A Reading Agent

Tamitha Carpenter
tamitha@cs.brandeis.edu
(617) 736-2718

Richard Alterman*
alterman@cs.brandeis.edu
(617) 736-2703

Department of Computer Science
and Center for Complex Systems
Brandeis University
Waltham, MA 02254

Abstract

Recent work in agency has explored the interactive nature of goal-driven behavior ("activity"). An interactive agent is responsive to events and is affected by and effects the context in which it exists. Our contention is that interaction is also an important characteristic of a reader. By treating reading as an activity, the *reading agent* can interact with the text, achieving goals and planning what to read, when to read, and how to read. This paper will discuss how reading within a context of activity provides goals and enables planning, thus creating the reading agent. The system described, SPRITe, reads natural language, primarily instructions, in order to facilitate other activities.

Introduction

Recent work in agency (e.g., Suchman, 1987; Agre, 1988; Chapman, 1990) has explored the interactive nature of goal-driven behavior ("activity"). When reading is viewed as an activity, the same issues come into play. In order to define a "reading agent", it is important to establish how an agent interacts with text, how reading interacts with action, and how activity changes the reading process.

For example, consider trying to read the instructions for assembling a lawnmower. If reading occurs in isolation from the assembling activity, the instructions must be read from beginning to end, they can only be understood in general terms (e.g., the reference to a "bolt" can only be concrete once the reader has experienced the bolt used in the lawnmower), and the understanding that results may or may not be adequate for the agent to successfully assemble the lawnmower without referring back to the instructions.

A *reading agent* has several properties that make the reading process active. These include:

- Using goals.
- Planning the reading activity.
- Reading in the context of a larger activity.
- Improving the reading skill.
- Responding to changes in the environment (interactivity).

Other systems have explored some of these properties. Maybury (1991) shows a planning system that *creates* instructions (as well as other types of text). In Ram (1991), knowledge goals were used to take advantage of information as it was read; however, the reading was not a planned activity, and so text was read sequentially from beginning to end. Several other systems have developed models of instruction usage (e.g., Chapman, 1990; Badler et al., 1990; Vere and Bickmore, 1990). However, these models have not taken the step towards building a reading agent that *interacts* with the instructions, using goals and planning strategies.

The properties of a reading agent have been implemented in a model called SPRITe (using Structure to Plan the Reading of Instructional TExt). Instead of reading sequentially, SPRITe uses reading goals and planning techniques to guide how it reads instructions. In addition, SPRITe is integrated in a model of agent intelligence known as **FLOABN** (Alterman et al., 1991). FLOABN works with and learns about mechanical and electronic devices and their instructions. SPRITe primarily interacts with FLOABN's activity planner, an adaptive planning system for engaging in activity (see Zito-Wolf, 1993, for details).

A Cognitive Reading Model

In Carpenter (1993), a protocol study was presented which studied 12 people using instructions. The subjects, students and faculty who had never before used a fax machine, were given the machine's instructions and asked to send a fax. The instructions were presented on a computer monitor one line at a time and the instruction usage was recorded. Subjects were also asked to talk aloud while sending the fax. Their comments were recorded and matched to the instructions showing on the monitor at the time each comment was made.

In spite of the length (approx. 70 pages) and difficulty of the instructions, every subject was able to use the instructions well enough to complete the requested task.

* This author is supported in part by a grant from Digital Equipment Corporation.

No one actually read the entire set of instructions. With one exception (a subject who read none of the instructions), the subjects all used the instructions periodically during the interaction until they found text that clarified whatever part of the task was at hand. This text was then read to whatever depth was necessary to extract enough content to continue with the activity.

The subjects used structural and contextual clues in their navigation of the text. The ability to use the structural properties of the instructions is due to their predictable nature. Van Dijk and Kintsch (1983) call the nature of a type of text (e.g., instructions, short stories, et cetera) its *superstructure*. That is, certain structural features (e.g., numbered lists) are used by instructions for specific purposes (e.g., listing a set of steps). The better a reader's knowledge of structure usage, the better he is able to predict the organization of a text.

We have endeavored to create a model of reading that has abilities similar to those portrayed by the subjects of the protocol study. Three primary conclusions from the protocol study reflect the design principles of SPRITe:

1. The subjects interleaved reading with action (e.g., see figure 1).
2. The subjects generally seemed to have some goal in mind, which caused them to skip or reread certain portions of text.
3. The subjects became increasingly confident and skilled in their usage of the instructions. For example, the subjects would often remember where a certain piece of text was located.

By following these behaviors, SPRITe is able to choose when to read, what to read, and how to read.

```
1   TEXT Use the keypad to enter the
         telephone number, then press the
         Start key.
         :
2   SARA "Use the keypad to enter the
         telephone number.  And then press
         the..."
         :
3   FAX  <dial beeps>
         :
4   SARA Then press the start key.  She
         looks for the start key.  This...
5   FAX  <beep>
6   TEXT Lift the telephone handset,  Using
         the telephone, press the Start key
         When the remote ready tones are
         heard, then replace the handset.
```

Figure 1: An excerpt from the protocol study illustrating the interleaving of reading and action. 'TEXT' refers to the line of text showing on the monitor during the current point of interaction and 'FAX' shows the sounds made by the device in response to the user's actions.

The Architecture of a Reading Agent

FLOABN's primary method of activity is based on adaptive planning (Alterman, 1988). The main components of FLOABN (see figure 2) are independent agents (cf. "Society of Mind": Minsky, 1986). Each is capable of running without the other, but in combination, they can accomplish more. This is because the action planner provides the reader with goals and activity against which a concrete understanding of text is made, and the reader provides access to new sources of knowledge that aid the activity's interaction.

In order for the exchange of information to be made successfully, the communication process between SPRITe and the action planner must be defined. For FLOABN's communication, we have defined a set of request and response frames -- a set of templates that, when filled out, provide a predictable structure for information. When the action planner requests information from SPRITe, it chooses one of about 10 request frames to fill out. A request frame suggests a reading goal and provides information such as which plan is in use, a trace of the activity, and the how the plan failed. Each request frame corresponds to a type of breakdown the action planner is likely to encounter, and is designed to provide information about the agent's activity that the reader is likely to need in order to read the instructions and help the action planner recover.

Consider the following "Find-part-ref" request:

```
(request :name    find-part-ref
         :device  fax-machine-001
         :plan    photocopier
         :part    ADF)
```

When this request was made, the action planner was attempting to use a fax machine by adapting its plan for using a photocopier. When the planner encountered the unfamiliar term "ADF" (produced by an earlier access to the instructions), it asked the reader for a definition.

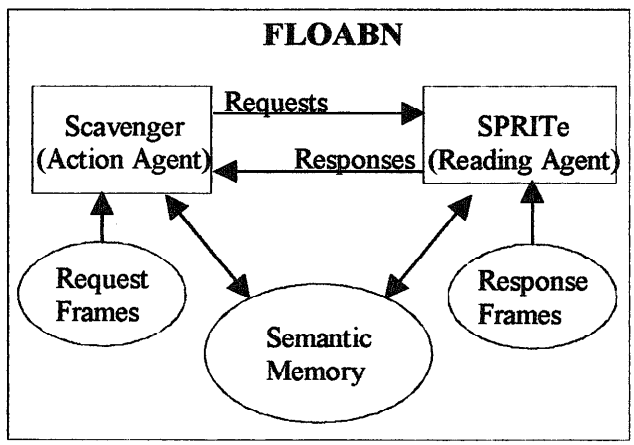

Figure 2: Communication between reading and action agents.

The reader has a list of about 20 response frames. Each request frame has a subset of associated response frames from which the reader must choose. For example, if the planner had made a "Find Object" request, the reader would choose one of the following:

- **Object location**— gives the location of object.
- **Alternative object**— indicates a different object can be used for the same task.
- **Missing object**— indicates the object does not exist and suggests plan modifications.

The "find-part-ref" request frame has only one associated response frame: term-definition. In order to fill out the term-definition response frame, SPRITe first reads the instructions and determines that the definition of ADF is "automatic document feeder". SPRITe then fills out the response frame as follows:

```
(response :name   term-definition
          :device fax-machine-001
          :plan   photocopier
          :term   ADF
          :def    automatic-
                  document-feeder)
```

With this new knowledge, the action planner can use the ADF as part of its interaction with the fax machine.

How to Read

SPRITe (see figure 3) reads instructions using several methods. The overall control of SPRITe is based on the same adaptive planning model as FLOABN's main system. When SPRITe first starts reading, it selects one of about 25 reading plans (e.g., see figure 5) based on the information provided by the request frame and the observable characteristics (i.e., top-level structure) of the text. As reading takes place, the selected plan is adapted to better suit the instructions and/or reading goal of the current situation. This allows SPRITe to react to differences in organization and style between various pieces of text. This also allows the same request frame to cause different reading behavior depending on the instructions and the context.

When SPRITe has successfully read the instructions, it stores three types of information:

1. SPRITe remembers the successful reading steps that were performed as a new plan. This plan is indexed and stored in plan memory.
2. SPRITe remembers the characteristics of the instructions that corresponded to the successful reading steps, thus building a representation of the instructions that will facilitate future usage.
3. SPRITe augments semantic memory with the knowledge it gained through reading.

SPRITe's ability to learn in these ways increases its ability to read effectively and efficiently in the future.

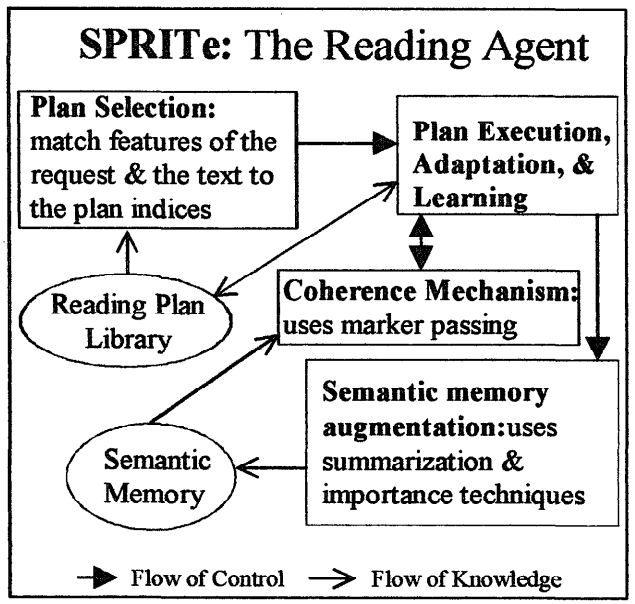

Figure 3: SPRITe -- Inside the Reading Agent

During the execution of reading plans, SPRITe must 'understand' the text it is reading. Understanding is accomplished, at least in part, by establishing *coherence*. SPRITe establishes two types of coherence:

- *textual coherence*: The connection of the meaning of text to other text being read.
- *coherence with activity*: The connection of the meaning of the text to the activity of the agent.

Even if the text is coherent with itself, if the connection of the text with the device and its activity cannot be established, the instructions are useless because they cannot be applied. Coherence of text with activity is not only the association of terms in the text with objects in the world (*external reference*), but also the association of the text with the agent's current interaction (i.e., results of plan steps taken), knowledge of how devices work, and possibilities for future action.

Currently, coherence is established using a modification of the marker passing techniques described in Norvig (1989). SPRITe's knowledge source is a semantic network in which basic concepts (e.g., hierarchy for types of money) are combined with detailed representations of approximately 50 devices.

Top Level Control

This section will describe in detail the process SPRITe uses to read instructions in the context of FLOABN using a simulated[1] Airfone for the first time.

To begin, FLOABN approaches the device, and has the expectation that it will work like a pay telephone.

[1]Simulations are based on actual usage of the device.

However, when FLOABN tries to execute its plan for using a pay phone by lifting the receiver, it doesn't work. FLOABN tries adapting its plan, but none of the steps in the pay phone plan is executable. So, FLOABN calls SPRITe with the following request:

```
(request  :name   next-step
          :device Airfone-001
          :plan   pay-phone
          :failed-steps
                  (lift-receiver
                   listen-for-
                      dial-tone
                   insert-coin))
```

This request frame has four required entries:

- the name of the request (which generally is the reading goal)
- what device FLOABN is trying to use
- which action plan FLOABN is using
- what plan steps have failed so far

A fifth optional entry, what plan steps have succeeded, is not present in this instance of the request frame.

SPRITe's algorithm is shown in figure 4. In step 1, SPRITe notes that the request type is "next-step". In step 2, SPRITe locates the Airfone instructions and builds a representation of them, which initially contains a list of the instructions' top-level text structures: 'title', 'italic-text', plain-text', 'enumerated-list', and 'big-text'.

Using Reading Plans

In step 3, SPRITe chooses a reading plan based on the reading goal and indices. The reading goal is matched against the name of the request frame, and the indices are function calls which probe the current situation (i.e., characteristics of the instructions and details from the request frame). For the current example, the plan in figure 5 was chosen because the goal, "next-step", matches the current request frame and because the index evaluates to true (the Airfone instructions are shorter than 2 pages). Since all the indices match, no future change is anticipated.

Step 4 executes the plan. Each plan has three tasks:

1. locate a likely location for relevant text
2. verify that the content of the selected location is going to help (optional)
3. extract the content of the text

The verify and extract steps require the coherence mechanism to build representations of the text at varying levels of detail, while the locate step uses SPRITe's knowledge of instructional text structure and usage to choose the most likely candidate for reading.

The *locate* step is responsible for finding a starting point in the instructions from which the rest of the plan can work and storing that location in the variable *location*. The locate step can occur in two ways: using the method in the :locate portion of the plan or using the representation that SPRITe has built of the instructions. In the current example, SPRITe has not developed its representation of the instructions yet, so the :locate method of the "find-steps_short-text" plan is used, which locates the enumerated list in the Airfone instructions. If no enumerated list had been found, the plan would fail, and SPRITe would either attempt adapting the plan or choosing another plan.

The *verify* step uses the coherence mechanism to determine whether the "dumb" search done by the locate step actually found an appropriate piece of text. The verify portion of the "find-steps_short-text" plan looks for a description component — a piece of text that describes the purpose or content of the list. According to SPRITe's knowledge of instruction superstructures, the description component is an optional first line of text in the list that is not preceded by the "\item" formatting command. In the Airfone example, the enumerated list does not have a description component. This is not a failing condition for this plan. Only if a description **does** exist but is not coherent with the reading goal would the verify step fail. If this were the case, SPRITe would go back to the locate step and find another location.

Finally, SPRITe performs the *extract* step. The call to "read-enum-list" is actually a call to a sub-plan that uses marker passing to establish coherence between the content of the enumerated list and the plan steps provided

1. Identify *request type*.
2. **If** instructions for current reading task have been used before:
   then locate previously constructed representation
   else build an initial representation for the instructions
3. Select *reading plan* based on *request frame* and characteristics of the instructions.
   a. If any unmatched indices occur, anticipate future change.
4. Execute *reading plan* and adapt.
   a. Sequence through steps to locate text, verify position, and extract content.
   b. If failure occurs, adapt and either restart **step 4** or continue.
   c. If adaptation is unsuccessful restart **step 3**.
5. Remember successful plan steps and store in plan memory.
6. Prepare and send *response frame*.
7. Return control to ACTION AGENT (or to READING AGENT during a recursive call).

Figure 4: SPRITe's top-level control.

```
(plan
  :name    find-steps_short-text
  :goal    next-step
  :indices (short-text)
  :plan
    (:locate
       (assoc
          '\begin(enumerate) text)
     :verify
       (if (not (next-type location
                           '\item))
           (skim (first location)
                 'find-steps))
     :extract (read-enum-list location
       (getf :failed-steps request)
       (getf :successful-steps
             request))))
```

Figure 5: The reading plan chosen in the Airfone example.

by the request frame. This instance of "read-enum-list" skims the first item (line 5) of the enumerated list and discovers the coherence between the instruction "insert credit card" and the failed-step "insert-coin".

Step 4 also includes the provision for adaptation. In the Airfone example, the plan was successfully executed without the need to adapt. If adaptation had been necessary, SPRITe would have first examined any unmatched indices, as well as features of the text and the request frame that were not utilized by the indices. Using this information, SPRITe would have selected either a new locate step (possibly with a related verify step) or a new extract step from a library of location and extraction techniques. The new step would be inserted into the plan and execution would proceed.

Adding to Plan Memory

SPRITe performs step 5 of its procedure and creates a new plan. The new plan, shown in figure 6, is a copy of the original plan with a few changes. First, the text that was used to develop this plan is remembered (line 2). Although this plan could be used with any text, it will receive preferential treatment when the Airfone instructions are used again. Second, the plan has been modified to specifically find the *first* step of an action plan. This is reflected in the creation of a new index (line 5) which remembers that no *successful-steps* were provided in the request frame. Also, the extract step (line 7) has been shortened to skim only the *first* item in the enumerated list. In addition, the verify step was dropped, since it was not successfully executed.

SPRITe then prepares the following response frame:

```
(response :name   replace-step
          :device Airfone-001
          :plan   pay-phone
          :old-step (insert-coin)
          :new-step
                  (insert credit-card))
```

Control is then returned to the action planner, which *proceduralizes* the "insert credit-card" step into its own plan and continues interaction with the Airfone. (For discussion of the proceduralization of instructions, see Alterman et al., 1991.)

The Airfone procedure continues to deviate from the expectations provided by the pay telephone action plan. After inserting the credit card, the action planner is still unable to lift the receiver. Control is given to SPRITe with the following request frame:

```
(request :name   next-step
         :device Airfone-001
         :plan   pay-phone
         :successful-steps
                 (insert-credit-card)
         :failed-steps (lift-receiver
                  listen-for-dial-tone))
```

SPRITe selects the plan shown in figure 5 again, because both its goal and its index match. The plan created by the previous episode (figure 6) is not selected, even though the :text field matches the current text, because the new index (no :successful-steps field in the request frame) is not matched.

Unlike the previous episode, SPRITe does not use the :locate step provided by the plan, since SPRITe's current representation of the Airfone instructions contains the enumerated list. The :verify step is tried again, with the same results as before. Finally, the :extract step skims the list items, recognizing that the "insert credit card"

```
  (plan
1 :name    find-steps_short-text-001
2 :text    Airfone-instructions
3 :goal    next-step
4 :indices ((short-text text)
5           (null (getf :successful-
                        steps request)))
6 :plan
    (:locate (assoc '\begin(enumerate)
             text)
7    :extract (skim-item (assoc '\item
                                location)
             (getf :failed-steps
                   request))))
```

Figure 6: A new reading plan created during interaction with the Airfone.

item matches the successful step "insert-credit-card".

SPRITe then looks at the next item in the list: "lower door handle over card". This is interpreted as a "close" step, similar to the close step required to play a cassette in a tape deck. A new plan is created in the same way as the plan in figure 6. However, the :locate step is changed to reflect the usage of SPRITe's representation of the instructions. Control is then returned to the action planner with the new information.

With this information, the action planner is finally able to lift the receiver, but is immediately thwarted once again when the "listen for dial tone" step fails. This time, when SPRITe receives control, it selects the new plan (shown in figure 7). The new plan is selected over the original plan because the :text field matches. The enumerated list is selected from SPRITe's representation of the instructions, and the list is perused.

SPRITe matches the first two items in the enumerated list with the successful steps given in the request frame. The third item in the instructions says to "observe lighted display". SPRITe interprets this as a reading instruction. However, the display is blank since the action planner has already lifted the receiver.

So, SPRITe reads the entire instruction, which indicates that the display would have said when to lift the receiver. Since the action planner already successfully lifted the receiver, SPRITe interprets this instruction as being equivalent to the step of "lift-receiver". The next item in the list, "press green DIAL TONE button", is then interpreted as an activation step. FLOABN is now able to complete interaction with the Airfone without further help from the instructions.

Conclusion

This paper has described an intelligent reading agent, SPRITe. SPRITe uses planning techniques to perform efficient, goal-driven reading. It develops plans to read a given set of instructions in the context of engagement with a particular device. By planning, adapting, and storing new reading plans, SPRITe builds a library of concrete methods for reading which over time improves SPRITe's ability to read both the texts from which the methods were created, and with instructions in general.

SPRITe is embedded in a larger agent, FLOABN. Combining the activities of reading and action produces an agent more capable than the separate activities could be. The action provides *context* and *goals* for reading, which allow reading to be selective and produce concrete representations of the text. In return, reading provides a detailed knowledge source to the activity planner.

```
(plan
  :name    find-steps_short-text-002
  :text    Airfone-instructions
  :goal    next-step
  :indices ((short-text text))
  :plan
    (:locate
       (or (getf :enum-list inst-rep)
           (assoc '\begin(enumerate)
                  text))
     :extract
       (read-enum-list location
          (getf :failed-steps request)
          (getf :successful-steps
                request))))
```

Figure 7: Another reading plan created during interaction with the Airfone.

References

Agre, P.E. (1988). The dynamic structure of everyday life. Technical Report AI-TR 1085, MIT Artificial Intelligence Laboratory.

Alterman, R., Zito-Wolf, R., & Carpenter, T. (1991). Interaction, Comprehension, and Instruction Usage. *The Journal of Learning Sciences*, 1(3&4):361-398.

Alterman, R. (1988). Adaptive Planning. *Cognitive Science*, 12:393-421.

Badler, N., Webber, B., Kalita, J., & Esakov, J. (1990). Animation from instructions. In Badler, N., Barsky, B., & Zeltzer, D. (Eds.), *Making Them Move: Mechanics, Control and Animation of Articulated Figures*, pp. 51-93. Morgan Kaufmann, Los Altos, CA.

Carpenter, T. (1993). Using Instructions -- A Protocol Study. Presented at the Third Annual Meeting of the Society for Text and Discourse.

Chapman, D. (1990). Vision, instructions, and action. Technical Report AI-TR 1024, MIT Artificial Intelligence Laboratory.

Maybury, M.T. (1991). *Planning Multisentential English Text Using Communicative Acts.* PhD thesis, University of Cambridge, pp. 178-192.

Norvig, P. (1989). Marker passing as a weak method for text inferencing. *Cognitive Science*, 13(4):569-620.

Ram, A. (1991). A Theory of Questions and Question Asking. *The Journal of Learning Sciences*, 1(3&4).

Suchman, L. A. (1987). *Plans and Situated Actions.* Cambridge University Press.

van Dijk, T. & Kintsch, W. (1983). *Strategies of discourse comprehension.* Academic Press.

Vere, S. & Bickmore, T. (1990). A basic agent. *Computational Intelligence*, 6:41-60.

Zito-Wolf, R. (1993). *Case-Based Representations for Procedural Knowledge.* PhD thesis, Brandeis University.