Machine Translation, 10:1-2, 139–180 (1995)

Toward a Lexicalized Grammar for Interlinguas

CLARE VOSS voss@cs.umd.edu
BONNIE J. DORR bonnie@cs.umd.edu
Department of Computer Science, University of Maryland, College Park, MD 20742

Received September 1,1994; Revised July 15, 1995

Abstract. In this paper we present one aspect of our research on machine translation (MT): capturing the grammatical and computational relation between (i) the interlingua (IL) as defined **declaratively** in the lexicon and (ii) the IL as defined **procedurally** by way of algorithms that compose and decompose pivot IL forms. We begin by examining the interlinguas in the lexicons of a variety of current IL-based approaches to MT. This brief survey makes it clear that no consensus exists among MT researchers on the level of representation for defining the IL. In the section that follows, we explore the consequences of this missing formal framework for MT system builders who develop their own lexical-IL entries. The lack of software tools to support rapid IL respecification and testing greatly hampers their ability to modify representations to handle new data and new domains. Our view is that IL-based MT research needs both (a) the formal framework to specify possible IL grammars and (b) the software support tools to implement and test these grammars. With respect to (a), we propose adopting a **lexicalized grammar** approach, tapping research results from the study of *tree grammars* for natural language syntax. With respect to (b), we sketch the design and functional specifications for parts of ILustrate, the set of software tools that we need to implement and test the various IL formalisms that meet the requirements of a lexicalized grammar. In this way, we begin to address a basic issue in MT research, how to define and test an interlingua as a computational language — without building a full MT system for each possible IL formalism that might be proposed.

Keywords: interlingua, machine translation, lexicon, lexicalized grammar

1. Introduction

In this paper we present one aspect of our research on machine translation (MT): capturing the grammatical and computational relation between (i) the interlingua (IL) as defined **declaratively** in the lexicon and (ii) the IL as defined **procedurally** by way of algorithms that compose and decompose pivot IL forms.

The distinction between (i) and (ii) corresponds to the well-known software engineering principle of separating *data* and *algorithms* in system designs. The distinction also shows up in linguistic analyses in the contrast between constraints on *representations* and constraints on *derivations.* We examine here the need to extend this underlying principle to the process of constructing an interlingua *explicitly,* in terms of its definition in the lexicons and pivot algorithms, rather than *implicitly,* as is currently done. We explore both the motivation and the groundwork needed for developing a theory of the interlingua in formal terms.[1]

We begin by examining the interlinguas in the lexicons of a variety of current IL-based approaches to MT. Even with the review limited to a brief survey of lexical

IL forms, the overall finding is clear: no consensus exists among MT researchers for defining the Lexical Component of an IL or its equivalent with respect to the levels of representation in an MT system.

In the section that follows, we explore the consequences of this missing formal framework for MT system builders who develop their own lexical IL entries. We describe two issues common to all approaches: first, determining the coupling of the IL representation language to other representation languages in the MT system, and second, scaling up the IL representation language. With respect to coupling, we present a few examples of variants on an IL formalism and point out that these changes require adjusting the mapping between the IL and syntactic argument structure. With respect to scaling, we find the complexity arises from the problem of *interlocking* — the difficulty introduced by interactions between the declarative and procedural components of the IL when a change to one component drastically affects the functionality of the other, and vice versa.

Our view is that IL-based MT research needs both (a) the formal framework to specify possible IL grammars and (b) the software tools to implement and test these grammars (Dorr and Voss, 1994). Software tools developed in conjunction with a formal framework would support rapid IL respecification and testing and would facilitate the construction of ILs to handle new data and new domains.

With respect to (a), we propose a **lexicalized grammar** approach, tapping research results from the study of *tree grammars* for natural language syntax. Our approach defines the IL as a formal, computational language with two separate components. First, the *Lexical Component,* the finite set of IL forms in the entries of language-specific lexicons, is defined in terms of *tree languages.* Second, the *Pivot Form Component,* the algorithms that compose and decompose the pivot IL forms, is defined in terms of tree transformation operations.

With respect to (b), we sketch the design and functional specifications for parts of ILustrate, the set of software tools needed to implement and test the various IL formalisms that meet the requirements of a lexicalized grammar. It is expected that this tool would aid IL developers in analyzing and implementing the approaches covered in the survey section of this paper. Ultimately, as the tool is further enhanced, we expect to be able to use it for other IL approaches.

In this way, we begin to address a basic issue in MT research — how to define and test an interlingua as a computational language without building a full MT system for each possible IL formalism that might be proposed.

2. Variation Across Approaches

In this section we focus on the declarative portion of the IL i.e., the Lexical Component in IL-based MT systems. We examine five current research approaches to the level of representation for the lexical IL forms: lexical-textual, lexical-ontological, lexical-semantic, lexical-syntactic, and tiered.[2] All five approaches push the limits of two traditional assumptions implicit in IL research.

The first assumption is that the lexical IL forms that feed into the Pivot-Form Component exist at one predefined depth, or level of representation, beyond which further analysis does not occur. (Indeed in the classic transfer and IL "pyramid" diagram in (Hutchins and Somers, 1992, p. 107), one can even "see" this depth of analysis metaphor.) The second implicit assumption is that adequate translation is achieved only through exhaustive coverage at a single level of representation (Nirenburg et al., 1992).

2.1. Lexical-Textual IL Forms

In the MT system Mikrokosmos, lexical entries are subdivided into three zones that correspond to syntactic, semantic, and text meaning representation (TMR) information (Levin and Nirenburg, 1994; Onyshkevich and Nirenburg, 1995). We will explore here briefly the TMR, which corresponds to the Lexical Component of the interlingua, i.e., it is the formal basis for the interlingua in Mikrokosmos. The TMR defines the acceptable lexical IL forms (or lexical-textual forms) and, via composition and decomposition of those forms, it also defines the full range of pivot forms that may appear during translation.

The unique characteristic of the TMR-based interlingua is that it is a collection of *microtheories* of meaning. These microtheories include meaning facets such as aspect, modality, evidentiality, speech acts, reference, speaker attitudes, stylistics factors, and temporal relations, as well as a "who did what to whom" component of meaning. These microtheories or components of meaning, when taken together, give the TMR-based IL its expressive strength. This approach challenges the assumption that meaning ought to be stratified, derived from one level only to the next. What we see here is that various sorts of knowledge can be defined, composed, and decomposed within the same formalism.

What is less clear, however, is the way these microtheories are defined at the lexical level. We can ask within this framework, for example, whether microtheories have Lexical Components of their own. If they do, or of those that do, what determines the microtheory-based meaning that a particular lexical item contributes to pivot IL forms? (Along the same lines, we could ask how the Lexical Component hands over to the Pivot Form Component the job of coercing or construing a microtheory-based meaning component.) The relevant unit of analysis within this framework may not be the lexical entry's full TMR zone, but rather the clause or component within the zone that occurs there by virtue of contributing to one (or perhaps more) microtheory.

2.2. Lexical-Ontological IL Forms

Among AI researchers working on multilingual and MT systems, one of the most strikingly nonminimal approaches to lexical IL representations built in an ontological or conceptual framework is the current work of DiMarco, Hirst, and Stede

(1993); Stede (1993); Hirst (1995). Their approach has been to develop two-part definitions by splitting lexical meaning along two levels of representation. The first is a *conceptual* level for meaning components that are language-independent, i.e., interlingual, configurations of concepts, roles and associated fillers. The second is a *linguistic* level for meaning components that are language-specific structures and features tuned to capture fine connotational and denotational distinctions. The conceptual components are stored in a KL-ONE-style taxonomic knowledge base (a KB with pointers from the lexical entries), and the linguistic components are stored in the relevant lexical entries.

Applying a two-component view of the IL to this research, we can see that the ontology is an indirect way of building a relational IL lexicon — the lexical IL forms are placed in well-defined relations to one another, not directly within the MT lexicon data structures themselves, but rather in the KB. This is a first step in isolating the Lexical Component of the IL. The single ontology containing all the lexical IL forms presents a framework for exploring the space of lexical IL forms, a prerequisite to a formalization of the Lexical Component.

This current research is distinguished from the other AI approaches to MT research in that it takes seriously the challenge of separating the conceptual ontology from linguistic knowledge. It does not assume that the conceptual relations in the KB are identical with thematic roles needed in lexical semantics. Rather it uses cross-linguistic semantic data, such as incorporated meanings, to guide the representations; and it specifically addresses the problem of nonredundancy (as happens when MT developers scale up their systems without being able to check if they have created a nonredundant set of IL primitives).

2.3. Lexical-Semantic IL Forms

The MT system UNITRAN (Dorr, 1990; Dorr, 1993) takes the theory of (Jackendoff, 1983; Jackendoff, 1990) for the interlingua. Specifically, Dorr developed a modified, computational version of Jackendoff's "lexical-conceptual structures" (LCSs) as the formalism for lexical IL forms and pivot IL forms.[3]

Although Jackendoff embedded his work in a psychological framework and has argued that his theory's semantic structures are conceptual structures (i.e., equating semantic and conceptual levels of representation), we note that these assumptions do not carry over to Dorr's work. UNITRAN assumes only that the LCS formalism provides a "syntax" or notation for encoding the lexical and sentential semantics, i.e., the interlingua. In other words, UNITRAN is theory-neutral with respect to the relation of semantics and conceptual structures.

Dorr's approach differs dramatically from the one discussed in Section 2.2. This work assumes that each individual lexical IL form is a (i) single, connected, annotated graph, that is a (ii) language-specific, (iii) semantic structure (iv) located in the lexicon. This is in marked contrast to DiMarco et al.'s two-part structures where the language-independent conceptual component is located in an ontology and the language-specific linguistic component is stored in the lexicon. Several

limitations to the Lexical Component in UNITRAN's interlingua are a function of the gaps in Jackendoff's theory that Dorr relied on. For example, lexical entries for quantifiers, *not, and,* pronouns, and (in)definite articles — all central to research in logical semantics — were not covered directly by Jackendoff.[4]

What distinguishes this research from previous efforts to build a semantic IL is that the lexical IL formalism is parameterized. This provides a constrained set of IL forms that map via simple linking rules to and from the syntactic structures within the MT system.

2.4. Lexical-Syntactic IL Forms

Recent work by Nomura, Jones, and Berwick (1994) has focused on the development of an interlingua at a lexical-syntactic level of representation. This work draws on the formal linguistic research of Hale and Keyser (1993) who use Lexical Relational Structures (LRSs) as the basis for lexical IL forms. One of the stated goals of this approach is to delimit the space of LRSs available for the Lexical Component of the IL. They criticize Dorr's use of LCS theory, for example, on the grounds that the space of LCSs is not constrained and so does not allow for a properly constrained mapping between sentential syntactic forms and full LCS-based IL pivot forms.

Nomura et al. attempt to show the value of their delimited LRS space in terms of a well-defined and constrained mapping between the LRSs and sentential syntactic structure. Indeed, within Hale and Keyser's work, the LRSs are also called "lexical syntactic structures" — making it clear that the broader shared research agenda is to push their current syntactic formalism down from the sentential level into the lexical level.

The formal linguistic focus of this new research, with its strong theory-internal guiding assumptions, clearly sets this work apart from the lexical research on interlinguas at the other levels of representation. These researchers will need to examine whether the interlocking problem will be significant for their framework. In particular, when the LRSs are instantiated as LRS-based pivot forms, it may be discovered that the analysis trees in Hale and Keyser's formal work need to be supplemented with derivation trees and possible subsequent revisions to the LRSs.

2.5. Tiered Lexical IL Forms

The tiered lexical IL forms in the work of Dorr and Voss (1993), and more recently in experimental form in (Dorr et al., 1994) reflect the trends mentioned above. This approach rejects the assumptions that (i) the interlingua exists at a specific level of representation and (ii) that adequate translation is achieved only through exhaustive coverage at that level.

With respect to the first assumption, the goal of the tiered model is to decouple the notions of an interlingua as a computational language and as a level of representation. Consequently, the tiered form contains information derived from several

levels of representation. For example, each constituent structure within a tiered form is *typed,* i.e., has a type value paired with it. The types are a small set of ontological categories from a *conceptual* level of representation.

For each predicator in a tiered form, there is an associated *semantic field,* such as a *locational, temporal,* or *possessional* field. These reflect the view that the patterns of lexical semantic structures in a variety of fields are shared because they are based on mappings from a few basic fields, such as location (Anderson, 1971; Jackendoff, 1983; Langacker, 1987; Talmy, 1985). Syntactic information is also encoded indirectly in the lexical forms. For each subcategorization frame where a verb appears, there is a distinct tiered lexical IL form. This mapping between frame and form indirectly preserves the information that is present in syntactic alternations through translation.

With respect to the second assumption, the tiered approach challenges the notion that all of the deepest, i.e., conceptual, knowledge is available in the interlingua. The underlying problem is partly theoretical and partly practical. Theoretically, the difficulty arises from the assumption that IL constituents are equivalent to conceptual categories. This implies incorrectly that the meaning of a sentence is a rich knowledge structure. If this were indeed the case, then what would be the basis for bounding that structure?[5] If we, as developers, are forced to design our IL representations on the basis of unbounded structures, we would face the practical limitation that no representation, even if called conceptual, would capture the full meaning of an item in an MT lexicon.

The central idea is that the depth of the IL representation in an MT application is constrained enough to overcome the practical concern described above.[6] For example, in this approach, a verb's lexical IL form is taken as a highly reduced, language-specific sketch of an event or state. The IL primitives within such forms are constants at the IL level. Their meaning is grounded in the *full,* i.e., nonsketch, version of events and states stored in a knowledge representation and reasoning (KRR) system within the MT system. The KRR system is then available at run-time (via the Pivot-Form Component) to help interpret those reduced sketches. In short, the primitives in the lexical IL forms for any one natural language comprise only a subset of the KRR's full set of concepts.[7]

3. Variation Within An Approach

The current state of the art in IL-based MT systems is that no consensus exists on delimiting the interlinguas being used. In the last section we surveyed the lexical IL forms in several IL-based approaches and found substantial variation among them. In this section we make the argument that, even *within* a particular IL approach, there will also be variation in IL forms at development time and at scaling time due to the open-endedness of IL theory on:

- the coupling of the IL to other components in the surrounding MT system
- the interactions between the declarative and procedural components of the IL

Within our framework approach, we distinguish two types of formal limitations on the scaling: (i) those built into the framework and (ii) those specified by an IL grammar. With respect to (i), the framework, as discussed later in the paper, places an outer bound on the IL syntax.[8] With respect to (ii), IL developers themselves (i.e., not us, the framework developers) specify the IL syntax for their set of IL forms. It is also their decision to change the IL syntax they have specified when scaling their MT system as required by cross-linguistic data. The IL developers working within this framework are the ones who define the set of "possible" IL forms.

3.1. Coupling of IL with Its Mapping to Syntax

Here we define a *coupling problem* that arises as the lexical IL forms or the algorithms for the IL pivot forms are revised to handle new data. First note that while an IL grammar specifies a set of acceptable IL forms, the specification is independent of the instantiation of the IL in an MT system. The grammar does not define the relations that, at MT runtime, map those IL forms to and from the other formalisms in the MT system. So we say that an IL formalism is coupled to its instantiation in an MT system because changes to the form of the IL will affect the mappings that must be instantiated between the IL and syntactic argument structure.

In this section we explore one way of experimenting with changes to an IL formalism and show an example where the coupling problem arises within a tiered approach (see Section 2.5). The bottom line of this exercise is straightforward — without a formal basis for defining the IL (be it in terms of a lexicalized grammar or otherwise), we cannot adequately:

- specify the mappings between the IL and other representation languages in an MT system
- compare such mappings across MT systems
- evaluate whether a particular mapping is predictive, i.e., yields the generalizations expected

Consider first the sentence *He put the socks in the drawer.* Within any of the approaches discussed in the last section, we will find that the verb *put* takes three syntactic arguments and that some set of procedures has been developed for converting each argument into its IL form and for mapping these forms into the lexical-IL form for *put.* For example, a one-to-one mapping from each syntactic argument into one position in the *put* lexical-IL form would result in the pivot-IL form in Figure 1.[9] The arguments corresponding to *he, the socks, the drawer* map to positions labeled i, j, and k, respectively. We read this figure, in a paraphrase, to mean, he caused the socks to go into the drawer.

Now consider what is involved when the verb in this sentence is changed to *stuffed,* in *He stuffed the socks in the drawer.* If we wish for the IL to capture the fact that

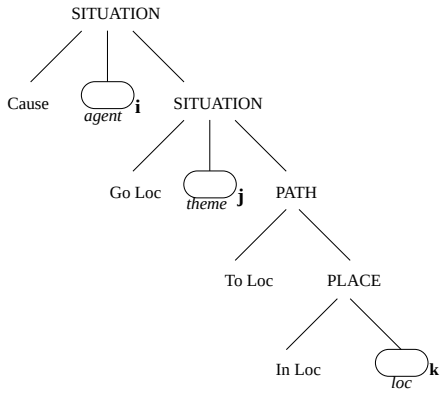

Figure 1. "He put the socks in the drawer" - no extractions

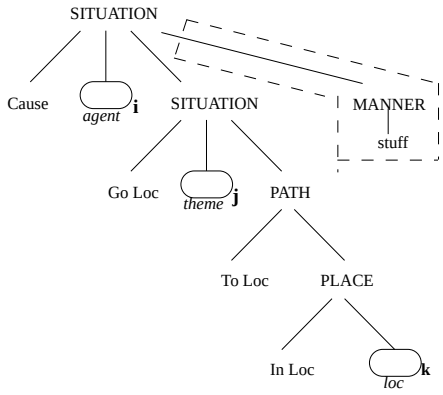

Figure 2. "He stuffed the socks in the drawer" - no extractions

he is still *putting* the socks in the drawer, but just doing so in a particular manner, namely, with a forcing, stuffing action, the structure in Figure 1 can be added to, as in Figure 2. Here the one-to-one mapping of arguments between syntax and the pivot-IL form is retained; only the surrounding lexical-IL form corresponding to the verb has been augmented.

In the process of defining the lexical-IL form for *stuff* as a modified version of the lexical-IL form for *put,* we noted that the CAUSE primitive is ambiguous, allowing for two interpretations. In a logical Cause(X,Y), either (i) the X is a trigger of situation Y and so a participant in X is not logically required to be involved in the resulting logical consequence Y, or (ii) X may both be the logical trigger of situation Y and include a participant in the consequence Y as well.[10] In particular, when we think of stuffing the socks into the drawer, we want the agent in X to take on both the trigger and the participant roles, as in sense (ii). This way we see that the stuffing is the manner of the agent's activity, as in Figure 3. Here we preserve the underlying result that socks go into the drawer, but we decouple the agent's action from the logical consequence.[11] We can read this figure, in a paraphrase, to mean, what he did by stuffing was he caused the socks to go into the drawer.

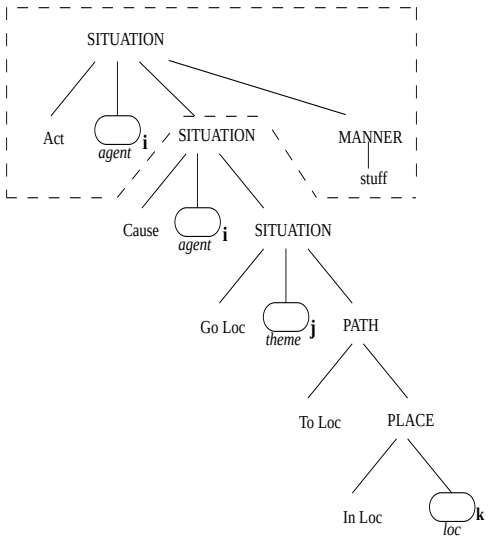

Figure 3. "He stuffed the socks in the drawer" - 1 extraction

However, while we have gained a representation that contains a clear separation of action and result that is tidy for logical inferencing in a knowledge representation system, we have also lost the simplicity of the one-to-one mapping between the syntactic surface form and the pivot IL form. There is a clear tradeoff at work here.

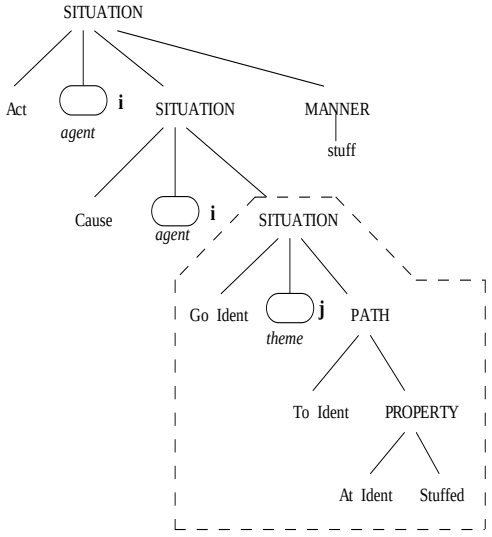

Figure 4. "He stuffed the drawer" - 1 extraction

This is one of the core issues of the debate over what belongs in the interlingua for MT. We can opt to augment the IL forms to facilitate the processing that must be done on occasion by the reasoner in the KR component of an MT system — or we can opt to keep the IL forms as transparently close as possible with a minimal mapping to and from syntactic forms.

We call the representation in which action and result are separated a form with *one extraction*. Overloading of verb meaning arises not only in cases where the activity and result are potentially individuated, but also in cases where a verb meaning has other related senses. For example, in the sentence *He stuffed the drawer* (as in Figure 4), the verb has the meaning of causing the drawer to change states by becoming stuffed, in the sense of *full.*[12] Paraphrasing the figure, we can read it as, what he did in his stuffing action was he caused the drawer to become stuffed (full). The verb in this sentence provides two types of meaning, both the stuffing action and the stuffed full state.

When we continue scaling up the set of representations, another type of challenge that arises systematically is the issue of what to do with optional modifiers, as with the phrase *with socks* added onto the previous sentence. Two possible representations are given in Figures 5 and 6. In one case the activity of the agent is elaborated, in the other both the action and the result are elaborated.[13] As the lexical IL forms being considered become more complex, as in Figure 6, the mappings between the IL and syntactic structure also need reconsideration.

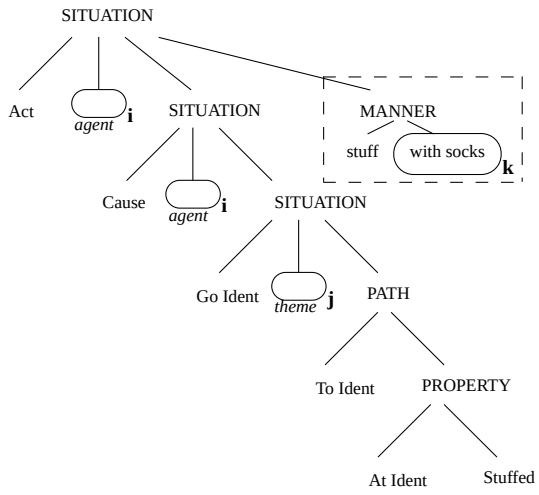

Figure 5. "He stuffed the drawer with socks" - 1 extraction

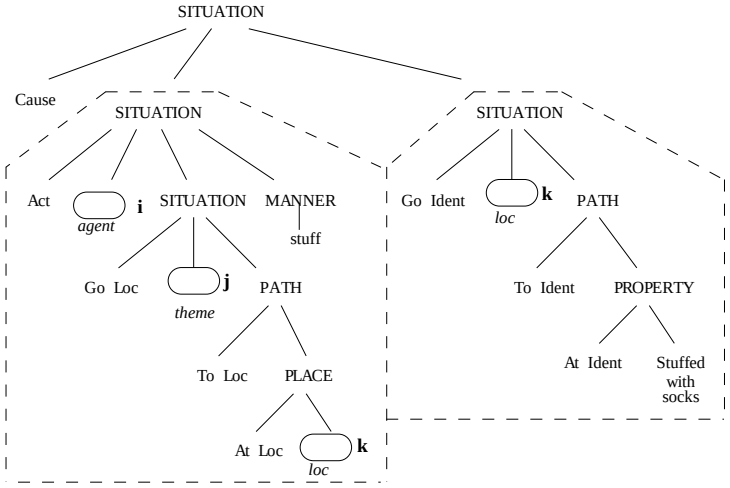

Figure 6. "He stuffed the drawer with socks" - 1 extraction

In examining different sentences, we see that, as the set of syntactic arguments changed, different IL forms were considered that, in turn, made different demands on the MT runtime mapping between the syntactic and IL forms. We started out by focusing on *stuff*'s change-of-location sense because the syntactic context that we were examining had all the components for that interpretation. Implicit in using that IL form was the need for a one-to-one mapping between syntactic and IL arguments. We then shifted our focus to *stuff*'s change-of-state sense because the next syntactic context had the argument needed for that interpretation and kept the same simple mapping. This scenario of (i) a simple mapping and (ii) the verb having a distinct lexical IL form for the two syntactic contexts, however, fails to capture clear inferences between these two related uses of the same verb: a drawer becomes stuffed if someone stuffs it and if a drawer is stuffed, then someone stuffed it. This failing becomes apparent when we shift to the last sentence and its *with socks* phrase. Thus, for the last figure, we also consider a way of including both senses, i.e., the change-of-location and the change-of-state predicates, in the same IL form. Now we clearly see the coupling of the IL and its mapping to syntax: in order to consider using this last IL form with its richer internal semantics of subevents, we need to change the mapping between the syntax and IL.[14]

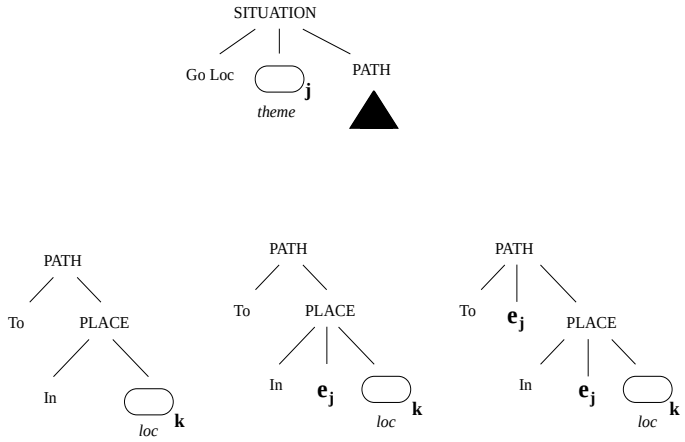

Figure 7. "The socks went into the drawer" - 0,1,2 extractions

It is also worth observing that the difficulty of assessing this tradeoff in representational forms is not restricted to verbs. In Figure 7 we see one subcomponent from the IL forms (the top tree) with three possible Path representations (the three bottom trees), corresponding roughly to variants available for the spatial preposition *into*. Two questions that arise are: (1) Should the IL form hold one argument, preserving a simple mapping to and from syntax, or should it reflect inferable

logical arguments? and (2) If we include inferable logical arguments, should this information be available within each substructure in the IL form?

The IL form for *into* with 0 extractions (leftmost PATH tree in Figure 7) has one argument to be resolved (*loc*) in composing its internal structure. The mapping is straightforward: the argument position is filled by the IL form for the direct object in the PP headed by *into.* However, this PATH IL form does not contain within it the logical subject as has been proposed elsewhere (e.g., see Herskovits (1986)). If we opt to include logical subjects of these predicates in the IL form, then we must determine where the subjects will come from. Figure 7 includes PATH IL forms with 1 and 2 extractions, the center and rightmost trees respectively with one and two logical subject positions (labeled e_j). The algorithm to fill these positions will have to search the range of pivot IL forms permitting PATHs as substructures and properly identify that subject. For example, in the sentence *the socks went into the drawer,* the e_j within the IL form for *into* will need to corefer to the socks. However in a sentence such as *the children poured cups of juice into the drawer,* the e_j within the IL form for *into* will need to corefer to the juice.

These examples highlight the fact that an evaluation of an IL formalism must ultimately be done in conjunction with a second evaluation that assesses the instantiated mappings that operate on the IL forms. At this point however, without the logical prerequisite of a formal framework for defining the IL itself, we cannot adequately specify the mappings — let alone evaluate how well they capture underlying generalizations across levels of representations.

3.2. Interlocking of the IL Components

At MT definition time, the decisions concerning the two components of the IL grammar are frequently *interlocking,* i.e., a change to one component drastically affects the functionality of the other, and vice versa. (At MT run-time, by contrast, the relation between the components is fixed and one-way, with the pivot-form algorithms accessing the lexical IL forms.) We review the interlocking problem here because it is indirectly another source of within-approach variation in MT systems: some IL developers may resolve interlocking problems within an IL grammar by favoring a Lexical Component change, such as altering lexical IL forms, while others may choose a Pivot Component change, such as adding in new pivot form operations.

Consider the English words *about*, *by*, *down*, *in*, *off*, *on*, *out*, *over*, *under*, and *up*. These appear in two syntactic constructions: the verb-particle construction (VPC, a complex with a verbal element and a particle) and the verb-prepositional phrase construction (VPP, a verb plus PP that may be an argument or an adjunct to the verb) (Lindner, 1981). These two constructions are illustrated here as (1) and (2), respectively:

(1) a. The intruder shot out the light.

b. The troops ran down the rations.

c. They looked up the address.

(2) a. The dog shot out the door.

b. The troops ran down the hill.

c. They looked up the chimney.

Note that a parser, using syntactic categories alone without semantic features, cannot determine the correct interpretation of these sentences. It might arbitrarily produce just one analysis for the NP-V-P-NP sequences, or it could overgenerate and produce two or more analyses, including possibly one VPC and one VPP for each of the above sentences. If we assume an overgenerating parser, then once the range of parse structures is known, the IL developer must establish the lexical IL forms and the pivot operations for translating these sentences.

An interlocking problem may arise with the IL forms for these sentences when the IL developers start their work at the Lexical Component. They build the lexical IL forms for the language-specific MT lexicons and then write pivot-form algorithms such that they are compatible with the lexical IL forms.[15] During algorithm development, they must take into account the range of sentential contexts, including the VPCs and VPPs, where the verbs and spatial prepositions (as listed above) may appear, as well as the range of parses for those contexts; this then may force them to revise prior decisions about the IL forms and pivot operations.

One IL developer might take a strictly compositional approach to building IL forms for verbs first as the "seeds" of the pivot construction or deconstruction process; the IL forms for the prepositions and particles are then built later on the assumption that their forms will attach to, but not overlap with, the verbs' forms (as in a jigsaw puzzle). Another developer might instead work with algorithms that permit operations beyond strict composition; in this case, the developer would build IL forms for specific verb-preposition and verb-particle combinations so that the IL form for each word "links together" with the other in an overlapping (possibly redundant) fashion.

For example, consider the construction of the lexical forms for *run* and *down,* as in sentences (1)b and (2)b. Assume that the IL developer has chosen to construct lexical semantic forms following Jackendoff (1983) for the Lexical Component and opts to compose those forms using a substitution operation in the Pivot Component. Two separate senses would be assigned to each of these words. For *run* in (2)b, one sense would be a motion predicate GO LOC that included a PATH argument and a manner modifier. For the other sense, in (1)b, *run* would have a main CAUSE predicate with a change of state predicate GO IDENT that included a PROPERTY argument. For *down,* the sense of direction in (2)b would be a spatial PATH predicate and the other sense of being diminished in (1)b would be a PROPERTY term.

Given these forms, the substitution operation in the Pivot Component would compose these senses pairwise in building pivot IL forms for these *run* + *down* sentences (see Figure 8). Either *down*'s PROPERTY-type lexical form would attach

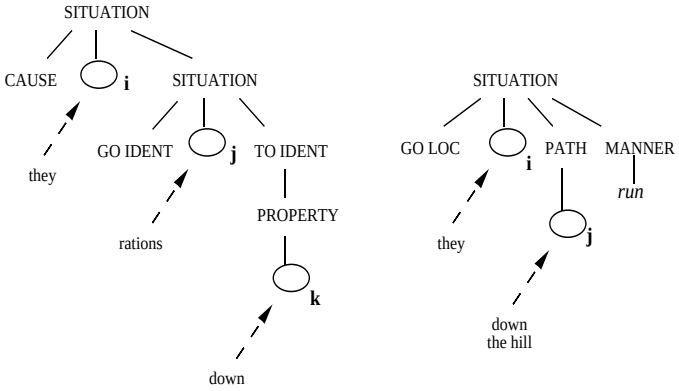

Figure 8. Lexical IL forms for *run* in:
VPC sentence (1)b *They ran down the rations.* (left side)
VPP sentence (2)b *They ran down the hill.* (right side)

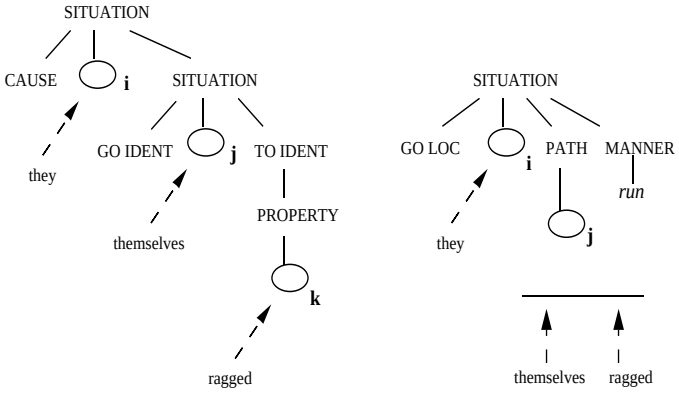

Figure 9. Lexical IL forms for *run* from:
VPC sentence (1)b, as used with sentence (3) (left side)
VPP sentence (2)b, as used in sentence (3) (right side)

at the PROPERTY position in *run*'s causative change of state IL form, or *down*'s PATH-type lexical form would attach at the PATH position in *run*'s motion IL form. The substitution (or attach) operation places one IL form at an "open" position within another IL form. The original lexical forms are unchanged in this operation and no new terms are introduced in making the attachment.

Now consider the situation where *run* appears in another context:

(3) They ran themselves ragged.

Figure 9 shows two lexical IL forms for *run* based on the possibilities already established within the current framework as illustrated in Figure 8. Here the IL developer finds that using either of these forms is problematic in constructing a pivot IL form for the sentence in (3). The causative change of state sense of *run* (Figure 9, left side) may compose with the phrases in (3): there are three open positions where the lexical IL forms for *they, themselves,* and *ragged* can be properly attached. However, even if the composition succeeds,[16] this sense of *run* is missing a manner of motion meaning, precisely the meaning in the GO LOC sense of *run*.[17] This second sense of *run* (Figure 9, right side) will also fail to compose with the phrases in (3): it has two open positions (one for a logical subject and one for a path) but none of the lexical IL forms for these phrases will attach at the PATH-type position.

In order to handle (3) with a literal reading (i.e., that they were actually running), some change to the IL for *run* is needed. We describe here briefly only one of several possible changes as an example of the interlocking of the two IL components. IL developers could argue that the difficulty above stemmed from failing to capture the shared component of meaning for *run* in (2)b and (3). Their solution might be to reduce the lexical IL form for the motion sense of *run* down to a minimal shared core: an activity predicate MOVE with a logical subject and manner modifier, but without a PATH argument (as had been in the *run* for (2)b.)

With that change to the Lexical Component however, there is a concomitant need to change the Pivot Component. Recall that in the original scenario with sentences (1)b and (2)b, the substitution operation was sufficient for attaching a PATH-type (or PROPERTY-type) form into a PATH-type (or PROPERTY-type) argument position. Now, with the PATH position gone from the motion *run* IL form in its reduction to a MOVE predicate, the IL developer must figure out how to create an attachment site for PATH-type phrases in building pivot IL forms for sentences such as (3).

In order to carry through with the proposed change to Lexical Component for *run,* the IL developers must specify one or more Pivot Component operations, that unlike substitution, will build the structure to connect PATH phrases, such as *down the hill,* into a pivot IL form also holding the revised motion IL form for *run.* For example, given the IL form for (2)b in Figure 10 containing the predicate MOVE (boxed), both the additional connecting edge from the PATH argument to the MOVE predicate must be accounted for.[18]

Similarly, given the possible pivot IL form for (3)b in Figure 11, the structure surrounding the new MOVE predicate (boxed) must be accounted for by some

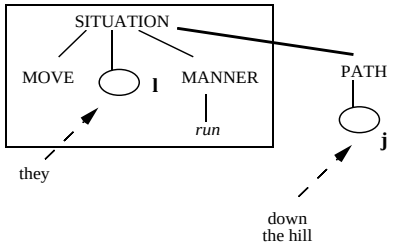

Figure 10. Sentence (2)b *They ran down the hill,* with revised lexical IL form for *run*

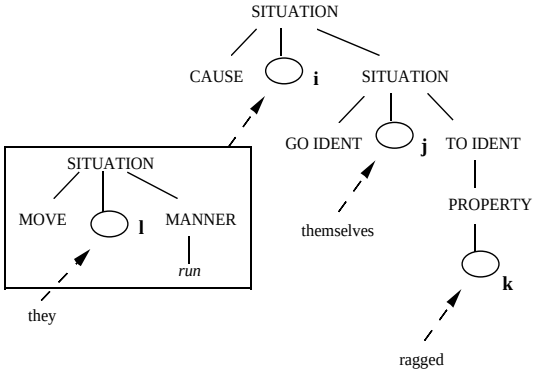

Figure 11. Sentence (3) *They ran themselves ragged,* with revised lexical IL form for *run*

combination of lexical IL forms for the phrases *themselves* and *ragged* and the structures introduced by Pivot Component operations in creating one connected pivot IL form. Note that it is not obvious what part of the IL grammar licenses the inclusion of the CAUSE and GO IDENT predicates in Figure 11. We will return to this issue in the next section.

In short, the choice of lexical IL forms affects the properties of pivot-form algorithms and vice versa. Without IL prototyping support tools that allow for rapid reconstruction and testing of new lexical IL forms or pivot-form algorithms, the interlocking problem will remain in IL-based MT systems.

4. Formalization of the IL Components

This section provides a more detailed justification for our view that IL-based MT research needs a formal framework to specify IL grammars. Although interlinguas have been developed at many research sites, there are currently no tools to assist MT developers in establishing a formalism for describing interlinguas.[19] It is our objective to define a framework within which IL-MT researchers would be able to develop and test new IL representations.[20] It is not our intent to build a new, complex interlingua or even to argue for one specific IL form over another. Rather the framework approach is to enable developers to build interlinguas and overcome the coupling and the interlocking problems that may arise in scaling up their systems. In this section we explore ways in which a *two-component* framework may be given a formal standing and serve as the basis for comparing different types of interlinguas, such as those in Section 2.

Recall that the two components in our framework are (i) the set of IL structures associated with the entries in each language-specific lexicon of an MT system and (ii) the operations associated with the composition and decomposition of the pivot IL forms built at translation run-time. We start with a "tree" view of the two components in the framework, using different tree types to describe the lexical and pivot IL forms. We then look at formal rewrite-rule grammars and ask to what extent this traditional "rewrite" view captures the implicit grammar in the two IL components. These two views, taken together, give us a notation for representing pivot IL forms in terms of constituent lexical IL forms and pivot tree-rewrite operations. We end the section by adding one more constraint to this notation that then gives us our current working formalism for defining a "lexicalized IL grammar."

4.1. "Tree" View of Components in Framework

We view the formal description of (and support tools for) the Lexical Component to be complementary to, yet distinct from, that of the Pivot-Form Component.[21] That is, we seek standalone descriptions for each component where the yoking of the components is evident in the description formalism. The "standalone" requirement that each component be described separately, distinguishes lexical from non-lexical

(i.e., derived) forms. The "yoking visibility" requirement is that the Pivot-Form Component operations be specified in terms of Lexical Component objects.

The starting point for our working formalism was to make the yoking relation evident in our notation. We use one tree type for the Lexical Component's constituent IL structures and another tree type for the Pivot Component's operations in a derivation. This way the trees in Lexical Component can "appear" within the Pivot Component's trees. For example, in Figure 12, the Lexical Component's IL forms for *run* from Figures 9 and 11 (abbreviated here as I_1 and I_5) are identifiable elements within the Pivot Component's derivational history trees for the idiomatic and the literal readings of *they ran themselves ragged* from Section 3.2.

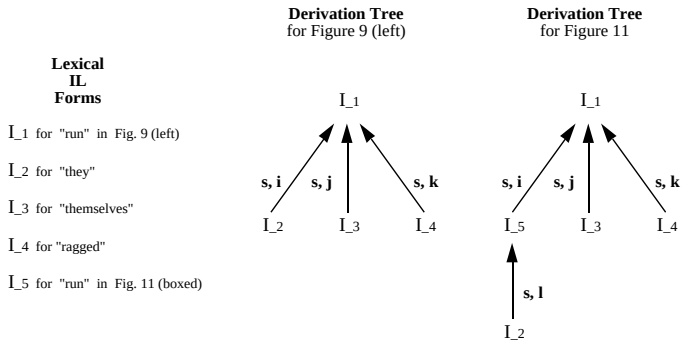

Figure 12. Derivation Trees for *They ran themselves ragged*
(s = substitution operation; i,j,k,l = addresses in lexical IL forms)

For the Lexical Component, the "tree" view of the lexical IL forms is simply specified as constituent or analysis trees whose nodes are labeled with IL terms and whose subtrees are IL constituents (as shown in Section 3). In a given MT system, each language-specific lexicon will contain only those analysis trees for words (or phrases or other lexical units) identified for that language, not the full set of possible analysis trees defined by the Lexical Component.

For the Pivot Component, derivational history or derivation trees show the trace of steps taken in applying pivot operations to a pivot IL form.[22] The derivation trees are not pivot IL forms themselves, rather they each have a corresponding pivot IL form that they encode indirectly. The nodes in a derivation tree are labeled with their corresponding pivot forms' constituent lexical IL forms, i.e., elements from the Lexical Component. Each edge is labeled and directed, identifying the particular operation of the source node's lexical IL form on the target node's IL form. The edge also holds addressing information for the composition and decomposition site in the target node of the corresponding pivot IL forms. The structure of the tree indicates the partial ordering of the pivot operations (noted as edge labels) on lexical IL forms (noted as node labels).

With these tree types, we see both lexical IL constituent structures and pivot IL operations together accounting for pivot IL forms.[23] Thus these trees meet our yoking visibility requirement: they make clear which operations in the Pivot Component work with which forms from the Lexical Component.

4.2. Rewrite-Rule View of Framework Components

The derivation trees in the last section provide a concise way of notating the set of lexical IL forms and pivot operations that go into the construction of a pivot IL form. This notation uses identifiers to *name* lexical IL forms (such as I_i, I_j, ...) and pivot operations (such as **s** for substitution) with a partial ordering on them, but it does not specify their *content:* (i) the structures internal to the lexical IL forms and (ii) the pivot operations' modifications to IL forms. The second step in building a working formalism was to define an outer bound on the form of IL constituents in each component using traditional rewrite grammars.

With respect to (i), the structures internal to the lexical IL forms are built before MT runtime and so are static during translation. For descriptive purposes, their stored entry forms may have been encoded directly enough to serve as their own specifications. However, since we are interested in sharing lexical resources built by different people for different natural languages, we need a way to compare pre-existing lexical IL forms and evaluate when it may be possible to transform them into a specified canonical lexical IL form.

One way to represent an MT system's set of lexical IL forms is to model their internal structure as trees built in the derivation of a *context-free string grammar* (CFSG).[24] These rewrite-rule grammars built in advance of the lexical form construction and, where possible, grammars built after the fact from a set of pre-existing lexical IL forms,[25] will define both a set of derivation trees as the lexical IL constituent trees and a string language of IL terms from the frontiers of those derivation trees. These may in turn allow an IL developer to use available parsing techniques and tools for tasks related specifically to the lexical IL forms, such as validating the well-formedness of new lexical structures and extracting or restructuring components in these forms.[26]

With respect to (ii), the pivot operations on IL forms, we looked to the rewrite rules in *context-free tree grammars* (CFTGs, as opposed to the CF string grammars for the Lexical Component above) as a way of specifying the operations. Generally the algorithms in MT systems are described in terms of tree-to-tree operations. For example, consider several operations used in pivot composition and illustrated in Figure 13.[27] Tree B may *attach* as the properly typed argument within a predicate in tree A, substituting a tree for a single node. (This substitution was the composition operation in Figure 12.) If a predicate in tree A is already present in tree B at composition time, i.e., they share this component of meaning, then tree B may *overlap* with tree A. If a typed argument within a predicate in tree A can recursively expand to tree B, then tree B may *adjoin* to tree A at that argument's position. When attaching fails, tree B may *connect* as an argument within a predicate in tree

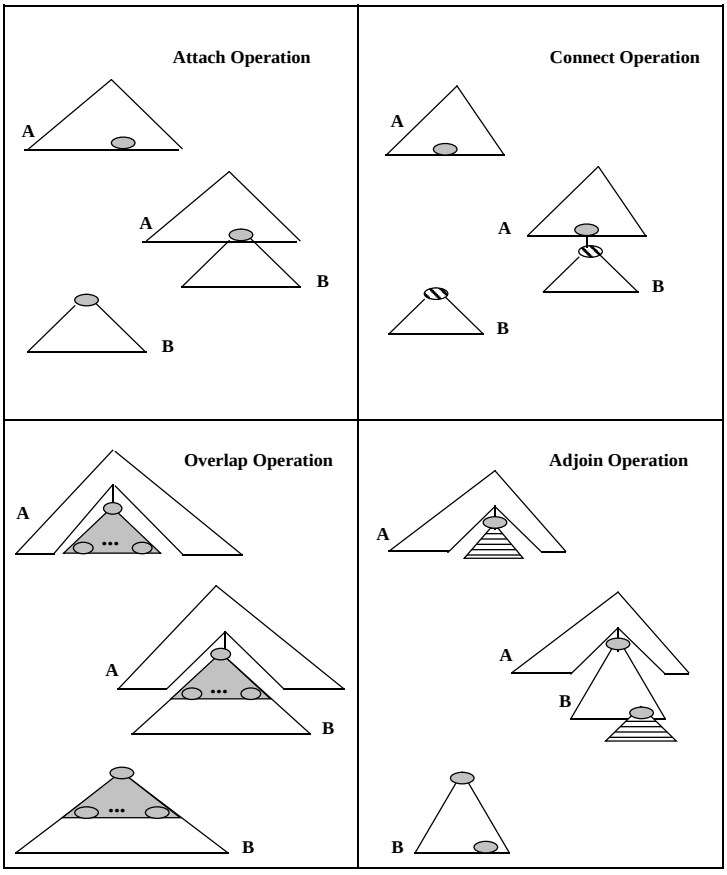

Figure 13. Various Tree-to-Tree Operations

A, introducing a new edge between unchanged trees. (In Figure 10, the Path-type tree connects to a lexical IL form of *run*.) These operations can all be formulated as context-free tree rewrite rules involving the operations of tree replacement and tree composition (as in (Schimpf and Gallier, 1985)), modified as needed for complex nodes with type checking.[28]

Context-free tree grammars also provide a way of formalizing the derivational history trees for pivot IL forms discussed above. In simplified terms, we can define the set of lexical IL forms (nodes in the derivational history trees) as the terminal alphabet of a CFTG G and redefine the set of pivot tree-to-tree operations (labels in those trees) as production rules in that G. Then G generates the set of pivot IL forms. While algorithms for CFTGs may prove useful for MT,[29] it is an open research question whether interlinguas do indeed require the full power of CFTGs. In particular, as we discuss in the next section, the tree-to-tree operations as presented so far in this paper — whether defined by rewrite rules in CFTG notation or algorithms for edge labels in derivational history trees — are lacking an explicit constraint needed for *lexicalized* IL grammars. One additional modification is needed in establishing our working formalization of a lexicalized IL grammar for our framework.

4.3. Toward a Lexicalized Grammar for Framework Components

Our working formalization thus far has involved: CFSG derivation trees to define the set of IL forms in the Lexical Component, CFTG tree replacement and composition operations as the basis for defining tree-to-tree operations in the Pivot Component, and derivational history trees (alternately encoded as CFTGs) for displaying the runtime yoking between the Lexical and Pivot Component elements. Together these meet the "standalone" and "yoking" requirements for formalizing our two-component approach that we stated earlier.

Throughout this discussion however we have not directly stated what makes an IL grammar "lexicalized" beyond defining the grammar as having a Lexical Component. In this section we review the notion of a lexicalized grammar with its lexical "anchors", as originally proposed for defining the syntactic structures in natural language. We then describe the remaining constraint we must add to our progression defining a lexicalized grammar for interlinguas in our framework. With this constraint in place, we work through several analyses of a spatial expression to show what an open research question looks like using our framework. The section ends with a summary diagram of our working definition for the framework's components.

4.3.1. Lexicalized Grammars As Originally Defined

The formal definition of a *lexicalized grammar* (Abeillé, Schabes, and Joshi, 1990; Schabes, Abeillé, and Joshi, 1988; Schabes, 1990) was originally stated, not for an MT system's interlingua, but for natural language (NL) syntax as:

> (i) a finite set of structures, each associated with lexical items; each lexical item will be called the *anchor* of the corresponding structure; the structures define the *domain of locality* over which constraints are specified; and (ii) an operation or operations for composing the structures.

A structure (in contrast to a grammar, as above) is said to be *lexicalized* when there is a lexical item realized in the structure. That is, the set of structures in (i) are lexicalized and the operations in (ii) are defined on lexicalized structures.

Although the definitions above were introduced in the context of research extending the formalism of tree adjoining grammars (TAGs) to *lexicalized* tree adjoining grammars (LTAGs), the definition given for a lexicalized grammar (LG) is more general than that for LTAGs. Since we are using the LG definition above as the basis for our definition of a LG for interlinguas, it is worth briefly spelling out the relations among LTAGs, TAGs, and LGs, to avoid possible confusion later.[30]

In simplified terms, a TAG is defined as a finite set of elementary trees and two tree operations, specifically substitution and adjunction. When TAGs have been used to define NL syntax, its elementary trees have encoded syntactic structures which are elaborated down to the lexical items (as leaves) and the tree operations compose the structures. LTAGs are TAGs whose elementary trees must be lexically anchored, as stated in part (i) of the LG definition above. The main difference between LTAGs and TAGs lies in the way that their elementary trees individually contribute to the syntactic structure of a full sentence. In LTAGs the elementary trees need not be fully expanded to lexical items, although they must include a lexical anchor, its lexicon name. Nonterminal nodes in the frontier of LTAG elementary trees are marked and filled by substitution. LTAGs are more specific than LGs in that LTAGs are restricted to specific tree operations, as compared with the LGs' set of composition operations in (ii).

The languages of TAGs, LTAGs, and LGs for syntactic structures are sentences, the sets of terminal strings from the frontiers of trees derived in those grammars. The task of parsing a sentence in a TAG or LTAG requires retrieving the elementary trees with the syntactic structures for words in the sentence and composing those structures into one parse tree. Computational tools that are based on the CKY algorithm for parsing strings in CF languages have been modified by Vijay-Shankar and Joshi (1985) for TAGs and extended for LTAGs by Schabes (1990). The task of decomposing a derived tree can be carried out for TAGs as a substructure recognition problem using a tree stack automata algorithm by Vijay-Shankar, Weir, and Joshi (1987b).

4.3.2. Extending the Notion of Lexicalized Grammars to Interlinguas

With the addition of one further constraint to our working definition, we will show here that the two components in our framework for defining an IL grammar are the "IL analogs" to components (i) and (ii) in the TAG/LTAG/LG framework for defining grammars of NL syntax. In the latter case, component (i) contains

the lexical-syntactic structures that can be composed via the operation(s) in (ii) into a full parse for a NL sentence. In former case, i.e., our framework, the Lexical Component contains the lexical IL forms that can be composed via operations of the Pivot Form Component into a full pivot IL form for a NL sentence. What remains unclear in this comparison is the role of the anchor in (i) within our framework.

In our Lexical Component, there is no term within a lexical IL form that is the obvious IL analog to being an anchor within a lexical-syntactic form. Our lexical IL forms cannot contain their own language-specific lexicon name as one of their internal terms, while lexical-syntactic forms do exactly this with their anchor doubling as lexicon name and terminal label within their own lexicon structure.

For the purposes of understanding what makes a grammar lexicalized, however, the anchor's critical role lies in identifying a lexicon name with one or more lexical structures (rather than with a label that is only an intermediary between the lexicon name and the structures). We can say that a grammar is lexicalized when the structures identified by lexically-named anchors, together with the operations in (ii), are sufficient to build the derivation trees of the grammar. Or put another way, a grammar is not lexicalized if it allows structures into the derivation tree at composition time that are not lexically anchored but were introduced by the operation itself.

We thus need first to define what constitutes a "lexically anchored" structure for an IL grammar and second, we must add to our IL grammar definition the explicit constraint that pivot operations may not introduce structure that is not lexically anchored. Since by definition a lexical IL form must have a lexicon name, we take the implicit anchor of a lexical IL form to be its lexicon name.[31] Given that a lexicon name may identify more than one lexical IL form (as when it is polysemous), an anchor may have more than one IL form. The explicit constraint on pivot component would be, in the derivational history trees, to disallow operations adding nodes that do not originate in the lexical IL forms being operated on.

4.3.3. An Example

In order to make our discussion more concrete, we look at one English sentence where we can readily see several ways for determining which words should anchor which portion of the sentence's semantics. In the sentence:

The car roared down the hill.

we understand that there is one event where a particular car was both roaring and going down a hill at the same time. That is, we could paraphrase this sentence, depending on our focus, as either:

The car roared going down the hill.

The car went down the hill roaring.

Since Talmy (1985), sentences such as this one have been used to show differences in cross-linguistic semantics. Here we see that English can express the concomitant sound and motion of a car in one clause. Levin and Rappaport-Hovav (1995) cite sentences in French and Japanese where, by contrast, these two components

of meaning require separate clauses. The *lexical anchoring problem*[32] for the IL developer is determining how the English words in the *roar down* sentence ought to anchor the motion semantics that is implicit in English, given that such semantics is needed independently for translations into languages such as French and Japanese.

In Figure 14 the words *roar* and *down* anchor the basic lexical IL forms I_6 and I_8 in the example sentence. The IL forms for the phrases *the car* and *the hill* are substituted into argument positions in I_6 and I_8 respectively. Since the basic sense of *roar* is assumed here to be a predicate for emitting noise with no sense of motion, as when it appears in the simple sentence *the thunder roared,* there is no substitution site in that sense's IL form for *down*'s Path IL form to connect to. Consequently a pivot operation (labeled c1 in the figure) that permits a Path to modify a Situation would be needed for this pivot IL form to compose.

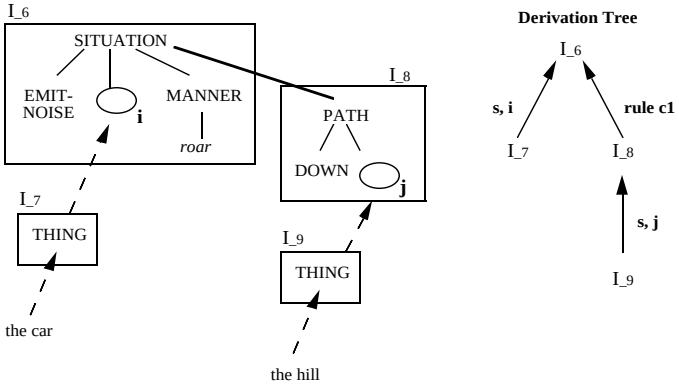

Figure 14. Basic lexical IL form for *roar* and *down* in pivot IL form and its derivation tree for sentence *the car roared*

However, even if there were independent justification for the pivot operation that connected the forms anchored by *roar* and *down,* the resulting pivot IL form in this case is difficult to interpret. Is the noise or the car in the process of making noise the entity traveling down the path? Had the sentence been *the mother shouted down the hill,* with another verb of emitting noise, we would want its pivot IL form to mean only that the noise, and not the mother in the act of shouting, was traveling down the hill.

Another possibility an IL developer may consider in representing the example sentence is for *roar* to anchor an IL form with a non-basic, or extended sense (i.e., moving with a loud, deep sound), while also using the basic sense of *down.* By creating a Path type position in *roar*'s IL form, this arrangement makes a site available for *down*'s IL form to connect to. While this simplifies the composition to substitution at each step, as can be seen in the derivation tree in Figure 15,

the two issues raised with respect to Figure 14 remain open here as well. What is going down the path? Should other verbs of sound emission also be assigned a comparable extended IL form with a Path type position added to their basic IL form?

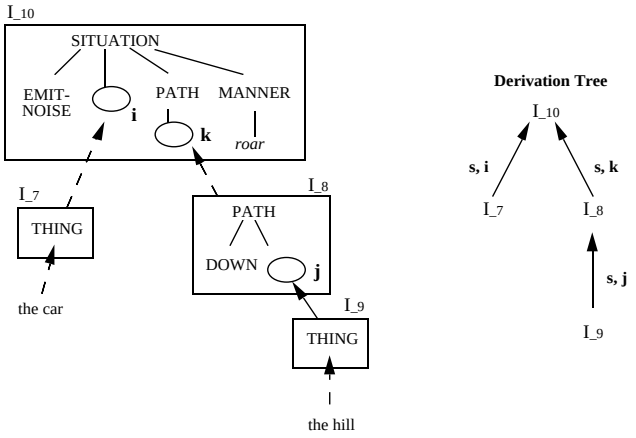

Figure 15. Extended lexical IL form for *roar* and basic lexical IL form for *down* in pivot IL form and its derivation tree for sentence *the car roared*

One way to encode the motion semantics for this sentence more directly, given that our framework dictates using only lexically anchored forms, is to place the semantic burden on *down* and have it anchor a structure that includes a motion predicate as well as its own basic path sense. For example, in Figures 16 and 17, *down* anchors a Situation type entity with a motion predicate. In the former case, the motion situation modifies the sound emission that is taken as the primary focus of the sentence's meaning. In the latter case, the motion itself is given primary focus and the sound emission is secondary.

Note that in a non-lexicalized approach, i.e., without the lexicalization constraint that all nodes in a pivot IL form be accounted for lexically, the anchoring problem and the interlocking problem run together. Since there is no need to distinguish pivot substructures that are anchored by lexical items from those that are not when no lexicalization constraint exists, IL developers will individually keep track of and decide on their own mix of lexically-based (anchored) and operationally-derived IL forms. For one word, they put its extended semantics in a separate lexicon entry, but for another word, they may have its extended semantics added in a pivot operation. Consequently in a non-lexicalized approach, we would expect to find a wider range of, and possibly a less consistent, encoding of patterns of sentential semantics when compared to a lexicalized approach.

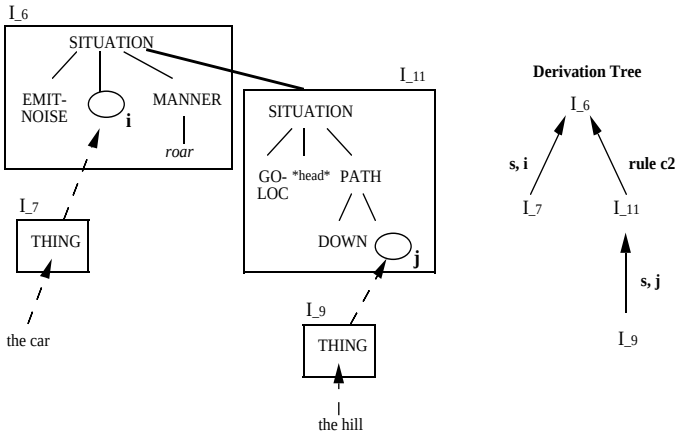

Figure 16. Basic lexical IL form for *roar* and extended lexical IL form for *down* in pivot IL form and its derivation tree for sentence *the car roared*[33]

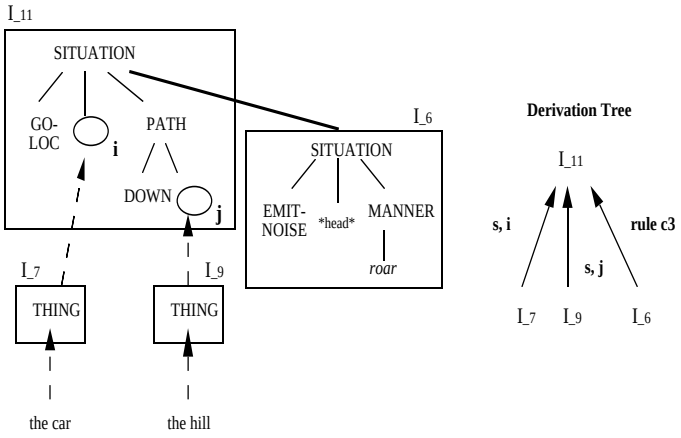

Figure 17. Extended lexical IL form for *down* and basic lexical IL form for *roar* in pivot IL form and its derivation tree for sentence *the car roared*

4.3.4. Summary

Central to our approach in proposing a framework for developing lexicalized IL grammars has been the idea that we, as framework builders, are working toward formalizing our two-component approach, identifying formal and computational tools that can assist us. These tools then enable the IL developers to specify, build and test their own representations. In Section 4.1 we described our use of derivational history trees for capturing the yoked relation between the Lexical and Pivot Components in specifying the pivot IL forms. The trees are defined in terms of nodes labeled by lexical IL forms and edges labeled by tree-to-tree operations (see Figure 18). In Section 4.2 we extended the specifications for the two-component framework by including rewrite-rule grammars to define the content of the nodes and edges in the derivation tree.

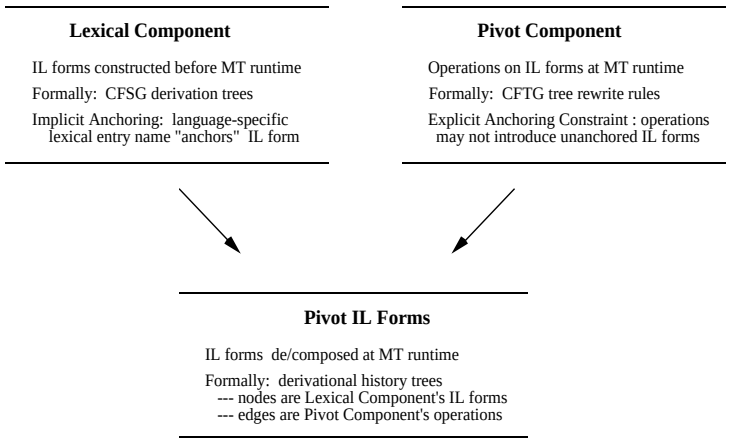

Figure 18. Components and Working Formalization for Framework

And lastly, after reviewing the notion of lexicalized grammars for NL syntax, we discussed the implicit anchoring in the Lexical Component and extended this principle to the Pivot Component by constraining pivot operations to be defined only on lexically anchored structures. We have thus presented our working definition for lexicalized IL grammars.

5. Testing the Approach

In this section we briefly introduce our work with ILustrate, a software tool to support development work during IL specification and testing cycles. We are currently in the beginning stages of implementation and have found that the many algorithms

used for parsing, recognition, and generation in areas outside of our specific focus (i.e., IL development tools) are, in fact, applicable to the design of ILustrate.

One of the goals of ILustrate is to assist in the scaling up of IL-based MT systems as the lexical IL forms or the algorithms for the pivot IL forms are revised to handle new data. ILustrate, in accord with our two-component view of the IL as a lexicalized grammar in MT systems, has two Specification Modules, one for the Lexical Component and one for the Pivot-Form Component. The output of the Specification Modules feed into the Design-Testing Modules for these Components. In addition to the reasons presented above, we add here other practical considerations for keeping this division.

5.1. Need for Tools Within an Approach

If we look at the MT research in progress even *within* one approach from Section 2, we see that (i) there is variation among researchers in their interpretation of a particular linguistic or conceptual theory for building the lexical IL forms, and (ii) there are limitations with respect to what phenomena are handled. For example, recently, Verrière (1994) has developed lexical IL forms for French following a lexical-semantic approach much like that of Dorr (1990). What we see however is that, although the forms are similar, there remain differences that make importing Verrière's French forms into another MT system, however similar, a time-consuming task that will require expertise in both the IL representation of each system as well as a knowledge of French. So one practical reason for designing ILustrate with a separate specification module for the Lexical Component is that it helps identify what types of declarative lexical IL variation exist between two MT systems. If a well-defined mapping can be established between the grammars of two systems' lexical forms, then a conversion algorithm can be built to adjust these forms before run-time. That is, by virtue of being able (i) to delimit the variation as lexical and (ii) to define a mapping between grammars of each system's lexical forms, we can scale up one MT system with lexical data from another system.

Since all theories imported for use as a basis for MT interlinguas are incomplete, invariably there will be another source of discrepancy between two research groups working even within the same approach: how they each choose to extend that theory to handle data outside the scope of the theory. For example, researchers who focus primarily on translations at the predicate-argument level (such as those working with Jackendoff's LCSs or Hale and Keyser's LRSs) will eventually have to scale up their formalism to cover logical semantic words, including boolean-logical words *and, or, not*, causal-logical words *if, then, because*, and quantifiers. The IL that includes this class of lexical entries must capture both their *inherent* semantic sense as well as their scope (or domain of locality) in *derived* forms. In a two-component view of the IL, the properties of logical operators can be specified declaratively in the Lexical Component as well as procedurally in the composition algorithms of the Pivot Component. (Recall from Figure 13 that several types of tree-to-tree

composition operations are possible, each allowing for different structural definitions of scope.)

The MT developer must specify how their logical terms will be interpreted. Thus another reason for designing ILustrate with a separate specification module for the Pivot Component is that eventually an MT system will need to be scaled up to include this class of entries and their scoping domains will need to be defined with respect to algorithms in that component.

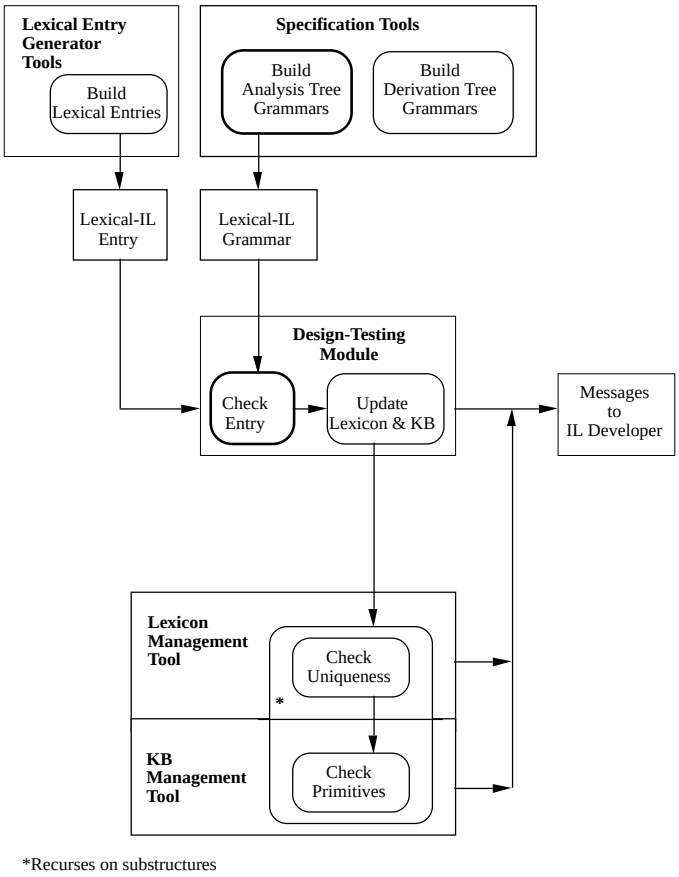

Figure 19. ILustrate Design: Lexical Component View

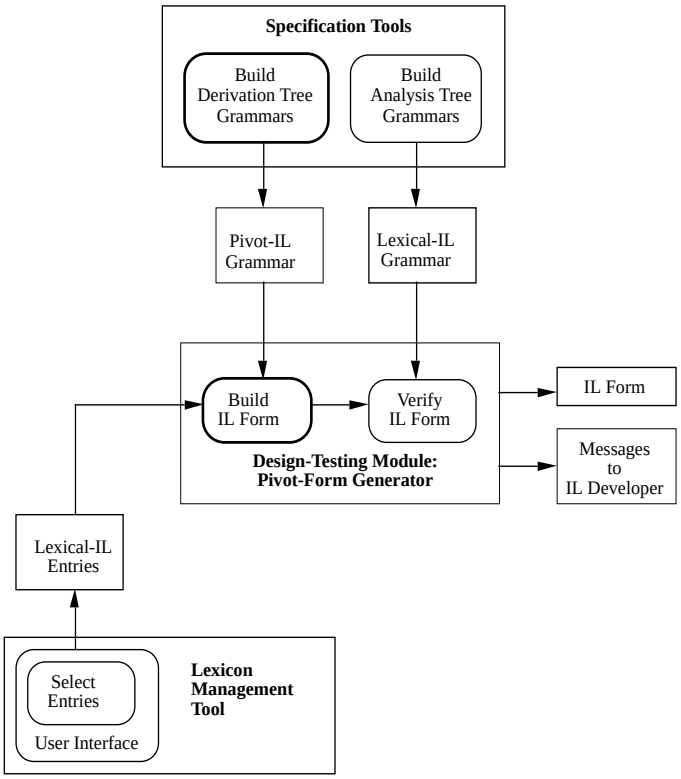

Figure 20. ILustrate Design: Pivot Form Component View for Composition

5.2. Lexical Component

In Figure 19, the Specification Tools box (in the topmost row of boxes) contains two modules. The "build analysis tree grammars" tool is the module used by the MT developer to specify the grammar of their lexical component entries. Once built, the Lexical-IL grammar can be used in the Design-Testing Module for a variety of functions. It may check the form of new lexical entries before they are added to the lexicon and knowledge base to ensure that new forms are grammatically consistent with lexical entries already created.[34] The Lexical-IL grammar may also be used to read in entries of one form and generate a second set of entries that is consistent with a different grammar.[35] This, for example, is the application we need to work with Verrière's data (mentioned above).

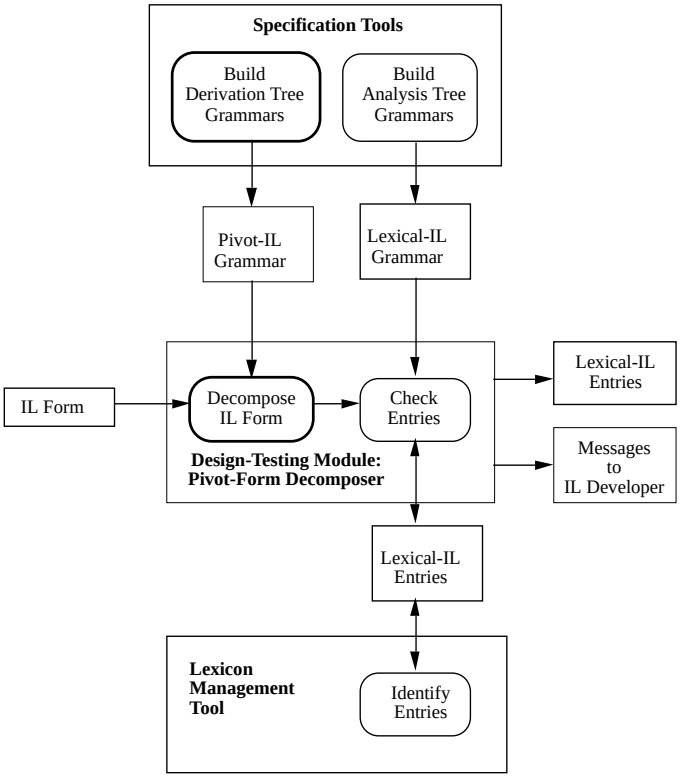

Figure 21. ILustrate Design: Pivot Form Component View for Decomposition

5.3. Pivot Component for IL Composition

The second module within the Specification Tools box is used by MT developers to specify the grammar of the Pivot Component operations in terms of derivation trees (see the "build derivation tree grammars" box in Figure 20). Once built, the Pivot-IL grammar is brought into the Design-Testing Module of ILustrate during pivot form composition and decomposition. For example, the grammar guides the building of the pivot form so the developer can supply new lexical entries and then test and modify their interaction with the algorithms of the grammar. This speaks to the interlocking problem described in Section 3.2 where the developer can be prompted to supply new lexical entries (such as for phrases whose meaning is not strictly compositional) and then can test for the correct pivot forms using either the new or older lexical entries.

5.4. Pivot Component for IL Decomposition

The Pivot-IL grammar, when brought into the Design-Testing Module during decomposition as in Figure 21, enables the developer to submit a test IL form and have it disassembled into lexical IL forms. As a side effect, this also provides a way of generating missing lexical-IL forms to be added to a new target language lexicon.

6. Conclusion

We have argued that no consensus exists among MT researchers on the level of representation for defining the IL. The consequences of this missing formal framework for MT system builders are that: (1) current interlinguas are defined by the MT system in which they are built and cannot be readily reused elsewhere; (2) modifications to an existing IL formalism often force the MT builder to adjust the mapping between the IL and other levels of representation (e.g., the syntactic structure); and (3) scaling up the IL representation language often results in the problem of *interlocking* — the difficulty introduced by interactions between the declarative and procedural components of the IL when a change to one component significantly affects the functionality of the other, and vice versa.

We have taken an approach that addresses these issues, relying on notions from well-known rewrite-rule grammars of formal language theory. While the connection between the grammar of formal theory and the grammar of the interlingua is not immediately obvious, we have shown that it is possible to view these notions in terms of a *lexicalized grammar* for interlingual machine translation. The strings of rewrite grammars are analogous to the trees of our lexical forms; moreover, the simple operation of attachment that occurs in the application of rewrite rules is analogous to a set of more complex operations with overlapping trees. We have shown that there is a correspondence between a formal grammar for a context-free language to a two-component formalism: (i) the production or rewrite rules in

the grammar correspond to the Lexical Component of the IL, and (ii) the implicit derivation operation of substring substitution maps into the Pivot Form Component of the IL. We claim that all IL system builders are operating under a similar set of constraints, regardless of their theoretical approach; thus, we are able to view the IL operations of any given approach within the framework of a lexicalized grammar.

One of our main objectives is to address the critical need to *decouple* the IL as a formal language (defined unto itself) from the representational formalisms used at other levels of description in the MT system. The specification of a formal framework for the IL is a logical prerequisite to defining a mapping between the IL and representations at other levels. Consequently we first anchor each level in its own primitive representations and requirements; and only then can we begin to talk about the relationships between the levels. This is analogous to the traditional view of programming languages: high-level languages are specified independently from the assembler languages into which they must be mapped; in turn, assembler languages are specified independently from the language of 0's and 1's into which they must be mapped.

We set about to find a means for testing an IL without building a full machine translation system. Given the view above, we have taken an approach that provides a framework within which the MT system builder can formally define the IL and can then define the mappings between the IL and other levels of representation. We believe this approach to be promising, especially when one considers scaling up the system. In particular, we have provided a framework that allows the builder to exploit systematic correspondences between levels (e.g., between syntax and lexical semantics); thus, when a given level is modified, the builder is able to accurately pinpoint the corresponding components at other levels that will also require modifications. This contrasts sharply with approaches based on an "incrementalist IL" (see, e.g., Frederking et al. (1994)) in which a modification to one component of the interlingua is likely to entail modifications to other seemingly unrelated components. In this contrasting approach, it is difficult for the MT builder to determine the full inventory of effects of a given change, however minor they may be.

A final issue that we have attempted to address within this framework is that of IL evaluation, i.e., determining how well a given IL captures underlying generalizations across levels of representations. Our approach allows the MT builder to assess both the adequacy of the IL forms as well as the adequacy of the mappings that operate on these forms.

Acknowledgements

We would like to express our gratitude to Doug Jones for helpful commentary and suggestions and also to three anonymous reviewers who provided substantial guidance toward shaping this article into its final form. The research reported herein was supported, in part, by Army Research Office contract DAAL03-91-C-0034 through Battelle Corporation, NSF NYI IRI-9357731 and Logos Corporation, NSF CNRS INT-9314583, Advanced Research Projects Agency and ONR contract

N00014-92-J-1929, Alfred P. Sloan Research Fellow Award BR3336, Army Research Institute contract MDA-903-92-R-0035 and Microelectronics and Design, Inc., and the University of Maryland General Research Board.

References

Abeillé, A., Y. Schabes, and A. Joshi. 1990. Using Lexicalized Tree Adjoining Grammars for Machine Translation. In *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki, Finland.

Anderson, J. 1971. *The Grammar of Case: Towards a Localist Theory.* Cambridge University Press, Cambridge, England.

DiMarco, C., G. Hirst, and M. Stede. 1993. The Semantic and Stylistic Differentiation of Synonyms and Near-Synonyms. In *Working Notes for the AAAI Spring Symposium on Building Lexicons for Machine Translation*, Technical Report SS-93-02, pages 114–121, Stanford University, CA.

Dorr, B. 1990. Solving Thematic Divergences in Machine Translation. In *Proceedings of the 28th Annual Conference of the Association for Computational Linguistics*, pages 127–134, University of Pittsburgh, Pittsburgh, PA.

Dorr, B. 1993. *Machine Translation: A View from the Lexicon.* MIT Press, Cambridge, MA.

Dorr, B. and M. Palmer. 1995. Building a LCS-Based Lexicon in TAGs. In *Working Notes for the AAAI Spring Symposium on Representation and Aquisition of lexical Knowledge: Polysemy, Ambiguity, and Generativity*, Technical Report SS-95, Stanford University, CA.

Dorr, B. and C. Voss. 1993. Machine Translation of Spatial Expressions: Defining the Relation between an Interlingua and a Knowledge Representation System. In *Proceedings of the AAAI*, pages 374–379, Washington, DC.

Dorr, B. and C. Voss. 1994. The Case for a MT Developers' Tool with a Two-Component View of the Interlingua. In *Proceedings of the First Conference of the Association for Machine Translation in the Americas*, pages 40–47, Columbia, MD.

Dorr, B., C. Voss, E. Peterson, and M. Kiker. 1994. Concept Based Lexical Selection. In *AAAI 1994 Fall Symposium on Knowledge Representation for Natural Language Processing in Implemented Systems*, New Orleans, LA.

Frederking, R., S. Nirenburg, S. Helmrich D. Farwell, E. Hovy, K. Knight, S. Beale, C. Domashnev, D. Attardo, D. Grannes, and R. Brown. 1994. Integrating Translations from Multiple Sources within the Pangloss Mark III Machine Translation System. In *Proceedings of the First Conference of the Association for Machine Translation in the Americas*, pages 73–80, Columbia, MD.

Grimshaw, J. 1993. Semantic Structure and Semantic Content in Lexical Represntation. Manuscript, Rutgers University.

Hale, K. and J. Keyser. 1993. On Argument Structure and the Lexical Expression of Syntactic Relations. In K. Hale and J. Keyser, editors, *The View From Building 20: Essays in Linguistics in Honor of Sylvain Bromberger*. MIT Press, Cambridge, MA.

Herskovits, A. 1986. *Language and Spatial Cognition*. Cambridge University Press, Cambridge, England.

Hirst, G. 1995. Near-synonymy and the Structure of Lexical Knowledge. In *Working Notes for the AAAI Spring Symposium on Representation and Aquisition of lexical Knowledge: Polysemy, Ambiguity, and Generativity*, Technical Report SS-95, Stanford University, CA.

Hirst, G. and M. Ryan. 1992. Mixed-Depth Representations for Natural Language Text. In P. Jacobs, editor, *Text-Based Intelligent Systems*. Lawrence Erlbaum, Hillsdale, NJ, pages 59–82.

Hutchins, W.J. and H. Somers. 1992. *An Introduction to Machine Translation*. Academic Press Inc., San Diego, CA.

Jackendoff, R. 1983. *Language and Cognition*. MIT Press, Cambridge, MA.

Jackendoff, R. 1990. *Semantic Structures*. MIT Press, Cambridge, MA.

Jensen, K. 1987. Binary Rules and Non-Binary Trees: Breaking Down the Concept of Phrase Structure. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins Publishing Company, Philadelphia, PA, pages 65–86.

Kay, M., J.M. Gawron, and P. Norvig. 1994. *Verbmobil: A Translation System for Face-to-Face Dialog*, volume 33. CSLI Lecture Notes, Stanford, CA.

Langacker, R. 1987. *Foundations of Cognitive Grammar, Vol. 1: Theoretical Prerequisites*. Stanford University Press, Stanford, CA.

Levin, B. and M. Rappaport-Hovav. 1991. Wiping the Slate Clean: A Lexical Semantic Exploration. In B. Levin and S. Pinker, editors, *Lexical & Conceptual Semantics*. Blackwell Publishers, Cambridge, MA, pages 123–151. (Reprinted from *Cognition* (1991) 41).

Levin, B. and M. Rappaport-Hovav. 1995. *Unaccusativity: At the Syntax-Semantics Interface*. MIT Press, Cambridge, MA.

Levin, L. and S. Nirenburg. 1994. The Correct Place Of Lexical Semantics in Interlingual MT. In *Proceedings of Fifteenth International Conference on Computational Linguistics*, Kyoto, Japan.

Lindner, S. 1981. *A Lexico-Semantic Analysis of English Verb Particle Constructions with OUT and UP.* Ph.D. thesis, University of California, San Diego.

Mani, I. 1995. An Integrative, Layered Approach to Lexical Semantics and Its Application to Machine Translation. In *Working Notes for the AAAI Spring Symposium on Representation and Aquisition of lexical Knowledge: Polysemy, Ambiguity, and Generativity*, Technical Report SS-95, Stanford University, CA.

Nirenburg, S., J. Carbonell, M. Tomita, and K. Goodman. 1992. *Machine Translation: A Knowledge-Based Approach.* Morgan Kaufmann, San Mateo, CA.

Nomura, N., D. Jones, and R.C. Berwick. 1994. An Architecture for a Universal Lexicon: A Case Study on Shared Syntactic Information in Japanese, Hindi, Bengali, Greek, and English. In *Proceedings of Fifteenth International Conference on Computational Linguistics*, Kyoto, Japan.

Onyshkevich, B. and S. Nirenburg. 1995. A Lexicon for Knowledge-Based MT. *Machine Translation*, 10(1–2).

Pinker, S. 1989. *Learnability and Cognition: The Acquisition of Argument Structure.* MIT Press, Cambridge, MA.

Rappaport, M. and B. Levin. 1988. What To Do With Theta-Roles. In W. Wilkins, editor, *Thematic Relations.* Academic Press, New York, NY, pages 7–36.

Rosenfeld, A. 1990. Array, Tree and Graph Grammars. In H. Bunke and A. Sanfeliu, editors, *Syntactic and Structural Pattern Recognition: Theory and Applications.* World Scientific, Teaneck, NJ, pages 85–115.

Rosetta, M.T. 1994. *Rosetta: Compositional Translation.* Kluwer Academic Publishers, Dordrecht, The Netherlands.

Schabes, Y. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars.* Ph.D. thesis, University of Pennsylvania, Philadelphia.

Schabes, Y., A. Abeillé, and A. Joshi. 1988. Parsing Strategies with 'Lexicalized' Grammars: Applications to Tree Adjoining Grammars. In *Proceedings of 12th International Conference on Computational Linguistics*, Budapest, Hungary.

Schimpf, K. 1982. *A Parsing Method for Context-Free Tree Languages.* Ph.D. thesis, University of Pennsylvania, Philadelphia.

Schimpf, K. and J. Gallier. 1985. Tree Pushdown Automata. *Journal of Computer and System Sciences*, 30:25–40.

Sparck-Jones, K. 1983. Shifting Meaning Representations. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany.

Stede, M. 1993. Lexical Options in Multilingual Generation From a Knowledge Base. Manuscript, University of Toronto.

Talmy, L. 1985. Lexicalization Patterns: Semantic Structure in Lexical Forms. In T. Shopen, editor, *Language Typology and Syntactic Description 3: Grammatical Categories and the Lexicon.* University Press, Cambridge, England, pages 57–149.

Thatcher, J.W. 1967. Characterizing Derivation Trees of Context-Free Grammars through a Generalization of Finite Automata Theory. *Journal of Computer and System Sciences*, 1:317–322.

Verrière, G. 1994. Manuel d'utilisation de la structure lexicale conceptuelle (LCS) pour représenter des phrases en français. Research note, IRIT, Université Paul Sabatier, Toulouse, France, June.

Vijay-Shankar, K. and A. Joshi. 1985. Some Computational Properties of Tree Adjoining Grammars. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, Chicago, IL.

Vijay-Shankar, K., D. Weir, and A. Joshi. 1987a. Characterizing Structural Descriptions Produced by Various Grammatical Formalisms. In *Proceedings of the 25rd Annual Meeting of the Association for Computational Linguistics*, Stanford, CA.

Vijay-Shankar, K., D. Weir, and A. Joshi. 1987b. On the Progression from Context-Free to Tree Adjoining Languages. In A. Manaster-Ramer, editor, *Mathematics of Language.* John Benjamins Publishing Company, Philadelphia, PA, pages 389–401.

Voss, C., B. Dorr, and M. Ülkü Şencan. 1995. Lexical Allocation in Interlingua-Based Machine Translation of Spatial Expressions. In *Working Notes for IJCAI-95 Workshop on the Representation and Processing of Spatial Expressions*, Montreal, Canada.

Zwarts, J. and H. Verkuyl. 1994. An Algebra of Conceptual Structure: An Investigation Into Jackendoff's Conceptual Semantics. *Linguistics and Philosophy*, 17:1–24.

Notes

1. We distinguish here between a *theory* and its definition in terms of a *formalism.* Various notational variants may express the same information within a theory. For example, the theory of context-free languages can be expressed in the string formalism of rewrite rules, such as $X \rightarrow \alpha$, or alternately in the graph formalism of one-level trees with a root labeled X and a frontier labeled α. Note that although two notations may be *formally* equivalent in terms of a mathematically well-defined isomorphism, they may not be *computationally* equivalent in terms of the resources required for their representations.
2. This classification is our own, as are the labels we have chosen to identify each approach.
3. Dorr calls her lexical IL forms *root word* LCSs, or RLCSs, and her pivot IL forms *composed* LCSs, or CLCSs.

4. Recent work by Zwarts and Verkuyl (1994) suggests the places in Jackendoff's representation where reference and quantification may be installed. Mani (1995), following their insights, proposes computational lexical structures that are "layered" with a logical formalism at one level and an LCS-like formalism at another level of representation.
5. Indeed, as pointed out by Kay, Gawron, and Norvig (1994), a *lexicalized* event is but one viewpoint of a real-world event. For example, the British action of *slotting a ticket in the machine* when one gets on a bus or a train may be, in Swiss French, *invalidate the ticket,* and in Swiss German, *validate the ticket.* Pinker (1989) holds a similar view: his "autonomy of lexical semantics" states that not all cognitively meaningful concepts are the basis for lexical structures.
6. Note that this tiering is distinct from two other types of representations: (i) Hirst and Ryan's mixed-depth representation where for one sentence, different phrases can be incomplete, i.e., partially encoded rather than fully, in different ways at different levels of representation (Hirst and Ryan, 1992), and (ii) Sparck Jones' simultaneous use of multiple different representations of the same text (Sparck-Jones, 1983).
7. While we know of no other current MT research that shares our tiered approach, the distinction we draw between linguistically relevant IL forms and conceptually based KRR forms, e.g., the tiering between the IL and the KRR levels in (Dorr and Voss, 1993), is similar to the one drawn by Grimshaw (1993), Levin and Rappaport-Hovav (1995), and Pinker (1989). Like them, we have argued that the set of structures available for linguistic semantics (our IL forms) is constrained even though the constants within those structures (our IL primitives) may be of unlimited complexity conceptually. (In spite of agreement on this basic assumption, the specific lexical representations developed in each of these tiered approaches is different.)
8. The set of IL forms may require at most a context-free tree grammar to describe it, where the lexical forms are trees and the pivot operations are context-free tree operations (see Rosenfeld (1990); Schimpf and Gallier (1985)). That is, the framework-based tools allow the IL developers a way of building, testing, and revising their own IL specifications, within the bounds of the class of CF tree grammars.
9. In this approach, the label on each terminal or leaf node is taken as an IL primitive that is grounded in a term in the knowledge representation system. The nonterminal nodes are IL class types.
10. As an example of case (i), consider the sentence *John threw the bag across the room,* where we take the X in Cause (X,Y) to be John's action of throwing the bag and Y to be the bag's going across the room. For case (ii), consider *John dragged the bag across the room.* We take the X to be John's dragging the bag and Y to be both John and the bag going across the room. That is, in case (i) John is a participant in only X, but in case (ii) he is involved in both X and Y. This distinction is important for translating these sentences into languages such as French where manner of motion verbs (e.g., the *throw* in X) and directional phrases (e.g., *across the room* in Y) do not appear and cannot be translated into the same clause (for details, see Talmy (1985)). In generating a separate clause for Y, the MT system must note that the participants in case (i) and (ii) will be different. In the figures in the text, we explore a range of IL forms to capture the relations among the X and Y subevents and their participants.
11. Other researchers, outside of *computational* semantics, also address related issues. Levin and Rappaport-Hovav (1995) discuss lexical semantic forms in terms of situation templates for activities, states, achievements, and accomplishments. For example, their lexical semantic forms for an activity and a resulting state compose monotonically into an accomplishment form. Pinker (1989) proposes complex lexical semantic forms with subevent structures also appearing in the lexical forms, but where he also has additional argument positions outside the subevents.
12. Note that in Figure 4 the *theme* is the drawer, not the *socks* as it was in Figures 1–3.

13. The curious reader wishing to examine other lexical representations may look to Pinker (1989, pp. 236–238), as well as work by by Rappaport and Levin (1988) and Levin and Rappaport-Hovav (1991). The group of verbs that alternate between the change-of-location (location-as-direct-object) and change-of-state (locatum-as-direct-object) variant senses are assigned different "basic" forms by Levin and Rappaport Hovav when they entail motion **toward** and motion **away from** a location. For the toward-sense verbs such as *load,* the change-of-location is "basic" and the change-of-state sense is "extended" (Rappaport and Levin, 1988). For the away-from sense verbs such as *wipe,* the situation is reversed; the change-of-state is "basic"and the change-of-location is "extended" (Levin and Rappaport-Hovav, 1991). Figure 6 falls somewhere between the forms in (Pinker, 1989) and (Levin and Rappaport-Hovav, 1991), with fewer IL sites for arguments mapped from syntax than Pinker, and with a more complex activity subevent than Levin and Rappaport Hovav.
14. Indeed this also highlights the coupling of the IL and the knowledge representation and reasoning component (KRR) in an MT system. As the lexical IL forms are built with more internal structuring to denote subevent level information, as in Figure 6, one must ask how these IL forms will map to and from the event structures in the KRR component.
15. This seems to be the typical sequence of events in MT research. Most linguistically-motivated representations are developed by linguists independent of the processing mechanisms that will operate on those representations. Thus, MT researchers are likely to adopt the linguistic representations first because these have been formally specified and then, in a subsequent step, they develop the algorithms that operate on these representations, rather than the other way around. In practical systems it is rarely feasible to decide on representations beforehand.
16. The reason for our caveat, i.e., that this attachment *may* work, is that it depends on all 3 phrases being accessible at the time when the argument positions in *run*'s IL form are being filled. If, for example, the parsed structure for (3) made the resultative *ragged* a non-VP internal phrase and, in addition, the syntax-IL mapping algorithms did not access phrases outside the VP during the verb's IL form composition, then *ragged* would not be accessible "in time" to fill a position in *run*'s IL form.
17. With the resultative *ragged* in (3), there is both a reading where they do literally run and an idiomatic reading where they become ragged somehow. The causative change of state form from (1)b in Figure 9 (left side) does capture the idiomatic reading of (3), but it fails to capture the literal *run* reading also possible in (3).
18. This possible form is suggested by the structure of sentence (2)b's paraphrase *they went down the hill running.*
19. We also find no standard documentation of IL grammars. In comparing ILs across MT systems, it is difficult to assess how portable the ILs are.
20. In this paper we focus on the grammar for interlingual structures and do not take a position in the debate on the relation between the syntactic and semantic levels, whether or not one level is derived from the other. In our figures and discussions of IL forms here, we have purposely included structures proposed by advocates of both approaches. Those who wish to minimize the extent that syntactic distinctions formally express semantic ones (e.g., Jackendoff (1983), Jackendoff (1990)) cite, as evidence for distinct, incommensurate syntactic and conceptual levels, examples where (i) a wide range of syntactic distinctions can signal the same semantic relation (e.g., the *aktionsarten* telic/atelic distinction can be expressed syntactically through the choice of verb, preposition, adverbial, determiner in the subject, object, or prepositional object) and (ii) a range of semantic distinctions can be expressed in terms of the same syntactic position (e.g., the direct object can express thematic roles such as theme, goal, source, experiencer, patient, or beneficiary). Those who wish to maximize this relation, seeking to derive syntactic distinctions from those in semantics, focus on the linking regularities that associate arguments bearing certain semantic roles to particular syntactic expressions and take, as evidence for their approach, cross-linguistic data showing "striking similarities in the linking regularities" (Levin and Rappaport-Hovav, 1995).

21. There are both practical and theoretical reasons for this. Practically, this would facilitate the sharing of MT resources (e.g., lexical IL forms) and the documentation of an IL's MT system dependencies (e.g., the coupling of parser and IL derivation processes). Theoretically, this separation forces the IL developer to identify the basic lexical substructures over which linguistic generalizations may be stated. (Outside of MT research, for example, Levin and Rappaport-Hovav (1995) propose a limited set of such substructures, *lexical semantic templates,* for capturing general patterns of verb class and range of multiple meanings. Pinker (1989) identifies a small number of *thematic cores* for verbs.
22. We distinguish here between *derived trees* and *derivational history trees.* Derived trees in our framework are tree representations for pivot IL forms — they have been derived from lexical IL forms following the application of Pivot Component operations. Derivation trees, an abbreviated name for *derivational history trees,* show both the operations and components that go into obtaining a derived tree. For a formal description of derivation trees, see details of Project Rosetta (Rosetta, 1994) and Vijay-Shankar, Weir, and Joshi (1987a).
23. In so doing, we follow other researchers working with both analysis (or constituent) trees and computation-defined trees in developing representations, such as Jensen (1987) who uses both phrase structure and parsing structures in syntactic forms.
24. Typically the term *context-free grammars* refers to **string** grammars. That is, the production or rewrite rules of the grammar are of the form $\alpha \rightarrow \beta$, where both α and β are strings. As we will discuss, one can also have **tree** grammars where the α and β in the rewrite rule $\alpha \rightarrow \beta$ are trees. Context-free tree grammars, like context-free string grammars, replace the α structure with the β structure independent of the context surrounding α. For further details, see (Schimpf, 1982).
25. In order to construct such a grammar retroactively given a set of lexical IL forms, a first-pass procedure would be to convert each set of sister nodes $\beta_1, ..., \beta_n$ with parent α in those forms into a rewrite rule $\alpha \rightarrow \beta_1, ..., \beta_n$. This procedure could then be revised in many ways, for example, by writing more general rules together with well-formedness constraints. Listing out the actual range of parent-daughter configurations as rewrite rules is a heuristic first step in dealing with the lack of a specification.
26. Here we sketch one such task using a Lexical Component's grammar: to verify that a lexical IL form is well-formed in the grammar, use a bottom-up tree automaton for that CFSG to check if the lexical IL form (a tree) can be derived in G. Intuitively, a bottom-up tree automaton A_σ recognizes an input tree **t** by producing a state tree **t'** of the same shape as **t** but with its nodes annotated with state information. (See (Thatcher, 1967) and its references for formal characterization.) Initially A_σ assigns an initial state to each terminal node in **t'**: a terminal node with the label **f** would be assigned an initial state $\alpha_f(\lambda)$. Then, for each rule $H \rightarrow a_1 \ \ldots \ a_k$ in G, when the states $s_{a_1}, \ \ldots, \ s_{a_k}$ have been assigned to all of H's successor nodes in **t'**, the state $\alpha_H(s_{a_1} \ \ldots \ s_{a_k})$ is assigned to H's own node in **t'**. (For each rule in G there is a unique state. Since the set of rules in G is finite, a finite correspondence table can be used to record state-rule pairs.) The final output state of A_σ is the state assigned to the root of **t'**. The input tree is accepted if the state assigned to that root is a final state corresponding to S in G.
27. Dorr and Palmer (1995) also discuss these operations.

28. In CFTGs, the tree replacement operation, $t_1[d \leftarrow t_2]$, changes t_1 by first deleting the subtree rooted at node address d and then by placing tree t_2 at that d. (The operation is defined formally as prefixing the addresses of all the nodes in t_2 with the path from the root of t_1 to d, but otherwise leaving the t_2 configuration and its content unchanged.) The tree composition operation, $t_0[var(x_1) \leftarrow t_1, ..., var(x_n) \leftarrow t_n]$, changes the tree t_0 with n simultaneous tree replacements, so that each subtree (in t_0) rooted at node address $var(x_i)$ is replaced by tree t_i. The *attach* operation in Figure 13 is a one-node form of tree replacement with only the node at the attachment site d in t_1 being "replaced" by t_2. The *connect* operation is a zero-node form of tree replacement: no nodes in t_1 are changed, but t_2 is treated as the replacing tree in t_1 attaching a single node descendant of d (i.e., formally the addresses of all nodes in t_2 are prefixed with a path from the root of t_1 through d to the root of t_2, while leaving t_2's configuration and content unchanged). The *overlap* operation occurs when the subtree in t_A rooted at address d is to be replaced by t_B while also being identical to t_B*, a subtree in t_B rooted at t_B's root. There are two ways of invoking this change via rewrite rules. One way is simply to treat it as a simple replacement, $t_A[d \leftarrow t_B]$. The other way is to treat it as a tree composition operation, $t_A[var(x_1) \leftarrow t_1, ..., var(x_n) \leftarrow t_n]$, where the leaves $var(x_1), ..., var(x_n)$ in the subtree t_A rooted at d are replaced respectively by $t_1, ..., t_n$, the trees in the forest left when subtree t_B* is removed from t_B. The *adjoin* operation is a sequence of tree replacement operations: $t_1[d \leftarrow t_2[d_2 \leftarrow t_1/d]]$. Remove the subtree at d in t_1 (call this t_1/d) and at that d address, attach the tree t_2 that has had its own subtree t_2/d_2 replaced with t_1/d.
29. For example, we can adapt a CFTG recognition algorithm that verifies if a pivot IL form **t** is well-formed in a CFTG G: use a tree pushdown automaton A_τ on that G to recognize subtrees from G's alphabet in **t**'s derivation, thus decomposing an input pivot IL form into its constituent lexical IL forms. A_τ, like A_σ (mentioned in an earlier note), traverses the input tree **t** reconstructing its derivation in a state tree except that, here, the derivation to be tracked is guided by *tree* rewrite rules, rather than *string* rewrite rules. (See (Schimpf and Gallier, 1985) for a formal definition.) Initially A_τ assigns one read-head with a *tree stack* to each terminal node in the state tree. So the tree stack of each read-head starts out with a subtree consisting of a leaf. Then A_τ either *moves* read-heads or *reduces* a read-head. In *moves,* read-heads percolate up in the input tree, merge and combine tree stacks. In a *reduce* step, the read-head pops its tree stack in matching a (transition function version of a) rule in G_τ and pushes a new parent (possibly a tree as well) on its tree stack. A_τ accepts the input tree if it is able to read the entire input tree and end up with one read-head stack at the root with its tree stack holding S from G_τ.
30. Readers well-versed in TAG formalisms and families will find that we opted for oversimplification and succinctness in our description over lengthy details and precision.
31. Given that a specific lexical IL form may appear in more than one MT lexicon, an IL form may have several anchors corresponding to lexicon names in different languages.
32. Elsewhere in our work (Voss, Dorr, and Şencan, 1995), we use the term *lexical allocation* to include both (i) this *pre-runtime* problem (here, building MT lexicon definitions for entries with a range of possible *anchorings*) and (ii) a *runtime* version of this problem: given a semantic structure for a sentence, which portion of that structure should be allocated to (alternately, anchored by) which lexical entries in the target language?
33. In this figure and in the next one, the node marked *head* is the logical subject of the situation-type IL form and its value is resolved at runtime with the logical subject of its IL form's parent.
34. In some MT systems the lexicon and KB are combined; in others they are kept separate. The specification and testing modules for the Lexical Component of ILustrate are independent of this aspect of MT system design.
35. Nothing in principle preempts adding a human checker into the loop to adjust the entries as well.