Hierarchical PNF Networks: A Temporal Model of Events for the Representation and Dramatization of Storytelling

Erick B. Passos Cesar T. Pozzer* Anselmo A. Montenegro
Flávio S. C. da Silva** Esteban G. Clua

UFF - MediaLab *UFSM - LaCA **USP - IME

Abstract

Storytelling is an important feature in games and also other types of (semi) automated entertainment systems such as machinima and digital-TV. The majority of the current research in storytelling use precedence-based directed acyclic graphs, or even linear sequences, to model the ordering of events in a story. This approach makes it easier to plan, recognize and perform these events in real-time, but it is also too simple to represent complex human actions, which form the basis of the most interesting stories in this niche. PNF-Networks and Interval Scripting are frameworks to represent, recognize and perform human action that was proposed in the context of computer-aided theatre. In this paper we describe two extensions to this framework that were designed and developed to enable its use in larger scale storytelling systems: Hierarchical PNF-Networks and a template-based definition. Hierarchical PNF-Networks present lower complexity propagation heuristic while the definition language enables high-level and abstract description of the temporal structure of the actions and events that compose an interactive story or game.

Keywords: storytelling, PNF networks, interval algebra

Authors' contact:
{epassos,anselmo,esteban}@ic.uff.br
*pozzer@inf.ufsm.br
**fcs@ime.usp.br

1. Introduction

A strong trend in game design is to include storytelling elements and narrative structures to enhance the player experience. However, being interactive applications, games put several challenges into this integration. These challenges are also present in other applications of interactive storytelling such as experiments on machinima [Riedl et al. 2008, Jahala et al. 2008] and interactive TV [Pozzer 2005], and can be divided into three main categories: story modeling, generation/planning, and dramatization. Story modeling tackles the problem of representing the events that make up a story, creating a solid framework for planning and dramatization tools. Planning algorithms have been used to create coherent story models and to keep it in this fashion, even in the presence of user generated events. Dramatization systems are designed to present the storyline and to interact with the user. Besides, the control of virtual actors or other agents such as camera systems still present opportunities to enrich the user experience.

Many storytelling models are constructed as directed acyclic graphs (plans are an example of such graphs), which can represent only incomplete temporal relations such as precedence or causality [Charles et al. 2003, Ciarlini et al. 2005, Barros and Musse 2005]. Since these temporal models are simple, the integration with planning and real-time dramatization systems becomes more straightforward. However, these models are a poor representation of complex human actions, which are the most common type of events in stories. This is due to the fact that these models are especially weak regarding temporal relations between events, being unable to directly represent certain types of parallelism such as mutual exclusion [Allen and Ferguson 1994]. We believe that a richer representation can lead to stories with higher complexity and consequently to a more interesting user experience.

PNF Networks were introduced by Pinhanez [Pinhanez et al. 1998] as a human action recognition framework, which uses the qualitative temporal predicates of James Allen's interval algebra [Allen 1983] to label the relations between the events that compose an action. Unlike precedence-based directed acyclic graphs, the constraints of a PNF network can represent all possible temporal relations that are present in the real world. The problem with interval algebra networks is that the propagation of the temporal constraints is NP-Hard. However, Pinhanez proposed a heuristic for this problem, called PNF-Propagation, which is linear on the number of constraints for networks where the knowledge about the temporal status of each event is discrete, restricted to three possible values and their combinations: P (past), N (now) and F (future). For instance, an event that is known to have already happened is labeled P, while an event that is known to be happening is labeled N. An event labeled PN is currently happening or already happened, and an event whose status is unknown must be labeled PNF.

This framework, augmented with real-time sensor output and past information, was used to represent and recognize complex human actions in the context of computer theatre. Pinhanez also proposed a script language [Pinhanez et al. 1997] that combines PNF-Propagation with user-defined sub-actions to control

interactive experiments in real-time. The use of these techniques to storytelling purposes was mentioned but not implemented by the original authors.

In this paper, we propose two extensions to Pinhanez's PNF-Networks that were developed to make them suitable as a model and dramatization tool for interactive storytelling. The main contributions of our work are: a template-based definition language that is used to compose high level stories by using abstract, reusable descriptions of actions; and a hierarchical version of PNF-Networks, that is augmented with an adaptation of the original restriction heuristic. The adapted heuristic aims to avoid unnecessary traversal of unaffected nodes in the network, leading to a lower complexity that enables the use of larger networks/stories. This Hierarchical PNF-Network was designed, implemented and integrated with a game engine to permit further research on interactive storytelling.

The rest of the paper is organized as follows: Section 2 compares the proposed system with related work in the field. Section 3 explains PNF-Networks and the Interval Script paradigm in more detail, while section 4 explains the new features we are proposing to the original approach. Section 5 brings an analysis of how these features are related to common storytelling concepts and constructs, which is another contribution of the present work. Finally, section 6 concludes the paper and shows future directions of our research.

2. Related Work

This section is organized in two parts: the first covers some important previous research in storytelling modeling and planning techniques. But since this work is strongly based on the PNF-Networks temporal model and algorithms, we also include here the origins and more recent work in this subject.

Chris Crawford created an interactive storytelling system, *Erasmatron* [Crawford 2004], in which the user generates stories by using *verbs* that define the actions of the characters. Each verb has an associated set of *roles*, which can be linked to the characters and objects in the particular setting. These *verbs* represent the same concept as the actions in our framework, while the *roles* are similar to the variables in our template definition language. Although Crawford´s approach is more align with character-based emergent storytelling and PNF-Networks fit better plot-based storytelling, his ideas can still benefit from PNF-Networks to detect what actions a particular character has already performed.

Plans have become one the most common approach for storytelling research partly due to their similarities to story models that emerged from narratology studies [Barros and Musse 2007]. Several previous researches included planning as part of their storytelling systems [Riedl and Young 2003, Charles et al. 2003, Ciarlini et al. 2005, Barros and Musse 2005]. We consider our work complementary to these, since we are not proposing new planning algorithms, but the integration of the Hierarchical PNF-Networks framework with current planning approaches as a formal model for the real-time execution and detection of the events that compose an interactive story.

PNF-Networks are a model and recognition framework for complex human action. It is been implemented and tested in several experiments and also with computer-aided theatre plays. We strongly recommend reading some of Claudio Pinhanez's published work [Pinhanez 1999, Pinhanez et al. 1998, Pinhanez et al. 1997] to better understand the extended framework we propose in this paper.

Meyer [Meyer 2002] further experimented with computer-aided theatre, introducing the use of XML as the definition language for the PNF-Networks. In this paper, we propose further extensions to Pinhanez's work, and also analyze some challenges and opportunities that derive from its application in storytelling research.

3. PNF Networks and Interval Scripts

PNF-Networks are a symbolic framework for modeling high-level events such as human actions - and an algorithm for detecting the occurrence of such events in real-time - that is rich in its representation power and features linear computational complexity, making it suitable to runtime interactive systems. Such as other models for representing high-level events in a story, PNF-Networks decompose actions into simpler sub-actions, but the detection of their occurrence is based on temporal reasoning about the state of the other actions in the network with a heuristic called PNF-Propagation.

Interval scripts are a proposal for the integration of PNF-Networks with user-scripted actions, which are activated by a real-time engine that uses the PNF-Propagation heuristic to determine what actions should be started or stopped. In this section we explain some terminology and the main concepts behind PNF-Networks: action representation by the means of networks of temporal constraints; the PNF-Propagation heuristic; and the interval script approach for complex action activation.

3.1 Terminology

Informally, an action is something performed by some being (character, animal, machine) that can have some consequences. An event can be understood as the perception of an action, its consequences or a natural phenomenon. In this paper, however, this subtle distinction is not so important to the mathematical formalism. More important is the perception that both

events and actions happen in intervals of time, which states that if enough information is available, one can distinguish if an event (or action) is currently happening (present), already happened (past) or has not happened yet (future). For now on, we may refer to an event or action with the term *interval*.

The other important terminology is related to the temporal relations between these intervals. Each event or actions will be represented by a node in a graph corresponding to its interval. PNF-Networks are graphs where the nodes are connected by binary directional relations. Each relation represents a constraint that affect the temporal state of the *restricted* (the destination node of the directed edge) interval based on the state of the *restrictor* (the origin of the edge).

3.1 Action composition

Most human actions can be described by a composition of simpler sub-actions. This concept is key to storytelling systems and, for instance, lets describe the events that can form a common drama in love stories: the knight characters declares this love for the princess, who is to marry a prince from another kingdom. The component sub-events can be: a) establishing shot of the castle; b) knight declares love; c) prince comes and kisses the princess; and d) princess wonders about what just happened. In order to understand this drama, the viewer must see the characters perform each of these sub-events in some organized fashion. Even this simple example will be enough to show how different models drift in representation power.

Since the occurrence of each one of these sub-actions must happen in definite intervals of time, one must define temporal relations between them to properly model and control either its runtime execution or automatic detection. A simple approach to model the temporal relations of these intervals is to consider that the sub-actions always occur in sequence, which would lead to the directed acyclic graph shown in Figure 1, where each edge represent a precedence relation. When used to control automated characters and objects such as a virtual camera, this model shows the order in which the events must be performed. If used to detection purposes, the system must perceive that the events happened in the declared order, to properly conclude that the described drama has actually been presented to the viewer, in case a Finite State Machine will easily suffice.

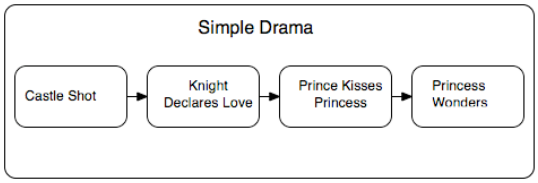

Figure 1: Sequential model for a common drama

The sequence in Figure 1 is enough when one just wants to use it as a high level representation for scripting purposes, but in many cases, some of the component sub-events can happen in parallel or in unknown order. For instance, this concept can be naively represented by the graph in Figure 2, where the two events of the knight declaring its love and the kiss between the prince and the princess are not tied to a specific order. However, this simple parallel model fails to capture an important temporal restriction of the real world: the impossibility of these two events to happen at the same time, which would otherwise lead to a inconsistent performance. The overall conclusion is that models for temporal relations between events that are based solely in precedence are not able to represent mutual exclusion situations properly [Allen and Ferguson 1994].

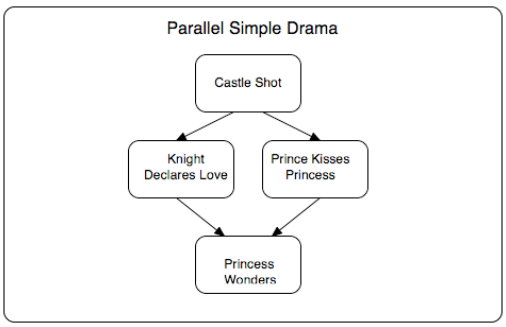

Figure 2: Naive parallel model of the drama event

Before coming back to the drama example, lets first introduce some concepts that are important to the understanding of PNF-Networks.

3.2 Temporal constraints: IA-Networks

To better represent high-level actions, one needs a richer model for temporal relations between the component sub-actions, which proper capture their complex nature. Instead of relying only in precedence and composition, the temporal relations defined by James Allen's Interval Algebra [Allen 1983] are a set of thirteen binary predicates, each representing a unique temporal relation between two intervals. Being an *algebra*, all possible knowledge about the temporal relation between any two events is proven to be a subset (empty included) of the predicates described by the *Interval Algebra*. These predicates are illustrated in Figure 3. One can notice that, except for the equal relation (the first on Figure 3), all other are pairs with an inverse version represented by an "*i*" at the beginning. For instance, the inverse of A <*Before*> B would be A <*iBefore*> B, which, concerning only the temporal relation, is the same as B <*Before*> A. However, the inverse relations are made necessary because the edges in an interval algebra graph are directed.

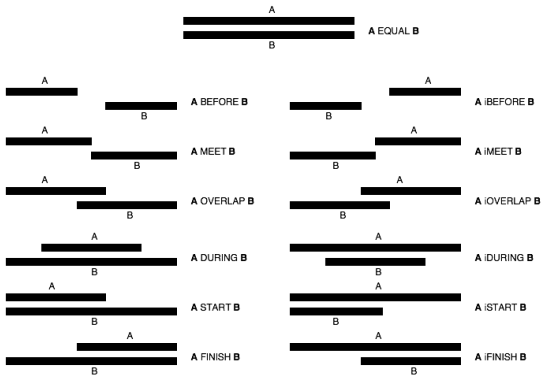

Figure 3: Allen's primitive temporal relations between two intervals

Two illustrate how these predicates better represent temporal relations between sub-actions, we can use them to define the most common relation in storytelling models: precedence. Saying that interval A precedes interval B implies that the first must end its execution before the second starts its own. But the actual start of interval B can either immediately follow the end of interval A or take some time to begin. This means that two predicates in Allen's algebra can be used to represent precedence: *Before* or *Meet*, which leads to three possible interpretations for the precedence relation:

- A <*Before*> B, where the start of interval B must take some time after the end of A to actually happen;
- A <*Meet*> B, where the start of interval B immediately follows the end of A;
- A <*Before*,*Meet*> B, where interval B either immediately follows or happens after some time A has finished, which is the usual meaning of precedence in more simple models.

These three possible representations for precedence are impossible to distinguish unless a more powerful set of predicates is used. In Interval Algebra, high level events (or long chains, sequences or other arbitrary sets of events) are represented by directed (possibly cyclic) graphs called IA-Networks, where the nodes are the sub-events and each directed edge is a subset of the thirteen predicates that compose Allen's Interval Algebra. These networks are useful when one wants a richer and finer grained representation of temporal relations. Figure 4 uses a possible IA-Network to the drama event of previous examples. It is important to notice that in IA-Networks the relation between the high-level event and the same types of predicates also represent its component sub-events. The "castle shot" camera sub-event marks the start of the "drama" event, so it is restricted by a <*iStart*> (inverse start) relation.

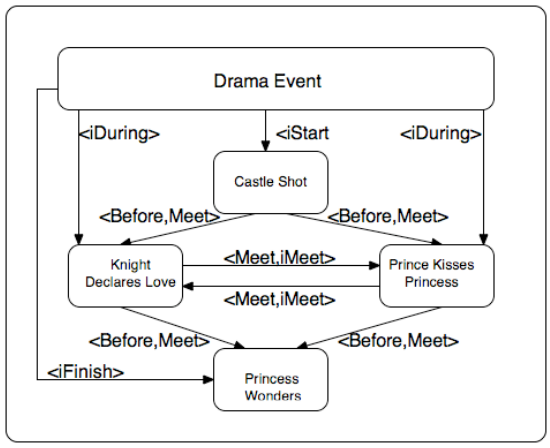

Figure 4 - Drama model represented by an IA-Network

3.3 PNF Propagation

The most important feature of an IA-Network is that it makes possible temporal reasoning, which means that one can use the knowledge about the temporal status of some nodes to infer about other nodes in the network. This is possible because each (directed) relation possibly restricts the status of the second node based on the temporal status of the first. For instance, if node A and B are related by the predicate *Meet*, and node/interval A is known to be happening now, one can conclude that node/interval B is yet to happen.

The problem with temporal reasoning in IA-Networks is that propagation algorithms for them are normally NP-Hard, which are not suitable for real-time systems, especially when used to represent complex stories (large IA-Networks). In order to solve this issue, Pinhanez [Pinhanez et al. 1998] proposed the simplification of IA-Networks by restricting the temporal knowledge about an interval to three discrete values and their combinations: P (past), N (now) and F (future). For instance, an event that is known to have already happened is labeled P, while an event that is known to be happening is labeled N. An event labeled PN is currently happening or already happened, and an event whose status is unknown must be labeled PNF.

At the same time, the temporal relations defined by Allen's interval algebra can be mapped to correspondent constraints also using discrete PNF values. A PNF constraint is a three-valued tuple, where each value is a combination of the values P, N and F. The first value in the tuple marks the possible status of the *restricted* interval given that the *restrictor* has in its current state the value P, the second value marks the possible status values for the presence of N in the origin, and finally, the third value restricts the destination in presence of F in the origin. The complete restriction is given by the sum set of the restrictions of the tuple. A conversion table between interval algebra relations and PNF constraints is given in Figure 5.

	P	N	F
E	< P ,	N ,	F >
B	< PNF ,	F ,	F >
iB	< P ,	P ,	PNF >
M	< PN ,	F ,	F >
iM	< P ,	P ,	NF >
O	< PN ,	NF ,	F >
iO	< P ,	PN ,	NF >
S	< PN ,	N ,	F >
iS	< P ,	PN ,	F >
D	< PN ,	N ,	NF >
iD	< P ,	PNF ,	F >
F	< P ,	N ,	NF >
iF	< P ,	NF ,	F >

Figure 5 - Mapping of Allen's primitives into PNF constraints, which defines a PNF-Restriction function

For instance, the relation *Meet* would be represented by the tuple <PN,F,F>. The first value PN means that if the origin interval is possibly in the past (P) state, the destination can either be happening now or already happen. The second value, F, implies that if the origin has the state N, the destination has to be in the future. Similar constraint happens with the third value. The sum set of the values needs to be used because sometimes one cannot know the exact status of a given interval, which will lead to more than one possible restriction path. This mapping defines a function that returns the value of the *restricted* interval (X_i) based on the current value of the *restrictor* (X_j) and the binary constraint between them (C_{ij}):

X_i = PNF-Restrict(X_j, C_{ij});

Given this 3-valued discrete representation of the status and restrictions for intervals (or events that happen during this intervals), Pinhanez developed an arc-consistency heuristic [Pinhanez et al. 1998] to propagate changes in a PNF-Network (IA-Network with PNF mapped relations). This arc-consistency heuristic is linear on the number of constraints in the network and computes the restricted temporal status for each node (interval).

Formally, a set of PNF values for all the nodes in a PNF-Network is a component domain. The goal of a restriction algorithm is to find the minimal domain that satisfies all the constraints in the network, given the fixed PNF values of the sensor nodes. The arc-consistency heuristic is conservative and always finds a domain that satisfies the constraints, but its not guaranteed that this domain is minimal. Code 1 shows the pseudo-code for this heuristic.

```
Input:   A PNF-Network N with values x1, x2,
         ... , xn, and the binary constraints
         Pij;
         W, a component domain of PNF values
         for N.
Output:  AC-R(W), a component domain that
         satisfies the constraints

Algorithm:
queue = all variables xi where  Wi ≠ PNF;
W_AUX = W;
while queue is not empty
   xq = pop(queue);
   for each xi in W
      X = PNF-Restrict(W_AUXq,Pqi);
      if (W_AUXi ≠ W_AUXi ∩ X)
         W_AUXi = W_AUXi ∩ X;
         push(xi, queue);
return W_AUX;
```

Code 1 - Arc-consisntency heuristic [Pinhanez 1998]

PNF-Networks coupled with arc-consistency are a framework for representing and detecting complex actions, given that the temporal (PNF) state of some nodes in the network are determined by the output of real-time sensors. Each time a sensor output changes, the PNF-Propagation routine is executed, resulting in a new state for the nodes in the network. However, this framework alone is not useful in a storytelling scenario, where one needs mechanisms to automatically control virtual actors and other AI controlled objects.

3.4 Interval Scripts

Interval scripts are an extension to PNF-Networks that enable its use as an engine for controlling the execution of automated characters or other objects by using the results of real-time detection of user (and other automated) actions with the described PNF-Propagation approach. An interval script is a PNF-Network with three types of nodes:

- Sensor - node whose temporal status is given by direct detection such as determining if an object is touching other. Its status cannot be changed by PNF-Propagation;
- Passive node - represents an event that cannot be directly detected. Its temporal status is solely given by PNF-Propagation;
- Actions - contains callback functions that are executed every time its temporal status should change. Its PNF status is defined by user code instead of PNF-Propagation.

The difference from basic PNF-Networks is the presence of action nodes, which are placeholders for user-generated scripts. It is important to notice that the interval script engine will never change the temporal status of any user-generated script (action node). Instead, whenever a change in the network implies a temporal change in such action node, the engine executes the callback function that is correspondent to the expected change on the temporal status.

- Start - callback that should contain the code to execute when the action is started. It is not necessary that the temporal status of the action to be N (now) after the execution of this callback;
- Stop - must contain the code to be executed when the action should be stopped. Similarly,

does not guarantee that the status is P (past) after its execution;
- State - function that should return the actual current temporal status of the action node. This status is the combination of the values P, N and F, as any other node in a PNF-Network.

Deciding what "start" and "stop" functions are executed, given the current state of the nodes in the network, does the control of the activation of action nodes. The steps to be taken in the heuristic proposed to this task in [ref] are:

1. Determination of the current states of all the nodes of the network;
2. PNF-Propagation of constraints to find the restricted desired states;
3. Thinning of the solution: the result of the last step can possibly contain more than one solution for the network, so a heuristic is used. For each action node, its desired state is computed as the intersection of its current state and the result of the PNF-Propagation. If the result of this intersection is empty, the state computed by the PNF-Propagation is used;
4. Execution of the necessary callback functions based on the thinning process, for all action nodes.

Table 1 describes what callback function to execute, given current and desired temporal states, where the later is the result of the thinning process. The first column denotes the presence of the given state, meaning that the current state of the action only needs to contain the denoted value. With the thinning process, for each action node, the heuristic actually tries not to call either the "start" or the "stop" callbacks, unless its actual current state contradicts the result of the PNF-Propagation.

Table 1 - Conditions for the activation of callbacks

Current	Desired	Callback
F	N,PN	Start
N	P	Stop
F	P	Stop

4. PNF Extensions

In this section we explain our main contributions to the existing framework: the template-based definition language and the hierarchical PNF-Propagation algorithm, which reduces the complexity for large networks.

4.1 Template-based definition language

One possible benefit of PNF-Networks is its ability to represent events based on previously-defined reusable sub-events. However, the whole system must be integrated with custom real-time game/graphic engines that implement the virtual sensors and the actual user-generated scripts with the callback functions. One of the most important features of such integration is the ability to represent reusable sub-events without having to specify the exact objects and actors that compose them. We developed a template system that provides for an easy to use method for describing and composing PNF-Networks in XML. The code below shows a simple network of two nodes and the causality constraints between them.

```
<network>
   <node name="cause" type="sensor" >
      <scene object="box" target="player" type="collision" />
   </node>
   <node name="consequence" />
   <constraint origin="cause" destination="consequence"
                type="b,m" />
</network>
```

Code 2 - Simple PNF-Network in XML

This XML sample shows the basic language to describe a network of PNF-Constraints. The <*node*> tag is used to describe all the intervals, with a required *name* attribute to define a label for each one. Besides being an id for the parsing and template mechanisms, this attribute is used in real-time to implement a query system for the temporal state of any node. The *type* attribute is optional for *passive* nodes and obligatory for *actions,* and *sensors*.

The *sensor* node type needs a special <*scene*> tag to define the attributes that specify the kind of sensor and the real-time objects that are subject to its "sensing" mechanism. Actual sensors are implemented by the customized graphic engines and the role of this XML specification is to describe its *type*, which can be *collision*, *proximity*, *look* or *grab*; and other attributes that are needed in real-time. For a sensor node, the necessary attributes are the *object* that should host the sensing mechanism and its *target*, which represents the label that another object must be marked with to properly activate the sensor. For instance, in the example above the real-time engine should host a collision sensor on the object named "box" and detect the collisions with objects labeled "player". By definition, sensor nodes are marked with the temporal state PF (past or future) when are not activated; and N (now) when activated.

The most useful features of the template system are its ability to reuse previously defined actions and the use of variables to describe the objects that compose them. The following XML sample shows the definition of a template, also showing how variables and action nodes are defined.

```
<template name="pass-through">
   <node name="pass" />
   <node name="begin" type="sensor" >
      <scene host="$1" object="$3" type="collision" />
   </node>
   <node name="end" type="sensor" >
      <scene host="$2" object="$3" type="collision" />
```

```
    </node>
    <constraint origin="begin" destination="pass" type="s" />
    <constraint origin="end" destination="pass" type="f" />
</template>
```
Code 3 - Sample template with wild cards

A template is similar to any other PNF-Network, but it is necessary to start with the definition of a *passive* node, normally one whose interval bounds the sub-network. Later on, when reusing this template to compose higher level structures, this passive node is the one to be connected by the constraints defined in the higher level network. The other difference is the inclusion of variables (marked with the *$* symbol) that can be replaced whenever the template is re-used in another definition.

This particular example shows a simple template that detects if a given object (represented by the variable $3) passes through two locations (respectively $1 and $2) in a particular order. The constraints guarantee that the passive node "pass" will be in the N (now) state as soon as the object touches the first sensor and in the P (past) state only after a collision with the second sensor (location) ceases to exist. Any other ordering of the events will collapse the network, meaning that the described event actually did not happen. Code 4 shows how to compose a higher level network by reusing previously defined templates and action nodes, that can be used to control automated scripts in the graphic engine.

```
<network>
    <node name="pass-instance" template="pass-through">
        <param name="$1" value="trigger1" />
        <param name="$2" value="trigger2" />
        <param name="$3" value="player" />
    </node>
    <node name="consequence" type="action">
        <scene object="npc" />
    </node>
    <constraint origin="pass-instance"
                destination="consequence"  type="m" />
</network>
```
Code 4 - Reuse of a template in a higher level network

The above example defines a simple network that activates the script represented by the "consequence" node immediately after the end of the event represented by the "pass-instance" node, which is an instance of the previously defined "pass-through" event. The activation is guaranteed by the MEET (m) constraint between the two nodes. The sample also shows how the variables can be replaced by the actual values to be used by the real-time graphic engine to implement the sensors. The definition of *action* nodes, those that control the execution of user-created scripts, also needs the specification of a scene attribute that indicates the object that should contain the callback functions *start* and *stop*. In our implementation, we use the message passing API of the Unity3D Game Engine [Unity Technologies 2009] to perform this operation.

When a template is instantiated, it acts as a wrapper for an independent sub-network. The constraints applied over the instanced node will be connected to the wrapper, instead of the internal nodes themselves. This wrapper acts as a barrier that detects when internal or external PNF-Propagations should be passed or not. This will happen only in the case where the temporal state of this wrapper changes. Figure 6 shows the graph representation for the hierarchical network presented in Code 3.

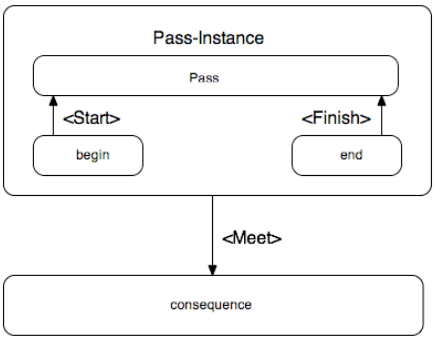

Figure 6 - hierarchical PNF-Network

4.2 Hierarchical PNF Propagation

The PNF-Propagation algorithm is linear on the number of constraints of the network; however, this can still be a limiting performance factor when considering the possibility of very large networks that represent complete narratives, especially with the control of low level virtual actors. Due to the structure of the most common narratives found in the literature of storytelling research, we decided to create the hierarchical PNF-Networks model, which can be though as a layered graph, with the highest level comprising a sequential or parallel plot, composed of nodes connected only by *meet* constraints. The lower levels of the graph can use more sophisticated PNF constructs, for instance to enable the control of multiple actors in a scene.

For instance, in the sample network of Figure 4, which represents an event in a plot, there is one *passive* node, the higher level "drama"; being all other nodes user-implemented *actions*. One can notice that all the "internal" nodes of that network are bound by the interval represented by the "drama" *passive* node. This pattern of using a passive node to bind all internal intervals is important to the definition of coherent hierarchical PNF-Networks.

This guarantees that internal nodes of two different sub-networks never overlap, and PNF constraints are necessary only between nodes in the same level/layer. The wrapper node encapsulates all the links of the internal sub-network (actually the first passive node) with the external one (the higher level network). Using this feature, the PNF-Propagation algorithm can be restricted to only the sub-networks that are to be affected by its execution. For each wrapper node that connects a sub-network to its container network, the

algorithm in Code 5 is executed every time a PNF-Propagation occurs at the container level, while Code 6 is executed after an internal PNF-Propagation is fired.

```
// pNode is the passive node of the internal network
// wNode is the wrapper node that connects with the
// external network
if (pNode.state != wNode.state)
{
   pNode.state = wNode.state;
   pNode.propagate();
}
```

Code 5 - blocking external PNF-Propagation

Whenever an external PNF-Propagation is fired, this algorithm will block its execution for the internal network, including any sub-network on levels below. This provides for a barrier that will prevent unnecessary traversal of sub-networks that are not to be affected anyway. The algorithm in Code 5 is the same; the only difference being the direction of the propagation, in this case to the external network.

```
// pNode is the passive node of the internal network
// wNode is the wrapper node that connects with the
// external network
if (pNode.state != wNode.state)
{
   wNode.state = pNode.state;
   wNode.propagate();
}
```

Code 6 - blocking internal PNF-Propagation

Simple structures like these shown here can be used to compose complex narratives and provide for a runtime engine that controls its execution. In the next section we will show how this extended version of the original PNF-Network framework relates to common storytelling concepts and constructs.

5. Integration with storytelling concepts

With this paper, we propose the use of the hierarchical PNF-Network framework to model and control some levels of an interactive storytelling system. In this section we describe how the PNF model relates to common concepts in storytelling research.

5.1 Planning and execution with plot-based storytelling

Several previous works have focused on the use of planning algorithms to generate narratives [Riedl and Young 2003, Charles et al. 2003, Ciarlini et al. 2005, Barros and Musse 2005]. When these algorithms are used with a plot-based approach, the output often is an ordered (partially or totally) set of events, composed of subjects, verbs and objects. There is still research to be done in this subject, but in this section we show a possible mapping of the output of a planner to a hierarchical PNF-Network.

The generated plot can be mapped to a high-level PNF-Network, with each verb corresponding to a previously defined action, which is by itself composed of others, lower level, sub-actions. The subjects and objects of the plot sentences are the parameters that shall replace the variables needed by these action templates to be complete. For instance, the following story can be directly mapped to the (high level) PNF-Network of Figure 7. The sentences are meant to represent a very condensed version of the movie Star Wars, episode IV: A new hope.

1. Imperial troopers kill Luke's uncle and aunt
2. Luke meets Obi wan
3. Yoda trains Luke in Jedi
4. Luke joins the rebels
5. Luke destroys the death star

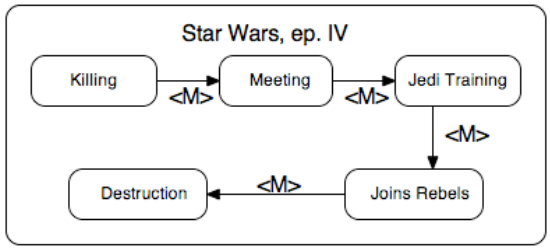

Figure 7 – Highest level of the Star Wars PNF-Network

In Figure 8, we chose to represent the last event of the plot, the destruction of the death star, because it highlights some features of the PNF-Network model. This event is composed of several sub-actions performed by the rebel and imperial spaceships, and ends up with the final destruction of the star-like weapon.

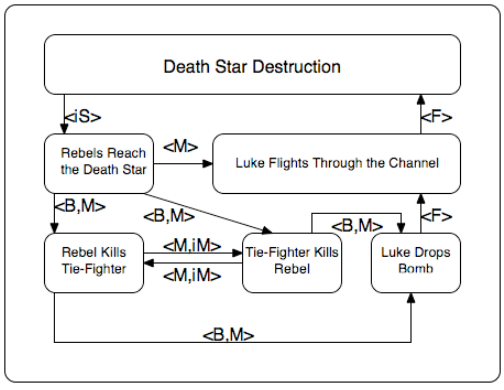

Figure 8 – Death star destruction PNF-Network

The actions that compose this mathematical model for the death star destruction can be realized in different ordering, but will always finish with Luke's precise bomb drop that explodes the imperial weapon. The "Death Star Destruction" node is the passive internal interval that bounds the execution of the others. The inverse start (iS) constraint to the "Rebels Reach the Death Star" action will guarantee that the later will be started as soon as a higher level PNF-Propagation changes the temporal state of the first to N

(now). The rest of the relations will bring parallel actions such as "Luke Flights Through the Channels" and a mutual exclusion between the two killing actions.

For now, our system is suitable for plot-based stories whose sentences are based in verbs that can be mapped to high-level PNF defined events. There is still work to be done to enable runtime planning in the case of user intervention. This is our main research interest at this moment.

5.2 Virtual cinematography

Many previous research, and this work is no exception, on modeling and planning for storytelling tend to describe the plots as high level events, leaving room for the use of different techniques to enhance the viewer experience. One important research area in interactive storytelling is the use of traditional cinematography concepts as automated agents.

We are experimenting with two approaches for the integration of cinematography agents with hierarchical PNF-Networks: camera actions and director agents. With camera actions we include action nodes in our PNF templates so that every time that template is used in a story it already comes integrated with the camera shots (as action nodes linked to pre-defined camera scripts). Figure 9 shows a PNF template for the sub-events of the drama that was analyzed in section 3, including camera actions that are tied to the sub-events with the constraint equal (E).

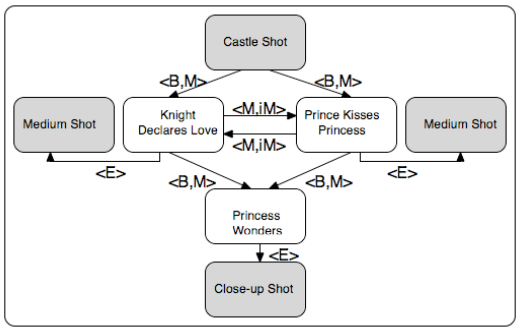

Figure 9 - Drama event with camera actions (gray nodes)

Another option, that is more flexible than the latter is to establish a communication protocol with an independent agent that chooses the camera shots to be used. The general idea is to send all *Start* and *Stop* messages to this director agent as well. The message to this director agent must include a reference to the PNF action that received it originally, so that useful information can be extracted from the mathematical model such as the name of the action (verb that possibly defines the type of shot to be used) and the parameters that link this action to the real-time characters and objects on the graphic engine. Figure 10 shows a schematic representation for this communication architecture.

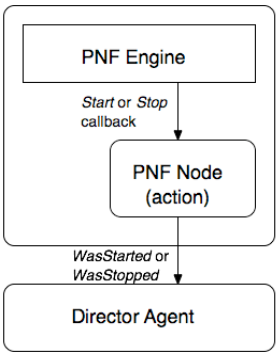

Figure 10 - Communication between the PNF engine and a director agent

5.3 Flashbacks and out-of-order execution

A narrative concept that helps the viewer to mind important events, while at the same time enhances the storyteller possibilities is the use of flashbacks [Bae and Young 2008], a replay of a part of the story already told/seen. The PNF-Networks original framework was not design with this possibility in mind, but the hierarchical feature of our model provides for partial independency of (lower level) events, meaning that the whole internal structure of a sub-network can be reset, leaving room for its (re) execution, possibly with a different approach from the director agent, highlighting the flashback timing with image filters or different shots.

This idea can be extended to out-order-execution if the first level of the network (plot) is controlled by another engine, which uses different decision rules to choose the ordering of the events. The plot events are now independent PNF-Networks that can be started in any desired order, but still benefitting from the features of the temporal model behind it at the lower levels.

5.4 Character-based emergent storytelling

Character based storytelling relies on the interaction of virtual actors that are agents directly driven by goals or assisted by some sort of planning. Creating coherent and interesting stories in real-time with this approach is still the objective of this research field, and one of the challenges is related to the way virtual actors perceive the environment.

PNF-Networks can be used with this approach to model and recognize the actions of all the virtual actors, providing for a flexible framework for the agents to perceive and react accordingly to its plan or drives.

These are some integration ideas made possible by Hierarchical PNF-Networks. We are still investigating if and how other key storytelling concepts can be used with this framework.

6. Conclusions and Future Work

With this paper we presented the Hierarchical PNF-Networks and its template-based definition language, two extensions over the original PNF-Network framework, a fine-grained representation for temporal relations between intervals that can be used to describe high level events such as complex human action. The two extensions made possible the use of PNF-Networks to represent large sets of interdependent events such as interactive. The XML schema used to describe the networks facilitates the description of a library of events and actions to be used by a storytelling system.

We are currently researching more extensions to Hierarchical PNF-Networks. For instance, we are trying to integrate planning algorithms to create and modify PNF-based stories in real-time. It is easy to adapt a current planning algorithm to only generate a PNF-Network that is solely composed of precedence relations such as *Meet*, but that approach would not make any use of the richer representation that these networks can bring to interactive storytelling models. Our goal is to create a system that can be used in multi-user scenarios such as emergent stories in massive games or user-influenced plot-based narratives for digital-TV.

References

ALLEN, J. F., 1983. Maintaining Knowledge about Temporal Intervals. 1983. *Communications of the ACM*, 26 (26), Pages 832-843.

ALLEN, J. F. AND FERGUSON, G., 1994. Actions and Events in Interval Temporal Logic. 1994. *Journal of Logic and Computation*, vol. 4 (5), Pages 531-579.

BAE, B., AND YOUNG, R. M., 2008. A Use of Flashback and Foreshadowing for Surprise Arousal in Narrative Using a Plan-Based Approach. *In: Proceedings of the 1st Joint International Conference on Interactive Digital Storytelling: Interactive Storytelling, 2008 Erfurt, Germany*. Pages 156-167.

BARROS, L. M. AND MUSSE, S. R., 2007. Planning algorithms for interactive storytelling. 2007. *Computers in Entertainment (CIE)*, vol. 5 (1), Article No. 4.

BARROS, L. M. AND MUSSE, S. R., 2005. Introducing narrative principles into planning-based interactive storytelling. *In: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology, 2005 Valencia, Spain*. Pages 35-42.

CAVAZZA, M. AND CHARLES, F. 2005. Dialogue generation in character-based interactive storytelling. *In: Proceedings of the AAAI First Annual Artificial Intelligence and Interactive Digital Entertainment Conference, 2005 Marina del Rey, CA, USA*

CHARLES, F., IBÁÑEZ, M. L., MEAD, S. J., BISQUERRA, A. F. AND CAVAZZA, M. 2003. Planning formalisms and authoring in interactive storytelling. *In: Proceedings of TIDSE'03: Technologies for Interactive Digital Storytelling and Entertainment, 2003 S. Göbel et al. eds. Fraunhofer IRB Verlag, Darmstadt, Germany.*

CIARLINI, A. E. M., POZZER, C. T., FURTADO, A. L. AND FEIJÓ, B. 2005. A logic-based tool for interactive generation and dramatization of stories. *In: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology, 2005 Valencia, Spain*. Pages 133-140

CRAWFORD, C. 2004. Chris Crawford on Interactive Storytelling. New Riders Games, Indianapolis, IN.

JAHALA, A., RAWLS, C. AND YOUNG, R. M. 2008. Longboard: A Sketch Based Intelligent Storyboarding Tool for Creating Machinima. *In: Proceedings of the 2008 Florida Artificial Intelligence Research Society Conference, 2008 Florida, USA*.

MEYER, T. A., 2002. Development of Computer-Actors within the Interval Script Paradigm. Unpublished Dissertation, Massey University. Available from:http://www.massey.ac.nz/~tameyer/research/computertheatre/docs/thesis.pdf [Accessed 10 June 2009].

PINHANEZ, C. S., 1999. *Representation and Recgnition of Action in Interactive Spaces*. PhD thesis, Massachusetts Institute of Technology.

PINHANEZ, C. S., MASE, K. AND BOBICK, A., 1998. Human action detection using PNF propagation of temporal constraints. *In: Proceedings of the 1998 IEEE Computer Society Conference in Computer Vision and Pattern Recognition, June 23-25 1998 Santa Barbara, CA, USA*. Pages 898-904.

PINHANEZ, C. S., MASE, K. AND BOBICK, A., 1997. Interval scripts: a design paradigm for story-based interactive systems. *In: Proceedings of the SIGCHI conference on Human factors in computing systems, 1997 Atlanta, Georgia, United States*. Pages 287-294

POZZER, C. T., 2005. *Um Sistema para Geração, Interação e Visualização 3D de Histórias para TV Interativa*. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro.

RIEDL, M. O., ROWE, J. P. AND ELSON, D. K., 2008. Towards Intelligent Support of Authoring Machinima Media Content: Story and Visualization. *In: Proceedings of the 2nd international conference on intelligent technologies for interactive entertainment, 2008 Cancun, Mexico*.

RIEDL, M. O. AND YOUNG, R. M., 2003. Character-focused narrative generation for execution in virtual worlds. *In: Proceedings of ICVS 2003: International Conference on Virtual Storytelling, 2003 Toulouse, France*. Pages 47-56

UNITY TECHNOLOGIES. 2009. *Unity3D Game Engine* [software, game development tool] Available from: www.unity3d.com [Accessed 01 January 2009].