Draft of DAMSL: Dialog Act Markup in Several Layers*

James Allen and Mark Core

October 18, 1997

Contents

*The structure of this annotation scheme was developed by the Multiparty Discourse Group at the Discourse Research Initiative (DRI) meeting at the University of Pennsylvania in March, 1996 and a follow up meeting at Schloss Dagstuhl in February 1997. Additional revisions and development of the manual have been greatly aided by suggestions and feedback from Johanna Moore and her research group at the University of Pittsburgh, Rebecca Passonneau, and Teresa Sikorski. The development of this manual was supported in part by NSF grant IRI-95-03312 and IRI-95-28998, the latter of which was under a subcontract from Columbia University. Thanks to Becky for the acronym for DAMSL. While we have tried to maintain the spirit of the decisions made at the Dagstuhl workshop, we have also had to make changes based on problems that have arisen as we used the scheme in practice. Thus it should not be taken as the final word on the DRI scheme, but could be useful as a starting point to reach a general consensus.

This manual describes a system for annotating dialogs. It marks important characteristics of utterances that indicate their role in the dialog and their relationship to each other. This annotation scheme has been defined in order to provide a top-level structure for annotating a range of dialogs for many different purposes. For any particular project, we would expect that the annotation scheme would be refined to provide further detail on phenomena of interest. By agreeing to this standard, however, we would increase the chances that annotations done in one project for one purpose would be reusable later in other projects and for other purposes.

1 Preliminaries

A dialog is a spoken, typed or written interaction in natural language between two or more agents. This scheme was developed primarily for two-agent task-oriented dialogs, in which the participants collaborate to solve some problem. Terms such as speaker and listener will be used in this manual, however, those working with typed dialogs or with dialogs with more than two participants should substitute "currently contributing participant" for "speaker". "listener" refers to the person that the speaker directs their utterance toward or to the person who responds to the speaker's utterance in the case where the speaker does not direct his speech at a single person.

A dialog is divided into units called turns, in which a single speaker has temporary control of the dialog and speaks/writes for some period of time. Within a turn, the speaker may produce several spoken or typed utterance units. While there are many possible ways to define utterances, here we based the notion of utterance on an analysis of the intentions of the speaker (the speech act). For each utterance, the annotation involves making choices along several dimensions, each one describing a different orthogonal aspect of each utterance unit.

In the unannotated dialogs as well as the examples in this manual, the start of utterance units will be preceded by an utterance name such as utt1 or utt2, and the turns will be marked to the left of the utterances (space permitting). The dialog excerpt below shows the formatting; the excerpt contains three utterance units, two spoken by u ("I need a" and "okay") and one by s ("so you've got the engines at Elmira and uh").

```
T1 utt1: u:  I need a
|  utt2:     okay
T2 utt3: s:  so you've got the engines at Elmira and uh
```

In spoken dialogs, when one speaker interrupts the other, speakers will sometimes talk at the same time. Words of overlapping speech are marked with numbered square brackets with the number in parenthesis next to the right bracket. You can match up overlapping sections by finding all the bracketed text with the same index. Consider the two utterance units below. Here, "assuming" and "okay" are marked as overlapping since they are both are bracketed with number 1 brackets.

```
T1 utt1: s: uh would take two hours <sil> [ assuming ](1) you have
|  |         an engine at Bath
T2 utt2: u: [ okay ](1)
```

In the middle of a sentence, a speaker will sometimes pause long enough for a second speaker to interject a few words and then the first speaker will finish their sentence. This is marked by empty square brackets as shown below.

```
T1 utt1: s: so you've got the engines at Elmira and uh [](1) Avon
T2 utt2: u: [Avon](1)
```

Cases where one speaker interrupts the sentence of another will be handled as shown above. Sometimes interruptions such as "mm-hm" or "okay" come between sentences that would be one utterance if not interrupted. In the example below, *s* makes a three utterance answer that is interrupted by an acknowledgment. You are allowed to group a continuous series of utterances together into a segment to accommodate cases where several utterances form one answer, request, etc.

Segments can receive any label that an utterance can. However, a segment must be composed of a continuous set of utterances, so in the example below "okay" and "and we need" must be included in the ans segment. Since the segment is labeled as an answer, the individual utterances of the segment (utt2-utt5) do not also need to be labeled as answers. However, utt3 and utt4 should be given Answer tags (see section 2.4) of "no" since unlike utt2 and utt5 they are not an answer to a question. Except for acknowledgments such as "okay" and "mm-hm" and repaired utterances such as utt4, utterances within a segment should have the characteristics with which the segment is labeled. Section 5 discusses how to tag dialogs using the dat annotation tool and how to form and tag segments with the tool.

```
     T1 utt1: u: where are the engines
ans| T2 utt2: s: there's an engine at Avon
   | T3 utt3  u: okay
   | T4 utt4  s: and we need
   | |  utt5  s: I mean there's another in Corning
```

You will be classifying each utterance unit along several dimensions (using *Utterance-Tags*) in order to record the unit's purpose and role in the dialog. Generally, the dimensions are orthogonal, and you can find examples of any possible combination of labels. We will explicitly point out a few places where this does not seem to hold.

Sometimes you will be unsure of whether or not a label should be applied to an utterance. One option you will have is to apply the label with an "uncertainty modifier". Section 5 discusses the dat annotation tool and how to mark uncertainty using it.

A special dimension called *Communicative-Status* is used to mark utterances that appear to be abandoned in mid-stream or which are impossible to understand. In general, if utterances are marked as being defective in one of these ways, you don't need to complete the rest of the annotation for that utterance.

2 Utterance-Tags

The utterance tags all indicate some particular aspect of the utterance unit itself, summarizing the intentions of the speaker (i.e., why the utterance was spoken) and the content of the utterance. These tags can be classified into four main categories:

- Communicative Status - records whether the utterance is intelligible and whether it was successfully completed.
- Information Level - a characterization of the semantic content of the utterance.
- the Forward Looking Function - how the current utterance constrains the future beliefs and actions of the participants, and affects the discourse.
- the Backward Looking Function - how the current utterance relates to the previous discourse.

Note that utterances do not need to always have a component at each level. For instance, some utterances may have no Forward Looking Function, while others might have no Backward Looking Function.

2.1 Communicative-Status

This utterance tag records certain features of an utterance unit such as whether it was interpretable. Unlike the other categories where a choice needs to be made between options, here the features are independent of each other. Since these features only mark exceptional circumstances, most utterance units have no features marked in this classification. The features marked are

- Uninterpretable
- Abandoned
- Self-talk

Uninterpretable

The utterance unit is not comprehensible. It may or may not be Self-talk depending on how it sounds. These utterances are rare and are usually word fragments although sometimes bad grammar and bad pronunciation/typing can make an utterance impossible to understand. An utterance may also be semantically ill-formed although syntactically perfect.

Abandoned

Sometimes a speaker will make an error in their utterance or change their mind about what they are going to say, and simply abandon their utterance. Such fragments are marked as abandoned only if they provide no content to the dialog, i.e., the import of the dialog would not change if these utterance units were removed. If the utterance does provide content to the dialog, it should not be marked abandoned and should be annotated as usual. In the example below, the abandoned utterance "so I pick up" does not provide content to the dialog and so is labeled abandoned.

```
Abandoned utt1: u: so I pick up ..
          utt2:    can I take oranges um on tankers from Corning
```

Note that just because an utterance is broken off does not mean it is abandoned as defined here. If it contributes content to the dialogue, then it is not marked as abandoned. For instance, consider

```
utt1: s: You'll need two boxcars.
utt2: u: okay, and we-
```

Utt2 is not marked as abandoned as it serves at least to acknowledge utt1, and possibly to accept it. Thus it contributes content to the dialogue.

Self-talk

The utterance unit consists of one speaker talking to him or herself. We assume this does not occur in typed dialogs. This is impossible to detect without listening to the actual speech. Just because an utterance is labeled as Self-talk doesn't mean that the other agent can't hear it, or can't use the information. It simply means that the speaker appears to not be intending to communicate what is being said.

Discussion

While most of the categories are fairly straightforward, more examples and explanation are needed concerning what makes an utterance Abandoned. If a complete clause is uttered then it should not be considered abandoned even if some extension to the clause is broken off. In the dialog excerpt below, "it's four hours" is a clause. *u* never gets to start the new clause introduced by "and" but "it's four hours and" should not be labeled abandoned, since the content of "it's four hours" has been communicated in the dialog. "how long" should be labeled Abandoned because it does not play a part in the dialog.

```
                 utt1: u: mm <click> okay
Reassert         utt2:    four hours from Avon to Bath
Action-directive utt3:    and then I guess attach that to the boxcar to Corning
Reassert         utt4:    it's four hours and
Abandoned        utt5:    how long
Info-request     utt6:    it is two hours from Bath to Corning
```

In this example, "um then go to Corn-" would be labeled abandoned even though it was broken by *s* instead of *u*.

```
Action-directive utt1: u: then take one more boxcar from Bath to Corning
                           <click> fill up with oranges
Abandoned        utt2:    um <sil> then go to [Corn-](1)
Reject(utt1)     utt3: s: [there are no](1) boxcars available with Bath
                           unless you're talking about the ones that are
                           filled with bananas
```

2.2 Information-Level

The Information-Level annotation provides an abstract characterization of the content of the utterance. In task-oriented dialogs, we can roughly divide utterances into those that address the task in some way, those that address the communication process (Communication-management), and those that do not fall neatly into either category (Other-level). In addition, we can subdivide the first category into utterances that advance the task (Task) and those that discuss the problem solving process or experimental scenario (Task-management).

Information-Level

- Task ("Doing the task")
- Task-management ("Talking about the task")
- Communication-management ("Maintaining the communication")
- Other-level

Task

The majority of utterances in most domains involve performing the task that is the reason for the dialog, such as making an airline reservation or scheduling a meeting. Utterances whose content is at the Task level directly move ahead (or attempt to move ahead) the goals of the domain. Each different domain must specify clearly what the task is and characterize the types of activities that occur while doing the task. It is not possible to make the level distinction without this information. As an example, in TRAINS the domain involves developing plans to move trains and cargo from one city to another. Examples of utterances at the Task level include questions about the state of the world (e.g., "Is there a train at Avon?", "When will it arrive?"), suggestions of routes (e.g., "Let's send the engine at Bath to Corning"), utterances discussing the plan (e.g., "That won't work because there's a long delay in Pittsburgh"), and utterances that ask about general properties of the domain such as questions about capabilities (e.g., "How many boxcars can an engine carry", "Can two trains run on the same track at the same time?").

Task Management

Utterances at this level explicitly address the problem solving process and experimental procedure. This includes utterances that involve coordinating the activities of the two speakers (e.g., "Are you keeping track of the time?", "Let's work on getting the train to Avon first", "Forget about that problem for a while"), asking for help on the procedures (e.g., "Do I need to state the problem?") or asking about the status of the process (e.g., "Are we done?"). In domains where the task involves building a plan such as TRAINS, it is important to distinguish between utterances that concern the plan, which are Task level, and utterances that involve the problem solving process, which are Task-management level. For instance, consider a situation in a TRAINS dialogue where a plan has been proposed; a question "will that work?" will be a Task level utterance because evaluating possible plans is part of the task.

Communication-management

Utterances at this level include conventional phrases that maintain contact, perception, and understanding during the communication process, and include greetings (e.g.,"hello"), closings ("Good Bye"), acknowledgments (e.g., "Okay", "uh-huh" or repeating part of what the speaker said), stalling for time (e.g., "Okay", "Let me see"), or signals of speech repairs (e.g., "oops") or misunderstandings ("sorry?", "huh?"). They also might address the communication process explicitly, say to establish the communication channel (e.g., "Are you there?", and answering with "I'm here"), to address communication problems (e.g., "I didn't hear/understand what you said"), or to explicitly manage delays or maintain the turn (e.g., "Wait a minute").

Other-level

This tag indicates that the utterance unit does not fall neatly into the Task, Task-management, or Communication-management category even though it may be relevant to the dialog. Jokes, non-sequitors, and small talk would fit into this category.

Discussion

In coding this dimension, you should remember that every utterance has a Communication component in some sense, but that utterances should be marked at the Communication-management level only when they make no direct contribution to solving the task. In other words, Communication-management level utterances are concerned exclusively with maintaining the conversation and if they were removed, the conversation might be less fluent but would still have the same content relative to the task and how it was solved. For instance, the greeting "hi" could be considered at the Task level in the sense that it starts the process of performing the task. Removing the utterance, however, would have no significant effect on the task or the way it was performed, thus we know its function is mainly at the Communication-management level.

Sometimes it is tricky distinguishing between Task-management and Communication-management, as utterances at either level may not add content directly relevant to the Task. For example, consider the utterance "Sorry for taking so long" which causes problems because it can be interpreted as apology both for not answering quickly enough (Communication-management) or for slowing down the interaction (Task-management). If you think that such an utterance has a component of both readings, however, then it should be labeled as Task-management.

Making the distinction between Task and Task-management can also be tricky in dialogues where the task involves solving a problem and/or developing a plan. Since the task involves building and evaluating plans, utterances such as "does the plan work?", and "how does that sound" should be marked at the Task level. Utterances referring to the current problem to be solved should also be labeled Task: "what is the problem", "the problem is we need to get two boxcars of oranges to Avon", "we have to get two boxcars of bananas to Corning", "is there a time limit", etc. The utterances that are the most complicated appear to have interpretations at both levels. For instance, the utterance "does that solve the problem?" could be intended as a request to check if the plan should work (a Task level utterance) or might be intended to ask whether the session is done (i.e., a paraphrase of "Are we done with the task?" which is an Task-management level utterance). In cases where it seems to do both, we will mark it at the Task-management level.

Note that when answering a question at one level, the answer is usually, though not always, at the same level. Question-answer pairs can occur naturally at these levels as seen in the following examples:

```
Task: utt1: u: How long does it take to get to Corning?
Task: utt2: s: Three hours.

Task-management: utt1: u: Do I have to state the problem?
Task-management: utt2: s: Yes.

Communication-management: utt1: u: Can you hear me.
Communication-management: utt2: s: Yes
```

2.3 Forward Looking Function

The purposes behind an utterance are very complex. This dimension characterizes what effect an utterance has on the subsequent dialogue and interaction. For instance, as the result of an utterance, is the speaker now committed to certain beliefs, or to performing certain future actions? Note it is often difficult to determine what actions the speaker intended to perform with an utterance. Also the effect that an utterance has on the subsequent interaction may differ from what the speaker initially intended by the utterance. For this reason, annotators are allowed to look ahead in the dialog to determine the effect an utterance has on the dialog. Note, there are several decision trees presented in this section that instruct annotators on how to make labeling decisions. The decision trees ask questions about the speaker's actions but the annotator may need to see how the listener responds to the speaker in order to determine what the speaker's actions were.

Often, there are many different effects simultaneously achieved by an utterance. To allow for this, the coding in this dimension allows eight different aspects of every utterance to be coded. Specific constraints on how many aspects you can code at a time and the criteria for choosing between cases where several seem applicable will be specified in the domain-specific annotation instructions. As a default case, you can assume that you should code all aspects that are applicable.

Forward Looking Function

- Statement
  - Assert
  - Reassert
  - Other-statement
- Influencing-addressee-future-action
  Open-option
  Action-directive
- Info-request

- Committing-speaker-future-action
  - Offer
  - Commit
- Conventional
  - Opening
  - Closing
- Explicit-performative
- Exclamation
- Other-forward-function

2.3.1 Statement

The primary purpose of statements (utterances having a tag in the statement aspect) is to make claims about the world as in utterances such as "It's raining outside" or "I need to get cargo there" (the world includes the speaker) and in answers to questions. As a rule, the content of statements can be evaluated as being true or false. Note that the speaker does not have to be strongly claiming that something is true or false. This classification also includes weak forms of statement such as hypothesizing or suggesting that something might be true. Note also that we are only coding utterances that make explicit claims about the world, and not utterances that implicitly claim that something is true. As a intuitive test as to whether an utterance makes an explicit claim, consider whether the utterance could be followed by "That's not true". For example, the utterance "Let's take the train from Dansville" presupposes that there is a train at Dansville, but this utterance is not considered a statement. You couldn't coherently reply to this suggestion with "That's not true".

Figure 1 shows a decision tree for the statement aspect. The top level is an applicability condition that tests whether an utterance should be coded along this aspect. We only code utterances that make an explicit claim about the world. If the statement aspect is applicable, the decision tree then indicates how to select the appropriate tag. The key distinction for the Assert tag is that the speaker is trying to change the beliefs of the hearer. If this is not true, then a further distinction is made depending on whether the claim has been made previously in the dialogue or not.

2.3.2 Info-Request

The Info-request aspect is simply a binary dimension where questions and other requests for information are marked. Utterances that introduce an obligation to provide an answer should be marked as Info-request. Note, answers can

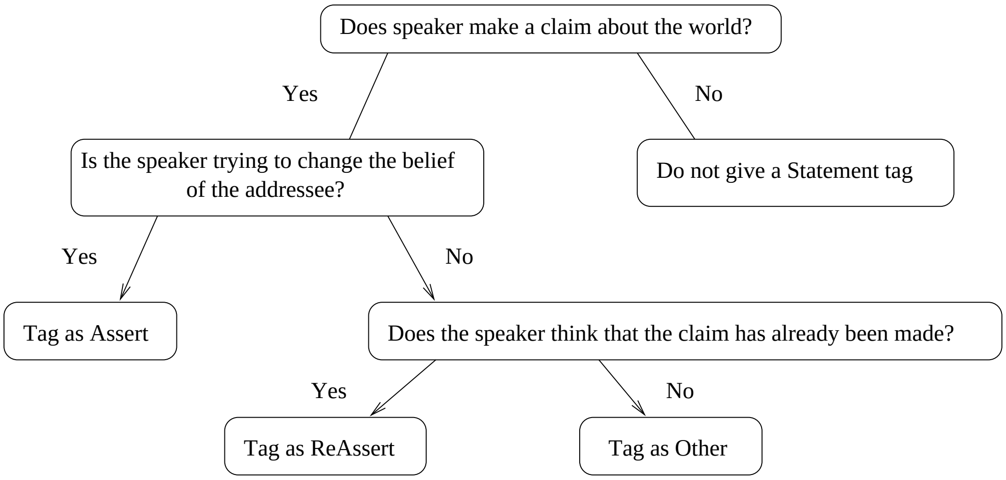

Figure 1: Decision Tree for Statement Aspect

be nonverbal actions providing information such as displaying a graph. Info-request includes all questions, including yes/no questions such as "Is there an engine at Bath?", "The train arrives at 3 pm right?", and even "The train is late" said with the right intonation. The category also includes wh-questions such as "When does the next flight to Paris leave?" as well as actions that are not questions but request information all the same such as "Tell me the time". Requests for other actions that can be used to communicate, such as "Show me where that city is on the map" are also considered Info-Requests. Basically, any utterance that creates an obligation for the hearer to provide information, using any form of communication, is marked as an Info-Request.

2.3.3 Influencing-addressee-future-action (Influence-on-listener)

The primary purpose of this aspect is to directly influence the hearer's future non-communicative actions, as in the case of requests ("Move the train to Dansville" and "Please speak more slowly") and suggestions ("how about going through Corning"). There are many verbs in English that describe variations of these acts that differ in strength, including acts like command, request, invite, suggest and plead. These distinctions in strength are not fully captured in the current annotation scheme.

A rough test to see if an utterance is in this aspect is to see whether the hearer could coherently respond with "I can't do that". This test includes some utterances such as "tell me the time" that do not belong in this category because they only involve requesting information. In addition, there are some examples where an utterance is in this class but the test fails. For instance, in TRAINS,

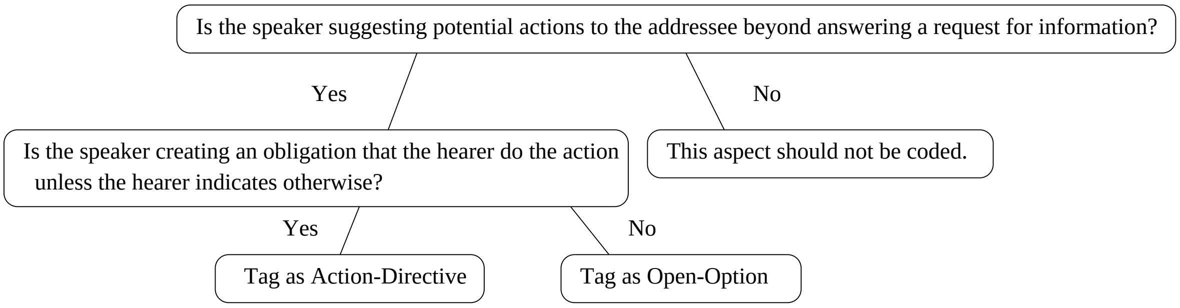

Figure 2: Decision Tree for Influencing-addressee-future-action

say the participants are discussing how to get some oranges to Bath, and one says "There's an engine at Avon", suggesting that they could use that engine to move the oranges. This utterance falls into this aspect but could not be followed by "I can't do that", although the variant "I can't use that engine" is fine. Questions only belong in this class if they suggest a course of action in addition to asking a question. For instance, in TRAINS, the question "how long will it take if we go through Corning" is sometimes used to suggest that they move a train through Corning. All questions obligate the hearer to reply, but this is not sufficient in itself to mark a question as having the Influencing-addressee-future-action aspect.

This annotation scheme makes the distinction between an Action-directive, which obligates the listener to either perform the requested action or communicate a refusal or inability to perform the action, and an Open-option, which suggests a course of action but puts no obligation on the listener.

As stated above, Action-directives may vary in strength, from commands, such as "Open the door", to pleading, such as "I beg you not to go to the party", but in each case the hearer is obligated to either perform the action or respond to the request. While not all Action-directives are responded to, not responding would be considered to be rude. Open-options, on the other hand, can be responded to but also can be ignored without any negative effect since no obligation beyond normal conversational constraints is placed on the listener.

For example, the first utterance below is an Open-option (abbreviated here as OO) because B does not need to address it and can coherently answer with utt2.

```
utt1 OO         A: There is an engine in Elmira
utt2 Action-dir B: Let's take the engine from Bath.
```

On the other hand, in the following example utt1 is an Action-directive and B should explicitly refuse the suggestion if it is not adopted.

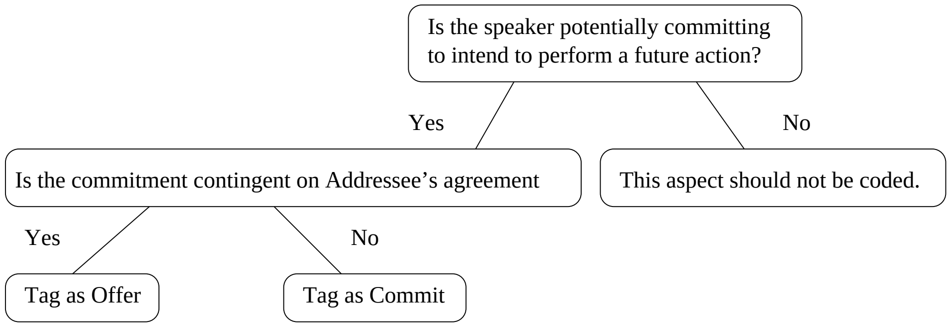

Figure 3: Decision Tree for Committing-speaker-future-action

```
utt1  Action-dir   A: Let's use the engine in Elmira.
utt2 Reject(utt1) B: No
utt3 Action-dir   B: Let's take the engine from Bath.
```

An example of an Open-option utterance from a furniture purchasing domain is shown below. Here what the speaker means is "we could buy my red sofa or my blue sofa".

```
Action-directive A: Let's buy the living room furniture first.
                 B: OK
OO, Assert,Offer    I have a red sofa for $150 or a blue one for $200.
```

Here is another example of an Open-option utterance from the TRAINS domain.

```
Assert A: I need to get the train at Avon to Bath.
OO     B: You could go through Corning.
```

2.3.4 Committing-speaker-future-action (Influence-on-speaker)

The defining property of utterances with this aspect is that they potentially commit the speaker (in varying degrees of strength) to some future course of action. The only distinction made within this aspect is whether the utterance's commitment is conditional on the listener's agreement or not. Commits that are conditional on the Addressee's agreement include what are called offers in English where the speaker indicates willingness to commit to an action if the hearer accepts it. The prototypical case of a commit not dependent on listener agreement is a promise, although this category may include other weaker forms of commitment such as "I'll probably be at the meeting at 3". The speaker is not making an offer here, he or she is just not making a very strong commitment.

Here are some examples:

Typical Offers:

```
Shall I come to your office
I'm free at 3  (in context of setting up a meeting)
```

Offers with explicit conditions:

```
I'll be free after four if my meeting ends on time
I can meet at 3 if you're free
```

A weak Commit:

```
Maybe I'll come to your party
```

Regular Commits:

```
I'll come to your party
I promise that I'll be there
```

Utterances that accept a previous request or Open-option will typically be a commit, as in the following dialogues:

```
Assert: B: I don't know what to do Saturday night
OO:     A: You could go to Bob's party with me
Commit  B: Great I'll see you there

Action-directive:  A: Take the engine from Corning to Bath.
Commit:            B: OK
```

The remaining forward looking functions are relatively rare. They include (1) conventional conversational actions such as greeting and saying goodbye, (2) explicit performatives where the speaker explicitly declares what is performed, as in firing someone by saying "You're fired", or, in TRAINS, quitting the task by saying "I quit", (3) exclamations such as "Ouch", (4) forward looking functions not captured by the current scheme such as holding/grabbing the turn by saying an utterance such as "Right" or "Okay".

- Conventional-opening: Is the utterance a phrase conventionally used to summon the addressee and/or start the interaction (e.g., "Can I help you?", "hi")
- Conventional-closing: Is the utterance a phrase conventionally used in a dialog closing or used to dismiss the addressee (e.g., "Good-Bye")
- Explicit-performative: Is the speaker performing an action by virtue of making the utterance (e.g., "Thank you", "I apologize")
- Exclamation: Is the utterance an exclamation (e.g., "Ouch")

- Other-forward-function: Is the speaker performing an action not captured by any other Forward Looking Function (e.g., signaling an error by saying "Opps")

Note that acts that are conventional openings or closings can be coded on other aspects as well. For example, at the start of a dialogue the utterance "Can I help you" could serve both as a Conventional-opening and as an offer.

A good intuitive test for the explicit performative class is whether you can insert the word "hereby" before the verb. For example, since "You're hereby fired" has approximately the same meaning as "You're fired", we know this is an explicit performative. The utterance "I want to go to the store" is not since "I hereby want to go to the store" is difficult to interpret, and in contexts where it might be understandable, the meaning would have changed.

2.3.5 Discussion

Note that the Forward Looking Function is a characterization of what effect the utterance has on the dialogue, even though the actual form of the sentence might look like something else. Each utterance could have multiple tags in this aspect depending on how many functions it simultaneously performs. As an example, in the right context, an utterance such as "There is an engine at Avon" could be both an Assert and an Open-option (stating the possibility of using that engine to move some cargo).

Other examples arise in dialogs where the agents are performing a task together. For instance, if the two agents are trying to agree on how to furnish a room, a suggestion like "Why don't we use the red sofa for the living room" is both a Offer (roughly, if you agree then I will use the sofa) and an Open-option (roughly, I state the possibility of you using that sofa). If this utterance is accepted by the hearer, then the net effect is that both participants are committed to using the red sofa.

Note that the previous version of this scheme had a separate category for conventional acts like thanking and apologizing. These acts now fall into different categories based on the tests presented. For instance, an utterance like "I'm sorry" is an Assert as it makes a claim about the world, which you can see by observing that it could be followed by the response "No you're not". The utterance "Thank you", on the other hand, cannot be responded to in such a way and only falls into the explicit performative category (observe that "I hereby thank you" is fine).

The Info-request and Influencing-addressee-future-action (Influence-on-listener) dimensions are similar in that they both apply to suggests and requests (Info-requests request communicative actions and Influencing-addressee-future-action utterances request non-communicative action). Table 1 gives examples of how various questions, imperatives, and indirect requests should be labeled in these aspects.

Example	Info-request	Influence-on-listener
What is the time?	X	
Tell me the time	X	
Can you tell me the time?	X	
Do you know the time?	X	
Go to Avon		Action-directive
There is a engine at Bath		Open-option
Use the engine at Bath		Action-directive
How long does it take to go from A to B	X	Option-option

Table 1: Examples of how to label Info-request and Influence-on-listener

Short utterances such as "okay" and "yes" can have many different interpretations depending on how they are being used. In fact, they can be Asserts, Commits and even Other-forward-functions. If the utterance conveys information in answering a question, then the utterance is only an Assert, as in the dialogue

```
Request-info  Utt: s: is there a train at Chicago?
Assert        Utt: u: yes.
```

On the other hand, if the utterance commits the speaker to a certain action, say in response to a request or invitation, then it should only be labeled as a Commit

```
Action-directive Utt: u: tell me if the route gets too long
Commit           Utt: s: okay
```

Cases where the speaker utters an "okay" to accept a request and then performs the act requested are also marked as a commit, even though the action to which the speaker is committing is performed immediately after, e.g.,

```
Action-directive  Utt: u: can you tell me the time?
Commit            Utt: s: okay.
Assert            Utt:    three o'clock.
```

Utterances such as "okay" can also be used not only to convey information but to manage the dialog, possibly to hold the turn, or to signal the introduction of a new topic. In these cases, the Forward Function of these utterances is not directly captured in the annotation scheme, and they should be marked as Other-forward-function. This category serves as a place holder for groups that want to analyze such utterances. Some communicative functions for these utterances are also captured in the annotation of Backward Looking Function that is described next.

2.4 Backward Looking Function

Backward Looking Functions (shown in figure 4) indicate how the current utterance relates to the previous discourse. For example, an utterance might answer, accept, reject, or try to correct some previous utterance or utterances. To capture these Backward Functions, we need to both identify the type of Backward Function and indicate the previous stretch of discourse that is being affected.

The previous utterance unit or set of units being responded to by the current utterance will be called the antecedent. The relations we consider here are local, and their antecedent typically is either the previous utterance unit or at least, in the previous turn. For the moment, we will assume that antecedents are single utterances near the current utterance, and in our examples, these will be marked in parentheses after the Backward Function tag. Note, if any Backward Looking Function is given to an utterance it must also be given a Response-to tag indicating its antecedent utterances. See section 5 for details on how to specify antecedent utterances in Response-to tags using the dat annotation tool. Constraints on what can be an antecedent and how to annotate complicated cases will be discussed after the basic set of tags is introduced.

There are four semi-independent aspects coded in this dimension. The first concerns relations affecting agreement about the task or whatever the topic of discussion is. The second concerns relations that signal understanding (or non-understanding). The third marks utterances that answer previous information requests. The fourth concerns the relationships between the informational content in utterances and is not developed further in this version of the annotation scheme. It provides a place holder for individual projects to define their own set of relations for their specific domains. The possibility of developing an abstract domain-independent level will be considered for later versions of this manual.

2.4.1 Agreement

The agreement aspect codes how the current utterance unit affects what the participants believe they have agreed to, typically at the task level. These relations occur in contexts where one agent has made some kind of proposal such as a request that the hearer do something, an offer that the speaker do something, or a claim about the world. The current utterance then indicates the other participant's view of the proposal. In general, the agent may explicitly accept or reject all or part of the proposal, or may simply be noncommittal on the proposal, or may leave the proposal open by requesting additional information or exploring the consequences.

Assuming the current speaker directly addresses another speaker's proposal, they could respond in one of the five ways shown in figure 5. Note, the label of Maybe applies to cases like the one in the figure where the speaker explicitly states that they cannot give a definite answer at that moment.

Note that the Accept-part utterance implicitly rejects part of utt1 however

Backward Looking Function

- Agreement
  - Accept
  - Accept-part
  - Maybe
  - Reject-part
  - Reject
  - Hold
- Understanding
  - Signal-non-understanding
  - Signal-understanding
    Acknowledge
    Repeat-rephrase
    Completion
  - Correct-misspeaking
- Answer
- Information-relation

Figure 4: Backward Looking Function

```
Context:          utt1: A: Would you like the book and its review?
Accept(utt1)            B: Yes please.
Accept-part(utt1)       B: I'd like the book.
Maybe(utt1)             B: I'll have to think about it
                        (intended literally rather than a polite reject)
Reject-part(utt1)       B: I don't need the review.
Reject(utt1)            B: No thanks
```

Figure 5: Various responses to an offer

this aspect only codes what is explicitly (accepted/rejected) addressed by the response. A response like "I'll take the book but not the review" will be segmented into two utterance units; one marked as Accept-part and the other as Reject-part.

Agreement can apply to cases other than proposals, as shown in the two examples below. In the first example, utt1 is an Open-option as it simply presents a possible option for solving a problem. Utt2 is still considered an accept though. Note, accepts are often words such as "alright", "yes", and "okay" as well as repetitions.

```
Open-option  utt1: s: we can unload them and then reuse the boxcars
                      on the way to Corning
Accept(utt1) utt2: u: alright
```

Accepts also can occur in response to Asserts, indicating that the information conveyed is accepted.

```
Assert       utt1: s: boxcars don't travel by themselves
Accept(utt1) utt2: u: okay
```

Accepts can also be used as a response to an information request, as in the example below. Note, utt3 is marked as an answer as discussed later.

```
Info-Request utt1: u: can you tell me the time?
Accept(utt1) utt2: s: yes
Answer(utt1) utt3:    it's 5 o'clock
```

The Hold tag applies to the case where the participant does not address the proposal but performs an act that leaves the decision open pending further discussion. This includes cases such as counter-proposals and questions that request additional information in order to help the participant make a decision (i.e., one sense of clarification request). This tag does not apply to cases where the speaker explicitly expresses uncertainty such as uttering "maybe".

In the dialog below, u's request is vague since there are two possible paths so s makes a clarification request by asking a question.

```
Action-directive         utt1: u: take the train to Corning
Info-request, Hold(utt1) utt2: s: should we go through Dansville or Bath
Assert, Answer(utt2)     utt3: u: Dansville
```

Instead of asking a question, a responder may make a clarification statement as shown below. This is also marked as a hold.

```
Info-Request utt1: u: how long will that take
Hold(utt1)   utt2: s: you want to go from Avon to Dansville
Answer(utt1) utt3: s: that's 5 hours
```

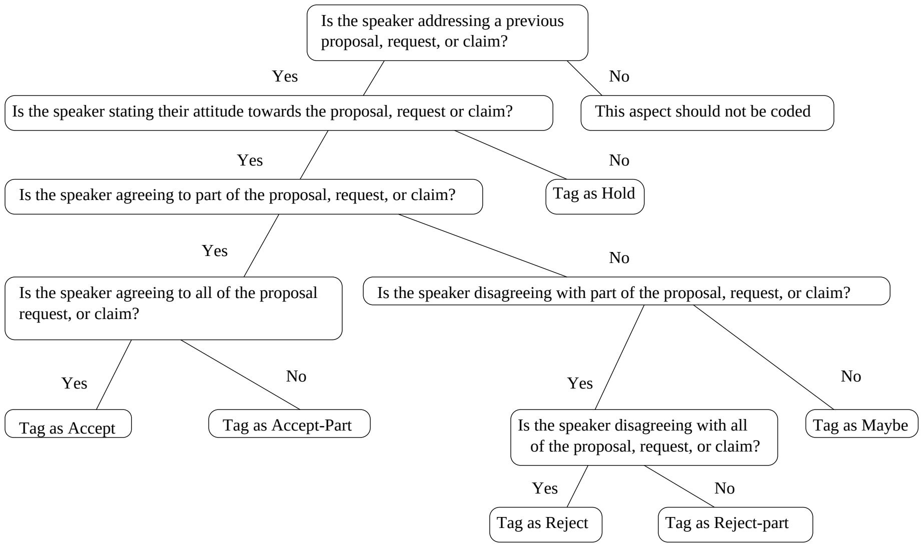

Figure 6: Decision Tree for Agreement Aspect

In the examples below, s's suggestion of an alternative route should be as marked as a Hold if you think that s is leaving the original option open, but as a Reject-part if not.

```
Action-directive        utt1: u: take the train to Avon via Bath
Open-option, Hold(utt1) utt2: s: How about we go via Corning instead
```

```
Action-directive                    utt1: u: take the train to Avon via Bath
Action-directive, Reject-part(utt1) utt2: s: Go via Corning instead
```

2.4.2 Understanding

This aspect concerns the actions that speakers take in order to make sure that they are understanding each other as the conversation proceeds. There are many levels of "understanding", ranging from merely hearing the words to fully identifying the speaker's intention. Here we group most of these levels together so that if the hearer is said to have understood the speaker, then the hearer knows what the speaker meant by the utterance.

Utterances that explicitly indicate a problem in understanding the antecedent are labeled as Signal-non-understanding. As an applicability test for Signal-non-understanding, you should be able to roughly paraphrase a Signal-non-understanding utterance as "What did you say/mean?". Note that not all clarification questions signal non-understanding. Clarification questions labeled as

Hold in the Agreement aspect involve acquiring additional information about how or why something was requested or proposed, and do not signal misunderstanding. Below are some examples of questions that are Signal-non-understanding (SNU) utterances.

```
Context: utt1: A: Take the train to Dansville
SNU            B: Huh?.                    (i.e., What did you say?)
SNU            B: What did you say?.       (i.e., What did you say?)
SNU            B: to Dansville?            (i.e., What did you say?)
SNU            B: did you say Dansville? (i.e., What did you say?)
SNU            B: Dansville, New York?     (i.e., What did you mean?)
SNU            B: Which train?             (i.e., What did you mean?)
```

On the other hand, responses that query how to comply with the speaker's request/proposal, or that question its desirability ("why are we doing this") are marked as Hold acts at the Agreement level and are not marked as Signal-non-understanding:

```
Context: utt1: A: Take the train to Dansville?
Hold(utt1)     B: through Avon?      (i.e, How shall we take the train)
Hold(utt1)     B: to get the oranges? (i.e., Why are we taking the train)
Hold(utt1)     B: Should it leave immediately?
                  (i.e., When should we take the train)
```

Utterances that explicitly signal understanding are marked with a Signal-understanding tag. Note that any utterance that doesn't explicitly indicate non-understanding implicitly indicates understanding. You do not need to mark such cases. Rather, there are some specific mechanisms used to explicitly signal understanding that we are interested in. Note in many cases, such utterances may also count as Accept acts at the understanding level. However, the examples below are Signal-understanding utterances (acknowledgments) that are not acceptances.

Acknowledgments are utterances consisting of short phrases such as "okay", "yes", and "uh-huh", that signal that the previous utterance was understood without necessarily signaling acceptance, as in

```
Context:   utt1: s: Take the Avon train to Dansville
Ack(utt1)  utt2: u: Okay
Hold(utt1) utt3     But wouldn't using the Bath train be faster?
```

Sometimes an acknowledgment interrupts a sentence as in the example below.

```
           utt1: u: if I take the engine and a boxcar from Elmira
Ack(utt1)  utt2: s: yes
           utt3: u: how long will that take
```

Another common case involves acknowledgments that are performed while the other agent is still speaking, which are often called backchannel responses. An example here is

```
          utt1: u: The we take the engine at Avon [to Bath](1) for the oranges
Ack(utt1) utt2: s: [uh-huh](1)
```

Of course, many times an acknowledgment also counts as an acceptance in which case it will be marked at both levels. In cases where it is uncertain whether the acknowledgment accepts the antecedent, then the utterance can still be marked unambiguously as an acknowledge, and can be labeled as a possible accept using the uncertainty modifier at the agreement level, as in

```
Assert                    utt1: s: It would take two hours [assuming](1) you have
                                   an engine at Bath
Ack(utt1), Accept(utt1)? utt2: u: [okay](1)
```

The Repeat-rephrase tag is used for utterances that repeat or paraphrase what was just said in order to signal that the speaker has been understood. Like acknowledgments, Repeat-rephrases do not necessarily make any further commitment as to whether the responder agrees with or believes the antecedent.

```
                      utt1: s: do you need the bananas [in boxcars](1) at Bath
Repeat-rephrase(utt1) utt2: u: [the bananas](1)
```

Sometimes a listener will show understanding by finishing or adding to the clause that a speaker is in the middle of constructing. Such phenomena are marked with the Completion tag as shown in the example below. Here u completes s's sentence; s then goes on to finish the sentence himself using the suggested completion. Note, completions also include cases where another speaker makes an extension (but not completion) to the current phrase being uttered.

```
utt1: s: so you've got the engines at Elmira and uh [](1) Avon
utt2: u: [Avon](1)
```

The Correct-misspeaking tag is used for utterances that by offering a correction indicate that the hearer believes that the speaker has not said what he or she actually intended. In the example below, u misspeaks by saying "engine E" instead of "engine E one" and s offers a correction.

```
                     utt1: u: so we should move the engine at Avon engine E to
Corr-misspeak(utt1) utt2: s: engine E one
Accept(utt2)        utt3: u: E one to Bath
```

This category only applies to cases where another speaker makes a correction. If u had corrected himself then no Correct-misspeaking label would be applied to this section of the dialog. There is currently no dimension in this scheme for annotating such speech repairs.

2.4.3 Answer

The Answer aspect is simply a binary dimension where utterances can be marked as complying with an info-request action in the antecedent. A standard question answer is shown below.

```
Info-request         utt1: u: can I take oranges um on tankers from Corning
Assert, Answer(utt1) utt2: s: no you may not they must be in boxcars
```

Most questions are answered with one or more declarative sentences although it is possible to answer a question with an imperative as shown in the direction giving example below. Note this imperative-looking answer is also marked as an Assert act as its Forward Function is to provide information rather than to influence u's future action. In fact, answers by definition will always be asserts. The answer is also an Open-option because it describes one option for u's future action.

```
Info-request                     utt1: u: How do I get to Corning?
Assert, Open-option, Answer(utt1) utt2: s: Go via Bath.
```

Questions generally cannot answer questions and are usually not answers. Mostly they are used for clarification and an answer is given later.

```
             utt1: u: How long will it take to go from Corning to Avon
             utt2: s: Which route do you want to take
Answer(utt2) utt3: u: go through Bath
Answer(utt1) utt4: s: it'll take 6 hours
```

Sometimes a speaker will answer an implicit or indirect question. The general rule of thumb is to consider whether the implicit or indirect question was obvious enough to obligate the hearer to respond with the information (i.e., the antecedent is an Info-request). The example below is similar to the previous one except that u has not asked a question and we would not mark utt1 as an Info-request. Thus s's utterance is an Action-directive, a request, and is not considered an Answer that gives information and makes a suggestion.

```
Assert           utt1: u: I need to get the train to Corning.
Action-directive utt2: s: Go via Bath.
```

Of course there are borderline cases where these distinctions are a matter of degree. For instance, if u says “I don't know how to get oranges to Corning”, in many contexts this strongly suggests an implicit question and implies that a response like “You could get them from Bath” is an Answer. These difficult cases are left to the annotator's intuition, but the two utterances should be annotated consistently, so that if the second is an Answer, the first is an Info-request.

Consider an unambiguous case where we wouldn't mark an information request and hence don't mark an answer. Here A is trying to avoid going to a meeting, but B does not cooperate. Because we do not consider utt2 as a request for information, we don't mark utt3 as an answer.

```
utt1: A: I should be at the meeting.
utt2: A: Luckily, I don't know what time it is.
utt3: B: It's 3 o'clock.
```

Note that clarification requests and other utterances that refuse to answer an Info-request are not considered answers. For instance, consider the dialogue fragment:

```
Info-request         utt1: u: How can I get oranges to Corning?
Assert, Reject(utt1) utt2: s: I don't know.
```

Here *s* states that he doesn't know the answer (and hence can't answer the question). Thus it is treated as a Reject just as in cases of requests for non-communicative actions as in

```
Action-directive     utt1: u: Please open the door
Assert, Reject(utt1) utt2: s: I can't, my arm is broken.
```

Occasionally, speakers ask a question and then answer it themselves. Even though it is the same speaker throughout, we still mark it as an info-request/answer pair, as in

```
Info-request utt1: u: Will I be able to take the tanker plus four boxcars
Answer(utt1) utt2: u: No
```

2.4.4 Information-Relations

The fourth aspect of the Backward Looking Function is the Information relation, which captures how the content of the current utterance relates to the content of its antecedent. This category is not currently elaborated and will be the subject of future study. For the moment, this aspect provides a hook for individual groups to experiment with their own schemes for encoding this information.

2.4.5 Antecedents

Typically responses follow directly after the utterances that they are responding to. Sometimes, however, they are further apart and separated by a series of clarification requests, confirmations, and elaborations. Only those responses that still have a close contextual connection to their antecedent should be annotated. Unfortunately, we do not yet have a precise definition of "close contextual connection." For the short term, here are some guidelines for refining intuitions about how close an utterance(s) and its response must be. Assume that utterance X by s is a proposal, a claim, a question or some other act that can be responded to. Then an utterance can be part of an antecedent to X if

- every utterance by the responding agent between utterance X and the current utterance can be marked as a Hold at the Agreement level;
- or the content of the entire interaction from X to the current utterance could be paraphrased as a single claim, proposal or question that is being responded to.

In such cases, the antecedent is marked as containing all the utterances from X up to the response. We expect that this definition of allowable antecedents may have to be modified after we have some additional experience coding and have seen where it breaks down.

Consider some examples of common cases that satisfy this definition.

```
Open-option              utt1: u: We could use that train.
Info-request, Hold(utt1) utt2: s: The one at Dansville?
Assert, Answer(utt2)     utt3: u: yes
Accept(utt1-utt3)        utt4: s: Okay.
```

In this case, the option accepted is to use the train at Dansville, and the content of utt1-utt3 could be paraphrased as "We could use the train at Dansville".

```
Open-option                   utt1: u: We could use the train at Dansville.
Open-option, Hold(utt1)       utt2: s: Could we use one at Avon instead?
Assert, Reject(utt2)          utt3: u: No, I want it for something else.
Open-option, Hold(utt1-utt3)  utt4: s: How about the one at Corning then?
Assert, Accept (utt4)         utt5: u: Okay.
Accept(utt1-utt5)             utt6: s: Okay.
```

In this example s produces two alternatives before an agreement is reached. The content of the antecedent for utt6 could be paraphrased as "We could use the train at Corning".

This example shows the initial speaker adding further elaboration before the question is answered.

```
Info-request                 utt1: u: How long will it take to get to Avon?
Info-request, Hold(utt1)     utt2: s: With engine E one?
Assert, Answer(utt2)         utt3: u: yes
Info-request                 utt4:    and going by way of Bath.
Assert, Answer(utt1-utt4)    utt5: s: Six hours.
```

Note that utt4 is classified as an Info-request even though it doesn't look like a question out of context. But it is said to influence u's future action and involves the act of providing information, hence it satisfies the definition of Info-request. This interpretation is further strengthened by considering that utt4 certainly is not making any claim about the world, nor committing u to any future action. The content of segment utt1-utt4 can be paraphrased as "How long will it take engine E one to get to Avon by way of Bath".

With keyboard dialogues, we need to generalize the definition of antecedent somewhat as a turn might contain several different interactions, each of which are then addressed in the following turn: in a keyboard dialogue, a turn may be divided into several antecedents, each of which can be responded to in the next turn.

Consider the following keyboard dialogue from the Coconut domain where participants must agree on a set of furniture with which to furnish two rooms. The dialog shows how in keyboard dialogs, several responses (each with its own antecedent) may occur in the same turn. Note, Assert is abbreviated (AS) here.

```
Open-option,Offer    utt1: M: i do have a lamp-floor, blue.  i have a green
                               table (200) and four chairs for 75 a piece.
AS                   utt2:    sorry I am taking so much time.
AS                   utt3:    I lost a chair. Meghan is finding it

AS,Accept(utt2-utt3) utt4: D: Not a problem with the time
AS                   utt5:    sorry about the typo, my brain forgets that
                               my fingers don't function as quick as it does.
Accept-part(utt1)    utt6:    the lamp and table sound good
```

We also may want to allow such structures in spoken dialogues, if we start to find cases where the more restrictive rule is a problem. Certainly a model of higher-level segmentation would predict that examples like the one above could occur naturally in spoken dialogue.

If we switch the example around a bit, however, we find a case involving a cross-serial dependency that can occur in keyboard dialogue but is extremely rare and possible disfluent in spoken dialogue:

```
Open-option,Offer        utt1: M: i do have a lamp-floor, blue.  i have a green
                                   table (200) and four chairs for (75)
                                   a piece.
Assert                   utt2:    sorry I am taking so much time.
Assert                   utt3:    I lost a chair. Meghan is finding it

Accept-part(utt1)        utt4: D: the lamp and table sound good
Assert,Accept(utt2-utt3) utt5:    Not a problem with the time
Assert                   utt6:    sorry about the typo ...
```

There may also be cases where you would like to mark more than one response relation for an utterance. We may consider adding this capability if there turns out to be a significant need. For the moment, however, if an utterance appears to realize several different responses, then you should annotate the one you feel is most important and relevant to the dialogue, and note the other interpretations using a Comment tag.

2.4.6 Discussion

There are a couple issues common to all the Backward Looking Functions. The first concerns whether a person can respond to themselves. Although this is rare, speakers sometimes answer their own questions as shown below or reject/accept their own utterances. In the example below, *u* answers her own question ("will I be able to take the tanker plus four boxcars") and then goes on to ask another ("I need two right").

```
utt1: u: <sil> I guess <sil> with one engine <sil>
           will I be able to take the tanker <sil> plus four boxcars <sil>
utt2: u: no
utt3: u: I + need <sil> two right +
```

Another issue concerns acknowledgments. Utterances such as utt2 are clearly acknowledging understanding/hearing what the previous speaker said. It is a difficult issue whether utt2 is accepting utt1. If you do decide that utt2 is an accept or an accept with uncertainty, you still need to mark it as an acknowledgment since acceptances by their definition indicate acknowledgment.

```
utt1 u: um engine two from Elmira to Corning
utt2 s: okay
utt3 s: let's pick up oranges in Corning
```

3 Multi-Dimensional Problems

The dimensions of the DAMSL annotation scheme are mostly independent however there are a few dialog phenomena that form distinct patterns across the dimensions and should be annotated consistently. Acknowledgments and accepts are one such phenomenon. If an utterance is labeled as an acknowledgment but not an accept then it must be labeled as Communication Management at the Information Level dimension because it signals hearing/understanding of the message but says nothing about its content. If an utterance is an accept (and by definition an acknowledgment) then it should be labeled with the same Information Level as its antecedent as shown in the examples below:

```
Task              utt1: u: take the boxcars to Corning
Task,Accept(utt1) utt2: s: okay
```

```
Task-Management              utt1: u: let's work on the oranges plan first
Task-Management,Accept(utt1) utt2: okay

Comm               utt1: u: I'm turning up the microphone so I can hear you
                             better
Comm,Accept(utt1) utt2: s: okay

Other              utt1: u: that noise is the telephone next door
Other,Accept(utt1) utt2: s: okay
   (assuming the telephone had nothing to do with the dialog)
```

Check questions are a second phenomenon that forms a distinct pattern across two types of labels. Check questions such as the example below are similar to yes/no questions but can be answered with words such as "right", "okay" in addition to "yes" and "no". The questions ask for confirmation of a fact, so unlike standard questions their Statement tag is Other-statement instead of none. Check questions are labeled Other-statement instead of Assert because the speaker is asking for confirmation, not trying to change the belief of the addressee. The speaker does not think the claim has already been made; if it had they would not need to ask for confirmation. Responses to check questions are both answers and asserts. The pair of a check question and a positive answer also resembles an accepted assert (such as shown in utterances utt3 and utt4) so a positive answer to a check question should also be labeled as an accept. Note, if utt3 is said with the proper intonation then it can be a check question as well.

```
Info-Request, Assert utt1: u: and it's gonna take us also an hour to load
                               boxcars right
Answer,Accept(utt1)  utt2: s: right

Assert       utt3: u: and it's gonna take us also an hour to load boxcars
Accept(utt1) utt4: s: right
```

Note, in general answers to questions are always asserts. Words such as "okay" and "right" when not answers are not asserts. However responses that make a claim about the world are asserts. This even includes responses such as "that's right".

4 Tagging Cue Words and Speech Repairs

Spoken language contains many elements that function mainly to manage the interaction (i.e., turn taking) and to maintain reliable communication. Some of these elements include cases where the speaker is stalling for time while they think (keeping the turn in other words), signaling a speech repair/topic shift, or making a confirmation. This section describes how to label such phenomenon with the current annotation scheme.

As discussed before, cue words include words such as "okay" that, depending on the context and the way they are pronounced, can serve several different purposes in the dialog. When used as a response to a proposal, request or statement, it has the Backward Function Accept and a Forward Function such as Commit. If it is used simply to signal understanding, then it has the Backward Function Acknowledge and no Forward Function. On the other hand, if it is used to hold a turn or signal a topic shift, it has a Forward Function not captured with the current scheme except by the Other-forward-function label.

Consider the following example that contains two instances of "okay", one as an acceptance and the other as a cue word to maintain the turn. *u* first accepts *s*'s directive with an "okay". Since the effect of this act is that the speaker is committed to get a tanker, it is marked as a Commit. *u*'s second utterance seems to be further confirmation that utt1 was understood (and might even be labeled as an Accept instead of a Repeat-rephrase). Utt4 is a cue word uttered presumably in order to keep the turn or to signal a continuation of the topic. Presently, it would be labeled as Other-forward-function. Given its position in a series of utterances by *u*, and its intonation, it clearly is not a response to anything.

```
Action-directive       utt1: s: so I'm assuming you'll also be taking a
                                 tanker from [Corning](1)
Commit,Accept(utt1)    utt2: u: [oh](1) okay
Repeat-rephrase(utt1)  utt3:    take a tanker there
Other-forward-function utt4:    okay
Assert                 utt5:    so its two hours ...
```

Sometimes, cue words like the one above will not be broken into their own utterance units. For example, *u*'s third utterance above might have been "Okay, so its two hours ...". In this case, the most important aspect of the utterance unit is the Assert act, and the cue word's individual functions would be ignored.

Consider another example. Cue words such as "alright" may maintain a turn, confirm understanding, make an acceptance, or may signal a discourse event such as a speech repair or topic switch. In the example below, *u* utters the first "alright" with the intention to signal the description of a new plan. The second "alright" signals the second start of the restarted phrase. The current annotation scheme does not distinguish between these cases, and both are marked as Other-forward-function.

```
Assert                  utt1: u: ah three p.m. that's not gonna work
Other-forward-function  utt2:    alright um <breath>
Abandoned               utt3:    if I take
Other-forward-function  utt4:    alright
```

```
Assert                  utt5:    if I take the engine and a boxcar from
                                  Elmira
Ack(utt1-utt5)          utt6: s: yes
Assert                  utt7: u: uh I just use it
```

As previously noted, the utterance segmentation will affect how you label cue words. For instance, utt3-utt5 might have been a single unit (as shown below), in which case the speech repair would be ignored in favor of the completed act.

```
Assert, Task  utt1: u: if I take <sil> alright <sil> if I take the engine
                        <sil> and a boxcar <sil>  from Elmira
```

Sometimes a speaker is interrupted and their utterance is not completed. In these cases, you have to decide whether the utterance was abandoned in the sense that the dialog continues as though it were never said. In the example below, *u* never completes her sentence but *s* treats it as an information request (as if *u* wants him to complete her sentence), so utt1 is not marked abandoned.

```
Info-Request         utt1: u: so that's
Assert, Answer(utt1) utt2: s: loading the orange <sil> juice will take
                                another hour
Accept(utt2)         utt3: u: okay
```

5 Mechanics of Annotating

A new dialog annotation tool (dat) is available on our web site:

http://www.cs.rochester.edu/research/trains/annotation

The site contains instructions on how to setup and run dat. When you start dat, you can give it a dialog file to open. You can also chose **Open** from the **File** menu to open a dialog. dat expects files to be in a format called DAMSL (Dialog Act Markup in Several Layers) which is a variant of SGML tailored to this application. The dat tool can convert raw text files into DAMSL and the man pages of dat contain more information about this format.

The sets of tags discussed previously can be applied through menus, buttons, and text fields (in the bottom half of the dat window) once an utterance or segment is selected. dat allows you to select utterances by clicking on them. Clicking on another utterance while holding the shift key allows multiple utterances to be selected. Once you have selected one or more utterances, you may select various Utterance-Tags for the utterance(s). In addition, if the utterance(s) are a response then choose the appropriate Backward Function labels and indicate what was responded to, using the Response-to field. In this field, type the names of the utterances that a response pertains to, and separate names by dashes (NOT spaces). In the example below, you should type "utt1-utt2" in the Response-to field of utt3 since it accepts both utt1 and utt2.

```
T1  utt1 u: take the engine to Corning
|   utt2 u: there will be a tanker there and we will take that to Elmira
T2  utt3 s: okay
```

The Communicative-Status tags are implemented as menus near the middle of the screen. If a tag applies then select "yes" or "maybe" from its menu depending on how certain you are. All of the other dialog tags have question-mark buttons next to them allowing uncertainty to be encoded.

Once you are done annotating an utterance or a set of utterances, click the "Apply" button. Click "Reset" to eliminate any changes you made since you last clicked "Apply".

We encourage you to listen to any speech associated with each utterance. To do so, select the utterance unit and click the "Play Speech" button.

Note, tagging a series of utterances with a label such as Assert means that you have a series of Asserts. If you would rather a series of utterances be considered one unit then form a segment from the utterances and label the segment. To form a segment, first select a series of adjacent utterances and then choose **Define Segment** from the **Group** menu. You can click on this new segment and label it just like an utterance.

When you wish to stop annotating, select **Save As** from the **File** menu. The dialog box is similar to the one you saw when opening a file. Edit the filename to create a new file to save your changes in. Selecting **Save** afterwards will save in this new file. You may open this file later in order to finish an annotation. You quit dat by selecting **Quit** from the **File** menu. To attach notes to the dialog, use a comment tag or select **Dialog Attrs** from the **Edit** menu to make general comments.

The organization of tag descriptions in this manual was picked to simplify the explanation but this does not mean it is easiest to annotate in this order. You may annotate in any order you wish.

6 Appendix: Outstanding Issues

These are temporary decisions that may be reconsidered later.

Agreement to Asserts

Consider

```
A: It's raining
B: yes.
```

Does B's utterance have a Forward Function? Intuitively, it seems that it does and we will for the moment label such acts as Assert.

Conditional Commits

There are some commits that are conditional but not conditional on a decision of the hearer, such as "I'll be there if the package arrives on time". At present, we would mark this as a Commit even though it does not seem to belong in the same group as promises which are unconditional.