The Role of Language in Cognition: A Computational Inquiry

Jill Fain Lehman, Allen Newell, Thad Polk, and Richard Lewis

Carnegie Mellon University

In preparing this manuscript, as in preparing the talk on which it is based, we were asked to tell George, from our particular standpoint, "how the mind works." We have clearly taken some license in recasting his question as "What is the role of language in cognition?" Still, the relationship between language and cognition has been a central concern for George throughout his career. Indeed, an answer to the question is central to any theory of mind.

Our standpoint is computational. As a result, our methodology in contemplating the issues is, first, to reason from the architecture, and second, to examine existing systems. Nevertheless, we take seriously George's injunction to "stand back a little from your most recent work," and so are willing to speculate and extrapolate on his behalf. We begin our discussion of the role of language in cognition by examining the prevailing point of view.

1. Language as Transducer

A basic tenet of cognitive science is *the problem space view* which states that thinking occurs in internal, task-oriented problem spaces that use internal, task-oriented operators on internal representations of the situation. This characterization of cognition is one of the field's important contributions, in part, because it allowed computation to be brought to the enterprise of understanding the nature of mind.

When applied to our question, the problem space view yields the paradigm of language as transducer—the process of comprehension transforms utterances from the external world into the internal representation that task-oriented operators require. The complementary process of generation tranforms the result of those task operations back into external linguistic forms:

Essentially all systems that contain both natural language and problem solving components have taken this view (for example, UNDERSTAND (Hayes & Simon, 1975)). The transducer paradigm is epitomized by the natural language front-end.

What does the transducer paradigm say about the role of language in cognition? It provides a simple and direct answer: language and cognition share a structure, which we call *the situation model*. By delivering a representation of the situation to the task, language has its effect on cognition through the encoding of their shared model and through any subsequent structures added to long-term memory based on that encoding. In essence, the transducer paradigm is equivalent to a weak form of the Whorfian Hypothesis (Whorf, 1956): language influences cognition, but does not determine it.[1]

---

[1]In a recent attempt to quantify this influence, (Hunt & Agnoli, 1991) appeal to current practice in psychology and enumerate, as significant, effects at the 50 msec level.

Given this longstanding view, is any other answer possible? We go to the architecture and ask!

2. The Architecture Replies

The Soar architecture has been described fully in (Newell, 1990). Figure 2-1 allows us to summarize briefly. Following the numbers in parentheses, we can characterize the architecture by a long-term memory composed of patterns that deliver their associations to working memory, thereby defining the current problem-solving context (1). Problem solving occurs in problem spaces (2) by a process of state-to-state transition from an initial state to a desired state. Transitions occur via the application of operators, one transition per *decision cycle* (3). In the elaboration phase of a decision cycle, knowledge flows into working memory in the form of preferences for new problem spaces, operators, or states. Once all patterns have delivered their associations, a fixed decision procedure is invoked. If there is an unequivocal next step, it is taken (4). Otherwise the architecture impasses (5) and a new goal is established to attain the knowledge to resolve the impasse. The new goal gives rise to a new problem space (6), and problem solving continues. Once the impasse is resolved by reaching a desired state in the subspace(7), Soar's learning mechanism captures the conditions that lead to the impasse and the results of problem solving in a new association which is added to long-term memory (8).

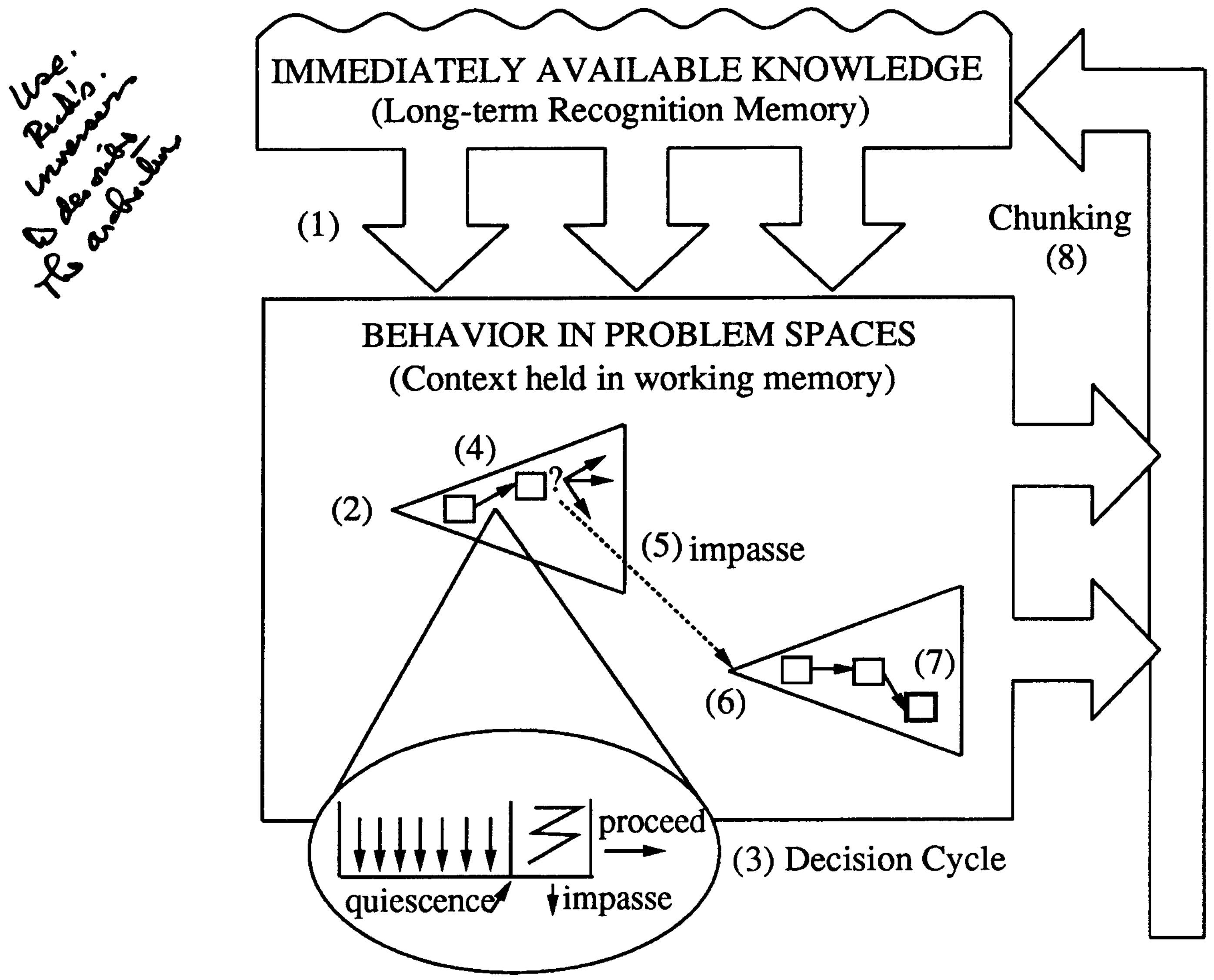

Figure 2-1: The Soar architecture

Task → Task
Task → Language
Langu → Language
Lang → Task.

To understand the architecture's reply, let us first separate cognition from language by reifing the former in a Task problem space defined by non-linguistic task operators, and the latter in a Language problem space defined by comprehension and generation operators. We can then imagine the sort of typical, hierarchical goal stack that arises in Soar as a result of impasses. As shown in Figure 2-2, within the goal stack we may find either or both of the two possible relationships between language and thought: a Task space may impasse into a Language space (A) or a Language space may impasse into a Task space (B). Although this is simply a structural reply, it is not without functional consequences.

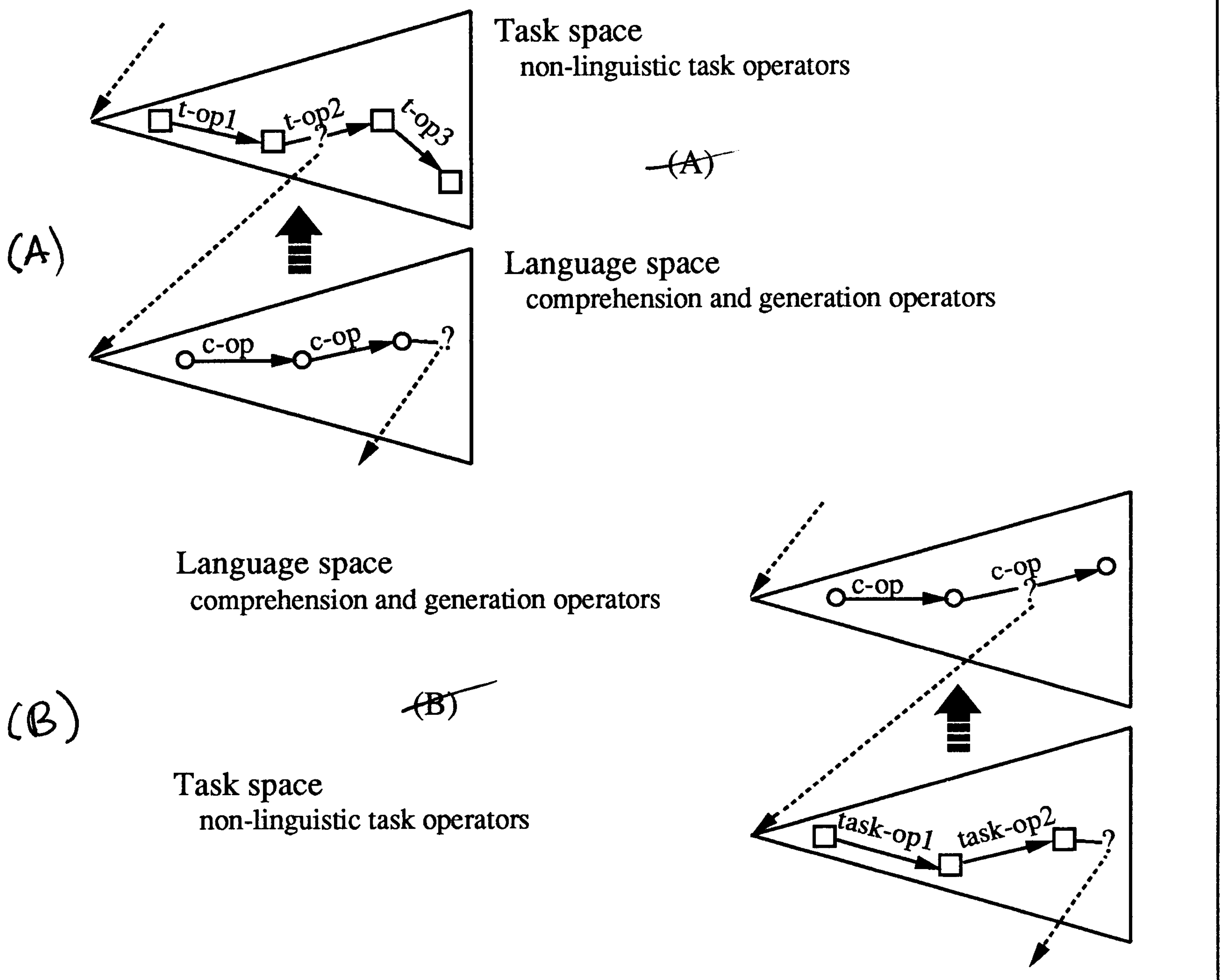

Figure 2-2: Possible relationships between language and cognition.

An examination of the impasse from Task to Language (Figure 2-2, (A)) reveals the realization of the transducer paradigm in Soar. The general dynamic is straightforward; we trace it out simultaneously in Figures 2-2 and 2-3. The Task space, in this case the Blocks-world space, contains task operators which perform task-related transformations on the state. For example, t-op1 alters the state by producing the situation model in the top-right portion of Figure 2-3. Some of these task operators may be transduction operators that are proposed when linguistic input appears on the state (for example, t-op2). The transduction is implemented in the Language space by operators that transform linguistic input from the external world to a non-linguistic form (the middle situation model of Figure 2-3). Once the content of the utterances has been captured, operators in the task space continue to apply to this non-linguistic representation (for example, t-op3 and the bottom situation model in Figure 2-3). A similar process occurs when transduction is required for generation.

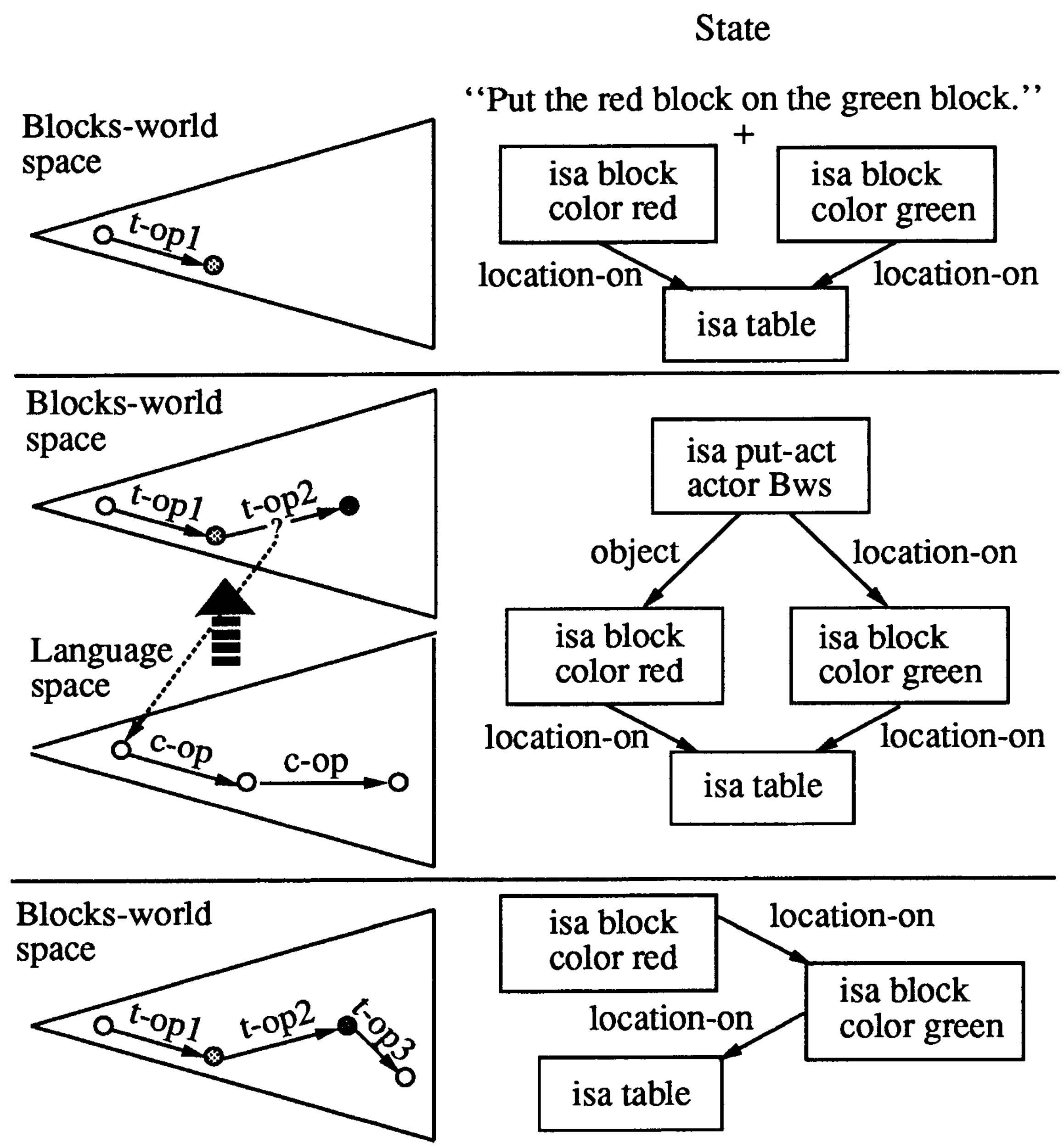

Figure 2-3: Language transduction in the blocks world.

Note that the role of language in this case is essentially ancillary. Granted, a weak Whorfian view is supported by the architecture because task operations proceed based on the model delivered by Language. Still, if there were some other method for generating the relevant piece of situation model (such as remembering it), then no impasse would arise and task operations alone would be adequate to reach a desired state. The Whorfian Hypothesis is weakened further by the observation that whatever Language operations are used, they may occur in the context of, and be influenced by, a pre-existing situation model that has resulted solely from task operations.

That the transducer paradigm can be found among the potential behaviors of the Soar architecture should not be surprising; Soar clearly takes the problem space view, and it is from this view that the transducer paradigm naturally evolves. What may be surprising, however, is that the structural reply in Figure 2-2 has two other, very different, functional consequences, as well.

3. The impasse from Task to Language: Linguistic Task Operators (LTOs)

In the example above, and in the transducer paradigm in general, there are two types of operations: task operations and transduction operations. The former are carried out by task operators in the Task space using task knowledge to take a non-linguistic portion of the state into a new non-linguistic portion of the state. The latter are carried out by linguistic operators in the Language space using linguistic knowledge to take a linguistic input from the external environment into a non-linguistic portion of the state. That task operators are non-linguistic is an assumption of the paradigm, not the Soar architecture. Indeed, as shown in Figure 3-1, the architecture admits the possibility of a linguistic task operator (LTO) that uses knowledge about language to take a non-linguistic state into a non-linguistic state.

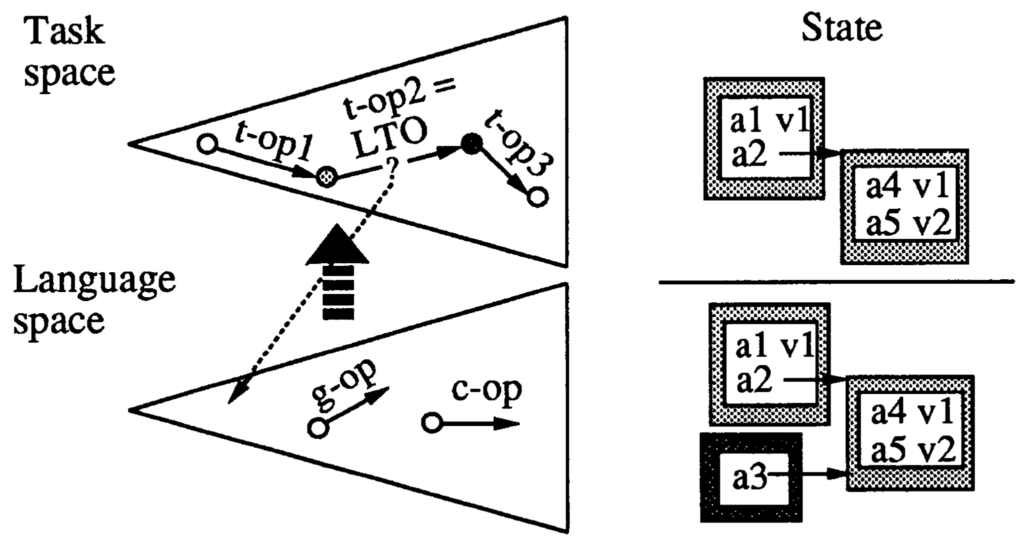

Figure 3-1: Linguistic task operators.

The process begins as it did in Figure 2-3 with the top situation model produced by the non-linguistic task operator t-op1. The bottom model is then produced by an LTO, a task operator implemented in the Language space. This t-op2 is different from the t-op2 in Figure 2-3 because it does not require the existence of an utterance from the external environment in order to produce a change to the situation model. The disorganization of operators in the Language space in Figure 3-1 points to another difference between the t-op2's. In the blocks world case, we know what linguistic knowledge is needed to implement the transduction. At this point in the discussion, it is unclear what linguistic knowledge implements an LTO.

Observing that the architecture admits the possibility of LTO's is a far cry from either demonstrating that LTO's exist or explaining what LTO's mean. To do so, we must look beyond the architecture's reply and employ our second methodological tool. Thus, we turn to VR-Soar, a system that solves syllogisms, to find the answers.

3.1. A ~~brief~~ digression: VR-Soar and the categorical syllogism task

The syllogism task is probably familiar: given two premises, we must state a conclusion that both links the end terms and necessarily follows from the premises. Two specific syllogisms and their general forms are shown below:

Premise 1: All artists are barbers	All A are B
Premise 2: All barbers are chefs	All B are C
Response: All artists are chefs	All A are C

Premise 1: All artists are barbers	All A are B
Premise 2: Some barbers are chefs	Some B are C
Response: ?	?

VR-Soar is a computational theory of human syllogistic reasoning that seeks to explain individual differences in this task by predicting individuals' behavior on all sixty-four types of syllogisms (Polk & Newell, 1988). As is evident from the examples, some syllogisms are easy and some are not, the latter providing evidence that solving syllogisms is a genuine reasoning task in which we expect to find task operators as well as language operators. Nevertheless, systems capable of performing the task tend to fit the transducer paradigm (for example, resolution theorem provers and model-based systems such as (ref Johnson-Laird's new book ?).

The general organization of VR-Soar is shown in Figure 3-2. The Task space has one task operator, negate-conclusion, and three transduction operators that are implemented in the Language spaces (read-input, generate-conclusion, and respond). The static impasse structure in the figure makes it clear that there are ample opportunities for Task to Language impasses during actual problem solving.

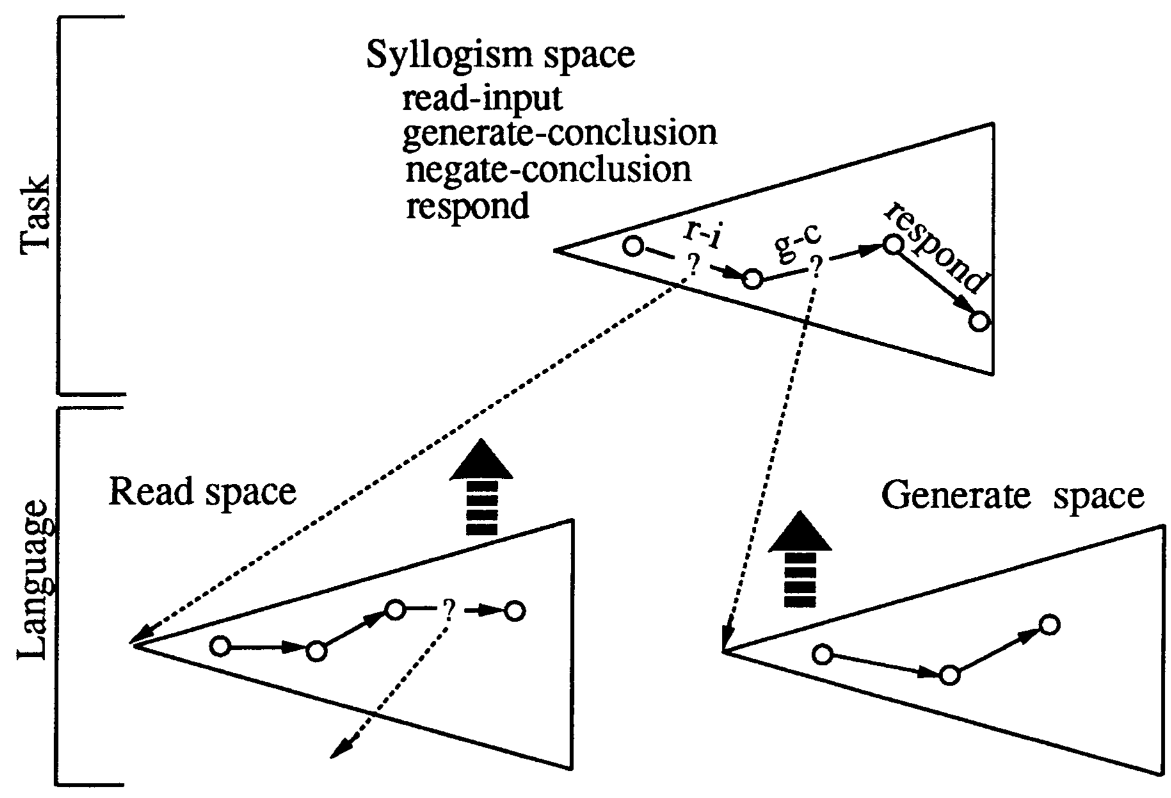

Figure 3-2: VR-Soar, a system for solving syllogisms.

VR-Soar uses two methods for solving syllogisms: the basic method and the falsification method. In the basic method, the system simply generates a conclusion from the single, cumulative situation model created by reading the premises. Consider the first example we saw above and the situation model produced after each premise:

Read: All artists are barbers.	(A B)
Read: All barbers are chefs.	(A B C)
Generate: All artists are chefs.	
Respond: All artists are chefs.	

Reading the first premise produces a model with a single element that has both the property of being an artist and of being a barber. The lack of explicit quantification in the model is a key piece of VR-Soar's theory. Reading the second premise in the context of the existing model augments the single element with the property of being a chef (that is, comprehension of the premise results in every barber already in the model being made a chef). From this cumulative model it is straightforward to generate the conclusion *All artists are chefs* and respond accordingly.

The basic method is not guaranteed to generate conclusions that necessarily follow from the premises. The falsification method, through the use of the negated-conclusion operator, is. If a negated conclusion leads to a contradiction then the original conclusion must necessarily follow from the premises. Similarly, if the negated conclusion does not lead to a contradiction, the original conclusion was not valid. Consider how this method is used in our second example:

Read: All artists are barbers.	(A B)
Read: Some barbers are chefs.	(A B C) (B C) (B)
Generate: Some artists are chefs.	
Negate: No artists are chefs.	
Comprehend: No artists are chefs.	(A B -C) (B C) (B)
Re-read: All artists are barbers.	no change
Re-read: Some barbers are chefs.	no change
Respond: No valid conclusion.	

As in the previous example, reading the first premise creates the model (A B) which is then augmented by reading the second premise. How a person interprets the word *some* is considered a source of individual differences. In the interpretation shown here, *some* means there is at least one that is not. Thus, comprehension has three effects on the situation model: it makes the existing artist-barber a chef, it creates a barber who is a chef but not an artist, and it creates a barber who is neither a chef nor an artist. From this cumulative model a conclusion linking the end terms can be generated: *Some artists are chefs.* At this point in the basic method VR-Soar would respond with the generated, albeit incorrect, conclusion. Using falsification, however, the next step is to negate the conclusion. Comprehending the negated conclusion then augments the model further—the artist is no longer a chef. Finally, the system re-reads the two premises. Since no inconsistency arises with respect to the situation model during re-reading, the system correctly responds, *No valid conclusion.*

Several interesting things occurred in these two examples. The first syllogism, solved by the basic method, did not require any task operations to build up the situation model and read off the conclusion. To make the point in a slightly different way, treating the language processes as transducers was sufficient for the task. In the second syllogism, however, falsification required two task operations: negating the conclusion and testing the situation model. Nevertheless, these task operations were accomplished via language processes. But that is exactly what LTO's are all about.

3.2. LTO's: Reprise

Before our digression, we had established two features of LTO's. First, we defined an LTO to be a task operator implemented in the Language space. Second, we noted that an LTO is distinct from a transduction operator because it does not require input from the external environment in order to produce a change to the state. Yet, at that point in the discussion, it remained unclear how LTO's could be implemented in the Language space, whether they really exist, and, most importantly, what they mean.

Given our analysis of VR-Soar the answers to these questions have become clear. An LTO is a task operation that is implemented as an act of generation followed by an act of comprehension. The generation process produces an utterance which comprehension uses to test and/or change the situation model. In the falsification example we saw two uses of LTO's. The first occurred in the implementation of the negate-conclusion task operator: the generation of the negated conclusion was followed by its comprehension, thereby performing the task operation of changing the situation model through linguistic means but without the existence of an utterance from the external environment. The second use of LTO's came in the act of re-generating the premises in order to perform the task operation of testing the situation model by comprehension.[2].

What, then, do LTO's mean? By virtue of performing task operations that are not mere transductions, using LTO's is truly thinking in language. Do LTO's therefore vindicate the strong Whorfian Hypothesis that language determines thought? If LTOs were the only kind of task operator available, the answer would be yes. But there are many other non-linguistic spaces for tasks (for example, visual spaces, mathematical spaces), filled with non-linguistic task operators. Thus, while LTO's may take their rightful place in the cognitive repertoire, we must acknowledge that they are but one of the techniques available.

4. The impasse from Language to Task: Taskification

In looking at the blocks world example and VR-Soar, we found two functional implications of the structural configuration produced by an impasse from Task to Language: language can influence the task situation model (weak Whorfian hypothesis) or language can peform the task (LTO-Whorfian hypothesis). The impasse from Task to Language being only half the story, we now turn our attention to the other half of the architecture's reply.

Consider the configuration shown in Figure 2-2(B). Here an operator in the Language space (in

---

[2]Since the re-generation of the premises happened via reading, isn't this a case of using utterances from the external environment, and, therefore, not an occurrence of an LTO? Functionally, we consider the re-reading to be an internal activity—that the premises are *read* was necessary the first time (the transduction), but that they are *read* is incidental the second time (they could, for example, have simply been remembered). To make the point a bit differently, suppose we had heard the premises initially and written them down. The act of writing them down is functionally equivalent to memorizing them. Thus, if recovering them in the latter case is an internal operation, then it is functionally internal in the former case as well

this case a comprehension operator) gives rise to an impasse that can only be resolved through task knowledge. To understand how such a configuration can occur, and what taskification means, we must digress once again, this time to understand how language comprehension occurs in Soar.

4.1. A brief digression: NL-Soar and the task of language comprehension

NL-Soar is the current realization of Soar's Language space. It is the set of problem spaces and operators that provide Soar with a comprehension capability that responds to the real-time constraint of 200 to 300 msec per word (Lehman, Lewis, & Newell, 1991a, Lehman, Lewis, & Newell, 1991b). In the mapping of Soar onto human cognition (Newell, 1990), the real-time constraint corresponds to a processing constraint of two to three operators per word. Meeting this processing constraint requires *recognitional comprehension* via total integration of the relevant knowledge sources. We achieve total integration through *comprehension operators* that are learned automatically through chunking.

A graphical trace of the operation of NL-Soar is shown in Figure 4-1. Comprehension is recognitional whenever a comprehension operator is available in the Comprehend space for a word in a given context. Under those circumstances all knowledge is brought to bear in a single operator application and NL-Soar's utterance and situation models are incrementally augmented (the utterance model captures the structure of the utterance, the situation model captures the meaning). However, when no comprehension operator is available for the current context, an impasse arises and the remaining problem spaces in NL-Soar implement the comprehension operator through deliberate problem solving. The problem spaces accessible via that impasse bring syntactic, semantic, and pragmatic knowledge to bear by the sequential application of relevant operators. When the impasse is resolved, chunks are built that avoid the impasse in similar, future contexts. These chunks become part of the comprehension operator, integrating in a single operation all the knowledge that was applied sequentially in order to resolve the impasse.

Suppose NL-Soar is given the sentence, *The artist is a barber* and assume there is no comprehension operator available for *barber*. Following the numbers in parentheses in Figure 4-1, let us look at the processing done by the system for that word when it is encountered in context. Extrapolating from the examples in Section 3.1, we know that at the point that *barber* is processed, the situation model contains only a single element with the property of being an artist. The lack of comprehension operator for *barber* creates an impasse in the Comprehension space (1). As a result, a link operator is proposed in the Construct space to tie *barber* into the utterance model as a predicate nominative (2). Before the link can be established, it must meet certain syntactic, semantic, and pragmatic conditions. Thus, another impasse arises leading to further processing in the Constraints space (3). In Constraints, NL-Soar performs a number of syntactic checks, for example, to make sure there is number agreement between the subject and the predicate nominative. Then the system must make certain that the link makes sense, that is, that artists are, in fact, the sorts of things that can be barbers (4).

The knowledge that satisfies the semantic constraint is unavailable in the Constraints space. So an impasse arises (5), operators in the Semantics space are brought to bear, and, as a result of chunking, knowledge from Semantics becomes immediately available in Constraints in future,

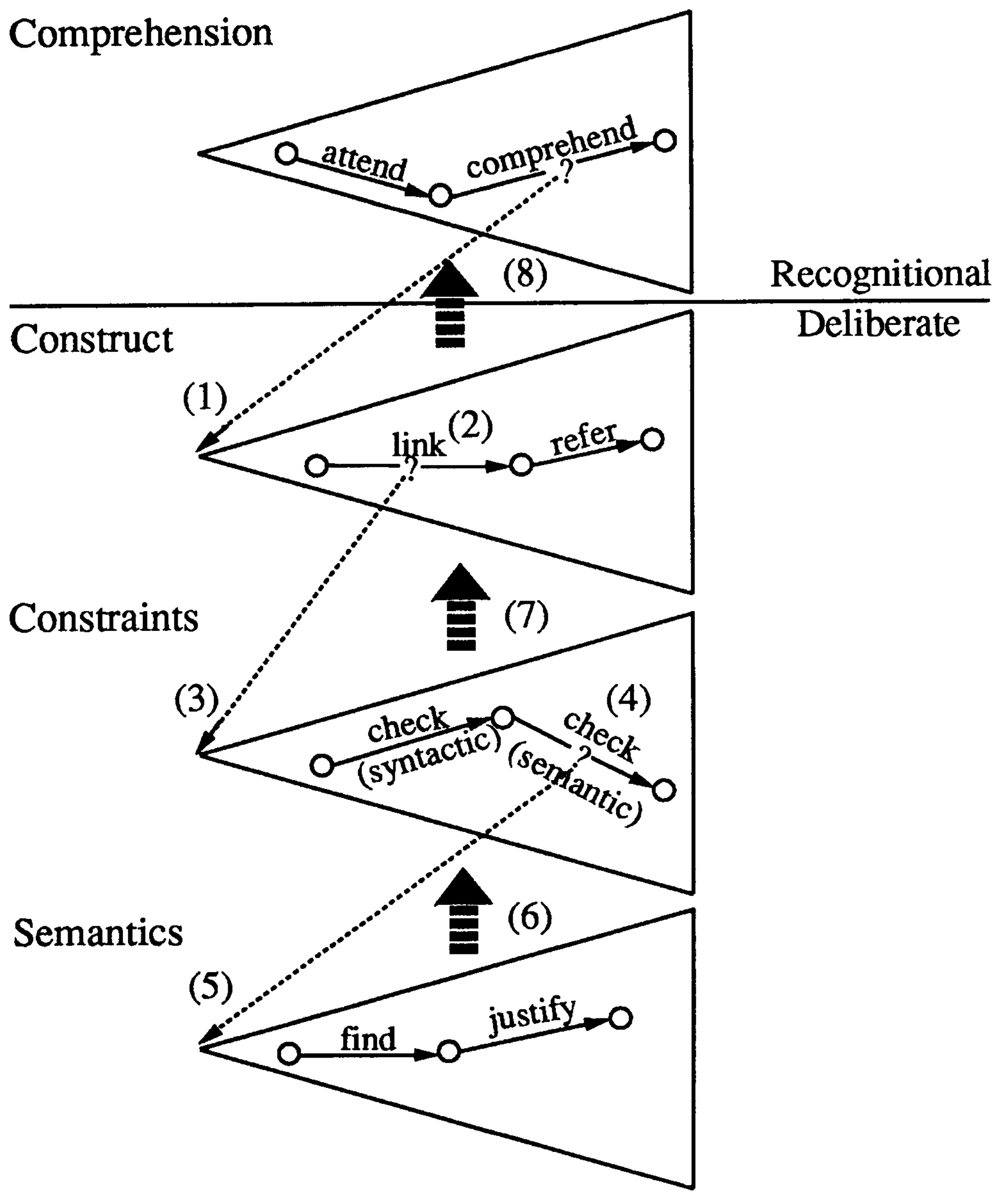

Figure 4-1: NL-Soar, a system for language comprehension.

similar contexts (6).[3] When all of the constraints have been passed, the impasse from Construct to Constraints is resolved, and chunking moves syntactic, semantic, and pragmatic knowledge into the higher space (7). Once the situation model has been augmented by the refer operator in Construct, the impasse from Comprehension to Construct is resolved, and chunking creates a piece of the comprehension operator for *barber* (8). The association that is learned during this last impasse resolution tests for all the conditions that determined the word's meaning in the general context (including the semantic condition that justified making the artist a barber). When those conditions are present in the future, the comprehension operator chunk will produce its changes to the utterance and situation models directly (including the change that adds the property of being a barber). In other words, chunking has moved knowledge from the lower spaces up into recognitional comprehension. But that is exactly what the impasse from Language to Task is all about.

[3]Although it is not obvious from the discussion, pragmatic knowledge is brought to bear in Semantics as well.

4.2. Taskification: Reprise

Before our second digression, we had raised two questions regarding the impasse from Language to Task: how could it arise? and what would it mean? What the NL-Soar example showed is the process by which independent knowledge sources become part of the conditions of a comprehension operator, and, thus, part of the relevant context for assigning a particular meaning. Because this process is essentially invariate over knowledge sources, we are now in a position to answer our questions: a Language to Task impasse arises whenever the task constrains the meaning of a word. The result of such an event is the incorporation of task specific knowledge into the comprehension operator, a process we call *taskification.*

To make the idea of taskification concrete, let us reconsider the second syllogism example and the second premise *Some artists are chefs*. It is certainly possible to assume, as we did implicitly in Section 3.1, that the interpretation a person gives to *some* in this task is just whatever the meaning of *some* would normally be for that person. It is also possible, however, to instruct someone to use a particular meaning of *some*, as in *by "some" we mean only that there is at least one*. How could these instructions be used by someone who did not, naturally, interpret *some* in this way? Figure 4-2 demonstrates.

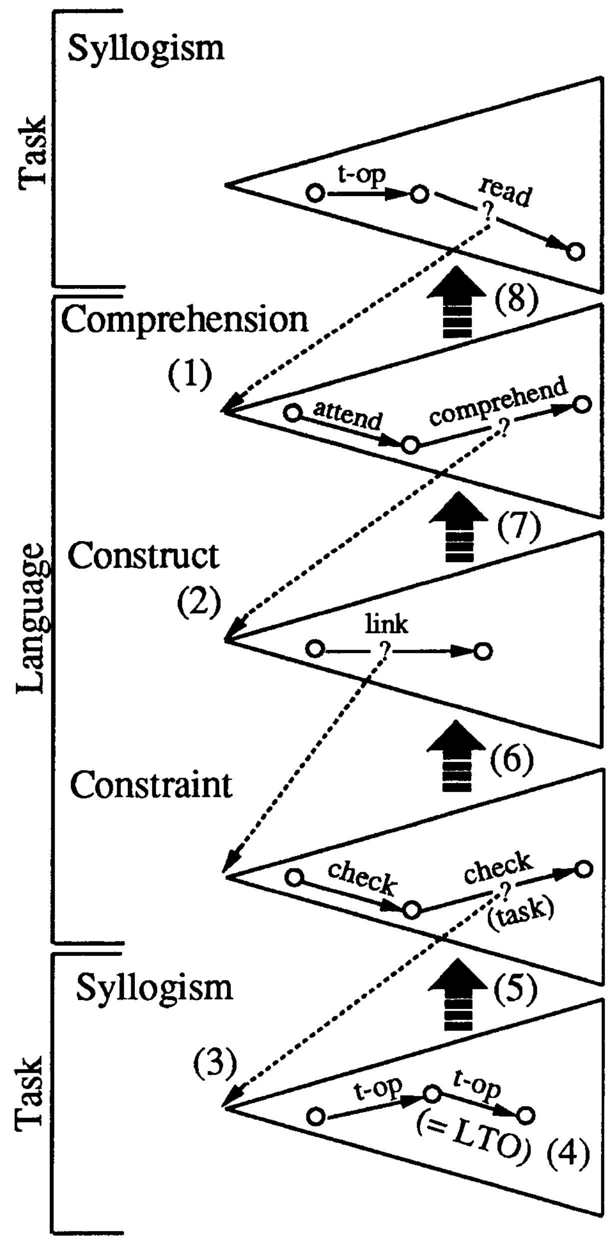

Figure 4-2: The taskification of language.

The reading of this premise occurs as a regular transduction in the context of the syllogism task (1). If the comprehension operator is already sensitive to this context, comprehension proceeds recognitonally, otherwise an impasse occurs (2). When constraint checking is performed during normal deliberate processing, a task constraint causes an impasse just as the semantics constraint did, above (3). Since the Language space does not have the knowledge to resolve the impasse, but the Task space does, the impasse is from Language to Task. Once the Task space has done whatever is needed to satisfy the task constraint (for example, by using an LTO to change the situation model according to the directions (4)), the Language-to-Task impasse is resolved, and the goal stack itself begins to unwind. As we follow chunking back up the problem space hierarchy (5, 6, and 7), the relevant features of the task will be included in the new piece of the comprehension operator for *some*.

It may well be conditional on task.

Abstracting away from this example, we can make a more general statement about what taskification means. As more task contexts arise, more and more of the task will move up into the Language space. Over the long term, the comprehension operator will contain more and more task-specific knowledge for the vocabulary of the task. Thus, as the *taskification* of the person's language proceeds, the apparent modularity between Task and Language disappears.

Refinements

5. Beyond Whorf: Predictions from the architecture

We began our exploration of the role of language in cognition by observing that current practice is to view language as merely a transducer, distinct from cognition and with limited potential for influencing cognition's path. We then countered this weak Whorfian view by appealing to the architecture. What we found there—LTO's and taskification—predicts a decidedly more active role for language than common practice allows.

In essence, LTO's show us that thinking in language is possible. Although this would seem to lead to a dominating role for language in cognition—the strong Whorfian Hypothesis that language determines thought—that simple conclusion is unwarranted. Since LTO's are not required for thought, they are only one possible method for performing a task operation. To the extent that non-linguistic means are used during problem solving, language will have no influence. Thus, the existence of LTO's creates the potential for a determining role for language: the LTO-Whorf Hypothesis, stronger than the weak Whorfian view but weaker than the strong.

good

How much stronger? How much weaker? In part, taskification decides. If we assume that an LTO is used only when it can do the job, then the more taskified the language is, the more often LTO's are applicable and the stronger is language's influence on cognition. At the same time, taskification means that the language through which the LTO's are implemented is, itself, partly task dependent, once again weakening language's role.

But other important limits

What the Soar architecture tells us, then, is two-fold: language can, in fact, determine thought, but its power to do so independent of the task itself varies over time and context. It is not clear that such conclusions would follow from other architectures. Strong modularity, for example, precludes the possibility of taskification and is essentially antithetical to the idea that Task and Language slowly blend into one. Thus, the architecture yields two novel predictions, which are our present to George on the occasion of his retirement.

↑ rephrase a little in terms of the initial set up - should relate to telling George something he has always wanted to know (!) about how the mind works

References

Hayes, J. R., and Simon, H. A. (1975). Understanding written problem instructions. In Gregg, L. W. (Ed.), *Knowledge and Cognition.* Potomac, MD: Erlbaum.

Hunt, E., and Agnoli, F. (1991). The Whorfian hypothesis: A cognitive psychology perspective. *Psychological Review, 90*(3), 377-389.

Lehman, J. Fain, Lewis, R., and Newell, A. (1991). *Natural language comprehension in Soar* (Tech. Rep.). Carnegie Mellon University Technical Report CMU-CS-91-117.

Lehman, J. Fain, Lewis, R., and Newell, A. (1991). Integrating Knowledge Sources in Language Comprehension. *Proceedings of the Thirteenth Annual Conferences of the Cognitive Science Society.* .

Newell, A. (1990). *Unified Theories of Cognition.* Cambridge, Massachusetts: Harvard University Press.

Polk, T. A. and Newell, A. (August 1988). Modeling human syllogistic reasoning in Soar. *Proceedings of the Annual Conference of the Cognitive Science Society.* .

Whorf, B. L. (1956). *Language, Thought, and Reality: Selected Writings.* Cambridge, Mass: Technology Press of M.I.T.

Table of Contents

List of Figures