Understanding of Stories for Animation

Hideo SHIMAZU *

Artificial Intelligence Laboratory,
3531 Boelter Hall, University of California,
Los Angeles, CA 90024, USA

Yosuke TAKASHIMA
Masahiro TOMONO

C&C Information Technology Research Laboratories
NEC Corporation
4-1-1 Miyazaki Miyamae-ku Kawasaki
Kanagawa 213 Japan

ABSTRACT

This paper presents the story understanding mechanism for creating computer animation scenarios. The story understanding mechanism reads a natural language story and creates its scenario for realistic graphic animations. This paper presents three types of hidden actions and relations of actions that must be discovered for realistic animations of stories but which are not explicitly described in the stories. They are: 1) causality check among actions; 2) interpolation of a continuous action beyond a sentence; 3) interpolation of hidden actions between neighboring sentences. This paper also describes the inference mechanism which recognizes the need for interpolation of these hidden actions. Multiple TMS is introduced in the mechanism. The knowledge base is action-oriented, hence it is independent of individual stories' domains.

1. Introduction

Recently computer animations have been widely used in many fields like conventional CAD, commercial films and movies. For these kinds of applications, high-level languages are now provided [Reynolds 82][Zeltzer 82]. However, because these languages are programming languages and are hard to use for untrained personnel, it is desirable to develop an easy-to-use computer animation system to encourage more wide-spread use of computer animation.

The authors have been developing *Story Driven Animation* (SDA) [Takashima et al. 87] which automatically generates the animation for a given story written in Japanese taken from a children's story book.

SDA consists of three modules: 1) story understanding; 2) stage directing; 3) action generating. The first module reads a story written in Japanese and makes an action-based scenario. The second module receives the scenario and modifies it for stage setting. The last module generates animations on a graphics display according to the precisely specified scenario given by the stage directing module.

SDA differs from previous natural language processing systems, such as summarizing [Young & Hayes 85][Lytinen & Gershman 86], depth understanding of stories [Dyer 83], question-answering [Wilensky 82][Harris 84] and story generating[Schank & Riesbeck 81], in that it interpolates hidden actions among sentences that must be discovered for realistic computer animations but are not explicitly described in the story itself.

Since **SDA** can accept curtailed expressions in input stories, story writers do not have to describe all the acts explicitly to get the desired graphic animation. Consider the following sentences in "The Hare and The Tortoise" in Aesop's fables:

(1). The hare ran.
(2). The hare looked back.
(3). The hare said, "the tortoise can never catch up with me".
(4). The hare lay down on the grass.
("The Hare and The Tortoise", Aesop's fable)

It is never thought that the hare would lie down while running with his face looking in the reverse direction. If he were to do so, he would do a dive and his neck would be broken! Any person can conjure up an accurate image of the hare's actions. Naturally lacking facts between sentences are interpolated using the human reader's common-sense. We imagine that the hare stopped between (1) and (2), and looked forward between (3) and (4). However, if an animation producing program does not have this common-sense, it produces strange graphic animations when these sentences are not more explicitly described. The above scenario is merely one example of SDA's ability to interpolate curtailed expressions of action. In order for a story understanding program to accurately accept input sentences, it is imperative that such a common sense be built into the program.

SDA story understanding mechanism was constructed based on action-oriented knowledge. Knowledge related to actors' actions is independent of the content of individual stories, and is common to

* The author is a member of NEC Corporation, and is now staying at the Artificial Intelligence Laboratory, UCLA during this academic year.

everyone since actors' movements are constrained by physical limitations of a human body. Although this story understanding approach is superficial, it allows for an extensive domain.

This kind of research which strictly infers occurrences of actions among natural language sentences has not been done yet. In this paper we present various types of hidden actions among sentences and the inference mechanism to identify and interpolate them. The whole SDA system is roughly described in [Takashima et al. 87].

2. Types of Hidden Actions to be Interpolated

There are three types of hidden actions and relations of actions to be interpolated among sentences. The above example shows a type of hidden actions to be interpolated. It can be stated as follows:

- **Continuity of different actions between sentences:** When an action in a sentence is not consecutive to any action in the previous sentence, bridging action(s) has/have to been found and added into the original text. The action "stop" between "run" at (1) and "look back" at (2) and the action "look forward" between "look back" at (2) and "lie down" at (4) are examples. SDA interpolates discriminately; if in the same context it must interpolate, if not in the same context, then it must not try to interpolate (e.g. "The frog lay down. *The next morning*, the frog went out.").

The following example includes the other two types of interpolations in it. The example is also from "The Hare and The Tortoise".

(5). The tortoise ran.
(6). The tortoise ran as kicking up a cloud of dust.
(7). The tortoise sweat while running.
(8). The tortoise stopped at the top of the mountain.

The proper interpretation of these sequences may be the followings:

The tortoise starts running at (5). The tortoise runs while kicking up a cloud of dust at (6). Here, the action of "kick-up-a-cloud-of-dust" must be caused by the action of "run". The tortoise sweats while running and kicking up a cloud of dust at (7), even though "kick-up-a-cloud-of-dust" is not specified in (7). Here, the action of "sweat" must be caused by the action "run", for the tortoise is not breaking into a cold sweat. The tortoise stops running at (8). At this time the tortoise also stops "kick-up-a-cloud-of-dust"-ing and "sweat"-ing because these two actions were caused by "run".

The other two types of hidden actions are:

- **Causality among actions:** When an action appears in a sentence, it must be verified whether it is independent of any other action or caused by other actions. If an action is caused by another action, it also ceases when its dependent action stops, (e.g. the relationship between "kick-up-a-cloud-of-dust" to "run" and "sweat" to "run"). This verification is done between neighboring sentences whose agent is the same.

  Continuity of an action beyond a sentence: An action is assumed to continue until it is explicitly ordered to stop if this action is a continuous type. Inference of the continuity of the action "kick-up-a-cloud-of-dust" from (6) to (7) is an example.

3. SDA Story Understanding Mechanism

In order to accurately understand an input story, SDA performs four distinct operations:

[1] **Extracting meanings of a sentence:** Each sentence is parsed, its meanings extracted, and the meanings put into an independent block called *world*. Because our target story has a simple form, the sequence of its sentences becomes a chronological sequence. Each sentence in a story includes several assertions. An assertion extracted from a sentence may not be true in context with time of its succeeding sentence. Therefore, an individual *world* is assigned to each sentence in order to store assertions which represent the situation inherent in a sentence. When a new *world* is created, it is linked to the sequence of *worlds* which is linked with each individual actors; the hare, the tortoise etc. Each *world* is compared with its previous *world*, and its assertions are added/deleted/modified in the following processes, [3] and [4].

[2] **Causality check among actions:** When an action assertion is put into a *world*, its causal relationship to other actions is checked. If it is dependent on another action, the causality link is connected between the action and its independent action.

[3] **Interpolation of hidden actions between neighboring sentences:** Each action of the present sentence is also checked for its continuity to actions in the previous *world*. If some action is hidden between the previous *world* and this sentence, it is identified and added into the present *world*.

[4] **Interpolation of a continuous action beyond a sentence:** When there exist actions which are not mentioned in the present sentence but should continue from the previous sentence to the present sentence, they are added into the present *world*.

Implementation

4.1 Implementing *world*

Each *world* consists of two stages: *present-state* and *post-state*. *Present-state*, the upper part of a *world*, holds assertions which represent the state of the moment when the sentence is uttered. *Post-state*, the lower part of a *world*, holds assertions which represent the state just after the time when the sentence is uttered. For example, the *world* of (5) has the assertion "the tortoise runs" in its *present-state*, the assertion "the-act-of the tortoise is run" in its *post-state* (see Figure 1).

the tortoise runs.	present-state
the-act-of the tortoise is run.	post-state

Figure 1 *World* of (5)

This means that the tortoise is running during the sentence (5) and afterwards continues to run. Each *post-state* of each *world* is independently monitored by Truth Maintenance System (TMS) [Doyle 79]. This structure is similar to Viewpoints in ART [Clayton 85]. TMS works well in accomplishing the continuity check of an action beyond a sentence.

4.2 Causality Check Among Actions

The dictionary contains action causalities of verbs. When a verb or a verb phrase is processed, the SDA parser consults the dictionary. The following is a part of the dictionary for the verb phrase, "kick-up-a-cloud-of-dust":

```
kick-up-a-cloud-of-dust
 if the-act-of *actor is run
 then            ;;; *actor is a variable, the agent of this action
    present-state:
          *actor kick-up-a-cloud-of-dust.
    post-state:
          true(the-act-of *actor is kick-up-a-cloud-of-dust)
           supported-by(
                in(the assertion-id of "true(the-act-of *actor is run)")
                out())
 else
    present-state:
          *actor kick-up-a-cloud-of-dust.
    post-state:
          true(the-act-of *actor is kick-up-a-cloud-of-dust) as-premise
```

If the tortoise kicks up a cloud of dust when it is running, the action of "kick-up-a-cloud-of-dust" is assumed to be *caused* by the "run" ction. Therefore, the corresponding assertion of the action "kick-p-a-cloud-of-dust" is supported by the "run" assertion. If the tor-oise is not running at the time, the assertion of "kick-up-a-cloud-of-dust" is justified as a premise, which means that the tortoise is kicking up a cloud of dust while standing at a point. This causality between different actions is used as a dependency directed link for TMS.

4.3 Interpolation of Hidden Actions Between Sentences

Interpolation of hidden actions is accomplished by using *goal directed search*. When a sentence is processed and its assertions are extracted, the system picks up each assertion, and then makes an inspection to determine whether the action of each assertion is continuous from the state of the previous *world* or not. The continuity is inspected by checking whether the pre-condition of the action is satisfied in the *post-state* of the previous *world* or not. Each verb is specified its *pre-condition* and *post-condition* in the dictionary. *Pre-condition* is the constraint to be satisfied just before the act of a verb. *Post-condition* is the state to be achieved just after the act of a verb. For example, the dictionary indicates that in order to "stop", the agent must be going on foot (*pre-condition*), and after the agent "stop"s, it must be standing (*post-condition*).

If an action in the present sentence is continuous from the *post-state* of the previous *world*, it is simply put into the present *world*. If it is not continuous, the system searches for a sequence of actions which bridge it (goal point) and the *post-state* of the previous *world* (starting points) by referring the *pre-condition/post-condition* of verbs in the dictionary. This search process is similar to the execution of STRIPS [Nilsson 80]. If a bridging sequence of actions is found, the abridged actions in the sequence are added into the original assertion. Then, the modified assertion is put into the *world*.

The sentence (2) "the hare looked back" is modified to "the hare stopped and looked back" in order to satisfy the *pre-condition* of the verb phrase "looked back". The related pieces of the dictionary are shown below.

```
look-back
      pre-condition:
            (
            the-state-of the agent is not in the reverse direction
            OR
            nothing is mentioned regarding the direction
            )
            AND
            the-state-of the agent is standing.
      post-condition:
            the-state-of the agent is in the reverse direction.

stop
      pre-condition:
            the-act-of the agent is go-on-foot.
      post-condition:
            the-state-of the agent is standing.
```

4.4 Interpolation of a Continuous Action Beyond a Sentence

Interpolation of a continuous action beyond a sentence is accomplished based on the assumption that actors' actions are assumed to continue until they are explicitly ordered to stop. This assumption is the same as the *persistence problem* in [Shoham 88].

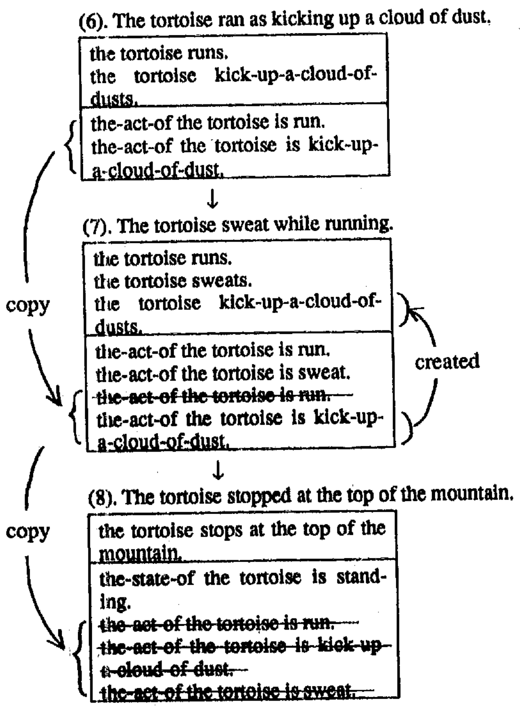
After a sentence is analyzed and its meanings are stored into both the *present-state* and the *post-state* of the present *world*, all the assertions in the *post-state* of the previous *world* are copied into the *post-state* of the present *world*. Then, the *post-state* of the present *world* is checked its consistency by TMS. This check prevents the over-copying of continuous actions from the previous *world*. TMS works according to the following monitoring rules:

- Duplication Elimination Rule: If there exist two or more same assertions in the present *world*, the copied one from the previous *world* is eliminated.
- Exclusive Action Elimination Rule: If there exist exclusive assertions, the one copied from the previous *world* is eliminated. The exclusion relations of actions and states are also defined in the dictionary. The exclusion relations are like the followings:

  exclusive(act-of run , state-of standing)

  exclusive(act-of run , act-of walk).

TMS compares each of the assertions in a *world* with each other. If two assertions cannot coexist, the one copied from the previous *world* is deleted.


(6). The tortoise ran as kicking up a cloud of dust.

the tortoise runs.
the tortoise kick-up-a-cloud-of-dusts.

the-act-of the tortoise is run.
the-act-of the tortoise is kick-up-a-cloud-of-dust.

(7). The tortoise sweat while running.

the tortoise runs.
the tortoise sweats.
the tortoise kick-up-a-cloud-of-dusts.

the-act-of the tortoise is run.
the-act-of the tortoise is sweat.
~~the-act-of the tortoise is run.~~
the-act-of the tortoise is kick-up-a-cloud-of-dust.

(8). The tortoise stopped at the top of the mountain.

the tortoise stops at the top of the mountain.

the-state-of the tortoise is standing.
~~the-act-of the tortoise is run.~~
~~the-act-of the tortoise is kick-up-a-cloud-of-dust.~~
~~the-act-of the tortoise is sweat.~~

Figure 2 *World* of (6), (7) and (8)

Figure 2 shows the *worlds* corresponding to sentence (6), (7) and (8).

Here, the *world* of (6) is already truth-maintained. After the sentence (7) is analyzed and its meanings are stored into the *world* of (7), the two assertions in the *post-state* of (6) are copied into the *post-state* of (7). TMS then deletes the duplicated assertion, "the-act-of the tortoise is run". The assertion "the-act-of the tortoise is kick-up-a-cloud-of-dust" remains in the *post-state* of (7).

Generally an assertion in a *post-state* corresponds to an assertion which presents the causal action in a *present-state* of the same *world*. For example, "the-act-of the tortoise is run" is corresponding to "the tortoise runs". When an assertion is added into the *post-state* of the present *world* by copied from the previous *world* and has no correspondence in the *present-state* of the present *world*, its corresponding assertion is created and put into the *present-state* of the present *world* by the system. Therefore, in this situation the corresponding assertion, "The tortoise kick-up-a-cloud-of-dusts" is created and added into the *present-state* of (7). The *present-state* of *world* of (7) shows that the tortoise is running while sweating and kicking up a cloud of dust (*present-state*), and afterwards continues to run while sweating and kicking up a cloud of dust (*post-state*).

After the meanings of the sentence (8) are stored into the *world* of (8), three assertions are copied from the *world* of (7) to the *world* of (8). Next, because "act-of run" and "state-of standing" are exclusive, the assertion "the-act-of the tortoise is run" is deleted by TMS according to the exclusive action elimination rule. Then, two other assertions in the *post-state* of (8) which were supported by the deleted assertions are subsequently eliminated according to the dependency-directed backtracking mechanism of TMS. Now, the *world* of (8) has only one assertion, "the-state-of the tortoise is standing", in the *post-state* of (8), which means the tortoise is standing and stopped kicking up a cloud of dust and sweating.

After all the story is processed and represented as chronological sequences of *worlds*, assertions in the *present-state* of each *world* are gathered and transformed into a scenario for the stage directing module.

5. System Configuration

Figure 3 indicates a high level view of the whole story understanding system. Each sentence is processed individually and its assertions are extracted by SENTENCE-PARSER. The sentence grammar in SENTENCE-PARSER is described using Definite

Clause Grammar in Prolog [Pereira & Warren 80]. The assertions are then put into the MORE-MEANING-EXTRACTOR which is based on forward-reasoning. Here as many assertions as possible are extracted from the inputs. For example, "If the weather is fine and it is night, then the background for the drama stage is colored in black with lots of stars", etc. The extracted assertions are put into a separate *world* for each sentence. Each *world* is monitored by TMS. Path #1 between neighboring *worlds* indicates the interpolation of a continuous action beyond a sentence. Path #2 indicates *goal directed search* to discover the path of transition between different actions. The whole story understanding system is implemented using Prolog on vax11/780.

6. Conclusion

This paper presents the story understanding mechanism for creating computer animation scenarios. The story understanding mechanism reads a natural language story and creates its scenario for realistic graphic animations. This paper presents three types of hidden actions and relations of actions that must be discovered for realistic animations of stories but which are not explicitly described in the stories. This paper also describes the inference mechanism which recognizes the need for interpolation of these hidden actions and relations. The transition in a story is reflected by chronological sequences of multiple *worlds*, each of which is monitored by TMS. A *world* holds extracted assertions representing the situation inherent in a sentence. Each *world* is compared with its neighboring *worlds*, and assertions in the *world* are added/deleted/modified in the following three processes:

- Causality check among actions.
- Interpolation of a continuous action beyond a sentence.
- Interpolation of hidden actions between neighboring sentences.

The knowledge base is action-oriented, hence it is independent of individual stories' domains. Currently, the story understanding mechanism works well for several fables.

Acknowledgements: The authors would like to express their appreciation for continuous encouragement from Kunihiko Niwa and Takashi Araseki. This research could not have been started without the encouragement of Kazumoto Iinuma. Stephen Berkov provided many useful comments.

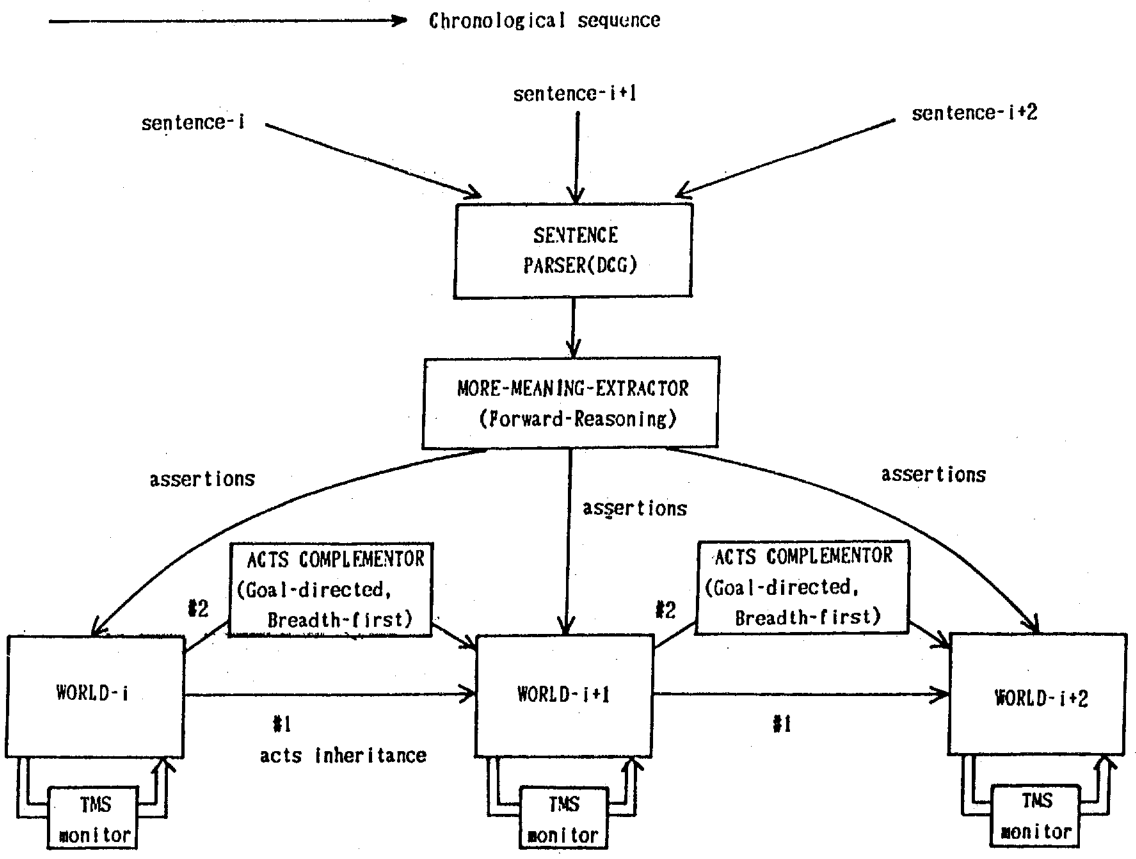


Figure 3 Whole Story Understanding System

References

[Clayton 85] B.D. Clayton, "ART Programming Tutorial, A First Look at Viewpoints", Inference, 1985.

[Doyle 79] J. Doyle, "A Glimpse of Truth Maintenance", Artificial Intelligence: An MIT Perspective, The MIT Press.

[Dyer 83] M. G. Dyer, "In-Depth Understanding", The MIT press, 1983.

[Harris 84] L.R. Harris, "Experience with INTELLECT: Artificial Intelligence Technology Transfer", THE AI Magazine summer 1984.

[Lytinen & Gershman 86] S. Lytinen & A. Gershman, "ATRANS: Automatic Processing of Money Transfer Messages", Proceedings AAAI-86, 1986.

[Nilsson 80] N. Nilsson, Principles of Artificial Intelligence", Tioga.

[Pereira & Warren 80] F. Pereira & D. Warren, "Definite Clause Grammar for Language Analysis -- A Survey of the Formalism and a Comparison with Augmented Transition Networks", Artificial Intelligence, 13, 1980.

[Reynolds 82] C. Reynolds, "Computer Animation with Scripts and Actors", ACM Computer Graphics, vol.16, no.13, July, 1982.

[Schank & Riesbeck 81] R. Schank & C.K. Riesbeck, "Inside Computer Understanding", Hillsdale, New Jersey: Lawrence Erlbaum Associates.

[Shoham 88] Y. Shoham, "Reasoning About Change", The MIT Press.

[Takashima et al. 87] Y. Takashima, H. Shimazu and M. Tomono, "Story Driven Animation", Proc. ACM SIGCHI and Graphics Interface '87 Joint Conference, 1987.

[Wilensky 82] R. Wilensky, "Talking to UNIX in English: An Overview of an On-line Consultant", Report No. UCB/CSD 82/104 internal report of UCB, 1982.

[Young & Hayes 85] S.R. Young & P.J. Hayes, "Automatic Classification and Summarization of Banking Telexes", the 2nd conf. on Artificial Intelligence Applications, Dec. 1985.

[Zeltzer 82] D. Zeltzer, "Motor Control Techniques for Figure Animation", IEEE Computer Graphics and Applications, Vol. 2, No. 9, November, 1982.