

The Parser LF Output

The logical form language is not a meaning in any traditional sense - rather it is a description of the semantic content that can be mapped to a meaning via contextual interpretation. For example, the LF representation might be fed through a reference resolution component and sense disambiguation and mapped to a prolog-style data-base query system. In another case, we might map to a frame-based representation (again after discourse interpretation). The motivation for this language is to have uniform representation of content of sentences independent of the knowledge representation used by the reasoning systems. Another consideration is the support of robust parsing and interpretation. The LF is designed so that the representation of “correct” fragments extracted from an utterance will be identical in form to the LF of the same phrases if we had produced the full parse. The key technique that enables this is the use of a “flat” unscoped representation. Since scoping cannot in general be determined independent of context, we do not attempt to encode scope in the LF form.

The logical form consists of a set of terms describing objects and relationships evoked by the utterance. One key term is speech act that was performed. For example, the logical form of

But the man wants to eat it

is as follows:

(LF::SPEECHACT V12 LF::SA_TELL :CONTENT w1 :MODS (b1))	s1 is a TELL speech act with content w1 and (discourse) modifier b1
(LF::F w1 (:* LF::WANT W::WANT) :ACTION e1 :EXPERIENCER m1 :TMA ((W::TENSE W::PRES)))	w1 is a wanting relation between m1 and e1, that holds at a time indicated by the present tense
(LF::THE m1 (:* LF::PERSON W::MAN))	m1 is some man identifiable in context
(LF::PRO i1 (:* LF::REFERENTIAL-SEM W::IT) :CONTEXT-REL W::IT)	i1 is a some object identifiable in context by pronoun "it"
(LF::F e1 (:* LF::CONSUME W::EAT) :THEME i1 :AGENT m1))	e1 is an eating relation between m1 and p1
(LF::F V176949 (:* LF::CONJUNCT W::BUT) :LF::OF w1)	w1 is related by a "but" relationship to previous context

Figure 1: The content of *But the man wants to eat it*

This document describes this output representation in detail.

1. Capturing Semantic Content

Building Blocks and Terms

The basic atoms of the logical form language consist of the following, each which will be described in more detail as we go along:

ATOMIC TYPES, atoms that denote classes of objects, be they physical objects, situations, abstractions, and so on (e.g., LF::PERSON, LF::SEND, LF::NUMBER). These are organized into a hierarchy in the LF Ontology.

TYPE FUNCTORS, functions that construct new types, including SET-OF, SEQUENCE-OF, and L (lambda), described below.

ROLE NAMES, which can be thought of as "slots" in a frame, labeled arguments to a predicate, or functions from one object to another.

VARIABLES, written in lower case, x, y, c

LOGICAL OPERATORS, operators, such as AND and OR

TERM CONSTRUCTORS: LF::THE, LF::A, LF::PRO, LF::IMPRO, etc. as described below.

Terms in the LF all have the exact same format, namely

($\langle term\ constructor \rangle \langle var\ name \rangle \langle type \rangle [\langle role \rangle \langle value \rangle]^$)*

where $\langle value \rangle$ can be

A variable

A list of variables

A list of pairs of form $(\langle role \rangle \langle value \rangle)$, which is currently used for the value of the TMA slot as described below.

Types

We can specify more complex type descriptions from simple ones. The most common is a construction that encodes more specific lexical information within a defined LF type.

($\cdot^ \langle atomic\ type \rangle \langle species \rangle$)*

For example, assuming LF::SELF-LOCOMOTE is a type classifying events involving a person walking, running, etc, then, $(\cdot^* LF::SELF-LOCOMOTE W::WALK)$ is a specialization of LF::SELF-LOCOMOTE associated with the word "walk". Similarly, $(\cdot^* LF::SELF-LOCOMOTE W::AMBLE)$ is the specialization associated with the word "amble". The semantic restrictions used in parsing do not typically distinguish between any specializations of LF::SELF-LOCOMOTE, but the more specific type information is retained that may be used by domain-specific reasoning engines. Complex types allow the information to be encoded without having to explicitly expand the generic LF ontology indefinitely.

Complex types can be constructed using a lambda function L:

($L \langle var \rangle \langle atomic\ type \rangle [:\langle rolename \rangle \langle var \rangle]^$)*

Allows the construction of complex type. For instance, If LF::MAP is a type (describing maps), then "map of there" might be the type $(L x (\cdot^* LF::MAP W::MAP) :OF V11)$, where OF is a role relation defined for LF::MAP and V11 is the ID for the pronoun "there". This construction is mostly used in plural forms (see below).

The types of plural forms are constructed using the SET-OF operator.

(SET-OF $\langle type \rangle$)

$(SET-OF (\cdot^* LF::MAP W::MAP))$ is the type corresponding to sets of maps. More complex sets can be described using the lambda construct. For instance, the set corresponding to "Maps of there" would be $(SET-OF (L z (\cdot^* LF::MAP W::MAP) :OF V11))$.

Another complex type is that of a sequence. As with SET-OF, we can specify an arbitrary set of constraints on the elements of the sequence.

(SEQUENCE-OF <type>)

The type of a sequence consisting of elements that satisfy the proposition. For example, an object of type (SEQUENCE (:* LF::VEHICLE W::TRUCK)) is a sequence of trucks.

(<OR> <type>*)

A disjunction of types providing a mechanism for capturing very simple uncertainty.

2. Speech Acts

Utterances are represented at the top level by a surface speech act captures the literal or surface speech act of the utterance¹. Using the example above with But the man wants to eat it, the top level form is

(UTT :TYPE UTT :ROOT V12 :TERMS ...)

where the value of :terms is a list of terms as shown in Figure 1. The root term, in this case V12 is defined as one of the terms as indicates the surface speech act, which in this case is a LF::SA_TELL speech act that corresponds to most declarative sentences, i.e.,

(LF::SPEECHACT V12 LF::SA_TELL :CONTENT w1 :MODS (b1))

The content w1 is defined by other terms. Discourse connectives such as but are treated as modifiers on the speech act. In this case, the term b1 defines the modifier as is defined as shown in Figure 1.

Surface Acts

The surface speech acts are listed below, together with the slots that can occur. Most of these acts allow :CONTENT and :MODS slots. In few cases, other slots are possible. The associated roles (except for :MODS which can occur with any term) are listed for each act.

W::SA_Wh-Question :content :focus	“Wh” questions	Where is the knife? When? What?	
W::SA_YN-Question :content	Yes no question	Is the knife in the kitchen?	
W::SA_Request :content	Imperative, typically a command	Get the knife.	
W::SA_Response :content	Responses to yes-no questions	Content	Example
		POS	Yes
		NEG	No
		UNSURE-POS	Maybe
		UNSURE-NEG	I don’t think so
		UNSURE	I don’t know

¹ Even when subsequently interpreted indirectly, the surface act influences the allowable forms of appropriate responses. For example, in response to the invitation *can you come to my party*, one can accept by saying *yes, i can*. But this acceptance would be inappropriate in response to the invitations *Please come to my party* or *Let me invite you to my party*. Each of these invitations would have a different surface act form. The fact that they are invitations is inferred by contextual interpretation.

W::SA_Tell :content	Assertions	The knife is in the kitchen
W::SA_Evaluate :content	Acts that express an opinion about something	Good, bad, excellent, OK, so-so, ...
W::SA_Ack :content	Acts that acknowledge or confirm	OK
W::SA_Request-comment :content	Acts that suggest an object or action	How about coming to my party. What about a beer. How about Toronto.
W::SA_Greet :content	Greetings and Goodbyes	Hello, Hi, Bye,
W::SA_Thank	Thanks	Thanks, thank-you,
W::SA_Welcome	Responses to thanks	You're welcome, not at all
W::SA_Discourse-Manage	Acts that help manage the conversation, grounding, etc	Just a second, uh-huh,

Figure 2: The Surface Speech Act Types

Discourse Adverbials

As mentioned above, the discourse adverbials relate the current utterance to the discourse context. Currently the parser does not analyze these into some deeper representation. Rather it just passes on a general classification (e.g., LF::CONJUNCT) and the actual lexical forms for use in contextual interpretation. These connectors can affect many aspects of the analysis, including not only tense, but also the discourse act performed and what collaborative problem solving act is inferred. For example, the discourse adverbial “And” as in “Then it left” is treated as a modifier of type (:* LF::CONJUNCT W::THEN) an the full LF for the speech act is

(LF::SPEECHACT V7263 W::SA_TELL :CONTENT V7035 :MODS (V6997))
 (LF::F V7008 (:* LF::CONJUNCT W::THEN) :OF V7263)

Some of the general classes are shown in Figure 3.

LF::CONJUNCT	And, and-then, so, but, ...
LF::SEQUENCE-POSITION	first, second, next, last, ...
LF::TOPIC-SIGNAL	by the way, anyways, ...
LF::POLITENESS	Please
LF::DEGREE-OF-BELIEF	Hopefully, Actually, in fact, ...
LF::INTERJECTION	You know, I guess, ...
LF::QUALIFICATION	Probably, originally, eventually, ...
LF::ADDITIVE	Too, also, ...
LF::REASON	So that, because, since,
LF::PREREQUISITE	If, just in case, ...
LF::QUALIFICATION	Maybe, ...

Figure 3: Some discourse adverbials

3. A Quick Overview of Basic Constructions

Simple Descriptions

Simple descriptions involve term constructors corresponding to definite and indefinite forms. For example:

the train -- (LF::THE x (:* LF::VEHICLE W::TRAIN))
 the trains -- (LF::THE x (SET-OF (:* LF::VEHICLE W::TRAIN)))
 a train -- (LF::A x (:* LF::VEHICLE W::TRAIN))
 trains -- (LF::A x (SET-OF (:* LF::VEHICLE W::TRAIN)))

Simple Events

The meaning of clauses is modeled as a relation between objects that are the arguments to the verb, and indicated by expressions using the formula constructor F. This is a neutral term that is used for any clause, whether it be main clauses, subordinates, complements, and so on. The use of the constructor F is determined by syntax rather than the semantic type of the event. The fact that a relation is claimed to represent the world would be captured in the speech act - the semantic formula describes the content of the sentence but makes no claim about the world. The event described in the sentence A man loaded the cargo would be:

(LF::F t1 (:* LF::FILL-CONTAINER W::LOAD) :AGENT m1 :THEME c1)
 (LF::A m1 (:* LF::PERSON W::MAN))
 (LF::THE C1 (:* LF::COMMODITY W::CARGO))

Note that events can occur in referring expressions as well. For instance, consider the following NP.

The loading of the cargo

(LF::THE t1 (:* LF::FILL-CONTAINER W::LOAD) :THEME c1)
 (LF::THE C1 (:* LF::COMMODITY W::CARGO))

This says that there is an LF::LOAD relation between m1 (defined as a man) and t1 (defined as the truck).

Modifiers

The LF of adjectives such as "red" are not treated as role predicates because they are time varying. A block might be red today, and green tomorrow. Furthermore, many adjectives take additional arguments and modifiers (as in "eager as a beaver", "ready to load"). Such propositions will use a named argument representation as we do with verbs. In addition, rather than introduce predicates for every adjective, we cluster them into ones that give alternate values on the same scale. Thus, "red" is realized by the type (:* LF::COLOR-REL RED). To attach a modifying phrase to a description, we use the :MODS role that takes a list of relation objects. For example:

The red truck

(LF::THE V53 (:* LF::LAND-VEHICLE W::TRUCK) :MODS (V57))
 (LF::F V57 (:* LF::COLOR-VAL W::RED) :OF V53047))

Typically, the single argument in unary relations is identified as the role :OF, as with the LF for red above. For binary functional relations, the arguments are typically identified as :OF and :VAL. For example, spatial prepositions use such arguments, as seen in the example:

The truck in the city

```
(LF::THE V27 (:* LF::LAND-VEHICLE W::TRUCK) :MODS (V20))
(LF::F V20 (:* LF::SPATIAL-LOC W::IN) :LF::OF V20357 :LF::VAL V240)
(LF::THE V240 (:* LF::DISTRICT W::CITY))
```

4. Noun Phrases in More Detail

There is a fair range of term constructors needed to handle the variety of noun phrases that occur. The ones defined so far are shown in figure 4.

THE	a definite form (we expect to be able to resolve it from context)
A	an indefinite form (we expect it to be introducing new object into context)
PRO	an pronoun form (we expect it to be resolved from local context)
IMPRO	an implicit anaphoric form (we expect it to be resolved from local context)
BARE	forms that have no specifier and are typically ambiguous between generic, kind, and indefinite interpretations
MASS	forms that refer to substances (e.g., some sand)
QUANTIFIER	“universally” quantified constructions (e.g., all trucks)
QUANTITY-TERM	a number or a quantity expressed in units (e.g., three pounds)
WH-TERM	“wh” terms as in questions (e.g., which trucks)

Figure 4: The Noun Phrase Term Constructors

Here we discuss the major classes of noun phrases and how they map into the LF forms.

Names

Names are treated as definite descriptions. A special role NAME-OF relates the object to its linguistic name. For instance, we have

```
John
(LF::THE x LF::PERSON :NAME-OF (W::JOHN))
```

The NAME-OF slot is a list to accommodate multi-word names such as "The New York Times":

```
The New York Times
(LF::THE x LF::PUBLICATION :NAME-OF (W::THE W::NEW W::YORK W::TIMES))
```

Although rare, this form can include modifiers as well, as in the phrase,

```
The other John
(LF::THE x LF::PERSON :NAME-OF (John) :MODS f1)
(LF::F V53605 (:* LF::IDENTITY OTHER) :LF::OF V53611)
```

Pronouns

Pronominal forms use the term constructor PRO. We also have a specially defined type in the ontology, LF::REFERENTIAL-SEM, that includes all types of objects that can be typically referred to (objects, events, some abstract objects). Pronoun LFs uses a special role called CONTEXT-REL that indicates information relevant for how the expression relates to the context. In general, the value of this slot is simply the lexical form of the pronoun that was used. For example, the pronoun it would have the LF

```
(PRO x (* LF::REFERENTIAL-SEM W::IT) :CONTEXT-REL W::IT)
```

The pronoun he would have the LF

```
(PRO x (* LF::PERSON W::HE) :CONTEXT-REL W::HE)
```

The pronoun them would be as follows (unless its type is further constrained by its surrounding context):

```
(PRO x (SET-OF (* LF::REFERENTIAL-SEM W::THEM)) :CONTEXT-REL W::THEM)
```

Modifiers for plurals

With plurals, some modifiers apply to each individual in the set, while others apply to the set itself. For example, in the NP The three red trains, the adjective red modifies each of the trains in the set, while three gives the cardinality of the set and it not a properties of the members. The cardinality of sets is captured in the role :SIZE.

The three red trains --

```
(LF::THE x (SET-OF (L y (* LF::VEHICLE W::TRAIN) :MODS (v1))) :SIZE V3)
(LF::QUANTITY-TERM V3 LF::NUMBER :VALUE 3)
(LF::F v1 (* LF::COLOR_VAL W::RED) :THEME y)
```

Comparatives and Superlatives

The comparative and superlative adjectives all refer to scalar operations on sequences ordered according to the scalar adjective. Thus, while *cheap* is an adjective that identifies an abstract location on a scale, say LF::COST, *cheaper* relates to objects on the LF::COST scale, and *cheapest* identifies the object on with the minimum cost. Note that *expensive* might refer to the same scale LF::COST, where the superlative, *most expensive*, refers to the object with the maximum cost.

The LF form captures these distinctions using a general ordering and maximizing/minimizing relation types. Each adjectives is classified in the lexicon by its direction on the scale (i.e., LESS adjectives produce a less-than relation for the comparative and selects the least value for the superlative), while MORE adjectives produce a greater-than relation for the comparative and a maximum value for the superlative.

The cheap computer --

```
(LF::THE C1 LF::COMPUTER :MODS (M1))
(LF::RELN (* LF::COST W::CHEAP) :of C1)
```

The cheaper computer --

```
(LF::THE C1 LF::COMPUTER :MODS (M1))
(LF::RELN (* LF::COMPAR-VALUE W::CHEAP)
:direction LESS :funct LF::COST :figure C1)
```

A computer cheaper than \$1000 --

```
(LF::THE C1 LF::COMPUTER :MODS (M1))
(LF::RELN (* LF::COMPAR-VALUE W::CHEAP)
 :direction LESS :funct LF::COST :figure C1 :ground C2)
(LF::THE C2 (SET-OF LF::COMPUTER) :size 2)
```

The cheapest computer --

```
(LF::THE C1 LF::COMPUTER :MODS (M1))
(LF::RELN (* LF::MAX-VAL W::CHEAP) :functn (* LF::COST-VAL W::CHEAP)
 :figure C1)
```

The cheapest of the computers --

```
(LF::THE C1 LF::COMPUTER :MODS (M1))
(LF::RELN (* LF::MAX-VALUE W::CHEAP) :funct (* LF::COST-VAL W::CHEAP)
 :figure C1 :ground C2)
(LF::THE C2 (SET-OF LF::COMPUTER))
```

Quantifiers *Note: this is under development and not always in agreement with current version of the parser*

Quantified phrases are complex, and our goal is to produce a logical form that retains enough the information, while not committing to a particular semantic treatment. We introduce a new term constructor, QUANTIFIER, which uses a special slot QUAN to specify the quantifier. For the simplest cases, the sense of the quantifier is placed in the QUAN slot, as in

All trucks ----

```
(LF::QUANTIFIER x (SET-OF (* LF::VEHICLE W::TRUCK)) :QUAN W::UNIVERSAL)
```

Quantifiers can get complicated, however, and for those we have a new term constructor LF::QSPEC which uses the quantifier in the “type” position and may have a role called the RANGE that specifies an expression that indicates the “reference set” from which the objects are drawn. For example,,

Most of the trucks ----

```
(LF::QUANTIFIER x (SET-OF (* LF::VEHICLE W::TRUCK)) :QUAN (Q1))
(LF::QSPEC Q1 W::MOST :RANGE V33)
(LF::THE V33 (SET-OF (* LF::VEHICLE W::TRUCK)))
```

Quantifiers may also take modifiers, that are listed in a MOD slot:

Almost all trucks ----

```
(LF::QUANTIFIED-TERM x (SET-OF (* LF::VEHICLE W::TRUCK)) :QUAN (Q1))
(LF::QSPEC Q1 W::ALL :MOD (* LF::QMODIFIER W::ALMOST))
```

Quantifiers can get arbitrarily complicated (e.g., Most but not all, more than seven and less than ten). We do not attempt any deep analysis in the LF, but just list them as modifiers.

Most but not all trucks ----

```
(LF::QUANTIFIER x (SET-OF (* LF::VEHICLE TRUCK) ) :QUAN (Q1 Q2))
(LF::QSPEC Q1 :QUAN W::MOST)
(LF::QSPEC Q2 :QUAN W::ALL :MOD (* LF::QMODIFIER W::NOT))
```

More (or Less) in Specifier Constructions

The word “more”, besides constructing comparatives, also serves in quantified constructions. There are several different interpretations that can be obtained:

Quantity Construction: This construction denotes a number constrained to larger (or smaller) number, as in *More than two trucks arrived*, which indicates a number of trucks greater than 2 arrived (similarly, we have *fewer than three trucks*).

More than two trucks ---

```
(LF::A V8470 (LF::SET-OF (:* LF::LAND-VEHICLE W::TRUCK)) :SIZE V8441)
(LF::QUANTITY-TERM V8441 LF::NUMBER :MODS (V8464))
(LF::F V8464 (:* LF::QMODIFIER W::MORE) :FIGURE V8441 :GROUND V8452)
(LF::QUANTITY-TERM V8452 LF::NUMBER :VALUE 2)
```

Quantifier Construction (a.k.a. In-addition-to): The construction modifies a set that is in addition to some other set (either explicit or implicitly defined). For example, *more trucks arrived*, we are talking about some trucks in addition to some already known trucks that arrived. We handle this using a quantifier sense of more. This can be modified with exact quantities, as in *Two more trucks arrived*, and *Two more of the red trucks arrived*.

Two more trucks --

```
(LF::QUANTIFIER V9640 (LF::SET-OF (:* LF::LAND-VEHICLE W::TRUCK))
:QUAN W::MORE :SIZE V9602)
(LF::QUANTITY-TERM V9602 LF::NUMBER :VALUE 2)
```

Whenever we have a *than* complement, we have the numeric interpretation. For instance, in *I have two more trucks than that*, we are talking about a number of trucks. And in the slightly awkward *two more trucks than that arrived*, we appear to be talking about a quantity of trucks, not additional trucks.

WH Terms

Wh-terms such as where, when, how, and so on play a central role in questions, and also appear in the complements of verbs like know, as in *I know where the truck is*.

For questions, the wh-terms appear in the LF using the WH-TERM constructor. For example

What's your plan

```
(LF::SPEECHACT V12087 W::SA_WH-QUESTION :FOCUS V14 :CONTENT V18)
(LF::F V18 (:* LF::IN-RELATION W::BE) :CO-THEME V11 :THEME V14
:TMA ((W::TENSE W::PRES)))
(LF::THE V11 (:* LF::PLAN-OBJECT W::PLAN) :POSS-BY V16)
(LF::PRO V16 (:* LF::PERSON W::YOUR) :CONTEXT-REL W::YOUR)
(LF::WH-TERM V14 (:* LF::REFERENTIAL-SEM W::WHAT))
```

For some wh-terms, like where, when and how, the question LF is captured with both a modifying relation and a wh-term. Thus “Where can we treat him” has an LF that is equivalent to “At what location can we treat him”.

Where was he treated

```
(LF::SPEECHACT V9120 W::SA_WH-QUESTION :FOCUS V9073 :CONTENT V9059)
(LF::WH-TERM V9073 LF::LOCATION :SUCHTHAT V9026 :CONTEXT-REL W::WHERE)
```

(LF::F V9026 (:* LF::SPATIAL-LOC W::WHERE) :OF V9059 :VAL V9073)
 (LF::F V9059 (:* LF::CURE W::TREAT) :THEME V9055 :MODS (V9026) :TMA ((W::TENSE W::PAST)
 (W::PASSIVE +)))
 (LF::PRO V9055 (:* LF::PERSON W::HE) :CONTEXT-REL W::HE)

WH-NP's as Complements

When wh-terms are used as complements to verbs like know or find out, we treat them as wh-terms descriptions with a special role :suchthat

I know what arrived.

(LF::F V21028 (:* LF::FAMILIAR W::KNOW) :THEME V21072 :COGNIZER V20915)
 (LF::WH-TERM V21072 (:* LF::REFERENTIAL-SEM W::WHAT) :CONTEXT-REL
 W::WHAT :SUCHTHAT V21086)
 (LF::F V21086 (:* LF::ARRIVE W::ARRIVE) :THEME V21072)
 (LF::PRO V20915 (:* LF::PERSON W::I) :CONTEXT-REL W::I)

Show me where the car stopped.

(LF::F V8536 (:* LF::SHOW W::SHOW) :ADDRESSEE V8555 :THEME V8559 :AGENT V8696))
 (LF::PRO V8555 (:* LF::PERSON W::ME) :CONTEXT-REL W::ME)
 (LF::WH-TERM V8559 (:* LF::SPATIAL-LOC W::WHERE) :SUCHTHAT V8607)
 (LF::F V8607 (:* LF::STOP-MOVE W::STOP) :THEME V8589 :TMA ((W::TENSE W::PAST)))
 (LF::THE V8589 (:* LF::LAND-VEHICLE W::CAR))
 (LF::IMPRO V8696 LF::PERSON :CONTEXT-REL W::*YOU*)

Mass Terms

Mass terms, such as "sand", have different properties than count nouns such as "truck". Whereas the LF type associated with truck, LF::TRUCK, can be viewed as a predicate that is true of any object that is a truck, it is not clear what the predicate LF::SAND is true of. The approach we take is that LF::SAND is true of any object that consists of the substance sand. Thus, a definite description "The sand" refers to some object that consists of sand, such the beach we are talking about. The indefinite form for mass terms refers to substances, and we use the term constructor LF::MASS.

The water

(LF::THE V8460 (:* LF::FOOD W::WATER))

Beer

(LF::MASS y (:* LF::FOOD LF::BEER))

Some beer

(LF::A V8499 (:* LF::FOOD W::BEER) :QUAN W::SOME)

NEEDS WORK: Note that some mass terms, like beer, can be coerced into other forms, like objects (as in a bottle of beer) and types of objects (I like many beers). Currently, we allow the indefinite count specifier LF::A for these expressions.

A beer

(LF::A y (:* LF::FOOD LF::BEER))

Beers

(LF::A y (SET-OF (* LF::FOOD LF::BEER)))

Bare Nouns

<<>>

Special Contextual Term**Demonstratives**

Demonstratives are treated as definite descriptions and the lexical item is placed in the CONTEXT-REL relation as done with pronouns. This allows reference resolution processes to have strategies specific for each word. For example

These trucks

(LF::THE V8288 (LF::SET-OF (* LF::LAND-VEHICLE W::TRUCK)) :CONTEXT-REL W::THESE)

That other truck

(LF::THE V8240 (* LF::LAND-VEHICLE W::TRUCK) :CONTEXT-REL W::THAT :MODS (V8235))
(LF::F V8235 (* LF::IDENTITY-VAL W::OTHER) :THEME V8240)

One

One used as a head noun indicates no restriction on the LF type, and the one is placed in the CONTEXT-REL to enable special referential processing.

The red one

(LF::THE x (* LF::ANY-SEM ONE) :MODS c1 :CONTEXT-REL ONE)
(LF::F c1 (*LF::COLOR-REL RED) :THEME x)

Nominalizations and Related Forms

Most verbs support forms that make various noun phrases. When a specifier is explicit we build the obvious form. For example

The destruction of the city

(LF::THE d1 LF::DESTROY :THEME c1)
(LF::THE c1 LF::CITY)

Gerund forms (e.g., destroying the city) also are referring expressions, though getting the right specifier is tricky. In "I liked destroying the city" we seem to have a definite reference, whereas "I like destroying the city" we seem to have a generic construction. We use the BARE term constructor for such forms. The exact nature of the expression will be determined by contextual interpretation.

I like watching the movie

(LF::F V10713 (* LF::EXPERIENCER-EMOTION W::LIKE) :THEME V10725 :EXPERIENCER V10618)
(LF::KIND V10725 (* LF::ACTIVE-PERCEPTION W::WATCH) :THEME V10758)
(LF::THE V10758 (* LF::ENTERTAINMENT W::MOVIE))
(LF::PRO V10618 (* LF::PERSON W::I) :CONTEXT-REL W::I)

Enumerated Sets

Conjoined noun phrases require the construction of a set of objects. A set is constructed for the conjoined phrase that uses the special role :MEMBERS to list the items involved. For example.

The large truck and small car

```
(LF::THE V27568 (LF::SET-OF LF::LAND-VEHICLE) :MEMBERS (V27640 V27560))
(LF::BARE V27640 (* LF::LAND-VEHICLE W::CAR) :MODS (V27612))
(LF::F V27612 (* LF::SIZE-VAL W::SMALL) :OF V27640)
(LF::THE V27560 (* LF::LAND-VEHICLE W::TRUCK) :MODS (V27557))
(LF::F V27557 (* LF::SIZE-VAL W::LARGE) :OF V27560)
```

4. Quantity Terms

There are a wide range of constructions that refer to quantities, and these constructions also interact with quantifiers. In this section we discuss the handling of quantities in detail. Numbers are a core case of quantity terms, but we also have constructions referring to quantities in terms of units (e.g., *three pounds*) and other constructions.

4.1 Numeric Quantities

Numbers are the most basic QUANTITY-TERMs. The LF of an expression like *five* is not simple an atom such as 5 because we need to handle modifiers, as in at least five, a few hundred, and not more than seven. These express constraints of values and can appear almost anywhere a simple number may. The system provides a capability to simplify purely numeric expressions into conventional form.

Five

```
(LF::QUANTITY-TERM V10665 LF::NUMBER :VALUE 5)
```

Numeric expressions that involve units, like hundred, thousand, are collapsed whenever possible to their mathematical representation. In certain case, such as *many hundred*, we must retain the linguistic form as discussed in the next section.

Five hundred

```
(LF::QUANTITY-TERM V12644 LF::NUMBER :VALUE 500)
```

Three hundred and five

```
(LF::QUANTITY-TERM V27178 LF::NUMBER :VALUE 305)
```

Two thousand three hundred and five

```
(LF::QUANTITY-TERM V27325 LF::NUMBER :VALUE 2305)
```

Many of the quantifiers in language primarily serve to give an approximate quantity. Thus we represent them as quantities rather than as the true quantifiers such as *all*, *every* and *no*. Here are some of the most common forms. Some, like *several* and *many*, indicate vague quantities.

Several

```
(LF::QUANTITY-TERM V5555 LF::NUMBER :VALUE W::SEVERAL)
```

Many

```
(LF::QUANTITY-TERM V5555 LF::NUMBER :VALUE W::MANY)
```

A few

```
(LF::QUANTITY-TERM V5555 LF::NUMBER :VALUE W::A-FEW)
```

These words can also be combined with number units, in which case

Several hundred

(LF::QUANTITY-TERM V21475 (* LF::NUMBER-UNIT W::HUNDRED) :VALUE V21472)

(LF::QUANTITY-TERM V21472 LF::NUMBER :VALUE W::SEVERAL)

Many thousand

(LF::QUANTITY-TERM V11262 (* LF::NUMBER-UNIT W::THOUSAND) :VALUE W::MANY)

Quantities are often also expressions as constraints of values rather than a specific value. We handle these cases by adding modifiers on the quantity terms as follows. For the binary numeric operators we use two roles: :FIGURE indicate the number being modified, while :GROUND is the number to which it is being compared. Note that the ground may not necessarily be a concrete number.

At least five

(LF::QUANTITY-TERM V12345 LF::NUMBER :MODS (M12428))

(LF::F M12428 (* LF::QMODIFIER W::MIN) :FIGURE V12345 :GROUND V12388)

(LF::QUANTITY-TERM V12388 LF::NUMBER :VALUE 5)

Approximately five

(LF::QUANTITY-TERM V12303 LF::NUMBER :MODS (M12344))

(LF::F M12344 (* LF::QMODIFIER W::APPROXIMATE) :FIGURE V12303 :IS GROUND)

(LF::QUANTITY-TERM V12309 LF::NUMBER :VALUE 5)

At most a few

(LF::QUANTITY-TERM V11364 LF::NUMBER :MODS (V11395))

(LF::F V11395 (* LF::QMODIFIER W::MAX) :FIGURE V11364 :GROUND V11386)

(LF::QUANTITY-TERM V11386 LF::NUMBER :VALUE W::A-FEW)

Greater than eight

(LF::QUANTITY-TERM V12050 LF::NUMBER :MODS (V12066))

(LF::F V12066 (* LF::QMODIFIER W::MORE) :FIGURE V12050 :GROUND V12060)

(LF::QUANTITY-TERM V12060 LF::NUMBER :VALUE 8)

Less than that

(LF::A V12219 LF::NUMBER :MODS (V8145))

(LF::F V8145 (* LF::QMODIFIER W::LESS) :GROUND V12193 :FIGURE V12219)

(LF::PRO V12193 (* LF::REFERENTIAL-SEM W::THAT))

4.2 Quantity Terms

There are many expressions of measurement in language, that combine a numerical quantity and some unit on a scale. This includes expressions such as three miles, many liters, several hours, eight days, and so on. These also map to terms that use the term constructor QUANTITY-TERM, as in

Three miles

(QUANTITY-TERM V25 (* LF::LENGTH-UNIT W::MILE) :VALUE V1)

(QUANTITY_TERM V1 LF::NUMBER :VALUE 3)

Several pounds

(LF::QUANTITY-TERM V26829 (:* LF::WEIGHT-UNIT W::POUND) :VALUE V26826)
 (LF::QUANTITY-TERM V26826 LF::NUMBER :VALUE W::SEVERAL)

At least ten dollars

(LF::QUANTITY-TERM V27069 (:* LF::MONEY-UNIT W::DOLLAR) :VALUE V27025)
 (LF::QUANTITY-TERM V27025 LF::NUMBER :MODS (M27106))
 (LF::F M27106 (:* LF::QMODIFIER W::MIN) :OF V27025 :IS V27068)
 (LF::QUANTITY-TERM V27068 LF::NUMBER :VALUE 10)

Quantity terms may take arguments that become the main semantic component of the phrase. For instance, *three gallons of water* is a quantity of water, not a watery set of gallons! These phrases can describe amounts of substances or sets of objects depending on the type of thing they modify. Here are a few examples:

Three gallons of the water

(LF::A V10131 (:* LF::FOOD W::WATER) :QUANTITY V10087 :SUBPART-OF V10096)
 (LF::QUANTITY-TERM V10087 (:* LF::VOLUME-UNIT W::GALLON) :VALUE V10086)
 (LF::QUANTITY-TERM V10086 LF::NUMBER :VALUE 3)
 (LF::THE V10096 (:* LF::FOOD W::WATER))

Several groups of people

(LF::A V10206 (LF::SET-OF (:* LF::PERSON W::PERSON)) :QUANTITY V10175 :SUBSET V10185)
 (LF::QUANTITY-TERM V10175 (:* LF::GROUP-UNIT W::GROUP) :VALUE V10172)
 (LF::QUANTITY-TERM V10172 LF::NUMBER :VALUE W::SEVERAL)
 (LF::A V10185 (LF::SET-OF (:* LF::PERSON W::PERSON)))

Three inches of snow

(LF::A V10458 (:* LF::PRECIPITATION W::SNOW) :QUANTITY V10431 :SUBPART-OF V10439)
 (LF::QUANTITY-TERM V10431 (:* LF::LENGTH-UNIT W::INCH) :VALUE V10430)
 (LF::QUANTITY-TERM V10430 LF::NUMBER :VALUE 3)
 (LF::BARE V10439 (:* LF::PRECIPITATION W::SNOW))

Two liters water

(LF::A V10538 (:* LF::FOOD W::WATER) :QUANTITY V10534)
 (LF::QUANTITY-TERM V10534 (:* LF::VOLUME-UNIT W::LITER) :VALUE V10533)
 (LF::QUANTITY-TERM V10533 LF::NUMBER :VALUE 2)

5. Times and Locations

Temporal Objects

Temporal expressions fall into two categories, those describing particular times according to some clock system (e.g., the time of day), and those describing durations of time (e.g., the length of a movie). Here we describe the first use, reference to temporal “locations”. Temporal durations are handled in the next section as an example of quantity terms.

Clock-times are often underspecified and require contextual processing. We say Saturday but don't mention which Saturday is meant, or 3 PM without mentioning the day. These terms serve to constrain the range of possible times that could be intended, and contextual interpretation

would identify the intended one. For LF::TIME-LOC we have a slot for each word that describes a clock-time attribute, including HOUR, MINUTE, SECOND, WEEKDAY (Monday, Tuesday, ...), TIME-OF-DAY (morning, evening, ...), MONTH (January, February, ...), AM-PM, DAY-OF-MONTH, YEAR. Thus we have the example

Monday July 4

```
(LF::THE V3 LF::TIME-LOC
      :DAY 4
      :MONTH (* LF::MONTH-NAME JULY)
      :DAY-OF-WEEK (* LF::DAY-NAME MONDAY)))
```

Five PM

```
(LF::THE V7039 LF::TIME-LOC :AM-PM (* LF::TIME-OBJECT W::PM) :HOUR 5)
```

The value of AM-PM could be W::AM, W::PM or W::INTL (for 24 hΣour international time).

Cardinality of Sets

Numbers and other quantities are also common in descriptions of sets of objects. These are represented by using a role :SIZE that can appear with any object that is a set type. Note that the size can be a symbol value as well, typically identifying an imprecise bound.

Five ideas

```
(LF::A V16777 (SET-OF (* LF::MENTAL-OBJECT W::IDEA)) :SIZE V16776)
(LF::QUANTITY-TERM V16776 LF::NUMBER :VALUE 5)
```

Several reasons

```
(LF::A V6906 (SET-OF (* LF::FACT W::REASON)) :SIZE V6903)
(LF::QUANTITY-TERM V6903 LF::NUMBER :VALUE W::SEVERAL)
```

Many thousand trucks

```
(LF::A V16741 (SET-OF (* LF::LAND-VEHICLE W::TRUCK)) :SIZE V16738)
(LF::QUANTITY-TERM V16738 (* LF::NUMBER-UNIT W::THOUSAND) :QUAN V16735)
(LF::QUANTITY-TERM V16735 LF::NUMBER :VALUE W::MANY)
```

5. Tense, Aspect and Modal Verbs

There is information expressed in the lexical and structural forms that identify tense, aspect and modality, that have not yet been expressed in the logical form language. All this information is important for contextual interpretation, and meaning so we package it up in a special structure that is the value of a new role :TMA, which can only appear in "F" structures. These TAM structures have the form (tense g-aspect l-aspect modality), where each position takes only a few values (where - indicates the value is unspecified):

TENSE - the tense derived form the verb, e.g., pres, past
 PROGR – a binary feature with + indicating the progressive
 PERF – a binary feature with + indicating the perfective
 NEGATION – a binary feature with + indicating negation

I want ice

```
(LF::F I1 (* LF::WANT W::WANT) :EXPERIENCER) i1 :THEME ice1 :TMA ((TENSE PRES))
(PRO i1 (* LF::PERSON W::I) :CONTEXT-REL W::I)
```

(BARE ic1 (:* LF::SUBSTANCE W::ICE))

I had not seen the ice

(LF::F V0 (:* LF::ACTIVE-PERCEPTION W::SEE) :THEME V3 :EXPERIENCER V8
:TMA ((TENSE PAST) (PERF +) (NEGATION +)))

(PRO V8 (:* LF::PERSON W::I) :CONTEXT-REL W::I)

(LF::THE V3 (:* LF::SUBSTANCE W::ICE))

Modal Auxiliaries

Modals also appear in the TMA structure in special slot MODALITY. The values of this feature range over the possible auxiliaries, clustered slightly based on their semantic meaning:

LF::ABILITY	W::CAN
LF::EMPHASIS	W::DO
LF::OBLIGATION	W::MUST W::SHOULD
LF::FUTURE	W::WILL, W::SHALL
LF::NECESSITY	W::HAVE
LF::POSSIBILITY	W::MAY, W::MIGHT
LF::CONDITIONAL	W::COULD, W::WOULD
LF::GOING-TO	W::GONNA, W::GOING-TO

Figure 5: The Modal Auxiliaries

I should have gone

(LF::F V20661 (:* LF::MOVE W::GO) :THEME V20496
:TMA ((W::TENSE W::PRES) (W::MODALITY (:* LF::OBLIGATION W::SHOULD)) (W::PERF +)))

(LF::PRO V20496 (:* LF::PERSON W::I) :CONTEXT-REL W::I)

I can't see it

(LF::F V57 (:* LF::ACTIVE-PERCEPTION SEE) :THEME V58 :EXPERIENCER V55
:TMA ((TENSE PRES) (MODALITY (:* LF::ABILITY CAN)) (NEGATION +)))

Elided Forms

Because we treat auxiliaries as augmentations to main verbs, we need a special treated of elided forms. We introduce a special LF for the elided verb phrase. This would serve as a signal to

I can't

(LF::F V6 (:* LF::ELLIPSIS) :THEME V2 :TMA ((TENSE PRES) (MODALITY (:* LF::ABILITY
W::CAN)) (NEGATION +)))

Main verb be

The main verb be has three main forms. The first use, LF::HAVE-PROPERTY, associates an object with a property which is realized by an adjective, PP, or other predicative form:

It was red

(LF::F h1 (:* LF::HAVE-PROPERTY W::BE) :THEME it1 :PROPERTY p1)
(LF::F p1 (:* LF::COLOR-REL W::RED) :LF::OF it1)

It is in the truck

(LF::F h1 (:* HAVE-PROPERTY W::BE) :THEME it1 :PROPERTY p1)
 (LF::F p1 (:* LF::SPATIAL-LOC W::IN) :LF::OF it1 :LF::VAL a1)
 (LF::THE a1 (:* LF::VEHICLE W::TRUCK))

The man was late

(LF::F h2 (:* LF::HAVE-PROPERTY W::BE) :THEME m2 :PROPERTY p2)
 (LF::THE m2 LF::MAN)
 (LF::F p2 (:* LF::EVENT-REL-TIME W::LATE) :LF::OF m1)

The second sense of be indicates a relationship between objects, which often is equality, but also might involve some contextually-defined relations. For instance, the utterance Three miles is four hours states that the relation “time to travel” relates three miles to four hours. The predicate LF::IN-RELATION is used for this, and the most uses of these sense involve be with an NP complement.

It is the best truck.

(LF::F be1 (:* LF::IN-RELATION BE) :THEME h1 :CO-THEME b1)
 (PRO h1 (:* LF::REFERENTIAL-SEM IT) :CONTEXT-REL IT)
 (LF::THE b1 (:* LF::VEHICLE TRUCK) :MODS (bd1))
 (LF::F bd1 (:* LF::QUALITY BEST) :LF::OF b1)

Three miles is four hours

(LF::F be3 (:* LF::IN-RELATION BE) :THEME h1 :CO-THEME b1)
 (LF::A V78 (SET-OF (:* LF::DISTANCE-UNIT MILE)) :QUAN 3)
 (LF::A V79 (SET-OF (:* LF::TIME-UNIT HOUR)) :QUAN 2)

The third sense is existence, and is typically seen in utterances like “there is the truck”.

There is a person with a broken leg

(LF::F V45 LF::EXISTS :ENTITY V46 :TMA ((TENSE PRES)))
 (LF::A V46 (:* LF::PERSON PERSON) :MODS (m1))
 (LF::F m1 (:* LF::ASSOC-WITH WITH) :LF::OF V46 :LF::VAL V47)
 (LF::A V47 (:* LF::BODY-PART LEG) :LF::OF m1 :MODS (b1))
 (LF::F b1 (:* LF::IN-WORKING-ORDER BROKEN) :LF::OF V47)

Main verb have

There are a number senses of the verb have. As shown in Figure 6.

Sense	Comment	Roles	Examples
LF:HAVE	Abstract possession	:CO-THEME :THEME	I have a car
LF::HAVE-PROPERTY	Relates an object to some situation	:THEME :PROPERTY	We have an emergency, I have a headache, The truck had a problem
LF::MAKE-IT-SO	An agent causing some situation	:AGENT :THEME	I had Fred go, They had the files destroyed
LF::CONSUME	An agent consumes some substance	:AGENT :THEME	I had some water, We had fish for dinner
LF::NECESSITY	An agent is required to do something	:THEME :ACTION	I had to go, The files had to be destroyed.

Figure 6: The senses of “have”

Here are a couple of examples uses some of the senses:

I have a car

(LF::F V444588 (:* LF::HAVE W::HAVE) :CO-THEME V44 :THEME V45
:TMA ((TENSE PRES)))

(PRO v44 (:* LF::PERSON W:I) :CONTEXT-REL W::I)

(LF::A v45 (:* LF::VEHICLE W::CAR))

We have an emergency

(LF::F h1 (:* LF::HAVE-PROPERTY HAVE) :THEME we1 :PROPERTY ha1)

(PRO i1 (SET-OF (:* LF::PERSON W:WE)) :CONTEXT-REL W::we)

(LF::A ha1 (:* LF::EVENT W::EMERGENCY))

I had Fred go

(LF::F mis1 (:* LF::MAKE-IT-SO W::HAVE) :AGENT i1 :THEME I1 :TMA ((TENSE PAST)))

(PRO i1 LF::PERSON :CONTEXT-REL I)

(LF::F I1 (:* LF::MOVE GO) :AGENT p2)

(LF::THE p2 LF::PERSON :NAME-OF FRED)

I had some water

(LF::F mis1 (:* LF::CONSUME HAVE) :AGENT i2 :THEME V45 :TMA ((TENSE PAST)))

(PRO i2 LF::PERSON :CONTEXT-REL I)

(A v45 (:* LF::FOOD WATER))