduced @[[560 570] [625 635]] *y* $[1 1 1]>
13 <Hyp #11828216 lateral @[[630 640] [705 715]] *y* $[1 1 1]>
14 <Hyp #11833616 high @[[270 1090] [840 1100]] *y* $[1 1 1]>
15 <Hyp #11834064 front @[[270 1090] [840 1100]] *y* $[1 1 1]>
16 <Hyp #11851688 glide @[[850 860] [940 950]] *y* $[1 1 1]>
17 <Hyp #11855444 labiovelar @[[850 860] [950 1060]] *y* $[1 1 1]>
18 <Hyp #11870120 reduced @[[940 960] [990 1010]] *y* $[1 1 1]>
19 <Hyp #11868806 nasal @[[1000 1015] [1100 1110]] *y* $[1 1 1]>
20 <Hyp #11879092 apical @[[1000 1015] [1100 1105]] *y* $[1 1 1]>
21 <Hyp #11875936 stop @[[1110 1115] [1130 1140]] *y* $[1 1 1]>
22 <Hyp #11888750 apical @[[1110 1115] [1130 1140]] *y* $[1 1 1]>
23 <Hyp #11896976 voiced_cons @[[1110 1115] [1130 1140]] *y* $[1 1 1]>
24 <Hyp #11911658 rhoticized @[[1130 1150] [1220 1230]] *y* $[1 1 1]>
25 <Hyp #11912586 fricative @[[1220 1230] [1340 1350]] *y* $[1 1 1]>
26 <Hyp #11937266 labial @[[1220 1230] [1340 1350]] *y* $[1 1 1]>
27 <Hyp #11935140 lateral @[[1200 1600] [1400 1650]] *y* $[1 1 1]>

Other possible hypotheses:
none
```

```
-*-*-*-*-*-*-*-*-*-
<Abducer syllable>:
     report

Data to be explained:
1 <Finding <Hyp #11739670 voiced_cons @[[20 30] [70 90]] *y* $[1 1 1]> stuck>
2 <Finding <Hyp #11741108 interdental @[[20 30] [70 90]] *y* $[1 1 1]>
     unexplained>
3 <Finding <Hyp #11750088 fricative @[[20 30] [70 90]] *y* $[1 1 1]>
     unexplained>
4 <Finding <Hyp #11747952 low @[[70 150] [20 680]] *y* $[1 1 1]>
     explained>
5 <Finding <Hyp #11763278 stop @[[250 260] [300 310]] *y* $[1 1 1]>
     explained>
6 <Finding <Hyp #11756588 apical @[[250 260] [300 310]] *y* $[1 1 1]>
     explained>
7 <Finding <Hyp #11748942 fricative @[[120 670] [320 680]] *y* $[1 1 1]> stuck>
8 <Finding <Hyp #11770824 apical @[[305 315] [440 455]] *y* $[1 1 1]> stuck>
9 <Finding <Hyp #11783218 rhoticized @[[470 490] [530 540]] *y* $[1 1 1]>
     unexplained>
10 <Finding <Hyp #11807854 high @[[480 670] [530 680]] *y* $[1 1 1]>
     explained>
11 <Finding <Hyp #11808184 front @[[480 670] [530 680]] *y* $[1 1 1]>
     explained>
12 <Finding <Hyp #11817514 reduced @[[560 570] [625 635]] *y* $[1 1 1]>
     explained>
13 <Finding <Hyp #11828216 lateral @[[630 640] [705 715]] *y* $[1 1 1]>
     explained>
14 <Finding <Hyp #11833616 high @[[270 1090] [840 1100]] *y* $[1 1 1]>
     explained>
15 <Finding <Hyp #11834064 front @[[270 1090] [840 1100]] *y* $[1 1 1]>
     explained>
16 <Finding <Hyp #11851688 glide @[[850 860] [940 950]] *y* $[1 1 1]>
     explained>
17 <Finding <Hyp #11855444 labiovelar @[[850 860] [950 1060]] *y* $[1 1 1]>
     explained>
18 <Finding <Hyp #11870120 reduced @[[940 960] [990 1010]] *y* $[1 1 1]>
     unexplained>
19 <Finding <Hyp #11868806 nasal @[[1000 1015] [1100 1110]] *y* $[1 1 1]>
     unexplained>
20 <Finding <Hyp #11879092 apical @[[1000 1015] [1100 1105]] *y* $[1 1 1]>
     unexplained>
21 <Finding <Hyp #11875936 stop @[[1110 1115] [1130 1140]] *y* $[1 1 1]>
     explained>
22 <Finding <Hyp #11888750 apical @[[1110 1115] [1130 1140]] *y* $[1 1 1]>
     explained>
23 <Finding <Hyp #11896976 voiced_cons @[[1110 1115] [1130 1140]] *y* $[1 1 1]>
     explained>
24 <Finding <Hyp #11911658 rhoticized @[[1130 1150] [1220 1230]] *y* $[1 1 1]>
     explained>
25 <Finding <Hyp #11912586 fricative @[[1220 1230] [1340 1350]] *y* $[1 1 1]>
     unexplained>
26 <Finding <Hyp #11937266 labial @[[1220 1230] [1340 1350]] *y* $[1 1 1]>
     unexplained>
27 <Finding <Hyp #11935140 lateral @[[1200 1600] [1400 1650]] *y* $[1 1 1]>
     explained>

Accepted hypotheses:
1 <Hyp #11765018 dh_ae_ts @[[-400 -40] [300 310]] *y* $[0.8 0.9 0.9]>
     explains 4 5 6 6
2 <Hyp #11802384 ri @[[470 490] [530 680]] *y* $[0.9 1 1]>
     explains 10 11
```

```
3 <Hyp #11805978 lone_vowel_syll @[[530 610] [570 700]] *y* $[1 1 1]>
     explains 12
4 <Hyp #11798646 li @[[260 990] [840 1100]] *y* $[0.9 0.9 0.9]>
     explains 13 14 15
5 <Hyp #11848974 wUn @[[850 860] [940 1250]] *y* $[0.9 0.9 0.9]>
     explains 16 17
6 <Hyp #11860580 dR @[[1040 1270] [850 2020]] *y* $[0.9 0.9 0.9]>
     explains 21 22 23 24
7 <Hyp #11917934 fUl @[[1220 1230] [1400 1650]] *y* $[0.8 0.8 0.8]>
     explains 27

Other possible hypotheses:
8 <Hyp #11741638 dh_ae_ts @[[20 30] [320 680]] *?* $[0.8 0.8 0.8]>
     could explain 4 7 8
9 <Hyp #11740520 dh_ae_ts @[[-630 -270] [70 90]] *?* $[0.8 0.8 0.9]>
     could explain 3 4
10 <Hyp #11745366 fUl @[[20 30] [130 390]] *?* $[0.8 0.8 0.9]>
11 <Hyp #11767048 wUn @[[-50 260] [300 310]] *?* $[0.8 0.8 0.9]>
     could explain 6
12 <Hyp #11773388 lone_vowel_syll @[[-50 300] [260 310]] *?* $[0.8 0.8 0.8]>
13 <Hyp #11775394 dh_ae_ts @[[120 670] [420 1330]] *?* $[0.8 0.8 0.8]>
     could explain 7 8 20 22
14 <Hyp #11777918 fUl @[[120 670] [380 980]] *?* $[0.8 0.8 0.8]>
     could explain 7 13
15 <Hyp #11792238 dh_ae_ts @[[-345 15] [440 455]] *?* $[0.8 0.9 0.9]>
     could explain 4 7 8
16 <Hyp #11793534 wUn @[[5 315] [440 455]] *?* $[0.8 0.8 0.9]>
     could explain 8
17 <Hyp #11797202 lone_vowel_syll @[[5 440] [315 455]] *?* $[0.8 0.8 0.8]>
18 <Hyp #11803060 dR @[[390 490] [530 540]] *?* $[0.8 0.8 0.8]>
     could explain 9

-*-*-*-*-*-*-*-*-*-
<Abducer word>:
     report

Data to be explained:
1 <Finding <Hyp #11765018 dh_ae_ts @[[-400 -40] [300 310]] *y* $[0.8 0.9 0.9]>
     explained>
2 <Finding <Hyp #11802384 ri @[[470 490] [530 680]] *y* $[0.9 1 1]>
     explained>
3 <Finding <Hyp #11805978 lone_vowel_syll @[[530 610] [570 700]] *y* $[1 1 1]>
     explained>
4 <Finding <Hyp #11798646 li @[[260 990] [840 1100]] *y* $[0.9 0.9 0.9]>
     explained>
5 <Finding <Hyp #11848974 wUn @[[850 860] [940 1250]] *y* $[0.9 0.9 0.9]>
     explained>
6 <Finding <Hyp #11860580 dR @[[1040 1270] [850 2020]] *y* $[0.9 0.9 0.9]>
     explained>
7 <Finding <Hyp #11917934 fUl @[[1220 1230] [1400 1650]] *y* $[0.8 0.8 0.8]>
     explained>

Accepted hypotheses:
1 <Hyp #11771560 thats @[[-400 -40] [300 310]] *y* $[1 1 1]>
     explains 1
2 <Hyp #11809662 really @[[460 500] [850 1090]] *y* $[1 1 1]>
     explains 2 3 4
3 <Hyp #11848596 wonderful @[[840 870] [860 2050]] *y* $[0.9 1 1]>
     explains 5 6 7

Other possible hypotheses:
4 <Hyp #11955636 wonderful @[[620 830] [1400 1650]] *?* $[0.9 1 1]>
     could explain 6 7
```