Programming the Story: Interactive Storytelling System

SeokKyoo Kim, SungHyun Moon and SangYong Han
Seoul National University, Seoul, Republic of Korea
E-mail: {anemone, shmoon, syhan}@pplab.snu.ac.kr

Juno Chang
Sangmyung University, Seoul, Republic of Korea
E-mail: jchang@smu.ac.kr

Keywords: interactive storytelling, planning, game programming

Received: September 1, 2009

A multi-story can be generated by the interactions of users in the interactive storytelling system. In this paper, we suggest narrative structure and corresponding Storytelling Markup Language. Actor, Action, and Constraint are declared and programmed using interactive storytelling system which generates the stories. Generated stories can be transformed to multimedia formats which are texts, images, animations, and others.

Povzetek: Opisan je sistem za generiranje mnogoterih zgodb.

1 Introduction

A term called 'Digital Storytelling' is being used in various sectors of society nowadays. Studies are being conducted not only in the academic fields that previously have a field of storytelling, such as literature, but also in media studies, computer engineering, and others, owing to its involvement with digital technology. Although the term has not been defined concretely, it is recognized as a storytelling in the digital era.

One definition of digital storytelling is a storytelling that is done by applying digital technology in the medial environment or as an expression means. In other words, digital storytelling in a broad sense indicates the case, wherein digital technology is applied to entire media environment in the production process of the audiovisual materials, or, at least, to creating story and discourse as a means of expression.

Interactive storytelling is one area of digital storytelling. This concept is a type of narrative using an interaction between the emotional and dramatic aspects of the story and the computer, indicating storytelling on which the user makes an influence so as to change the direction which the story proceeds in.

The closest examples are edutainment, videogames in which the progress and an ending of the story can be various depending on the choices made by the user. Especially in games, such a characteristic can be found in RPG (Role Playing Game) or adventure game, visual novels and others.

Studies on such interactive storytelling are not being actively conducted yet, but various attempts are now being made along with the development of online contents such as online game. This study gives a consideration to followings as essential components of interactive storytelling.

1) Narrative Structure
- The essential elements for processing stories and the structure to express such elements are required.

2) Script language to embody narrative structure.
- A system, where the narrative structure is expressed in the type of languages which authors or programmers can understand so that the computer can process it, needs to be arranged.

3) Story generator and authoring tool assisting in generation of story and narrative structure even without professional knowledge of programming languages.
- The development environment, which allows story makers to create stories easily even if they do not have knowledge of script languages, should be supported. The story generator interprets input information as narrative structure, converts it in script language, and then generates the story based on it. The authoring tool are implemented as the graphical user interface environment including the story generator.

Figure 1 shows the constituents of an interactive storytelling system.

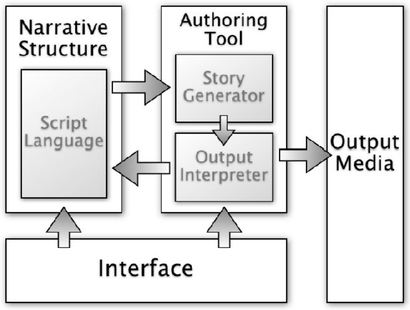


Figure 1: An Interactive Storytelling System

Among various important elements constituting an interactive storytelling system, the narrative structure and the interactive storytelling script language based on it are suggested in this paper. For this purpose, the existing script languages and their expression modes are described in section 2, and the constraint-based narrative structure is suggested in section 3. In section 4, SML (Storytelling Markup Language) is suggested for the system. This script language is fundamental to an interactive storytelling system and is used to develop story generator and authoring tools, making it possible to generate stories. The last section summarizes this study, discussing the method of utilizing this study and future tasks.

2 Related works

Languages that have been used in an interactive storytelling system so far can be largely divided into three types; natural language, logic programming language, and markup language derived from XML.

A natural language is the most suitable language to narrate actions, characters, events, and other source of story because it expresses a human language as it is. Although using a natural language provides convenience for users and increases accessibility or legibility, there are many difficulties compared to processing the existing programming languages. In previous studies, such a natural language system was used in integration with a speech recognition system, which was the system that shows the final stories by processing natural language came through the speech recognition system.

Since logic programming languages use an artificial intelligence planning technique in the narrative structure of an interactive storytelling system, programming languages that are compatible with logic programming are used. For example, there are STRIPS (STanford Research Institute Problem Solver) [2] and languages derived from STRIPS. STRIP, which was introduced to solve problems of AI, is the most suitable for expressing a planning algorithm of narrative structure, thus it is used in an interactive storytelling system [3] [4].

Lastly, there is XML (eXtensible Markup Language). HTML (Hyper-Text Markup Language), a subordinate concept of XML, is used as the standard output format of the World Wide Web all over the world, and many users, thus, know or can easily learn the format. Along with such an environmental factor, XML expresses all information in letter that users know, leading to high legibility. XML has been widely used as the standard to express information since proposed by W3c (World Wide Web Consortium), and it is even suggested that it might replace HTML. Many developing tools have already included the libraries dealing with XML and are continuously developing it. Accordingly, HTML has an advantage of being widely used.

Most of all, XML, a language with the property of generality, can add various formats and express almost any imaginable formats.

In regard to XML-derived languages used in an interactive storytelling system such as MPML (Multi-modal Presentation Markup Language) [5], AIML (Artificial Intelligence Markup Language) [6], APML(Affective Presentation Markup Language)[7], FML(Functional Markup Language)[8], BML(Behavior Markup Language)[9].

MPML is a markup language suitable for controlling actions of characters similar to the real world. MPML is a powerful language which can provide the control for behavior of second dimensional characters, the presentation flow, and the integration of external objects. FML and BML are designed to unify representational framework for Embodied Conversational Agents to produce multimodal behaviours of computer-generated characters in a broad range of circumstances. APML is another attempting version for ECA able to generate context-adapted behaviours based on Mind-Body interface; Mind which represents the personality of an agent, and Body reflects its appearance and expressive behaviours. However, they have some shortcomings to be the script language for generating stories which this study aims at, because it is a language for controlling the agent.

In the VISTA (Virtual Interactive Story Telling Agents) project [10], AIML was used to write programs. AIML is a XML style script language supporting for AI application program, and the Vista system used AIML interlocked with Prolog. AIML was used in the question-answer relationship applied to stories, while Prolog was used to generate various action rules. With regard to AIML, however, all the questions and answers should be defined in advance, and it is hard to produce various results inferred from the various conditions. Besides, there is another difficulty that general users who are not familiar with programming should know Prolog.

In addition to language to formalize narrative structure, authoring tools which assist to program the language and specify the information have been investigated and developed: INSCAPE[11] and PRISM[12] . In INSCAPE, an author writes an interactive story idea; prepares characters, props, and stages; and plan entire flow of story with those assets to achieve desired goals. It also adopted XML-style language, called ICML(Inscape Communication Mark-up Language) for underlying data model. It is designed to create interactive stories for edutainment, simulation, training, and other areas of nonlinear story. PRISM provides story map to set up interactive story in a similar way to INCACPE and it adopts hybrid narrative structure combining "condition based branching narrative" and "planning" methods to generate interactive story

3 Narrative structure

As examined previously, various script languages have been used in an interactive storytelling system. This paper suggests SML based on XML. Although there are already script languages derived from XML, such as MPML or AIML, these languages cannot defined the narrative structure or have difficulties in generating the variety of stories. In comparison, SML suggested in this study has following strength: it defines the narrative structure so that authors can intuitively program stories without difficulties and then can define the languages conforming to XML format according to this narrative structure.

3.1 Constraint based narrative structure

The previous studies have expressed stories mainly in STRIPS or Lisp format in order to solve problems based on AI planning techniques [1][4][13]. In regard to a planning algorithm of narrative structure, there are HTN (Hierarchical Task Network) and HSP (Heuristic Planning). STRIPS is used in the systems using HTN and HSP. The structure of HTN can be represented as a tree, in which the conclusion of story is a route and each sub-conclusion is a child. This employs a top-down method, which leads to good narrative coherence. In contrast, HSP generates the route from an initial state to a goal state, which makes it possible to generate stories flexibly [3]. It can be said that the method introduced in a text is a kind of HSP.

HTN constructs a story using a tree. The route node and each non-terminal node indicate the conclusion and the sub-conclusion of the story, respectively, while the terminal node indicates occurrence of a certain action. In regard to the way of constructing the story, the final conclusion is divided into several sub-conclusions, each of which is divided again into another sub-conclusion, and then the story is generated by solving each sub-conclusion. Here, several actions are collected to solve the smallest unit of sub-conclusion.

In other words, it is considered that the story is the combination of sub-conclusions and the lowest level sub-conclusion includes the collection of the specific character's actions. The story is generated using a divide-and-conquer planning technique based on AI, in which the problem is divided into the smaller units, which are then solved and combined. In such a structure, the conclusion cannot be reached unless the sub-conclusion is satisfied, which, in turn, guarantees the coherent flow of the story. In another aspect, the story is processed centering on the only actions that are preconditions for the conclusion, so only stories expressed on the route of HTN are formed, thereby causing the decrease in the degree of freedom.

In case of an ideal type of interactive storytelling, the story should have a high degree of freedom while maintaining its coherent flow. It is, however, not easy to realize an interactive storytelling system satisfying both of them, because these two are usually in a contradictory relationship [13]. This study aims at the direction of story generation that makes it possible for authors to adjust the degree of freedom and the coherence of stories.

The previous studies have maintained a causal relationship through the divide-and-conquer planning based on AI, but reached the limit for the degree of freedom. Therefore, this study suggests the form that escapes from such a frame by eliminating the hierarchical structure. In other words, contrary to the existing structures, the stories are seen as the continuous actions of the character after eliminating the step of sub-conclusion. Instead of the eliminated sub-conclusions, the constraint conditions such as a causal relationship or a temporal relationship between actions are declared in order to maintain the unity of the story. And through the degree of such constraint conditions, the degree of the coherence and the degree of freedom can be properly adjusted to the extent which the author wants.

The narrative structure suggested in this study is divided to constituent declaration and constraint declaration. The constituent declaration defines actor, action property, stage, and props, while the constraint declaration suggests the various condition-relation structures that can control the flow of the story and support the nonlinear process of the story.

3.2 Constituent declaration

In the constituent declaration, the basic constituents necessary for stories are declared. The constituents defined in the declaration include property, stage, actor, props, action and others. Figure 2 shows the constituents defined in the declaration and the interdependence among them.

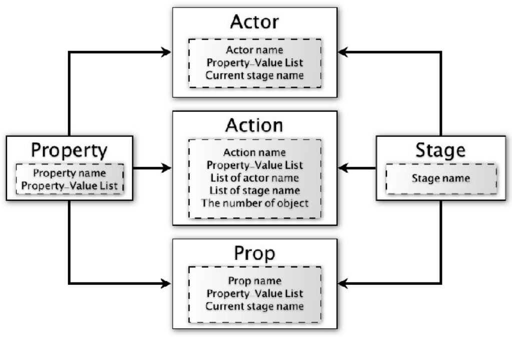


Figure 2: Declaration Constituents and Relationships

- Property

  A property expresses the characteristic of actors or props numerically. Actors or props can have the value corresponding to necessary properties for each as a numerical value.

- Stage

  A Stage displays the space where the story proceeds.

- Actor

  An Actor is the constituent that becomes the subject or the object of the story, which performs itself or is performed. There is a name for each actor, and it contains the types of properties that the actor has, the value of each, and the stage information that marks the space in which the actor is at the very beginning of the story.

- Props

Props refer to things that cannot be the subject but object of actions. Thus, things or actors who cannot be the subject of actions are defined as props.

- Action

  An action indicates the motion that actors can take. It displays actors who can carry out such an action, properties necessary for the action, stages where the action can occur, and how many objects of the action are.

In this constraint based narrative structure, properties commonly exist between actors and actions. This is the device for generating stories, wherein more logical choice can be made in deciding the motions of actors. Let's consider one example in order to understand uses of such a device. Followings are assumed; an actor 'A' has properties of talkative = "80" and aggressive = "30", an action 'attack' has properties of talkative = "30" and aggressive = "80", and an action 'talk' has properties of talkative = "80" and aggressive = "30".

```
<actor actorid="A">
   <propertyR propertyId="talkative">80</propertyR>
   <propertyR propertyId = "aggressive">30</propertyR>
</actor>
<action actionid = "attack">
   <subR subId="A"/>
   <propertyR propertyId = "talkative">30</propertyR>
   <propertyR propertyId = "aggressive">80</propertyR>
</action>
<action actionid = "talk">
   <subR subId="A"/>
   <propertyR propertyId = "talkative">80</propertyR>
   <propertyR propertyId = "aggressive">30</propertyR>
</action>
```

If an actor 'A' is in a situation where he or she has to perform one motion either 'attack' or 'talk', the possibility of carrying out 'talk' motion increases due to the property value. It is necessary to fully unitize this structure in a story generator in order to establish the balance between characteristics and actions of actors.

The constituents listed above are combined, thereby generating an event. In brief, an event means that a specific actor carries out a certain motion, wherein the defined form of the action determines the presence of an object.

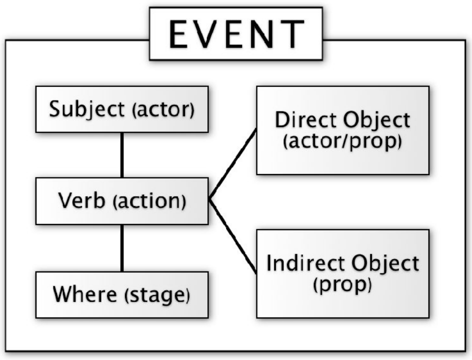


Figure 3: Components of event

Figure 3 shows the necessary components of an event. To put it concretely, an event is expressed as actor, stage, action, and props are combined in such form as "Specific actor does what action (something) (to someone) where". The generation of story in this structure means the generation of such an event. In addition, a special component called 'viewpoint' is placed in order to reduce the complexity in story generation. This indicates the stage at which the user is currently looking. The generation of all the stories is limited to the current viewpoint, which is in line with a play where audiences see only one stage. As a curtain comes down when the stage is changing, the change of viewpoint is required to change the stage in this structure. As an initial stage should be specified in a play, the viewpoint for a beginning point of the story should be specified in the constituent declaration as well.

3.3 Constraint declaration

In the constraint declaration, various constraint conditions that can control the direction of narrative flaw are declared. Such constraint conditions play a role of framework holding the direction in which the story proceeds, preventing the story from taking the wrong way. The degree of freedom is determined according to a dynamic of constraint conditions, and the types of constraint condition are shown below. The parts explained here is conceptual, so it is necessary to support a wildcard character ('?') or logical operation so as to control the stories more delicately when writing these in an actual script language.

1) Ending : (event) → (end)

This is a condition that ends the story. If a given event occurs, the story ends. The multiple endings can be setup by assigning several kinds of events.

```
<ending endingId = "HappyEnding">
   <preEventList>
      <preEvent>
         <preSub subtype=”actor” preSubId= "Monica"/>
         <actionR actionId = "love"/>
      <preObj objType=”actor” actorId = "Joseph"/>
    </prevent>
  </preEventList>
</ending>
```

2) Transaction : (pre event) → (post event)

Two events occur together like a transaction. This condition is set in order to induce another event to occur in sequence upon occurrence of a specific event. In script languages, the function, where post-event can refer to the subject and the object of pre-event, should be provided in order to make extensive forms of transaction condition. For example, let's assume the case where a wildcard character “?” designated as any actor, is given as a condition, and pre-event is defined as “?” hit “?”. It should be possible to define post-event for this case in the same construction with '[The object of pre-event] hit [the subject of pre-event]'. In this case, the subject and the object of post-event are determined at the time pre-event occurs.

```
<transaction transactionId = "HitAfter">
  <preEventList>
    <preEvent>
      <preSub  subType="actor" preSubId ="ALL” />
      <actionR actionId = "hit" />
      <preObj objType="actor” objId="All"/>
    </preEvent>
  </preEventList>
  <postEventList>
    <postevent>
      <postSub type="actor" postSubId="Robert"/>
      <actionR actionId = "hit"/>
      <postObj type="actor" postObjId="RS"/>
    </postEvent>
  </postEventList>
</transaction>
```

3) Transition : (event/action) → (change property)

Transition constraint is to change the value of a specific property upon occurrence of a designated event or action. The condition is set, for example, as the construction of 'reduce a property of an object, 'health', for a 'hit' action.' This is useful to define the motion accompanied with changes of a specific state.

```
<transit transitId = "Work">
  <preEventList>
    <preEvent>
      <preSub subType="actor"
                    preSubId ="Monica" />
      <actionR actionId = " work" />
    </preEvent>
  </preEventList>
  <postTransitList>
    <postTransit transiType="actor”
          transitSubId="RS" propertyId = "tiredness"
          value = "20"/>
  </postTrasitList>
</transit>
```

4) Induction : (condition of property) → event/action)

It can declare to perform an event or a motion when a specific attribute satisfies a certain condition. The condition is set, for example, as the construction of “eat rice when hungry is less than 30.” This constraint condition is assigned to make an actor perform a specific motion according to the change of state.

```
<induction inductionId = "Hungry">
  <preTransitList>
    <propTransit TransitSubId ="ALL"
              propertyId = "hungryness"
              value = "80" compare="GT"/>
  </preTransitList>
  <postEventList>
    <postEvent>
      <PostSub Type="actor" postSubId="RS" />
      <actionR actionId = "eat" />
    </postEvent>
  </postEventList>
</induction>
```

5) Must : (pre event) → (post event)

This is a constraint condition that executes various functions. The semantic view of this condition indicates that 'When pre-event is performed without occurrence of post-event, post-event must occur before the story ends.' In other words, this constraint condition is used if an event that occurs as a result must occur upon occurrence of an event corresponding to a cause in a causal relationship, Also, a negative option can be placed in post-event so that post-event should never occur when pre-event occurs. This can put a constraint so that another event can never follow when a specific event occurs. In addition, an event that must occur on any occasion can be indicated by setting only post-event not pre-event.

```
<must mustId = "EatAfter">
  <preEventList>
    <preEvent>
      <preSub subtype="actor" preSubId="Monica"/>
      <actionR actionId = "eat" />
    </preEvent>
  <preEventList>
  <postEventList>
    <postEvent>
      <actor actorId = "Monica" />
    <action actionId = "wash" />
  <props prosId = ”dish” />
  </postevent>
</must>
```

6) Ordering : (pre event) → (post event)

This is a declaration to adjust the logical flow of the story. It makes no difference even if events assigned here don't occur. It only indicates 'pre-event must occur prior to post-event at all times'. In other words, if pre-event doesn't occur, post-event wouldn't occur either.

```
<ordering orderingId = "WinAfter">
  <preEventList>
    <preEvent>
      <preSub subtype="actor"preSubId="ALL"/>
      <actionR actionId = "win" />
    </preEvent>
  </preEventList>
  <postEventList>
    <postEvent>
      <postSub Type="actor" postSubId="RS" />
      <actionR actionId="get"/>
      <props prosId="money"/>
    </postEvent>
  <postEventList>
</ordering>
```

7) Stage Change : (event) → (change stage)

This is a constraint declaration that changes the current stage of the subject or the object of an event when a specific event occurs. This is similar to an effect that makes characters on stage withdraw or makes characters appear on stage in a play.

```
<stageChange stageChangeId = "Work">
  <preEventList>
    <preEvent>
      <preSub Type="actor" preSubId="Monica"/>
      <actionR actionId="leave" />
    </preEvent>
  </preEventList>
  <targetStage>
    <ActorR actorId="Monica"/>
    <stageR stageid="office"/>
  </targetstage>
</stageChange >
```

In summary, the basic environment and constituent required in the story are defined in the constituent declaration, and the constraint items holding the outline of overall direction of story are assigned in the constraint declaration.

4 SML (Storytelling Markup language)

4.1 SML

The narrative structure with the purpose of an interactive storytelling was defined previously. However, the previously defined structure is simply a kind of an abstract data type. In order to generate stories, it is necessary to express above components in definite language so that the story generation engine can process them. The language for expressing narrative structure requires following characteristics.

First, the legibility should be good enough so that users can easily understand the meanings without any difficulty and add what they want. There would be no occasion to write codes, because the authoring tools are basically used. However, good legibility is necessary so that there would be no difficulties even in the case of writing codes directly.

Second, there should be good expressiveness. In the previously suggested structure, an actor or an action has subordinate constituents in complex form. In particular, the way the motion is defined demands information of properties necessary for the motion, actors who can perform the motion, and stages where the motion can occur. Each motion, however, has different number of such constituents.

The structures defined in the constraint declaration are combined by a relationship of 'AND' or 'OR' and demands a wildcard character, and a reference structure, thereby becoming more complex. The language should be able to express such structures that are defined as being complex and multilateral.

Third, the structure should be easy to deal with. It requires the operation of generating or parsing the codes written in the given language using an authoring tool or a story generator. The structure of the language should be easy to deal with, so that it would be easy to develop an authoring tool or a story generator that supports such a language. Such a characteristic of the language can be considered very important in case that an authoring tool, a story generator and a script language are independent from each other, which even makes it possible to develop various authoring tools or story generators to support the language afterwards.

The language should be defined with the above conditions in mind in order to achieve satisfactory results. When taking the structures of already existing languages into consideration, a XML (eXtensible Markup Language) format can be easily considered first. XML can be seen as a super-ordinate concept of HTML (Hyper-Text Markup Language). HTML is well known to many users because HTML is used as a standard output mode of World Wide Web. Many users, in turn, can read and write HTML. Thus, the XML mode can be considered to have quite excellent legibility.

In addition, XML itself is a language that can add various formats freely, leading to fairly good expressiveness. Also, since XML is widely known and used, there are many related libraries. Since it is easy to deal with languages with help of libraries, the XML format is, in a sense, the 'easy-to-handle structure'.

As described, XML satisfies all the conditions required as a language listed above. This study, thus, defines the previously defined narrative structure in the XML format, naming the language as SML (Storytelling Markup Language). The DTD (Document Type Definition) of SML is not included in a text, since the constituents and the constraint conditions of the previously presented narrative structure are large in quantity and complicated.

4.2 An example of SML

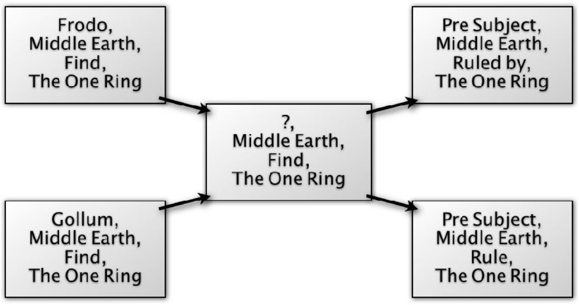
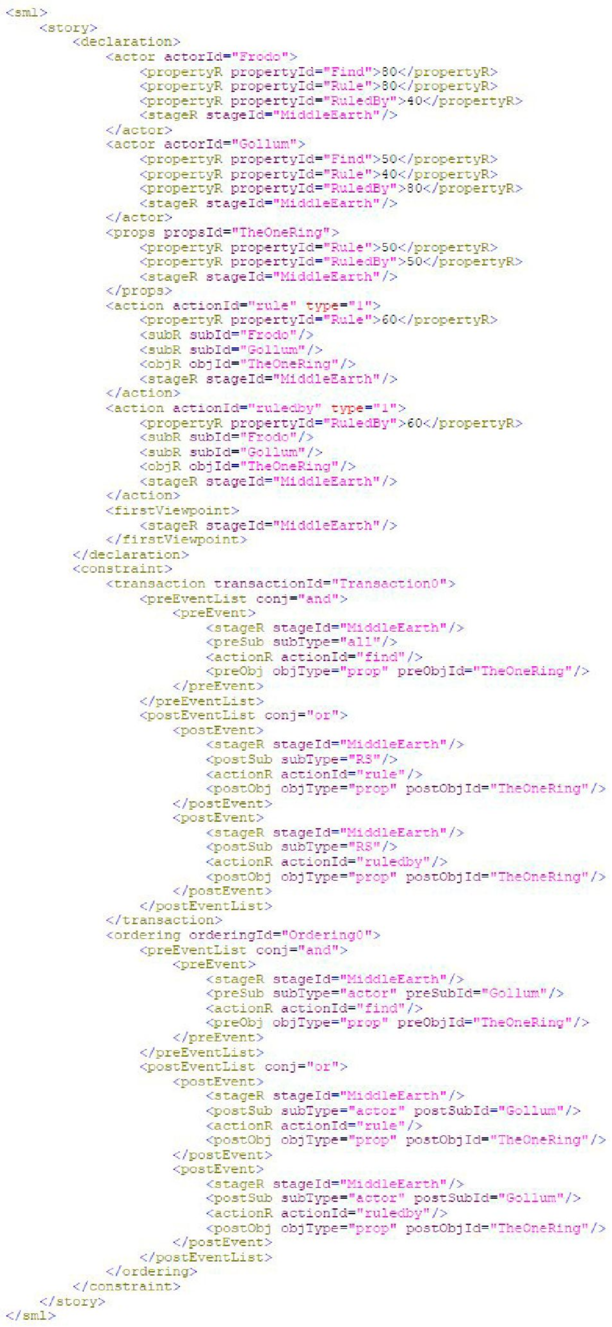

Frodo, Middle Earth, Find, The One Ring
Gollum, Middle Earth, Find, The One Ring
?, Middle Earth, Find, The One Ring
Pre Subject, Middle Earth, Ruled by, The One Ring
Pre Subject, Middle Earth, Rule, The One Ring

Figure 4: An Example of Story Structure.

Figure 4 is a drawing which illustrates some part of the story, while Figure 5 simply shows only the necessary part for the story illustrated in Figure 4 in SML. The event of 'Frodo finds the One Ring in the Middle Earth' or 'Gollum finds the One Ring in the Middle Earth' is set to occur for sure with constraint 'must' with no 'pre-event'. With constraint 'transaction', upon generation of an event that 'somebody finding the One Ring in the Middle Earth', 'the finder rules or is ruled by the One Ring.', in such a case, whether ruled by the One Ring or not depends on properties of Frodo and Gollum. The property value of 'ruling' is higher value than that of 'being ruled by' in case of Frodo, while the property value of 'being ruled by' is higher in Gollum. In such cases, an event, in which an action of 'ruling' occurs, has a higher possibility of being generated as for Frodo, while the possibility of generating an event, in which an action of 'being ruled by' occurs, increases as for Gollum.

4.3 Story generator

In previous sections, the narrative structure was defined, along with the language to express it. At this point, complier or interpreter is needed to interpret the codes written in the given language and to perform the operation. In other words, a story generator, which receives given SML codes and generates stories, is required.



```
<sml>
    <story>
        <declaration>
            <actor actorId="Frodo">
                <propertyR propertyId="Find">80</propertyR>
                <propertyR propertyId="Rule">80</propertyR>
                <propertyR propertyId="RuledBy">40</propertyR>
                <stageR stageId="MiddleEarth"/>
            </actor>
            <actor actorId="Gollum">
                <propertyR propertyId="Find">50</propertyR>
                <propertyR propertyId="Rule">40</propertyR>
                <propertyR propertyId="RuledBy">80</propertyR>
                <stageR stageId="MiddleEarth"/>
            </actor>
            <props propsId="TheOneRing">
                <propertyR propertyId="Rule">50</propertyR>
                <propertyR propertyId="RuledBy">50</propertyR>
                <stageR stageId="MiddleEarth"/>
            </props>
            <action actionId="rule" type="1">
                <propertyR propertyId="Rule">60</propertyR>
                <subR subId="Frodo"/>
                <subR subId="Gollum"/>
                <objR objId="TheOneRing"/>
                <stageR stageId="MiddleEarth"/>
            </action>
            <action actionId="ruledby" type="1">
                <propertyR propertyId="RuledBy">60</propertyR>
                <subR subId="Frodo"/>
                <subR subId="Gollum"/>
                <objR objId="TheOneRing"/>
                <stageR stageId="MiddleEarth"/>
            </action>
            <firstViewpoint>
                <stageR stageId="MiddleEarth"/>
            </firstViewpoint>
        </declaration>
        <constraint>
            <transaction transactionId="Transaction0">
                <preEventList conj="and">
                    <preEvent>
                        <stageR stageId="MiddleEarth"/>
                        <preSub subType="all"/>
                        <actionR actionId="find"/>
                        <preObj objType="prop" preObjId="TheOneRing"/>
                    </preEvent>
                </preEventList>
                <postEventList conj="or">
                    <postEvent>
                        <stageR stageId="MiddleEarth"/>
                        <postSub subType="RS"/>
                        <actionR actionId="rule"/>
                        <postObj objType="prop" postObjId="TheOneRing"/>
                    </postEvent>
                    <postEvent>
                        <stageR stageId="MiddleEarth"/>
                        <postSub subType="RS"/>
                        <actionR actionId="ruledby"/>
                        <postObj objType="prop" postObjId="TheOneRing"/>
                    </postEvent>
                </postEventList>
            </transaction>
            <ordering orderingId="Ordering0">
                <preEventList conj="and">
                    <preEvent>
                        <stageR stageId="MiddleEarth"/>
                        <preSub subType="actor" preSubId="Gollum"/>
                        <actionR actionId="find"/>
                        <preObj objType="prop" preObjId="TheOneRing"/>
                    </preEvent>
                </preEventList>
                <postEventList conj="or">
                    <postEvent>
                        <stageR stageId="MiddleEarth"/>
                        <postSub subType="actor" postSubId="Gollum"/>
                        <actionR actionId="rule"/>
                        <postObj objType="prop" postObjId="TheOneRing"/>
                    </postEvent>
                    <postEvent>
                        <stageR stageId="MiddleEarth"/>
                        <postSub subType="actor" postSubId="Gollum"/>
                        <actionR actionId="ruledby"/>
                        <postObj objType="prop" postObjId="TheOneRing"/>
                    </postEvent>
                </postEventList>
            </ordering>
        </constraint>
    </story>
</sml>
```

Figure 5: Example for expression of story structure using SML

When the story is viewed as the connection of continuous events, the generation of a high-quality story depends on how proper the order these events are sequenced in. When S_{tem} refers to a set having 'item' as domain, S_{event} can be expressed as followings.

$$S_{subject} = S_{actor}$$

$$S_{DO} = S_{actor} \cup S_{props}$$

$$S_{IO} = S_{actor} \cup S_{props}$$

$$S_{Event} = \{(x_{subject}, x_{stage}, x_{action}, x_{DO}, x_{IO}) \mid x_{subject} \in S_{subject}, x_{stage} \in S_{stage}, x_{action} \in S_{action}, x_{DO} \in S_{DO}, x_{IO} \in S_{IO}\}$$

In other words, a set of events can be viewed as a Cartesian product of domains; subject, stage, action, DO and IO, and one event is represented as a tuple of (subject, stage, action, DO, IO). In a wider prospective, if the tuple is viewed as node of graph, a story can be considered as a connection of nodes. Therefore, the generation of story can be understood as setting a proper path of the graph, wherein events are nodes.

From this point of view, this study excludes AI techniques and suggests a story generator using graph in which a node is an event, in contrary to previous studies.

As explained above, when an event is seen as a node in graph, a story can be one path connecting a series of nodes in graph. In this case, the story generation is soon down to the issue of setting the graph path in order that is not contrary to the constraint conditions. Thus, most of the problems to consider can be thought by converting them to graph problems.

First let's consider the expression of an event. Although it is said that an event can be seen as a node, nodes can be large in number according to the amount of data if all types of generable events are actually generated and the path between each node is specified as other graph problems. In case of events containing a wildcard character, in particular, the number of nodes will increase by geometric progression when all events are generated. Thus, even though it can be perceived as a graph problem in the conceptual aspect, the realization that uses memory on a practical level should be considered. In that,

It should be approached with the form in which not all events are generated as nodes but generated events are managed as nodes. Also, with managing the list of constraint conditions, it would be realistic to determine whether to set a route or not by comparing such a list and an event to be generated.

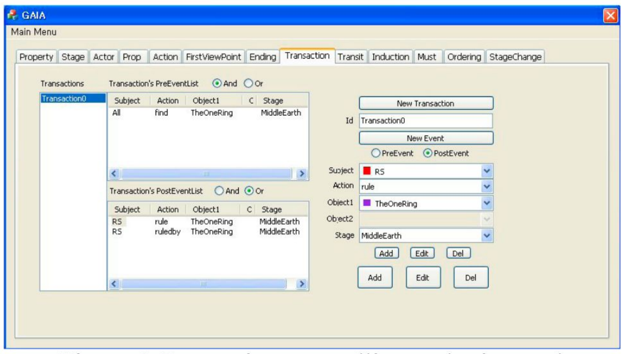
The problem of generating events can be seen as the problem of selecting one among many paths that can be chosen at the current node, which requires a function or a value that becomes a benchmark of path choice. In Prim's MST (Minimum Spanning Tree) algorithm, for example, the closest node in MST constructed so far becomes a benchmark of selecting a path.

This study considers the method which put a great deal of weight on the node having the highest value of property combination among available actions choices, which was also mentioned earlier in relation to [Figure 4]. Because when a specific person should select one among more than two actions, it is believed that it's more reasonable to carry out action suiting to one's propensity.

The condition of this choice needs to be processed by the concept of possibility in order to avoid the uniformity of the narrative. The problem of adhering to constraint conditions can be understood as a problem of dynamically managing the path between the nodes in graph, a problem of managing topological order, and a problem of preventing the path from repeating infinitely. If an event B cannot occur right after an event A in a conceptual aspect, then it can be understood as there is no path from node A to a node B. Moreover, when a temporal relationship exists between two events such as 'transaction', 'must', and 'ordering', the constraint conditions can be satisfied by establishing topological order. Also, the conflict among constraint conditions can be prevented by prohibiting the generation of indefinite repetition.

4.4 Example of story generation

The story was generated through a story generator based on the algorithm suggested above, after the construction of The Lord of the Rings story was simplified and then expressed in SML defined above, Figure 6 is the currently realized authoring tool, and Figure 7 shows one example of created stories.



Transactions: Transaction0

Transaction's PreEventList [✓ And] Or

Subject	Action	Object1	C	Stage
All	find	TheOneRing		MiddleEarth

Transaction's PostEventList And [✓ Or]

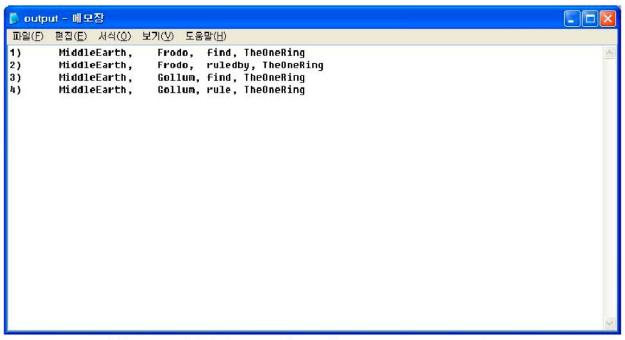
Subject	Action	Object1	C	Stage
RS	rule	TheOneRing		MiddleEarth
RS	ruledby	TheOneRing		MiddleEarth

Id: Transaction0

PreEvent [✓ PostEvent]

Suject: 🟥 RS
Action: rule
Object1: 🟪 TheOneRing
Object2:
Stage: MiddleEarth

Figure 6: Interactive storytelling authoring tool



```
1)      MiddleEarth,    Frodo,  find, TheOneRing
2)      MiddleEarth,    Frodo,  ruledby, TheOneRing
3)      MiddleEarth,    Gollum, find, TheOneRing
4)      MiddleEarth,    Gollum, rule, TheOneRing
```

Figure 7: Example of story generation

Depending on the property values of Frodo and Gollum, the story that is completely different from what we previous knew could be generated. Besides this, several kinds of different endings can be generated. The results are printouts in a plain text format. However, if it is expressed in script language and then combined with the output interpreter by extending this study further, it would be possible to express it in other media such as picture and animation. The output interpreter, which prints the generated text in the form of a series of drawings, is in development.

5 Conclusions

The text investigates the concept and the structure of an interactive storytelling system, and then suggests the narrative structure and the language to express it. The narrative structure suggested in this study cannot be considered as an ideal one for generating stories. However, it suggests a possibility in regard to how to express the structure constructed in such a way in language and what else is required to generate stories using the language that is expressed in such a way.

The narrative structure and SML suggested in this paper have following strengths, compared to exiting interactive storytelling systems.

First, the previous studies left much to be desired in relation to languages due to undue stress given to applications. On the other hand, this paper studies on languages, increasing expansion possibility of the system that can be combined and linked.

Second, the narrative structure of this study can add or delete the components as occasion demands, and even new narrative structure can be applied. For example, the narrative structure which is an application of data flow system using token can be made.

Third, various interactive storytelling systems using SML can be developed, thereby applied to games and educational multimedia system.

Currently, this study is expected to proceed in three directions; first is to improve the narrative structure and the functions of the language, second is to realize an interactive storytelling system which can generate the results not in text format but with various multi medias such as picture and animation by using the authoring tool and the output interpreter, and the last is to proceed in developing the script language and the control system that can control actions of NPC (Non Playable character) in MMORPG (Massively Multiplayer Online Role Playing Game) by applying SML, in cooperation with a domestic game company.

As shown in this study, SML shows a possibility to be the language that could be widely applied to the areas of game and multimedia.

References

[1] F. Charles, S. J. Mead, and M Cavazza, "User Intervention in Virtual Interactive Storytelling," *Proceedings of VRIC 2001*, Laval, France, 2001.

[2] R. Fykes, and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence 2*, pp. 189-208, 1971.

[3] M. Cavazza, F Charles, and S. J. Mead, "Interacting with Virtual Characters in Interactive Storytelling." *ACM Joint Conference on Autonomous Agents and Multi-Agent Systems*, pp. 318-325, 2002.

[4] L. M. Barros and S. R. Musse, "Introducing narrative principles into planning-based interactive storytelling," In *Proceedings of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, pp. 35-42, 2005.

[5] H. Prendinger, S. Descamps, and M. Ishizuka. "MPML: A markup language for controlling the behavior of life-like characters." *Journal of Visual Languages and Computing*, pp. 183-203, 2004.

[6] A.L.I.C.E. AI foundation, "Artificial Intelligence Markup Language (AIML)," *Technical Report*, URL: http://alice.sunlitsurf.com/TR/2001/WD-aiml/, 2001.

[7] B. DeCarolis, M. Bilvi, and C. Pelachaud. "APML, a Mark-up Language for Believable Behavior Generation," In *H. Prendinger and M. Ishizuka, editors, Life-like Characters. Tools, Affective Functions and applications*, pp. 65-85, 2004.

[8] B. Krenn and Gregor Sieber, "Functional Mark-up for Behaviour Planning: Theory and Practice," In *Proceedings of the AAMAS 2008 Workshop. Why Conversational Agents do what they do.* PP. 12 -16, 2008.

[9] S. Kopp, B. Krenn, S. Marsella, A.N. Marshall, C. Pelachaud, H. Pirker, K. R. Thorisson, and H. Vilhjalmsson, "Towards a Common Framework for Multimodal Generation: The Behavior Markup Language," In *The Proceedings of the 6th International Conference in Itelligent Virtual Agents.* pp. 205-217, 2006.

[10] E. Figa and P. Tarau, "The VISTAProject: An Agent Architecture for Virtual Interactive Storytelling," In *Proceedings of Technologies for Interactive Digital Storytelling and Entertainment*, pp. 106, 2003.

[11] N. Zagalo, S. Gobel, A. Torres, R. Malkewitz, and V. Branco, "INSCAPE: Emotion Expression and Experience in an Authoring Environment", In *Proceedings of Technologies for Interactive Digital Storytelling and Entertainment 2006*, pp. 219-230, 2006.

[12] Y. Cheong, Y. Kim, W. Min, E. Shim, and J. Kim, "PRISM: A Framework for Authoring Interactive Narratives", In *Proceedings of the 1st Joint International Conference on Interactive Digital Storytelling 2008*, pp. 297-308, 2008.

[13] F. Charles, M. Lozano, S. J. Mead, A. F. Bisquerra, and M. Cavazza. "Planning formalisms and authoring in interactive storytelling," In *Proceedings of Technologies for Interactive Digital Storytelling and Entertainment*, pp. 216-225, 2003.