Text and Discourse Understanding: The DISCERN System *

Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
risto@cs.utexas.edu

1 Introduction

The subsymbolic approach to natural language processing (NLP) captures a number of intriguing properties of human-like information processing such as learning from examples, context sensitivity, generalization, robustness of behavior, and intuitive reasoning. Within this new paradigm, the central issues are quite different from (even incompatible with) the traditional issues in symbolic NLP, and the research has proceeded without much in common with the past. However, the ultimate goal is still the same: to understand how humans process language. Even if NLP is being built on a new foundation, as can be argued, many of the results obtained through symbolic research are still valid, and could be used as a guide for developing subsymbolic models of natural language processing.

This is where DISCERN (DIstributed SCript processing and Episodic memoRy Network [18]), a subsymbolic neural network model of script-based story understanding, fits in. DISCERN is purely a subsymbolic model, but at the high level it consists of modules and information structures similar to those of symbolic systems, such as scripts, lexicon, and episodic memory. At the highest level of natural language processing such as text and discourse understanding, the symbolic and subsymbolic paradigms have to address the same basic issues. Outlining a subsymbolic approach to those issues is the purpose of DISCERN.

In more specific terms, DISCERN aims: (1) to demonstrate that distributed artificial neural networks can be used to build a large-scale natural language processing system that performs approximately at the level of symbolic models; (2) to show that several cognitive phenomena can be explained at the subsymbolic level using the special properties of these networks; and (3) to identify central issues in subsymbolic NLP and to develop well-motivated techniques to deal with them. To the extent that DISCERN is successful in these areas, it constitutes a first step towards building text and discourse understanding systems within the subsymbolic paradigm.

*To appear in R. Dale, H. Moisl and H. Somers (editors), *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text.* New York: Marcel Dekker.

2 The Script Processing Task

Scripts [6; 8; 26] are schemas of often-encountered, stereotypic event sequences, such as visiting a restaurant, traveling by airplane, and shopping at a supermarket. Each script divides further into tracks, or established minor variations. A script can be represented as a causal chain of events with a number of open roles. Script-based understanding means reading a script-based story, identifying the proper script and track, and filling its roles with the constituents of the story. Events and role fillers that were not mentioned in the story but are part of the script can then be inferred. Understanding is demonstrated by generating an expanded paraphrase of the original story, and by answering questions about the story.

To see what is involved in the task, let us consider an example of DISCERN input/output behavior. The following input stories are examples of the fancy-restaurant, plane-travel, and electronics-shopping tracks:

(1) `John went to MaMaison. John asked the waiter for lobster. John left a big tip.`

(2) `John went to LAX. John checked in for a flight to JFK. The plane landed at JFK.`

(3) `John went to Radio-Shack. John asked the staff questions about CD-players. John chose the best CD-player.`

DISCERN reads the orthographic word symbols sequentially, one at a time. An internal representation of each story is formed, where all inferences are made explicit. These representations are stored in the episodic memory. The system then answers questions about the stories:

```
Q: What did John buy at Radio-Shack?
A: John bought a CD-player at Radio-Shack.
Q: Where did John take a plane to?
A: John took a plane to JFK.
Q: How did John like the lobster at MaMaison?
A: John thought the lobster was good at MaMaison.
```

With the question as a cue, the appropriate story representation is retrieved from the episodic memory and the answer is generated word by word. DISCERN also generates full paraphrases of the input stories. For example, it generates an expanded version of the restaurant story:

```
John went to MaMaison. The waiter seated John. John asked the waiter for
lobster. John ate a good lobster. John paid the waiter. John left a big
tip. John left MaMaison.
```

The answers and the paraphrase show that DISCERN has made a number of inferences beyond the original story. For example, it inferred that John ate the lobster and the lobster tasted good. The inferences are not based on specific rules but are statistical and learned from experience. DISCERN has read a number of similar stories in the past and the unmentioned events and role bindings have occurred in most cases. They are assumed immediately and automatically upon reading the story and have become part of the memory of the story. In a similar fashion, human readers often confuse what was mentioned in the story with what was only inferred [3; 10; 11].

A number of issues can be identified from the above examples. Specifically, DISCERN has to (1) make statistical, script-based inferences and account for learning them from experience; (2)

store items in the episodic memory in a single presentation and retrieve them with a partial cue; (3) develop a meaningful organization for the episodic memory, based on the stories it reads; (4) represent meanings of words, sentences, and stories internally; and (5) organize a lexicon of symbol and concept representations based on examples of how words are used in the language and form a many-to-many mapping between them. Script processing constitutes a good framework for studying these issues, and a good domain for developing an approach towards general text and discourse understanding.

3 Approach

Subsymbolic models typically have very little internal structure. They produce the statistically most likely answer given the input conditions in a process that is opaque to the external observer. This is well suited to the modeling of isolated low-level tasks, such as learning past tense forms of verbs or word pronunciation [25; 27]. Given the success of such models, a possible approach to higher-level cognitive modeling would be to construct the system from several submodules that work together to produce the higher-level behavior.

In DISCERN, the immediate goal is to build a complete, integrated system that performs well in the script processing task. In this sense, DISCERN is very similar to traditional models in artificial intelligence. However, DISCERN also aims to show how certain parts of human cognition could actually be built. The components of DISCERN were designed as independent cognitive models that can account for interesting language processing and memory phenomena, many of which are not even required in the DISCERN task. Combining these models into a single, working system is one way of validating them. In DISCERN, the components are not just models of isolated cognitive phenomena; they are sufficient constituents for generating complex high-level behavior.

4 The Discern System

DISCERN can be divided into parsing, generating, question answering, and memory subsystems, each with two modules (figure 1). Each module is trained in its task separately and in parallel. During performance, the modules form a network of networks, each feeding its output to the input of another module.

The sentence parser reads the input words one at a time and forms a representation of each sentence. The story parser combines the sequence of sentences into an internal representation of the story, which is then stored in the episodic memory. The story generator receives the internal representation and generates the sentences of the paraphrase one at a time. The sentence generator outputs the sequence of words for each sentence. The cue former receives a question representation, built by the sentence parser, and forms a cue pattern for the episodic memory, which returns the appropriate story representation. The answer producer receives the question and the story and generates an answer representation, which is output word by word by the sentence generator. The architecture and behavior of each of these modules in isolation is outlined below.

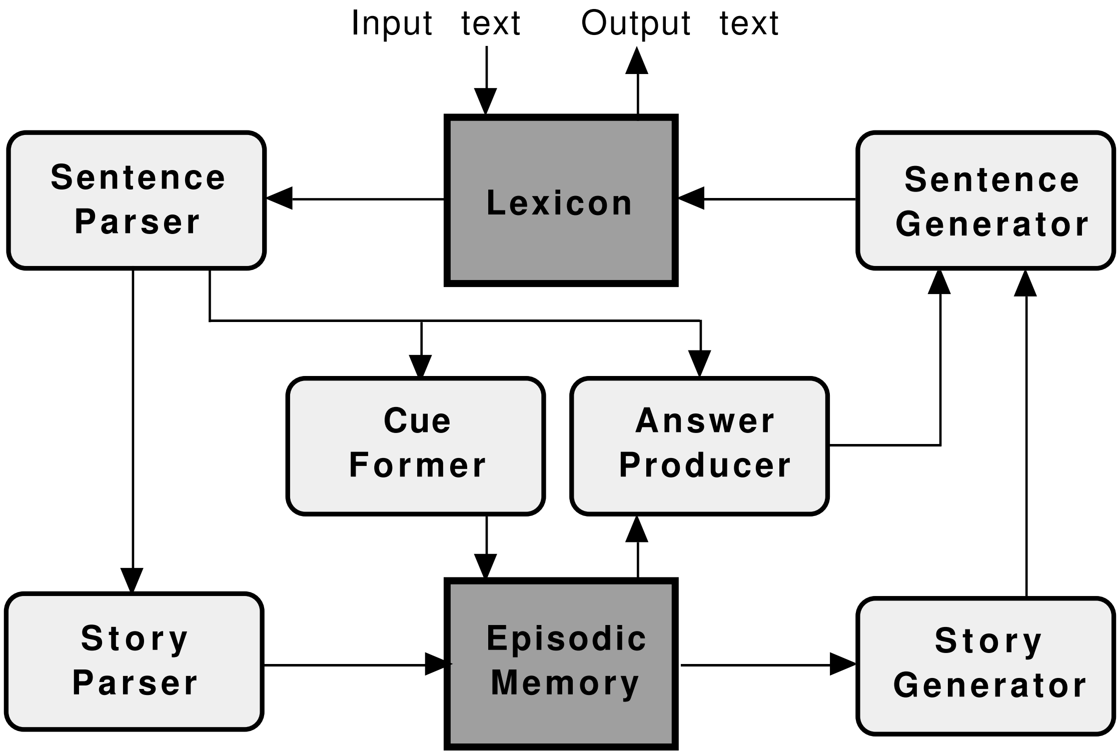


Figure 1: **The DISCERN architecture.** The system consists of parsing, generating, question answering, and memory subsystems, two modules each. A dark square indicates a memory module, a light square stands for a processing module. The lines indicate pathways carrying distributed word, sentence, and story representations during the performance phase of the system. The modules are trained separately with compatible I/O data.

5 Lexicon

The input and output of DISCERN consist of distributed representations for orthographic word symbols (also called lexical words). Internally, DISCERN processes semantic concept representations (semantic words). Both the lexical and semantic words are represented distributively as vectors of gray-scale values between 0.0 and 1.0. The lexical representations are based on the visual patterns of characters that make up the written word; they remain fixed throughout the training and performance of DISCERN. The semantic representations stand for distinct meanings and are developed automatically by the system while it is learning the processing task.

The lexicon stores the lexical and semantic representations and translates between them (figure 2; [20]). It is implemented as two feature maps [12; 13], one lexical and the other semantic. Words whose lexical forms are similar, such as `LINE` and `LIKE`, are represented by nearby units in the lexical map. In the semantic map, words with similar semantic content, such as `John` and `Mary` or `Leone's` and `MaMaison` are mapped near each other. There is a dense set of associative interconnections between the two maps. A localized activity pattern representing a word in one map will cause a localized activity pattern to form in the other map, representing the same word. The output representation is then obtained from the weight vector of the most highly active unit. The lexicon thus transforms a lexical input vector into a semantic output vector and vice versa.

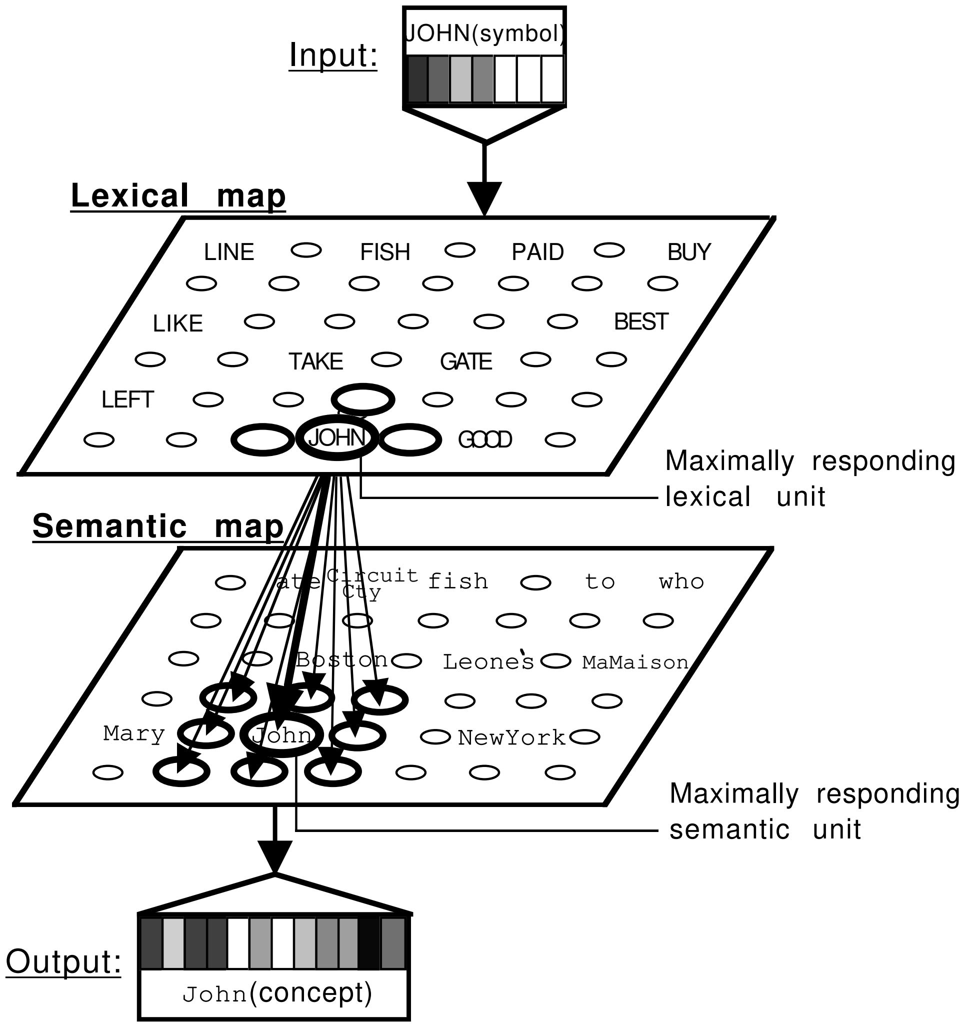


Figure 2: **The lexicon.** The lexical input symbol `JOHN` is translated into the semantic representation of the concept `John`. The representations are vectors of gray-scale values between 0.0 and 1.0, stored in the weights of the units. The size of the unit on the map indicates how strongly it responds. Only a small part of each map, and only a few strongest associative connections of the lexical unit `JOHN` are shown in this figure.

Both maps and the associative connections between them are organized simultaneously, based on examples of co-occurring symbols and meanings.

The lexicon architecture facilitates interesting behavior. Localized damage to the semantic map results in category-specific lexical deficits similar to human aphasia [4; 15]. For example, the system selectively loses access to restaurant names, or animate words, when that part of the map is damaged. Dyslexic performance errors can also be modeled. If the performance is degraded, for example, by adding noise to the connections, parsing and generation errors that occur are quite similar to those observed in human deep dyslexia [5]. For example, the system may confuse `Leone's` with `MaMaison`, or `LINE` with `LIKE`, because they are nearby in the map and share similar associative connections.

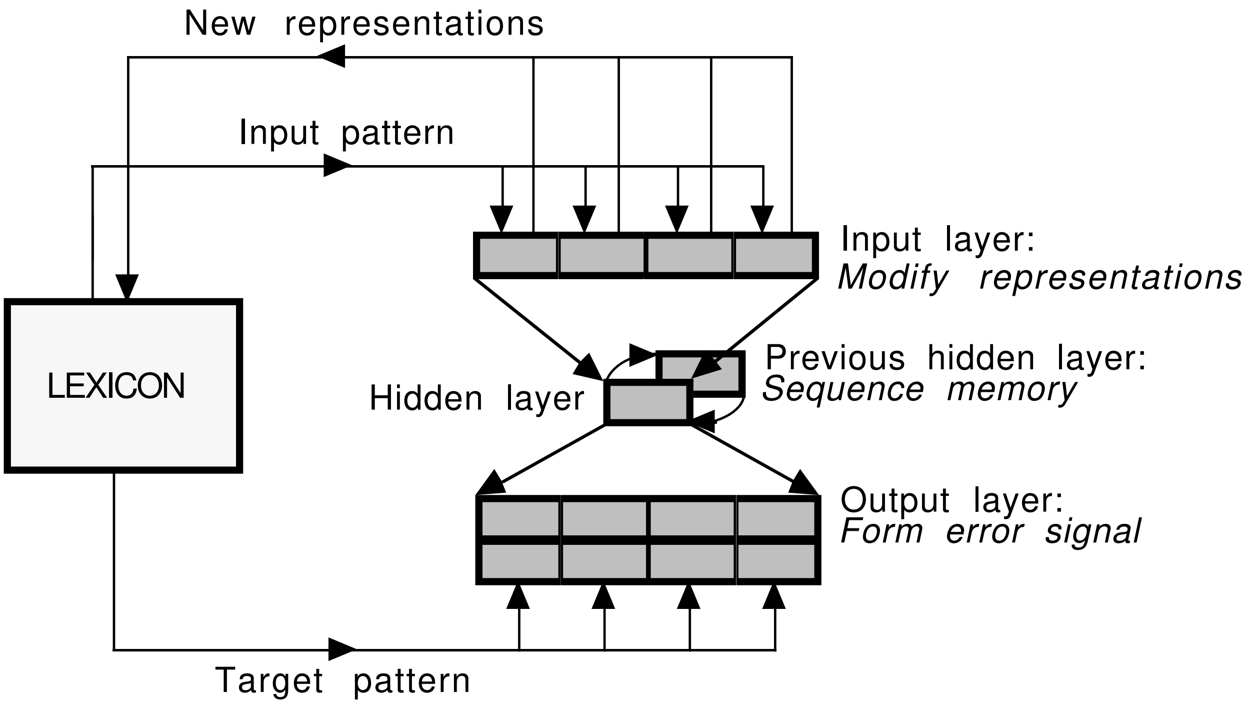


Figure 3: **The FGREP-module.** At each I/O presentation, the representations at the input layer are modified according to the backpropagation error signal, and replace the old representations in the lexicon. In the case of sequential input or output, the hidden layer pattern is saved after each step in the sequence, and used as input to the hidden layer during the next step, together with the actual input.

6 FGREP Processing Modules

Processing in DISCERN is carried out by hierarchically organized pattern-transformation networks. Each module performs a specific subtask, such as parsing a sentence or generating an answer to a question. All these networks have the same basic architecture: they are three-layer, simple-recurrent backpropagation networks [9], with the extension called FGREP that allows them to develop distributed representations for their input/output words.

The FGREP mechanism (Forming Global Representations with Extended backPropagation, [21]) is based on a basic three-layer backward error propagation network, with the I/O representation patterns stored in an external lexicon (figure 3). The input and output layers of the network are divided into assemblies (i.e. groups of units). The assemblies stand for slots in the I/O representation, such as the different case roles of the sentence representation, or the role bindings in the story representation. Each input pattern is formed by concatenating the current semantic lexicon entries of the input words; likewise, the corresponding target pattern is formed by concatenating the lexicon entries of the target words. For example, the target sentence `John went to Mamaison` would be represented at the output of the sentence parser network as a single vector, formed by concatenating the representations for `John`, `went`, and `MaMaison` (figure 7a).

Three types of FGREP modules are used in the system: non-recurrent (the cue former and the answer producer), sequential input (the parsers), and sequential output modules (the generators). In the recurrent modules the previous hidden layer serves as sequence memory, remembering where in the sequence the system currently is and what has occurred before (figure 3). In a sequential input network, the input changes at each time step, while the target pattern stays the same. The network learns to form a stationary representation of the sequence. In a sequential output network, the input is stationary, but the teaching pattern changes at each step. The network learns to produce a sequential interpretation of its input.

The network learns the processing task by adapting the connection weights according to the standard on-line backpropagation procedure [24]. The error signal is propagated to the input layer, and the current input representations are modified as if they were an extra layer of weights. The modified representation vectors are put back in the lexicon, replacing the old representations. Next time the same words occur in the input or output, their new representations are used to form the input/output patterns for the network. In FGREP, therefore, the required mappings change as the representations evolve, and backpropagation is shooting at a moving target.

The representations that result from this process have a number of useful properties for natural language processing. (1) Since they adapt to the error signal, they end up coding information most crucial to the task. Representations for words that are used in similar ways in the examples become similar. Thus, these profiles of continuous activity values can be claimed to code the meanings of the words as well. (2) As a result, the system never has to process very novel input patterns, because generalization has already been done in the representations. (3) The representation of a word is determined by all the contexts in which that word has been encountered; consequently, it is also a representation of all those contexts. Expectations emerge automatically and cumulatively from the input word representations. (4) Single representation components do not usually stand for identifiable semantic features. Instead, the representation is holographic: word categories can often be recovered from the values of single components. (5) Holography makes the system very robust against noise and damage. Performance degrades approximately linearly as representation components become defective or inaccurate.

After a core set of semantic representations have been developed in the FGREP process, it is possible to extend the vocabulary through a technique called cloning. Each FGREP representation stands for a unique meaning and constitutes a semantic prototype. In cloning, several distinct copies, or instances, are created from the same prototype. For example, instances such as `John`, `Mary`, and `Bill` can be created from the prototype `human`. Such cloned representations consist of two parts: the content part, which was developed in the FGREP process and encodes the meaning of the word, and the ID part, which is unique for each instance of the same prototype. They all share the same meaning, and therefore the system knows how to process them, but at the same time, the ID part allows the system to keep them distinct. The ID+content technique can be applied to any word in the training data, and in principle, the number of instances per word is unlimited. This allows us to approximate a large vocabulary with only a small number of semantically different representations (which are expensive to develop) at our disposal.

7 Episodic Memory

The episodic memory in DISCERN [16; 17] consists of a hierarchical pyramid of feature maps organized according to the taxonomy of script-based stories (figure 4). The highest level of the hierarchy is a single feature map that lays out the different script classes. Beneath each unit of this map there is another feature map that lays out the tracks within the particular script. The

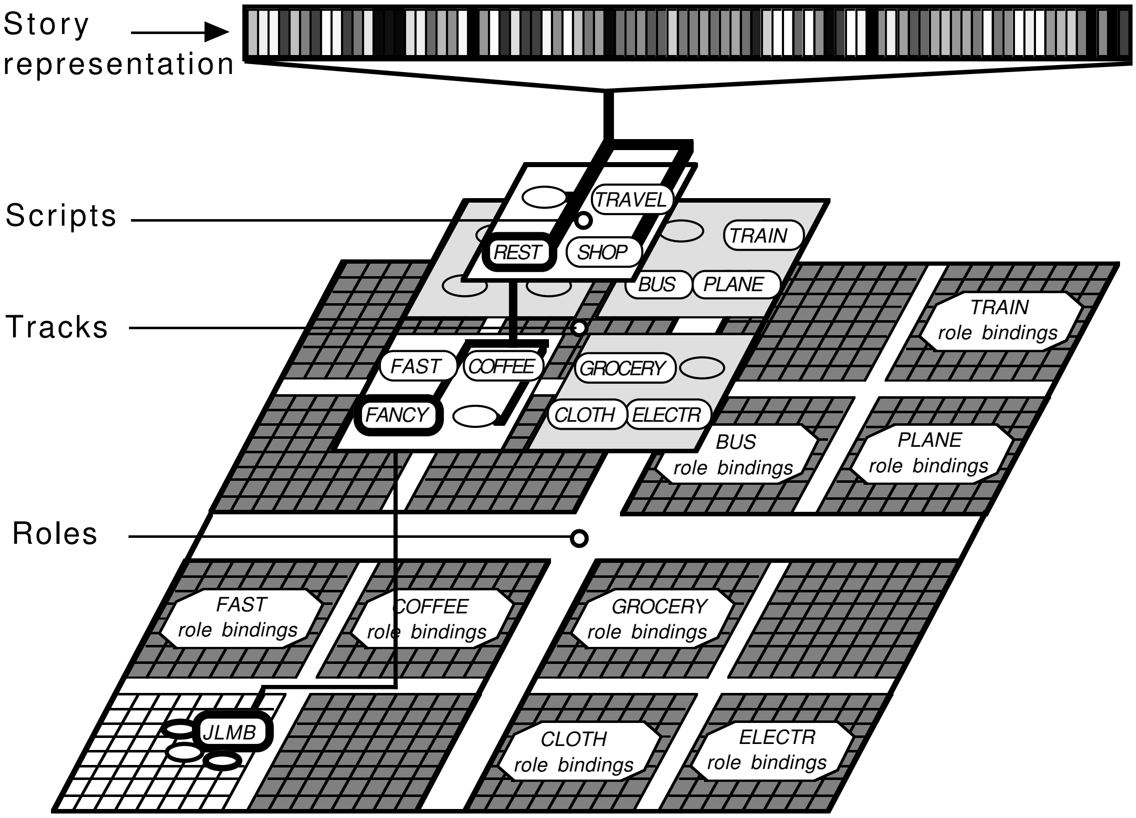
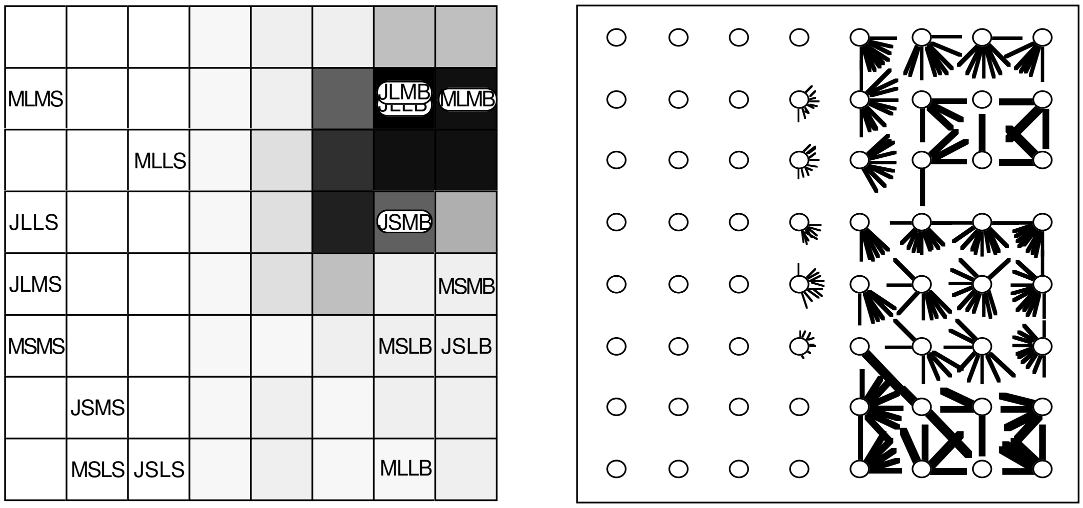


Figure 4: **The hierarchical feature map classification of script-based stories.** Labels indicate the maximally responding unit for the different scripts and tracks. This particular input story representation is classified as an instance of the restaurant script (top level) and fancy-restaurant track (middle level), with role bindings customer=`John`, food=`lobster`, restaurant=`MaMaison`, tip=`big` (i.e., unit JLMB, bottom level).

different role bindings within each track are separated at the bottom level. The map hierarchy receives a story representation vector as its input and classifies it as an instance of a particular script, track, and role binding. The hierarchy thereby provides a unique memory representation for each script-based story as the maximally responding units in the feature maps at the three levels.

Whereas the top and the middle level in the hierarchy only serve as classifiers, selecting the appropriate track and role-binding map for each input, at the bottom level a permanent trace of the story must be created. The role-binding maps are trace feature maps, with modifiable lateral connections (figure 5). When the story representation vector is presented to a role-binding map, a localized activity pattern forms as a response. Each lateral connection to a unit with higher activity is made excitatory, while a connection to a unit with lower activity is made inhibitory. The units within the response now "point" towards the unit with highest activity, permanently encoding that the story was mapped at that location.

A story is retrieved from the episodic memory by giving it a partial story representation as a cue. Unless the cue is highly deficient, the map hierarchy is able to recognize it as an instance of the correct script and track and form a partial cue for the role-binding map. The trace feature map mechanism then completes the role binding. The initial response of the map is again a localized



(a) (b)

Figure 5: **The trace feature map for fancy-restaurant stories.** (a) The organization of the map. Each story in the test set were mapped on one unit in the map, labeled by the role bindings (where `J=John`, `M=Mary`, `L=lobster`, `S=steak`, `M=Mamaison`, `L=Leone's`, `B=big`, and `S=small` (for size of tip)). The activation of the units (shown in gray-scale) indicates retrieval of the JLMB story. (b) Lateral connections after storing first JLMB and then MLLB. Line segments indicate excitatory lateral connections originating from each unit, length and width proportional to the magnitude of the weight. Inhibitory connections are not shown. The latter trace has partially obscured the earlier one.

activity pattern; because the map is topological, it is likely to be located somewhere near the stored trace. If the cue is close enough, the lateral connections pull the activity to the center of the stored trace. The complete story representation is retrieved from the weight vectors of the maximally responding units at the script, track, and role-binding levels.

Hierarchical feature maps have a number of properties that make them useful for memory organization: (1) The organization is formed in an unsupervised manner, extracting it from the input experience of the system. (2) The resulting order reflects the properties of the data, the hierarchy corresponding to the levels of variation, and the maps laying out the similarities at each level. (3) By dividing the data first into major categories and gradually making finer distinctions lower in the hierarchy, the most salient components of the input data are singled out and more resources are allocated for representing them accurately. (4) Because the representation is based on salient differences in the data, the classification is very robust, and usually correct even if the input is noisy or incomplete. (5) Because the memory is based on classifying the similarities and storing the differences, retrieval becomes a reconstructive process [14; 28] similar to human memory.

The trace feature map exhibits interesting memory effects that result from interactions between traces. Later traces capture units from earlier ones, making later traces more likely to be retrieved (figure 5). The extent of the traces determines memory capacity. The smaller the traces, the more of them will fit in the map, but more accurate cues are required to retrieve them. If the memory capacity is exceeded, older traces will be selectively replaced by newer ones. Traces that are unique, that is, located in a sparse area of the map, are not affected, no matter how old they are. Similar effects are common in human long-term memory [2; 22].

8 DISCERN High-Level Behavior

DISCERN is more than just a collection of individual cognitive models. Interesting behavior results from the interaction of the components in a complete story-processing system. Let us follow DISCERN as it processes the story about John's visit to MaMaison (figure 6). The lexical representations for each word are presented to the lexical map of the lexicon, which produces the corresponding semantic representation as its output (figure 2). These are fed one at a time to the sentence parser, which gradually forms a stationary case-role representation of each sentence at its output layer (figure 7a). After a period is input, ending the sentence, the final case-role pattern is fed to the input of the story parser.

In a similar manner, the story parser receives a sequence of sentence case-role representations as its input, and forms a stationary slot-filler representation of the whole story at its output layer (figure 7b). This is a representation of the story in terms of its role bindings, and constitutes the final result of the parse. The story representation is fed to the episodic memory, which classifies it as an instance of a particular script, track, and role binding, and creates a trace in the appropriate role-binding map (figures 4 and 5b).

The generator subsystem reverses the parsing process. The story generator network receives the story representation as its input and generates a sequence of sentence case-role representations. Each of these is fed to the sentence generator, which outputs the semantic representations of the output words one at a time. Finally, the lexicon transforms these into lexical words.

The sentence parser and the sentence generator are also trained to process question sentences and answer sentences. The cue former receives the case-role representation of the question (figure 7c), produced by the sentence parser, and generates an approximate story representation as its output (figure 7d). This pattern is fed to the episodic memory, which classifies it as an instance of a script, track, and role binding. The trace feature map settles to a previously stored memory trace (figure 5a), and the complete story representation (figure 7b) is retrieved from the weights of the maximally responding units at the three levels of the hierarchy (figure 4). The answer producer receives the complete story representation, together with the case-role representation of the question, and generates a case-role representation of the answer sentence (figure 7e), which is then output word by word by the sentence generator.

DISCERN was trained and tested with an artificially generated corpus of script-based stories consisting of three scripts (restaurant, shopping, and travel), with three tracks and three open roles each.

The complete DISCERN system performs very well: at the output, 98 percent of the words are correct. This is rather remarkable for a chain of networks that is 9 modules long and consists of several different types of modules.

A modular neural network system can only operate if it is stable, that is, if small deviations from the normal flow of information are automatically corrected. It turns out that DISCERN has several built-in safeguards against minor inaccuracies and noise. The semantic representations are distributed and redundant, and inaccuracies in the output of one module are cleaned up by the module that uses the output. The memory modules clean up by categorical processing: a noisy input is recognized as a representative of an established class and replaced by the correct representation of that class. As a result, small deviations do not throw the system off course, but

```
[ parsing input story: ]                                  (comment by DISCERN)
JOHN WENT TO MAMAISON .                                   (sequence of input words)
|John went _ _ _ MaMaison|                                (sentence parser output)
JOHN ASKED THE WAITER FOR LOBSTER .
|John asked waiter _ lobster _|
JOHN LEFT A BIG TIP .
|John left waiter big tip _|                              (figure 7a)

[ into internal representation: ]
|$restaurant $fancy John lobster MaMaison good big|  (story parser output; figure 7b)
[ storing into episodic memory: ]
[ image units (0,0), (0,0), (6,6): ]                      (location in the memory; figure 4)
|$restaurant $fancy John lobster MaMaison good big|  (episodic memory representation)
[ generating paraphrase: ]
|John went _ _ _ MaMaison|                                (story generator output)
JOHN WENT TO MAMAISON .                                   (sequence of output words)
|waiter seated _ _ John _|
THE WAITER SEATED JOHN .
|John asked waiter _ lobster _|
JOHN ASKED THE WAITER FOR LOBSTER .
|John ate _ good lobster _|
JOHN ATE A GOOD LOBSTER .
|John paid waiter _ _ _|
JOHN PAID THE WAITER .
|John left waiter big tip _|
JOHN LEFT A BIG TIP .
|John left _ _ _ MaMaison|
JOHN LEFT MAMAISON .

[ parsing question: ]
WHAT DID JOHN EAT AT MAMAISON ?
|John ate _ _ what MaMaison|                              (sentence parser output; figure 7c)

[ into cue: ]
|$restaurant $fancy John (?)  MaMaison good (?)|          (cue former output; figure 7d)
[ retrieving from episodic memory: ]
[ image units (0,0), (0,0), (6,6): ]                      (location in the memory; figures 4 and 5a)
|$restaurant $fancy John lobster MaMaison good big|  (episodic memory output; figure 7b)
[ generating answer: ]
|John ate _ good lobster _|                               (answer producer output; figure 7e)
JOHN ATE A GOOD LOBSTER .
```

Figure 6: **A listing of DISCERN output as it processes the example story.** Comments printed out by DISCERN are enclosed in brackets; otherwise all words stand for activity patterns, with "_" indicating the blank pattern. Output layers consisting of several assemblies are enclosed between bars.

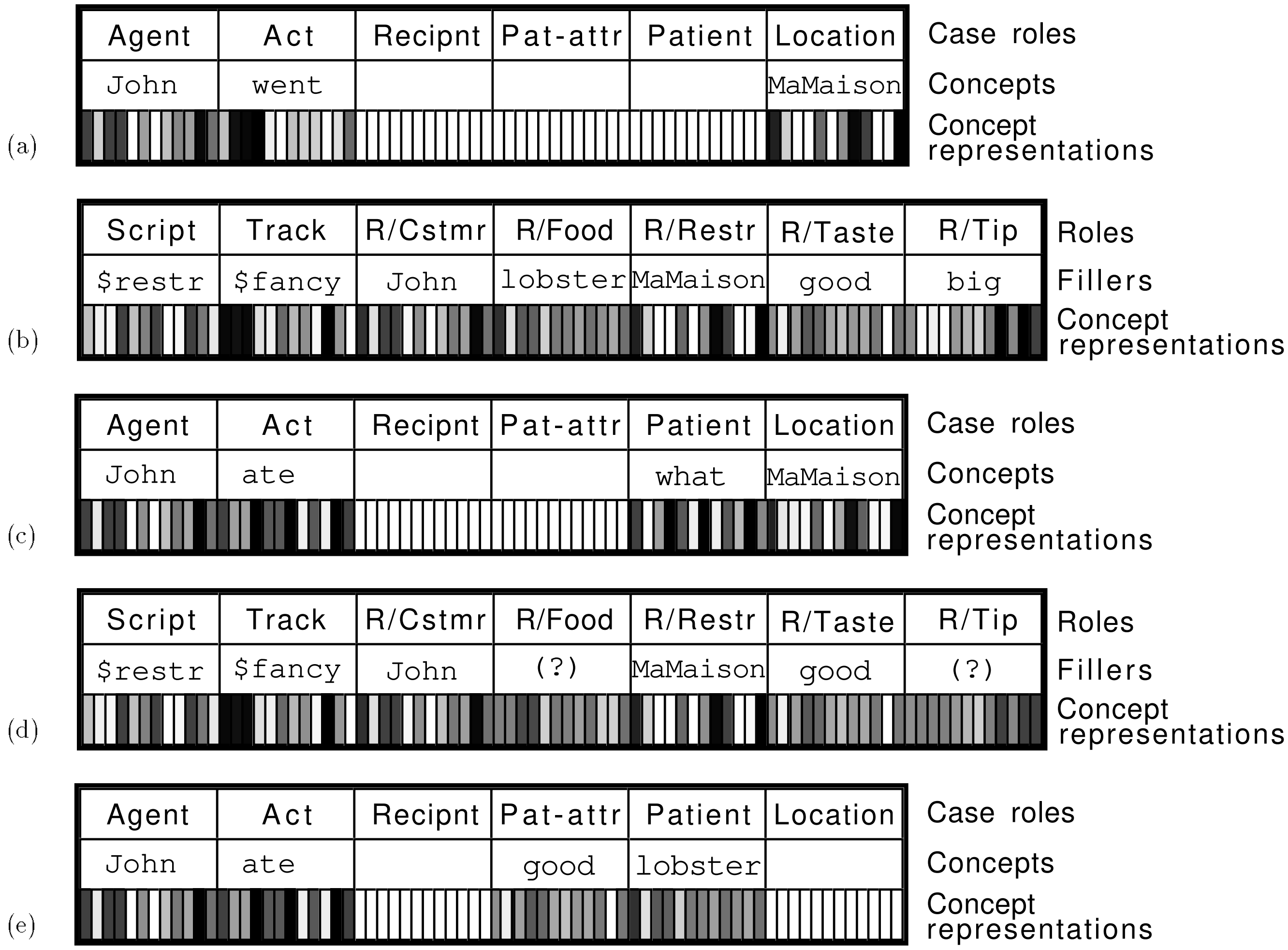


Figure 7: **Sentence and story representations** (a) Case-role representation of the sentence `John went to MaMaison.` The concept representations in each case-role correspond to the concept representations in the lexicon. (b) Representation of the story by its role bindings. The assemblies are data-specific: their interpretation depends on the pattern in the script slot. The role names R/... are specific for the restaurant script. (c) Case-role representation of the question `What did John eat at MaMaison?` Questions are represented as sentences, but processed through a different pathway (see figure 1). (d) Memory cue. Most of the story representation is complete, but the patterns in Food and Tip slots indicate averages of all possible alternatives. (e) Case-role representation of the answer `John ate a good lobster.`

rather the system filters out the errors and returns to the normal course of processing, which is an essential requirement for building robust natural language processing models.

DISCERN also demonstrates strong script-based inferencing [19]. Even when the input story is incomplete, consisting of only a few main events, DISCERN can usually form an accurate internal representation of it. DISCERN was trained to form complete story representations from the first sentence on, and because the stories are stereotypical, missing sentences have little effect on the parsing process. Once the story representation has been formed, DISCERN performs as if the script had been fully instantiated. Questions about missing events and role-bindings are answered as if they were part of the original story. If events occurred in an unusual order, they are recalled in the stereotypical order in the paraphrase. If there is not enough information to fill a role, the most likely filler is selected and maintained throughout the paraphrase generation. Such behavior automatically results from the modular architecture of DISCERN and is consistent with experimental observations on how people remember stories of familiar event sequences [3; 10; 11].

In general, given the information in the question, DISCERN recalls the story that best matches it in the memory. An interesting issue is: what happens when DISCERN is asked a question that is inaccurate or ambiguous, that is, one that does not uniquely specify a story? For example, DISCERN might have read a story about John eating lobster at MaMaison, and then about Mary doing the same at Leone's, and the question could be `Who ate lobster?` Because later traces are more prominent in the memory, DISCERN is more likely to retrieve the Mary-at-Leone's story in this case (figure 5b). The earlier story is still in the memory, but to recall it, more details need to be specified in the question, such as `Who ate lobster at MaMaison?` Similarly, DISCERN can robustly retrieve a story even if the question is slightly inaccurate. When asked `How did John like the steak at MaMaison?`, DISCERN generates the answer `John thought lobster was good at MaMaison`, ignoring the inaccuracy in the question, because the cue is still close enough to the stored trace. DISCERN does recognize, though, when a question is too different from anything in the memory, and should not be answered. For `Who ate at McDonald's?`, the cue vector is not close to any trace, the memory does not settle, and nothing is retrieved. Note that these mechanisms were not explicitly built into DISCERN, but they emerge automatically from the physical layout of the architecture and representations.

9 Discussion

There is an important distinction between scripts (or more generally, schemas) in symbolic systems, and scripts in subsymbolic models such as DISCERN. In the symbolic approach, a script is stored in memory as a separate, exact knowledge structure, coded by the knowledge engineer. The script has to be instantiated by searching the schema memory sequentially for a structure that matches the input. After instantiation, the script is active in the memory and later inputs are interpreted primarily in terms of this script. Deviations are easy to recognize and can be taken care of with special mechanisms.

In the subsymbolic approach, schemas are based on statistical properties of the training examples, extracted automatically during training. The resulting knowledge structures do not have explicit representations. For example, a script exists in a neural network only as statistical correlations coded in the weights. Every input is automatically matched to every correlation in parallel. There is no all-or-none instantiation of a particular knowledge structure. The strongest, most probable correlations will dominate, depending on how well they match the input, but all of them are simultaneously active at all times. Regularities that make up scripts can be particularly well captured by such correlations, making script-based inference a good domain for the subsymbolic approach. Generalization and graceful degradation give rise to inferencing that is intuitive, immediate, and occurs without conscious control, as script-based inference in humans. On the other hand, it is very difficult to recognize deviations from the script and to initiate exception-processing when the automatic mechanisms fail. Such sequential reasoning would require intervention of a high-level "conscious" monitor, which has yet to be built in the connectionist framework.

10 Conclusion

The main conclusion from DISCERN is that building subsymbolic models is a feasible approach to understanding mechanisms underlying natural language processing. DISCERN shows how several cognitive phenomena may result from subsymbolic mechanisms. Learning word meanings, script processing, and episodic memory organization are based on self-organization and gradient-descent in error in this system. Script-based inferences, expectations, and defaults automatically result from generalization and graceful degradation. Several types of performance errors in role binding, episodic memory, and lexical access emerge from the physical organization of the system. Perhaps most significantly, DISCERN shows how individual connectionist models can be combined into a large, integrated system that demonstrates that these models are sufficient constituents for generating sequential, symbolic, high-level behavior.

Although processing simple script instantiations is a start, there is a long way to go before subsymbolic models will rival the best symbolic cognitive models. For example, in story understanding, symbolic systems have been developed that analyze realistic stories in depth, based on higher-level knowledge structures such as goals, plans, themes, affects, beliefs, argument structures, plots, and morals (e.g. [1; 7; 23; 26]). In designing subsymbolic models that would do that, we are faced with two major challenges [18]: (1) how to implement connectionist control of high-level processing strategies (making it possible to model processes more sophisticated than a series of reflex responses), and (2) how to represent and learn abstractions (making it possible to process information at a higher level than correlations in the raw input data). Progress in these areas would constitute a major step towards extending the capabilities of subsymbolic natural language processing models beyond those of DISCERN.

Acknowledgements

This research was supported in part by the Texas Higher Education Coordinating Board under grant ARP-444.

Note

Software for the DISCERN system is available in the World Wide Web at URL http://www.cs.utexas.edu/users/nn/pages/software/nn-software.html. An interactive X11 graphics demo, showing DISCERN in processing example stories and questions, can be run remotely under the World Wide Web at http://www.cs.utexas.edu/users/nn/pages/demos/discern/discern.html.

References

[1] S. Alvarado, M. G. Dyer, and M. Flowers. Argument representation for editorial text. *Knowledge-Based Systems*, 3:87–107, 1990.

[2] A. D. Baddeley. *The Psychology of Memory.* Basic Books, New York, 1976.

[3] G. H. Bower, J. B. Black, and T. J. Turner. Scripts in memory for text. *Cognitive Psychology*, 11:177–220, 1979.

[4] A. Caramazza. Some aspects of language processing revealed through the analysis of acquired aphasia: The lexical system. *Annual Review of Neuroscience*, 11:395–421, 1988.

[5] M. Coltheart, K. Patterson, and J. C. Marshall, editors. *Deep Dyslexia.* Routledge and Kegan Paul, London; New York, second edition, 1988.

[6] R. E. Cullingford. *Script Application: Computer Understanding of Newspaper Stories.* PhD thesis, Department of Computer Science, Yale University, New Haven, CT, 1978. Technical Report 116.

[7] M. G. Dyer. *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension.* MIT Press, Cambridge, MA, 1983.

[8] M. G. Dyer, R. E. Cullingford, and S. Alvarado. Scripts. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 980–994. Wiley, New York, 1987.

[9] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[10] A. C. Graesser, S. E. Gordon, and J. D. Sawyer. Recognition memory for typical and atypical actions in scripted activities: Tests for the script pointer+tag hypothesis. *Journal of Verbal Learning and Verbal Behavior*, 18:319–332, 1979.

[11] A. C. Graesser, S. B. Woll, D. J. Kowalski, and D. A. Smith. Memory for typical and atypical actions in scripted activities. *Journal of Experimental Psychology: Human Learning and Memory*, 6:503–515, 1980.

[12] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78:1464–1480, 1990.

[13] T. Kohonen. *Self-Organizing Maps.* Springer, Berlin; Heidelberg; New York, 1995.

[14] J. L. Kolodner. *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.* Erlbaum, Hillsdale, NJ, 1984.

[15] R. A. McCarthy and E. K. Warrington. *Cognitive Neuropsychology: A Clinical Introduction.* Academic Press, New York, 1990.

[16] R. Miikkulainen. Script recognition with hierarchical feature maps. *Connection Science*, 2:83–101, 1990.

[17] R. Miikkulainen. Trace feature map: A model of episodic associative memory. *Biological Cybernetics*, 67:273–282, 1992.

[18] R. Miikkulainen. *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory.* MIT Press, Cambridge, MA, 1993.

[19] R. Miikkulainen. Script-based inference and memory retrieval in subsymbolic story processing. *Applied Intelligence*, 5:137–163, 1995.

[20] R. Miikkulainen. Dyslexic and category-specific impairments in a self-organizing feature map model of the lexicon. *Brain and Language*, in press.

[21] R. Miikkulainen and M. G. Dyer. Natural language processing with modular neural networks and distributed lexicon. *Cognitive Science*, 15:343–399, 1991.

[22] L. Postman. Transfer, interference and forgetting. In J. W. Kling and L. A. Riggs, editors, *Woodworth and Schlosberg's Experimental Psychology*, pages 1019–1132. Holt, Rinehart and Winston, New York, third edition, 1971.

[23] J. F. Reeves. *Computational Morality: A Process Model of Belief Conflict and Resolution for Story Understanding.* PhD thesis, Computer Science Department, University of California, Los Angeles, 1991. Technical Report UCLA-AI-91-05.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[25] David E. Rumelhart and James L. McClelland. On learning past tenses of English verbs. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2: Psychological and Biological Models*, pages 216–271. MIT Press, Cambridge, MA, 1986.

[26] R. C. Schank and R. P. Abelson. *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures.* Erlbaum, Hillsdale, NJ, 1977.

[27] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.

[28] M. D. Williams and J. D. Hollan. The process of retrieval from very long-term memory. *Cognitive Science*, 5:87–119, 1981.