A Structure for Modern Computer Narratives

Clark Verbrugge

School of Computer Science
McGill University
Montreal, Quebec, CANADA H3A 2A7
clump@cs.mcgill.ca

Abstract. In order to analyze or better develop modern computer games it is critical to have an appropriate representation framework. In this paper a symbolic representation of modern computer narratives is described, and related to a general model of operational behaviour. The resulting structure can then be used to verify desirable properties, or as the basis for a narrative development system.

1 Introduction and Overview

In order to analyze or better develop modern computer games it is critical to have an appropriate representation framework. Existing frameworks are demonstrably inadequate in this respect. In this paper a symbolic representation of modern computer narratives is described, and related to a general model of operational behaviour. The resulting structure can then be used to verify desirable properties of computer narratives, or as the basis for a narrative development system.

1.1 Motivation

Many modern computer games build gameplay based to varying degrees on computer narrative—game progression is defined through a narrative sequence of events. For some genres, particularly "adventure" and to a slightly lesser extent "role-playing" games (RPGs), the gameplay consists almost entirely of building/following a narrative, and a properly-constructed narrative structure is paramount.

As any avid game player is aware, narrative development is an imperfect process; mistakes are evident in unsatisfying plot holes and non-sequiturs. More severe consequences can include unwinnable situations, game unbalancing effects, or even program failure. A rigorous system for developing and describing narratives can aid in reducing or eliminating these problems.

1.2 Narratives and Graphs

In its simplest form, a narrative is a sequential presentation of events. States, important points or actions are necessarily described in order over the time of presentation—a linear plot graph, for example. In this sense a narrative is a total ordering of events.

However, the presentation of a narrative is often regarded as secondary to its internal semantics. While all narratives will unfold sequentially over time, most people understand the events to be "actually" ordered by one or more internally-consistent ordering frameworks. Internal, narrative time, for instance, can allow for two events to occur concurrently ("meanwhile…"); the relating of these two events is then an interleaving of the narrative's necessary ordering.

Plot DAGs Internal order-relations, such as narrative time or physical causality, produce a "plot DAG" (Directed Acyclic Graph). The plot DAG is a graph representation of the significant events/states in a narrative ordered by the internal system: each node represents a state, and directed edges between nodes represent necessary precedence. Since events in a traditional narrative do not repeat, the overall structure is acyclic. A simple example for the ubiquitous door locking "puzzle" found in adventure and RPG games is shown in figure 1.

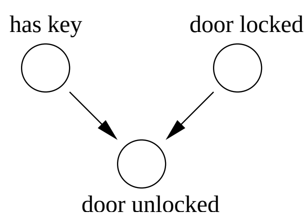

Fig. 1. A DAG structure describing a simple open door task. Multiple incoming edges form an "AND" relation—the door must be locked *and* the key possessed before it can be unlocked.

The above example illustrates a fundamental representation gap: in the plot DAG model, the door cannot be (infinitely) unlocked and re-locked, since that would imply a cycle. Narratives in computer games also tend to incorporate at least some amount of choice, and this is also not representable in the DAG system. Incoming edges in our plot DAG are combined as an "AND" relation, and so we cannot represent the possibility of the door perhaps also being unlocked by picking the lock. In a typical usage, the designer may avoid these issues by including this information as side-notes to their design, or they may attempt to abstract further (collapse the entire situation to just "get door unlocked"). With such limitations, however, plot DAGs are clearly an unsatisfying solution to the problem of representing computer narratives.

1.3 Roadmap

Section 2 discusses related approaches. Section 3 begins the presentation of our formalism. The initial, petri-net derivation is explained in subsection 3.1, and developed into a hypergraph model in subsection 3.2. The final structure, along with path and other dynamic actions is defined in subsection 3.3.

Section 4 discusses the use of this formalism for discovering various properties important to narrative development. The model is then applied to a more complex example in section 5, where we show how our model and approach can be used to describe the initial portion of an actual computer narrative. Directions for further research are then discussed in section 6, and conclusions drawn in section 7.

2 Related Work

There are remarkedly few academic computer science studies of modern computer games, and we are aware of no other formal attempts at representing computer game narratives. The concept of plot DAGs as a design tool for narrative computer games was discussed during the mid-1990's in the usenet discussion group `rec.arts.int-fiction` [BLM+94,OE94,BPA+95] and in an on-line trade journal [For97]. The idea is sometimes credited to the "Oz" group's work on interactive drama [Mat97]. Of course with DAG models cycles and choice cannot be represented, and must be attached as external information.

Approaches to analyzing *text* narratives, however, do exist. For example, Burg et al use constraint logic programming to analyze time-relations in a William Faulkner short story [BBL00]. High-level narrative development frameworks have been described by Strohecker and Brooks [Bro96,Bro97,Str97]; these systems are aimed at all aspects of (perhaps interactive) story development, and not the specifics of representation.

Low-level game "construction kits" also exist, automating the repetitive and/or stock tasks necessary to actually produce a working computer game. These are also typically not representation-oriented; the popular TADS [Rob87] system, for instance, represents narrative implicitly through the control structure of a high-level object-oriented language.

Our hypergraph model is derived by simplifying a form of Petri Net. Similar Petri Net models have been used in modelling "workflow," the abstract relations and precedence requirements of tasks in a business environment [vdA98]. Van der Aalst's workflow model is based on an enrichment of Petri Nets (transitions are augmented by the source of the firing impetus), and naturally does not discuss or define properties important to narratives per se.

3 Formalisms

Our interest is in a simple formalism that can model normal narrative progression (internal-dependencies), including cycles and choice. To keep our formalism simple, we do not attempt to explicitly model time[1]. Mechanics of player inter-

[1] E.g., a game action which must be completed in a specific real-time interval: "you have 30 seconds to defuse the bomb...". Real-time activities do occasionally appear

action (how actions and choices are actually presented and resolved) and other extra-narrative facets are also not considered.

The model we develop is derived from a form of *Petri Net*. The next subsection gives a quick introduction to Petri Nets, and is followed by an explication of our system.

3.1 Petri Nets

Definition 1. *A* Petri Net *is a 5-tuple,* (P, T, E, W, M)*, where* P *is a set of* place *nodes,* T *is a disjoint set of* transition *nodes* $(P \cap T = \phi)$*, and* $E \subseteq (P \times T) \cup (T \times P)$ *is a set of directed edges going between places and transitions. Edges are weighted* $(W : E \to \mathbb{N})$*, as are places via a "marking" or token-assignation function* $(M : P \to \mathbb{N} \cup \{0\})$*.*

Central to the model is the concept of a transition-node being "ready to fire," and the marking transformation that occurs through actual firing. In order for transition t to be ready to fire, there must be enough tokens in each incoming place p to satisfy the edges weights: $W((p, t)) \leq M(p)$. Actual firing conceptually removes tokens from the incoming places according to edge weights, and adds tokens to outgoing places according to edge weights. Note that firing is atomic; substeps in the firing of one transition cannot be interleaved with other transition firings. Also note that a place attached as both input and output to a firing transition will have its marking changed according to the difference between output weights and input weights.

Definition 2. *A* 1-Safe *Petri Net has edge weights of 1 in all cases,* $\forall (x, y) \in E, W((x, y)) = 1$*, and guarantees that markings are always either 1 or 0. If* $M_0 \xrightarrow{t_0} M_1 \xrightarrow{t_1} \ldots$ *is a sequence of markings generated by firing transitions* $t_0,\ t_1,\ \ldots$*, then:*

$$\forall i \geq 0,\ \forall p \in P,\ M_i(p) \in \{0, 1\}.$$

A variation on 1-Safe firing rules further requires all output places of a transition to be empty in order for it to fire. This does not affect expressive power, and for simplicity we will sometimes illustrate based on firing rules of either form. We will, however, explicity permit self-loops, and also assume that our nets are "T-restricted"—transitions do not map from or to empty place sets. The reader is referred to other texts (e.g., [Rei88]) for more comprehensive introductions to Petri Nets.

1-Safe Petri Nets model finite state systems, and are sufficient to represent all narrative structures in which we are interested (formal expressiveness results can be found, e.g., in [CEP95,Esp98]). For instance, an expanded version of the simple lock "puzzle," discussed in Section 1.2 is illustrated as a Petri Net in Figure 2. Transitions are drawn as small lines, places as circles and the marking by the presence/absence of a black dot in each place. To unlock the door, the

in computer narratives, but at least in narrative-centric games like adventure games, real-time situations are not well-liked [Ada98].

door must be currently locked and the key has to be available; firing the "Unlock door" transition changes the door state to "door unlocked," but does not affect the presence of the key.

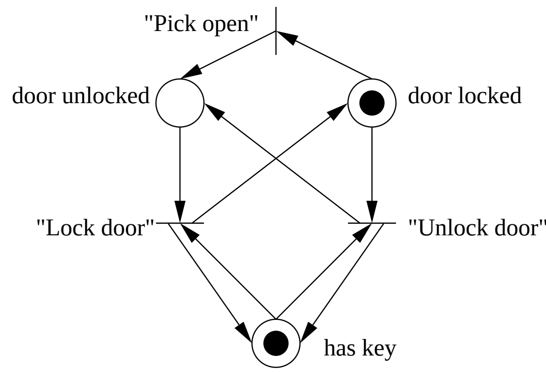

Fig. 2. A 1-safe Petri Net describing a simple lock "puzzle" in a narrative.

Note that as opposed to the DAG example in Figure 1, here the unlocking of the door is reversible—the graph is cyclic. Choice has also been incorporated; the lock is (un)lockable with the key, *or* it can be unlocked by picking the lock (firing the transition labelled "Pick open").

A Petri Net representation is quite flexible, and allows one to model more complex behaviours than a plot DAG. However, the bipartite structure and complex transition (firing) behaviour of Petri Nets adds unnecessary syntactic baggage to our model. In Figure 2, for instance, edges from the lock/unlock transitions are needed to restore the token within the "has key" place—this represents the fact that (un)locking the door does not alter the possession of the key. In essence, the "has key" place functions here not as a transformable state, but as required *context* for other actions. The next section introduces a simplified representation that formalizes our requirements without requiring explicit token movement.

3.2 Hypergraphs

Our formalism is based on a 1-safe Petri Nets, and is a kind of *directed hypergraph:*

Definition 3. *A* hypergraph *is a graph* (V, E)*, with the property that edges (*hyperedges*) can connect more than just 2 vertices:* $E \subseteq \mathcal{P}(V)$*.*

Definition 4. *A* directed hypergraph *is a directed graph* (V, E)*, with the property that (hyper)edges can connect more than one* tail *vertex to more than one* head *vertex:* $E \subseteq (\mathcal{P}(V) \times \mathcal{P}(V))$*. The* $tail : E \rightarrow \mathcal{P}(V)$ *and* $head : E \rightarrow \mathcal{P}(V)$ *functions extract the input or output sets from a given hyperedge.*

A directed graph has directionality assigned to each connection between a hyperedge and a vertex; note that some authors define a directed hypergraph by designating a single, distinct head for each hyperedge: $E \subseteq (\mathcal{P}(V) \times V)$ (e.g., see [AIN92]); we are not following that pattern.

Directed hypergraphs can model the structure of 1-Safe Petri Nets: a correspondence can be built between transitions and hyperedges, and nodes can be identified. A restriction exists in that multiple, identical Petri Net transitions (ie transitions with the same input and output place sets) cannot be represented; this does not affect expressive power. Note that every directed hypergraph can be trivially transformed into a 1-Safe Petri Net if we assume the variant firing rules discussed in subsection 3.1.

In order to model narratives succinctly, nodes that are both source and target of a hyperedge, *contexts* for other actions, should be abstracted out. This motivates the *context hypergraph* as a labelled, restricted variation of a directed hypergraph.

Definition 5. *A* context hypergraph *is a 6-tuple,* (V, E, C, L, L_V, L_E) *where:*

(V, E) *forms a directed hypergraph with the property* $\forall h \in E,\ \mathit{head}(h) \cap \mathit{tail}(h) = \phi$.
$C \subseteq (E \times \mathcal{P}(\mathcal{V}))$ *is such that* $(h, N) \in C \Rightarrow \forall n \in N,\ n \in \mathit{tail}(h)$.
L *is a finite set of labels.*
$L_V : V \rightarrow L$ *is a node labelling function.*
$L_E : E \rightarrow L$ *is a hyperedge labelling function.*

Remarks: A context hypergraph distinguishes nodes that function as *context* connections between a hyperedge h and some subset of tail(h). In the base hypergraph these are simple tail connections for a hyperedge; by identifying them with the C relation it becomes possible to use them for context purposes. An example is shown in figure 3.

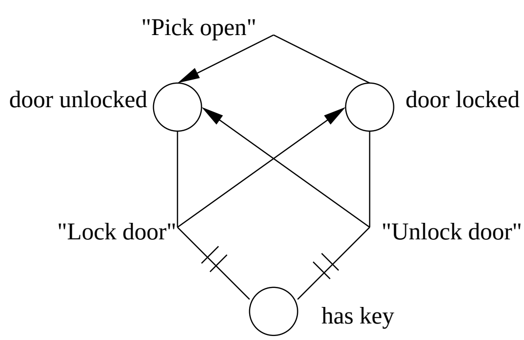

Fig. 3. A context hypergraph model of the same structure as figure 2. Context relations are marked by a line with two short lines intersecting it (eg, the two lines connected to the “has key node”).

3.3 Narrative Flow Graphs

Context hypergraphs form the syntactic basis for our model. We still need the concept of starting and ending nodes, and of course we need algorithms for discovering interesting and useful properties of our system. A simple context hypergraph is not yet sufficient for this. The *Narrative Flow Graph* defines the final representation.

Definition 6. *A* Narrative Flow Graph *(NFG) is a 4-tuple:* (H, a, w, ℓ)*, where* $H = (V, E, C, L, L_v, L_e)$ *is a context hypergraph,* $a \in V$ *is an identified starting, source node with no context connectivity:*

$$\forall h \in E,\ a \notin head(h)\ \wedge\ \forall (h, N) \in C,\ a \notin N$$

and $w, \ell \in V$ $(w, \ell \neq a)$ *are identified ending, sink nodes:*

$$\forall h \in E,\ w, \ell \notin tail(h)$$

The w *and* ℓ *nodes must not be simultaneously-reachable:*

$$\forall h \in E,\ |\{w, \ell\} \cap head(h)|\ \leq 1$$

Remarks: The addition of specific starting and ending nodes allows for paths to be defined in the structure. The initial node, a represents axiomatic precedence—all initial conditions are directly connected to a. Symmetrically, w and ℓ represent termination, either by *winning* or *losing* respectively. The final condition above ensures that a specific hyperedge does not lead to both win and lose at the same time.

For narrative games, the concept of winning and losing is important. For narratives per se, however, termination remains critical, but the distinction between winning and losing is unnecessary. In these cases w and ℓ can be equated, producing a simplified NFG:

Definition 7. *A* Simple Narrative Flow Graph *is an NFG* (H, a, w, ℓ) *such that* $w = \ell$.

The lock example is extended to a simple NFG in figure 4.

Remarks on the Formalism NFGs are a simplified representation of a particular form of 1-safe Petri Net. The advantage of this formalism is in its specificity to the task—through designated nodes and connectivity constraints NFGs formalize a general structure appropriate for narrative representation. There is an additional minor benefit in the representation of self-loops: ambiguity as to the readiness of such a transition in a 1-safe Petri Net (when not explicitly specified) is eliminated when viewed as a context in an NFG. Note that these differences do not alter the close relation to Petri Nets, and so results in that area remain trivial to transfer.

3.4 Traversals

In a regular graph, a path or traversal can be represented as an alternating sequence of nodes and edges, terminating at a destination node. A simple path is a path without any cycles (repeated nodes).

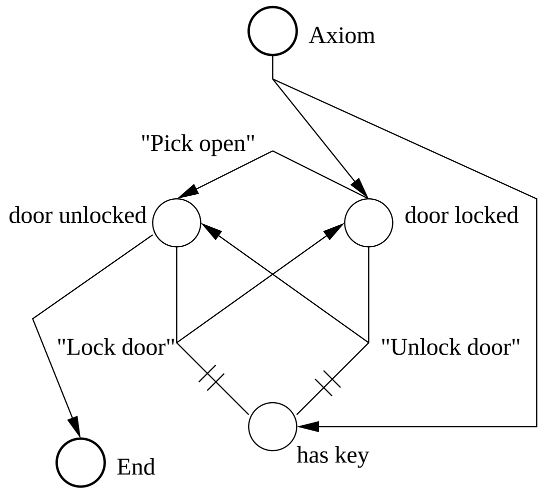

Fig. 4. A simple NFG of the same structure as figure 3. The "game" begins with the door locked and the key available. Once the door is unlocked (either by using the key or by picking), the door can be re-locked, or the game can end. There is no win or lose in this game, just termination.

For hypergraphs, due to the branching of hyperedges, traversal is more complicated, and (following [AIN92]) simple paths are most easily represented as minimal sub-hypergraphs connecting two sets of vertices. For example, in figure 4, one minimal *hyperpath* from {AXIOM} to {END} would include nodes {AXIOM, DOOR-LOCKED, DOOR-UNLOCKED, HAS KEY}, along with the UNLOCK DOOR, axiomatic, and terminal hyperedges.

While we can define traversals using hyperpaths, it may not be that a given hyperpath is a realisable traversal—due to their Petri Net origins, there is an implicit, necessary ordering in a traversal of our hypergraphs, and this is lost in a sub-hypergraph representation. In figure 5, for instance, a sub-hypergraph representing a traversal from A to E is shown (as is its Petri Net equivalent). The version on the left, however, is not actually realisable as a game—C is required for X, but it is also required for E. Context relations exist to permit such paths; the version on the right connects C to X using a context edge, akin to a bidirectional connection to a transition in a Petri Net, and so there is no difficulty in revisiting C before moving to E.

In order to verify path-based properties in our model, then, a simple sub-hypergraph model of traversal is insufficient. We need to consider the (non)existence of contexts. This is captured through the following definition of *flow:*

Definition 8. *Let $H = (V, E, C, L, L_v, L_e)$ be a context hypergraph, and let $X, Y, Q \subseteq V$ be non-empty subsets of nodes. Let $E_{Q\to Y} \subseteq E$ be a set of hyperedges between Q and Y:*

$$h \in E_{Q\to Y} \;\Rightarrow\; \mathit{tail}(h) \subseteq Q \;\wedge\; \mathit{head}(h) \subseteq Y$$

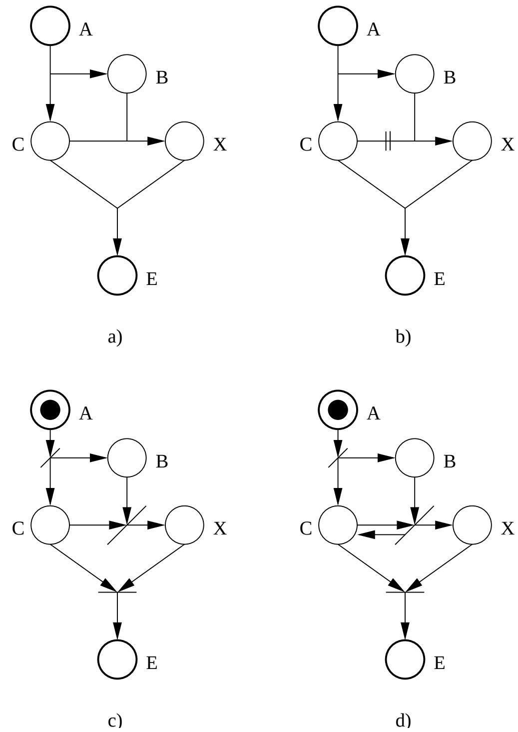

Fig. 5. Two small hypergraphs and their Petri Net equivalents. Graph a) does not represent a realisable path—in order to reach X a traversal must pass through C, but having done so, since C is also required for E, E cannot be reached. This is reflected in the Petri Net equivalent, c), where no sequence of firings will result in a token in the E place. In graph b), C is connected to X using a context relation; this does not "consume" C and so the path is realisable; it's Petri Net equivalent, d), can clearly have transitions fired to put a token in E.

There is a flow (X, Y) *representing a realisable traversal from* X *to* Y *if either* $X = Y$*, or if there exists a flow* (X, Q) *and a non-empty* $E_{Q \to Y} \subseteq E$ *such that:*

1. $\forall q \in (Q/Y),\ \exists h \in E_{Q \to Y}.\ q \in \mathit{tail}(h)$*, and*
   $\forall y \in Y,\ y \in Q\ \vee\ \exists h \in E_{Q \to Y}.\ y \in \mathit{head}(h)$
2. *For each* $q \in Q$*, let* $E_q = \{h \in E_{Q \to Y}.q \in \mathit{tail}(h)\}$.
   Let $E_{q/C} = \{h \in E_q.\ \exists (h, \{q\} \cup N) \in C\}$*. Then* $|E_q| - |E_{q/C}| \leq 1$*, and if* $q \in Y$ *then* $|E_q| - |E_{q/C}| = 0$.
3. *For each* $h_1, h_2 \in E_{Q \to Y},\ \mathit{head}(h_1) \cap \mathit{head}(h_2) = \phi$*, and for each* $h \in E_{Q \to Y},\ \mathit{head}(h) \cap Q = \phi$.

Note that given a pair of node sets (X, Y)*, there may be many possible flows from* X *to* Y*. In such cases it is convenient to describe a flow by* unfolding *it as sequences of node sets,* $(X = X_0, X_1, \ldots, X_n = Y)$ *for some finite or infinite* n.

Remarks: *Flows* are defined recursively, between two sets of nodes. Condition 1 of definition 8 ensures that a flow progresses by following hyperedges. Condition 2 shows the underlying Petri Net semantics: a particular node can be "used" at most once to progress to the next node set, and all other uses must be defined as *context* uses. If a particular node actually remains in a node set, then all uses must be *context* uses. Condition 3 guarantees 1-Safety.

Despite the apparent complexity of its specification, this is a relatively simple condition to recognize (or compute). In figure 5 b), for instance, the node sets involved in a flow between A and E would {A}, {B,C}, {C,X}, {E}. Sequences of node sets forming flows are analogous to sequences of Petri Net markings forming reachable states.

It will also be useful to have a notion of the "distance" between two node sets forming two ends of a flow. This notion is dependent on avoiding cycles, and hence is developed through the following definitions:

Definition 9. *A flow* (X, Y) *unfolded as* $(X = X_0, X_1, \ldots, X_n = Y),\ n \geq 0$ *is* simple *if it does not contain any cycles:* $\forall i \neq j,\ X_i \neq X_j$. *If each set of hyperedges between node sets consists of exactly one hyperedge:* $|E_{X_i \rightarrow X_{i+1}}| = 1$, *then the flow is* sequential.

Definition 10. *Let* $F = (X = X_0, X_1, \ldots, X_n = Y),\ n \geq 0$ *be a finite, simple flow. Then* F *has* length n, *expressed* $|F| = n$.

Remarks: Note that not all length 1 flows may be performed in a game as atomic or single actions—in our definition a length 1 flow may be comprised of concurrently following many independent hyperedges. *Sequential* length 1 flows, however, do indeed always follow one hyperedge.

Length is defined by the particular flow between two node sets. Since there may be many ways of reaching Y from X, it is not by itself sufficient to describe the "distance" between these node sets. The following definitions supply terminology for describing both minimal and maximal distances.

Definition 11. *Given a pair of node sets,* (X, Y) *and an NFG,* N, *the* distance *from* X *to* Y, *expressed* $D_N(X, Y)$, *is the smallest length over all possible simple flows in* N *between* X *and any* Y', *where* $Y \subseteq Y'$. *The* sequential distance, *expressed* $d_N(X, Y)$ *is the smallest sequential length over all possible simple flows between* X *and* Y'.

Definition 12. *Given a pair of node sets,* (X, Y) *and an NFG,* N, *the* separation *between* X *and* Y, *expressed* $s_N(X, Y)$, *is the length of the largest simple flow* $(X = X_0, X_1, \ldots, X_n)$ *such that* $\forall i \leq n,\ X_i \cap Y = \phi$. *Separation is always sequential.*

4 Narrative Properties

NFGs can be used as a design tool to describe narratives; section 5 illustrates such a usage. Narratives, however, must also satisfy semantic properties that are not necessarily trivially apparent in every syntactically-correct NFG. Below some interesting properties and associated analysis/verification strategies are discussed.

4.1 Pointlessness

In some narrative games there is the possibility of the game persisting after the player has performed actions that make the game as a whole unwinnable. For example, in the adventure game *The Hitchhiker's Guide to the Galaxy,* "...if you didn't pick up the junk mail at the very beginning of the game, it was unwinnable at the very end." [Ada98]. Since these games usually take considerable time to play through (often on a scale of days to weeks) such problems can be particularly vexing for players.

Ensuring a narrative reaches LOSE quickly if it cannot reach WIN is a desireable narrative property. This can be viewed as a form of reachability or path-length problem in the NFG formalism; below it is expressed in terms of separation.

Definition 13. *An NFG* $N = (H, a, w, \ell)$ *is* p-pointless *if for all flows* $(\{a\}, F)$*, either there exists a flow* $(F, \{w\} \cup Z)$ *for some* Z*, or* $s_N(F, \{\ell\}) \leq p$

REMARKS: The parameter p defines how "quickly" one must reach the LOSE state if the game is not winnable. A small value of p ensures a quick termination of a failed game.

Verifying this property in general involves determining reachability, a well-considered problem in Petri Nets [Esp98]. There are a variety of efficient solutions available; e.g. [PCP99].

4.2 Narrative Progress

It is possible to build a simple semantics for a DAG-based model by describing narrative progression as following a partial order on DAG subgraphs. A subgraph ordering, however, is too coarse for our purposes; first, it does not allow us to distinguish between progression towards the *win* as opposed to the *lose* nodes (and vice versa); second, two identical subgraphs, equated in a subgraph ordering, may not be at all equivalent with respect to reaching a goal due to the flow mechanics.

Instead, we base our semantic interpretation of movement through a narrative on the state space of the NFGs and flows.

Definition 14. *Given an NFG,* $N = ((V, E, C, L, L_v, L_e), a, w, \ell)$*, and two node sets* $X, Y \subseteq V$*, we define* $(N, X) \leq_w (N, Y)$ *iff* $d_N(X, \{w\}) \leq d_N(Y, \{w\})$*.*

REMARKS: This is trivially a partial order: reflexive, transitive, and antisymmetric. Distance is defined as the length of the shortest flow from a given node set to any superset of the target set. For NFGs, reaching the win state (w) is most important, and so our partial order is based on the distance to the w

node. Of course other orderings based on reaching other states are possible; e.g., a lose ordering: "$\leq_\ell$".

A partial ordering allows logical positions within a narrative to be compared: is player 1 "ahead of" player 2? Particular "actions" (flows of length 1) can also be categorized: actions forming increasing functions in this domain are known to increase the proximity to narrative conclusion.

5 An Extended Example

The examples presented so far are of a simple nature. In this section, the initial portion of an old, but nevertheless paradigmatic narrative-based computer game is described: "The Count," one of the well-known Scott Adams adventure games [Ada]. Note that although resource constraints prevent discussion of the entire game, these initial scenes illustrate non-trivial narrative requirements.

5.1 Game Introduction

This game, and indeed all the Scott Adams games, function in discrete time steps furthered by user interaction. At each time step the player is presented with a command line, which allows the player to (potentially) perform a narrative movement. Like many computer narratives, *The Count* is a first-person narrative, where the player is conceptually one of the game characters, and hence is modelled as one of the game objects.

Narrative Flow Graph Model Since there are many subtasks and many potential player actions, the actual NFG is both large and non-planar. We do not address graph drawing concerns in this paper, and so use an *ad hoc* graph structuring to organize and present the model. Also note that the particular structure we present is only one system for doing so—our presentation was developed experimentally, and we do not claim it is canonical in any way. Equivalence of different models is a non-trivial topic reserved for future work.

At least one node is defined for each object. Important objects in this example include you (the player), a sheet, and the end of the sheet (logically separate from the sheet); these are abbreviated in diagrams as **Y, S,** and **E** respectively. There are of course many other objects in the game; the majority, though are not important to the initial scenes, and so are not modelled here.

The state space of an object often includes its location within the game. Thus each object is represented by several mutually-exclusive states indicating its logical location in the game (which *room* it resides in). Relevant "rooms" in this example include the bed (starting location), the bedroom, the window ledge and the hall, as well as the player inventory. These are abbreviated **B, Br, L, H,** and **I**. Further state divisions of objects will be introduced as required. Also note that not all object-room combinations are allowable states (eg, **Y-I** is nonsensical).

Application of location states for objects can be seen in the effects of the user *take* and *drop* actions. These require the player to be in the same location or for the object to be in the inventory; example NFG fragments are shown in figure 6. There will be such a substructure required for each place from or to which each object can be moved.

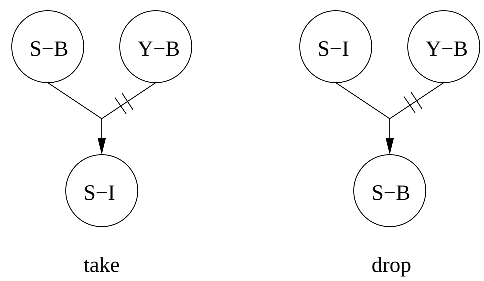

Fig. 6. NFG structure for representing effect of *take* and *drop* actions. On the left, while the player and the sheet are in the bed it is possible to move the sheet into the inventory. On the right is the symmetric situation—the sheet can be dropped into the bed provided it is in the inventory, and the player is also in the bed. Notice the context connection to **Y-B** in each case—taking or dropping an item does not affect the players location.

5.2 Game Tasks

Tasks within the game are basically organized according to location. Thus, as in the *take* and *drop* examples, the **Y-B, Y-Br, Y-H,** and **Y-L** nodes will frequently be required context connections when representing state transformations. In the sections below, the tasks and state space available in two rooms are described.

The player begins lying in bed. Not surprisingly, the bed also contains the sheet (the complete set of connections to the axiom would also include the intial states of all objects, in all rooms). A portion of the initial game situation is illustrated in figure 7.

The Bed The bedroom does not contain complex tasks. The only possibilities are to take or put objects here, and for the player to get up. One further possibility is to sleep. Sleeping three times terminates the game as a loss (player turns into a vampire). Note that sleeping in any room includes an immediate movement back to the bed; this can be easily modelled, but is not shown in figure 8.

The Bedroom The bedroom is by far the more complex of the rooms in this example. As well as the state of objects already mentioned, the room itself

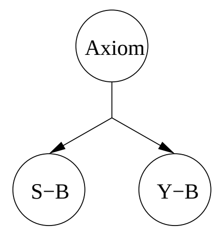

Fig. 7. Initial configuration of the extended example.

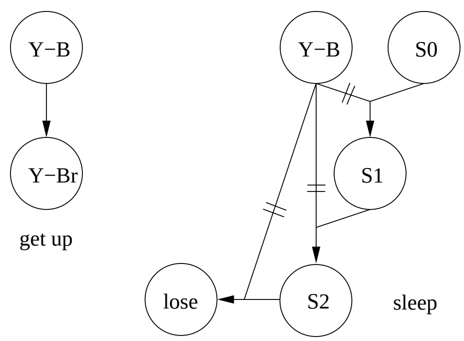

Fig. 8. Bed fragment. Here the player can "get up" and move out of the bed into the bedroom (left fragment), or they can sleep (right fragment). Having slept 3 times, they lose; in order to track this, 3 different "having slept" states are required (**S0, S1, S2**). This could also have been modelled by separate sleep actions, as opposed to sleep state nodes.

contains a window, which can be in open (**Wo**) or closed (**Wc**) state (initially closed). Figure 9 shows an NFG fragment for opening/closing the window and moving out to the ledge through the open window.

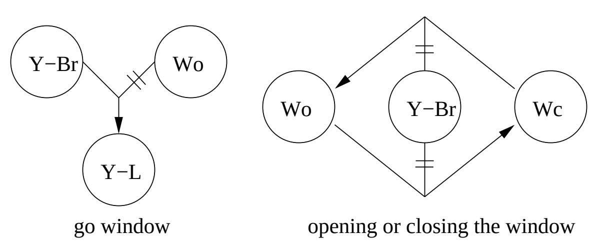

Fig. 9. Bedroom fragment. On the left is a structure allowing moving from the bedroom to the ledge, provided the window is open. On the right the window can be opened or closed.

A more complex state structure is in the way the sheet and the bed interact. It is possible to tie the sheet to the bed; this transforms the bed (**Be**) into one with a sheet tied to it (**Bs**), and creates the end-of-the-sheet object (**E**) in place of the sheet. The end can then (later) be used as a makeshift rope when on the ledge, through the open window. Tying the sheet can be done whether the sheet is in the bedroom itself, or in the player's inventory (a multi-room context)—see figure 10 .

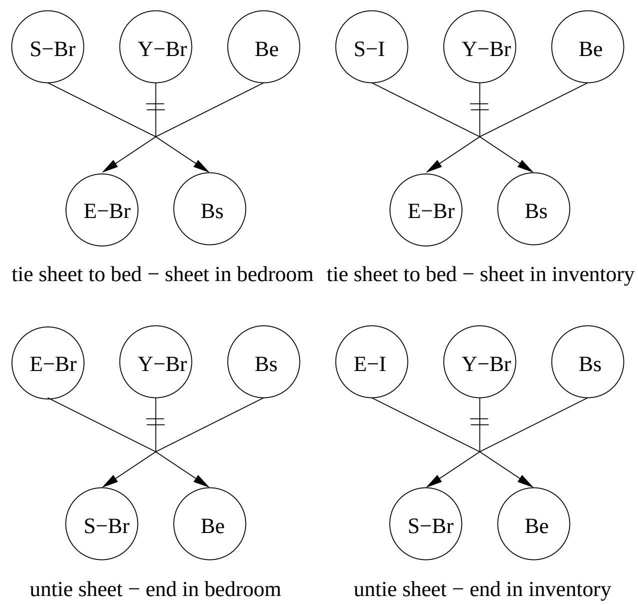

Fig. 10. Tying (top) and untying (bottom) the sheet. Tying can only be done in the bedroom, provided the sheet is in the inventory or bedroom itself. Untying is more complex and not all situations are shown; the sheet can be untied only when *you* are in the bedroom (**Y-Br** as context), but the end (**E**) can be in the bedroom, bed, ledge or inventory. This can be handled by further structures like the bottom two.

Further complexity is evident in how the tied sheet acts if certain other actions are taken. If the player takes the end of the sheet and ventures into the hall the sheet becomes "untied," restoring the sheet into the players inventory. This situation is shown in figure 11. As well, if the player sleeps, the tied sheet resets in a similar fashion (though the sheet appears in the bed, not the inventory); this is not shown in figure 8.

5.3 Counting Time

Although we have explicitly excluded the modelling of real time, there are several narrative events based on counting. For instance, after 29 moves, a door-bell

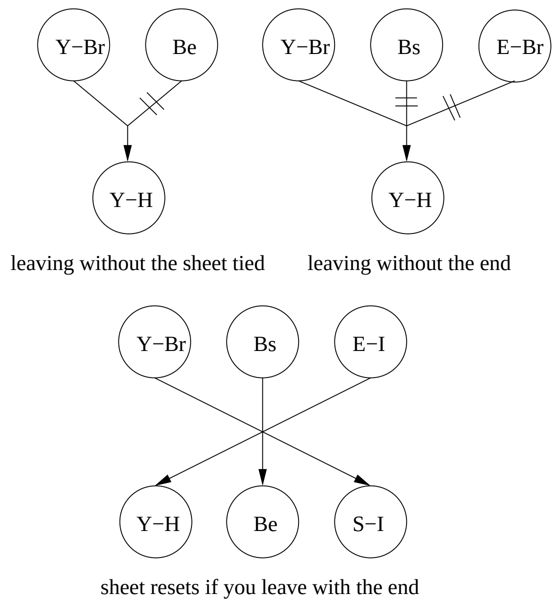

Fig. 11. Leaving the bedroom for the hall. If the sheet is not tied, the movement is straightforward (upper left). If the sheet is tied but the end is not in the inventory, it remains behind (upper right; this requires several similar structures replacing **E-Br** with **E-B** and **E-L**). If the end of the sheet is in the inventory, the sheet becomes untied, and the end is replaced by the original sheet.

rings elsewhere in the castle. After 64 turns the sun sets, and the player finds themselves back in bed the next morning (as if they'd slept). Static counts such as these can be explicitly modelled in our formalism by the appropriate number of state nodes, as we show for the sleep count (figure 8), or less-explicitly as a non-deterministic connection between normal behaviour and the post-count behaviour.

5.4 Pointlessness

The game fragment presented does not have a conclusive narrative goal. By adding winning and losing conditions, it becomes possible to discuss how p-pointless the narrative is. For example, figure 12 defines a win if the edge of the sheet is left on the ledge, and a lose if you go to the ledge without the sheet tied at all.

With this configuration, the aggregate NFG is 1-pointless—it is always possible to reach Win, unless in a condition where losing is the only possible action. The actual game would have a p-pointless value larger than 1, since sleep (also leading to Lose) is forced at several points, potentially leaving insufficient moves available to reach Win.

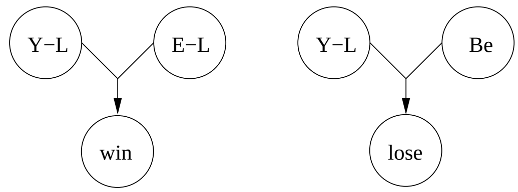

Fig. 12. Artificial winning and losing conditions. Dropping the sheet end on the ledge is a win; being there without having tied it at all is a loss. Other graph fragments allow for movement from the ledge back to the bedroom if neither condition is true.

6 Future Work

A number of issues remain to be addressed. Our formalism is an attempt to provide an initial infrastructure for narrative modelling and analysis; further analysis of complex computer narratives will expand and tune this model.

Further investigation of other formalisms, including variant forms of Petri Net (e.g., coloured and/or timed Petri Nets) is warranted. 1-safe Petri Nets can easily express the simple precedence relations (including cycles and choice) that form basic computer narratives; other formalisms, however, may allow for the expression of concepts more efficiently (e.g., counting), or which cannot be expressed at all in our formalism (e.g., real-time counters). The increase in complexity of such representations would have to be balanced against other factors, including readability and representative modularity/cohesion—any expansions of the model should be careful to avoid encompassing more than just the game narrative.

The NFG fragments illustrated in section 5 comprise a very small portion of the game. While these initial scenes are as complex as any subsequent scene, it is clearly more compelling to demonstrate the ability to represent and analyze a complete narrative.

The examination or use of larger examples raises other issues as well. An entire game would result in a very large, complex graph, and actively using such a graph for narrative development would be awkward for a human being. A higher-level environment automating many of the repetitive aspects of the formalism (eg implicitly modelling take/drop actions etc) is desireable for practical use.

An aspect we are actively working on is the overall semantics induced by narratives. In section 4 some interesting narrative properties were discussed, and a partial order semantics introduced. We are developing these ideas further, investigating how various game actions and potential playability problems or goals may be understood within a semantic framework.

7 Conclusions

Traditional, text narratives can be simply represented through a straightforward DAG structure. Narratives in computer games, however, require a more complex presentation system, including the ability to represent both cycles and narrative choice, two things that are not possible with a DAG format.

In this paper we presented an alternative representation framework, Narrative Flow Graphs, derived from a limited form of Petri Net. This framework addresses the existing representation gap, while providing a syntactically simple narrative description format. Using this format, we defined some simple properties pertinent to narrative development, and demonstrated the application of our formalism to a realistic example.

References

[Ada] Scott Adams. Scott Adams grand adventures (S.A.G.A). Web site: `http://www.msadams.com`.

[Ada98] Ernest Adams. The designer's notebook: "Bad game designer, no twinkie!". Online article from Gamasutra, March 1998. `http://www.gamasutra.com`.

[AIN92] Giorgio Ausiello, Giuseppe F. Italiano, and Umberto Nanni. Optimal traversal of directed hypergraphs. Technical Report TR-92-073, International Computer Science Institute, University of California at Berkeley, Berkeley, CA, 1992.

[BBL00] Jennifer Burg, Anne Boyle, and Sheau-Dong Lang. Using constraint logic programming to analyze the chronology in "A rose for emily". *Computers and the Humanities*, 34(4):377–392, December 2000.

[BLM+94] David Baggett, Felix Lee, Paul Munn, Greg Ewing, and Jason Noble. Plot in interactive works (was re: Attitudes to playing (longish)). discussion thread in rec.arts.int-fiction archives, October 1994.

[BPA+95] David Baggett, Andrew C. Plotkin, Julian Arnold, Dan Shiovitz, and Mark Clements. Plot DAGs revisited (was re: Game design in general.). discussion thread in rec.arts.int-fiction archives, September 1995.

[Bro96] Kevin M. Brooks. Do story agents use rocking chairs? The theory and implementation of one model for computational narrative. In *Proceedings of the fourth ACM International Conference on Multimedia*, pages 317–328, Boston, Massachusetts, 1996.

[Bro97] Kevin Brooks. Programming narrative. In *Proceedings of the 1997 IEEE Symposium on Visual Languages*, pages 380–386, 1997.

[CEP95] Allan Cheng, Javier Esparza, and Jens Palsberg. Complexity results for 1-safe nets. *Theoretical Computer Science*, 147(1–2):117–136, 1995.

[Esp98] J. Esparza. Decidability and complexity of petri net problems—an introduction. In *Lectures on Petri Nets I: Basic Models*. Springer-Verlag, 1998.

[For97] C. E. Forman. Game design at the drawing board. *XYZZY News*, (4), 1997.

[Mat97] Michael Mateas. An Oz-centric review of interactive drama and believable agents. Technical Report CMU-CS-97-156, School of Computer Science, Carnegie Mellon University, 1997.

[OE94] Magnus Olsson and Greg Ewing. Plot DAGs "undo", and finite automata (was: notes on "Annoy"). discussion thread in rec.arts.int-fiction archives, November 1994.

[PCP99] Enric Pastor, Jordi Cortadella, and Marco A. Pena. Structural methods to improve the symbolic analysis of petri nets. In *Proceedings of the 20th International Conference on Application and Theory of Petri Nets*, pages 26–45, 1999.

[Rei88] Wolfgang Reisig. *Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets*. Spring-Verlag, 1988.

[Rob87] Michael J. Roberts. TADS the text adventure development system. software, see `http://tads.org`, 1987.

[Str97] Carol Strohecker. A case study in interactive narrative design. In *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, Amsterdam, The Netherlands, 1997.

[vdA98] W. M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.