Early Cases of Bertillon, the Logic Programming Sleuth

Simone D.J. Barbosa, Edirlei Soares de Lima, Fabio A.G. da Silva, Antonio L. Furtado, Bruno Feijó
PUC-Rio – Departamento de Informática
Rio de Janeiro, RJ – Brazil
{simone, elima, faraujo, furtado, bfeijo}@inf.puc-rio.br

***Abstract*— We introduce in the present paper an operationally defined subclass within the genre of detective stories, specified on the basis of the logic programming model adopted in our Logtell project. Special attention is given to the treatment of communicative events. An SWI-Prolog plan-based tool was developed to compose consistent plots, conforming to the conventions of the genre. Seven criminal cases generated by the tool are described as illustration. It is shown, using the PlotBoard interface, how to run plan-based composition in interactive stepwise mode. We have also developed a storytelling system capable of representing the stories in the format of interactive comic books on tablet computers.**

Keywords*— Plot Composition; Communicative Speech Acts; Detective Stories; Logic Programming; Plan Generation; Plan Recognition; Comic Books.

I. Introduction

The basic objective of this paper is to propose a strategy to define a subclass of the genre of detective stories, on the basis of a logic-programming model introduced in [1] and more rigorously formulated in [2]. The definition specifies:

1. what can exist in a state of the underlying mini-world
2. how states can be changed, and
3. the factors driving the characters to act.

Informally speaking, such threefold specifications determine what kind of facts can hold in each state, what events (denoted by operations defined in terms of pre-conditions and post-conditions) can change a state, and what goals can motivate the acting characters to trigger events that compose consistent detective story plots. In turn, the composition process can be effectuated by a plan-generator algorithm.

What plots will be generated depends not only on how the genre was specified, but also on the specification of a suitable initial state, in which the entity instances (including characters and objects) that will figure in the plots are introduced, together with their initial properties (attribute-values and relationship connections with other entity instances). To accommodate characters with distinct personality traits, and to deal with a diversity of peculiar or even anomalous situations, we started to allow the generic pre-conditions and post-conditions of the event-producing operations to be complemented by what we call conditioner clauses, as one additional part of the initial state specifications. Choosing different initial states is thus a way to achieve a varied repertoire within a given genre – which is further enlarged by both (i) the ability of the plan-generator to backtrack to produce different plans to reach the same goals and (ii) the user-interaction features that can be built into the algorithm.

The reputable literary scholar, Todorov, has convincingly argued [3] that detective stories actually contain two stories: the story of the crime and the story of the investigation. According to him, the characters of the story of the investigation do not physically act very much, they mainly learn. In contrast, it so happened that in the Swords-and-Dragons genre, the first to be treated in our **Logtell** interactive plot-composition project [1], the plots were dominated by physical act scenes, such as the kidnapping of a princess amid violent combats between a villainous dragon and the brave knights who come to rescue her. Communicative acts, indispensable to extend our methods to other more subtle genres, were clearly missing. Accordingly, a second objective of the present paper is to provide an adequate set of operations for accomplishing such acts, thus advancing a study whose preliminary results were reported in [4].

The imaginative (and imaginary) protagonist of our detective stories, Bertillon, figures briefly in Sir Arthur Conan Doyle's *The Hound of the Baskervilles*[1], in probable allusion to a historic pioneer in Anthropometry:[2]

Dr. Mortimer: Recognizing, as I do, that you are the second highest expert in Europe ———"
Holmes: "Indeed, sir! May I inquire who has the honour to be the first?"
Dr. Mortimer: "To the man of precisely scientific mind the work of Monsieur Bertillon must always appeal strongly."

The remaining of the text is organized as follows. Section 2 describes our recently-developed non-physical event-producing operations. Section 3 discusses the specification of the detective stories genre and reports Bertillon‘s early feats. Two friendly user interfaces, to interactively produce and display the plots, are shown in Section 4. Section 5 has the conclusions. A fuller description of our logic-programming model, which adopts the Entity-Relationship model [5] to specify facts, the STRIPS method [6] to define events, and situation-objective

[1] cf. http://www.gutenberg.org/files/3070/3070-h/3070-h.htm
[2] http://en.wikipedia.org/wiki/Alphonse_Bertillon

rules to motivate the characters' behaviour, as well as details about the system's implementation, can be found in a previous technical report.[3]

II. Events involving non-physical acts

A. Information-gathering events

The *information-gathering* events [4] enable the various characters to mentally apprehend the state of the world. Without such events, one would have to assume that the characters are omniscient. Here we shall recognize a sharp distinction between the facts themselves and the sets of *beliefs* [7][8] of each character about the facts that hold at the current state of the world, which constitute, so to speak, their respective *internal states*. Beliefs can be right or wrong, depending on their corresponding or not to the facts. Moreover, we have taken the option that acquiring a belief does not cancel a previous belief. As a consequence, we allow a character to simultaneously entertain more than one belief with respect to the same fact, possibly with a different degree of confidence which depends on the provenance of the beliefs. We shall consider three types of information-gathering events, each type associated with a set of operations:

Communication events - operations: `ask, tell, agree, ask_event, tell_event, agree_event`

Perception events - operations: `sense, watch`

Reasoning events - operations: `infer, suppose`.

Operations `sense`, `ask`, `tell`, `agree`, `infer`, and `suppose` refer to beliefs on facts, whereas `watch`, `ask_event`, `tell_event`, and `agree_event` refer to some action event witnessed by a character. The operations are defined in terms of their pre-conditions and post-conditions [6]. The pre-conditions are logical expressions involving affirmed or negated facts and beliefs, whereas post-conditions denote the effect of the operation in terms of beliefs that are added or deleted to/from the current internal states of the characters involved. However, the specification of the operations is deliberately kept at a minimum, to be complemented by separate *conditioners* that express the peculiarities of the characters participating in the stories.

Within computer science, communication between characters immediately brings to mind the communication processes executed by software agents in multi-agent systems. In particular, the Agent Communication Language (ACL) consists of operations similarly defined by their pre- and post-conditions [9]; for an earlier more formal treatment, see for instance [10]. Software agents differ from fictional characters (and, ironically, from human beings in general) in that they are supposed to only transmit information in which they believe, to agents that still lack such information and need it in order to play their role in the execution of some practical service.

In contrast, certain characters are prone to lie, either for their benefit or even out of habit. In general they may ignore the conversational maxims prescribed by philosophers of language, such as [11]. The specification of our `tell(A,B,F)` operation does not even require that `A` has any notion of the fact `F` to be transmitted to `B`. It is enough that both characters are at the same location `L`; if they are not, a `current_place(A,L)` sub-goal is recursively activated, which may cause the displacement of the teller (character `A`) to `L`, where `B` currently is. And the only necessary effect of the operation is that `F` has been `told` by `A` to `B`. Whether or not `B` will believe in `F` will depend on the execution of the `agree` operation, which in turn depends on whether or not `B trusts A`. Another purpose served by `tell(A,B,F)` is to convey merely expressive speech acts [12]; the `F` parameter can then be any arbitrary sentence. The `B` parameter may remain unspecified, in which case `A` is addressing a general audience.

The `ask` operation is similarly defined, and its effect is just that `A` has `asked F` from `B`, who may respond or not. The fundamental character-dependent conditioners are established, respectively, by separate `will_tell` and `will_ask` conditioners.

Perception is the faculty whereby people keep contact with the world through their five senses (sight, hearing, touch, smell, and taste). At the present stage of our work we do not make such distinctions, and merely consider a generic `sense(C,F)` operation to apprehend any sort of fact `F` specified in the static schema as `perceptible`, with a variant version that makes provision for defective sensing. For correct sensing of a positive or negative fact `F`, `F` must be successfully tested. Distorted sensing (for instance, of certain colours by a daltonic subject) is accompanied by a side-remark on the true fact. In any case, besides the `sensed` clause, a `belief` clause is immediately added, since direct perception does not depend on a third party.

The `watch(C,E)` operation allows a character `C` to witness an event `E`, denoted as always in our system by some operation defined in the dynamic schema. Operations `ask_event`, `tell_event`, and `agree_event` allow another character `C'` to question `C` about `E`, that `C` reports the event, and that `C'` effectively agrees with its occurrence. As before, the definitions of these operations are completed by conditioners, respectively `sense_rule` and `watch_rule` clauses. For `sense`, it is required that, to ascertain a positive or negative fact `F` involving a person or object currently at place `L`, a character `C` must be at `L`, either originally or as the result of pursuing `current_place(C,L)` as a sub-goal. For `watch`, the `current_place` requirements depend on the type of event being watched, which justifies their being left to the special `watch_rule` clauses. For instance, the operation `go(A,L1,L2)` (required to update `current_place(C,L)` facts), can be watched partly by persons at `L1` (origin) and at `L2` (destination).

Both the agent of an event `E` and a character who watches the occurrence of `E` are, as expected, aware of the main effects of the event. Anticipating what we shall treat in our detective stories environment, if `A` kills `B`, or if `C` watches `A` killing `B`, then characters `A` and `C` will believe the facts `killed(A,B)` and `dead(B,true)`. And when the agent or a witness uses `tell_event` to inform another character `C'` and `C'` reacts with an `agree_event`, `C'` will also start believing in the facts caused by the reported event.

[3] ftp://ftp.inf.puc-rio.br/pub/docs/techreports/12_08_barbosa.pdf

Deduction, induction and abduction are complementary reasoning strategies. For deduction, if there is a rule $A \rightarrow B$ and the antecedent A is known to hold, it is legitimate to *infer* that the consequent B holds. In the case of induction, the systematic occurrence of B whenever A occurs may justify the adoption of rule $A \rightarrow B$. Abduction (cf. [13]) is a non-guaranteed but nevertheless useful resource in many uncertain situations: given the rule $A \rightarrow B$, and knowing that B holds, one may *suppose* that A also holds. This is a type of reasoning habitually performed by medical doctors to diagnose an illness from observed symptoms. The trouble is, of course, that it is often the case that more than one illness may provoke the same symptom, *i.e.*, there may exist other applicable rules $A_1 \rightarrow B$, $A_2 \rightarrow B$, ..., $A_n \rightarrow B$, suggesting different justifications for B. Thus in abduction, wherein the implication arrow is followed backward, one is led to formulate hypotheses rather than the firm conclusions issuing from deduction over deterministic rules.

Our `infer` and `suppose` operations utilize, respectively, deduction and abduction. Their conditioners can be the same rules of inference (`inf_rules`) to be traversed forward in the former case or backward in the latter. In our implementation of the `infer` operation, given a rule `P=>F` accepted by character `A`, the antecedent `P` furnishes the beliefs to be tested as pre-condition, whereas `A`'s belief in `F` will be acquired as the `added` effect (another addition being an `inferred` clause) upon a successful evaluation of `P`. In contrast, in the case of the `suppose` operation, the belief in the consequent will just motivate the addition of a `supposed` clause in a fact present in the logical expression of the antecedent. We must stress that the inference rules adopted by the characters in our story context do not pretend to be scientifically correct. Often originating from popular traditions, they may lead to far-fetched or absurd beliefs.

B. Directive and commissive events

In order to achieve a desired goal, a character `C` may need to resort to another character `C'` to perform an action that `C` is unable or unwilling to execute personally. Three operations were supplied to meet this requirement: `request`, `comply`, and `refuse`. Linguists [12] classify such communicative acts in the directive (the first one) and in the commissive categories (the two last ones).

As before, the specification of pre-conditions and post-conditions is complemented by conditioner clauses, named respectively `will_request`, `will_comply` and `will_refuse`. It is normal to specify that `C'` always complies to the requests of `C` if an `obeys(C',C)` relationship has been declared to bind them. On the other hand, compliance may be subjected to a reciprocal request: `C'` would, so to speak, negotiate with `C`, imposing a task as payment or compensation for the service to be rendered to `C`. Moreover, even if a character has complied to perform an action, it does not necessarily follow that the promise will be fulfilled, which is in consonance with the existence of characters who shamelessly lie. The `request` operation can be either directed to a specific character or left open, so that we achieve the generality afforded by the `cfp` (call-for-participation) operation of ACL [9].

C. Library-consulting events

As a dual process to plan-generation, plan-recognition is a no less invaluable resource for the composition of story plots. In principle, if we let the plan-generator run for an indefinite amount of time, it should produce all plots consistent with the given specification, even those unworthy of our attention. It is therefore sensible to provide a collection of story patterns extracted from pre-existing narratives of diverse provenance. Such collection, residing in a conveniently indexed library [14][15][16], could then contribute to create new plots by adaptation, or to match a few observed events against typical story patterns, thereby allowing to predict what the agents are trying to accomplish.

Our library items obey the format `user/main-agent/conclusive-message/goals/plan/complementary-test`. For our current purposes, the `plan`, the `test` and the `message` are of special interest. Our library-consulting operations, which are also incorporated in the plots produced by the planner, are named `collect`, `recognize`, and `try`. The first simply assembles in a list the events that have been directly `watched` by or `related` to a character. The `recognize` operation checks whether all events in the list of observations correspond to events in the `plan` component of a library item. If the pattern-matching succeeds, the logical expression in the `test` is passed to the `try` operation, (i) to exclude trivial cases of successful matches and (ii) to trigger additional events, such as recommendations for the detective to gather further information. Such events are then appended, thus becoming part of the generated plot. Finally, if both the pattern-matching and the subsequent test succeed, the `message` is composed and becomes available (in particular to the domain-oriented `expose` operation to be introduced in the next section, whereby the detective communicates his conclusions).

III. The Genre of Detective Stories

A. Related work

There have been a number of story generation systems which deal with plot compositing using a variety of different methods. One of the earliest examples of an automated story generation system is Tale-Spin [17], which demonstrated the effectiveness of using planning algorithms for the generation of coherent characters and plots. Other examples include Universe [18], which added restrictions and authorial goals to the planning algorithms in order to provide more control over the generated stories, and Minstrel [19], which proposed the reuse of parts of a story or an event library in order to generate new plots.

Mystery and, more specifically, detective stories do not constitute an unexplored field in digital entertainment. In fact, plot-driven detective stories (often called "whodunits", i.e. "who done [did] it?") have been extensively used as the narrative environment for many game-oriented systems.

One of the earliest examples of an automated storytelling system is the *Automatic Novel Writer* [20], which was programmed in FORTRAN and was capable of producing fairly long murder mysteries stories. The system relied on a simulation model where the behavior of individual characters and events was defined by probabilistic rules formulated by the authors, which progressively changed the state of the world. The system receives a description of the world in which the story is to take place as input, together with a description of character traits that define the murderer and the victim. The motives arise as a function of the events during the course of the story.

Another early example is the laserdisc game *Murder, Anyone?*,[4] played by two teams, whose goal is to guess the character Derrick Reardon's murderer, the motive and the method [21]. *Tea for Three* is a whodunit (inspired by Infocom's *Deadline* [22]) where the user plays the role of a detective who has to figure out (by seeking physical clues and talking to characters) if an apparent suicide was genuine, or if it was murder (in this case also disclosing who killed the victim, how, and why). It uses an architecture called Moe, designed to, with the use of adversary search, decide how and when to guide the user's experience. The interactive drama is broken down into abstract pieces called user moves, and the system is able to assess any complete sequence of user moves by the evaluation function for its dramatic quality.

A location-based pervasive game prototype was developed by [23], to be used during a car trip, including telephone and walkie-talkie interaction. The user plays the role of a detective (teamed up with a partner) who tries to uncover an organized criminal gang by investigating a series of crimes that seem to be related. Another prototype system developed to explore location-based interactive stories is *Who Killed Hanne Holmgaard?* [24], which is a historical murder mystery set during World War II. Two participants, each playing a different character in the story, should work cooperatively to unravel the mystery and solve the crime, while on the move.

U-DIRECTOR [25] has a different purpose: to create a director agent able to orchestrate in real time the events in a storyworld to improve the user's experience, coping with the uncertainty about the user's intentions and the absence of a complete theory of narrative. *U-DIRECTOR* models narrative objectives, storyworld state, and user state with a dynamic decision network that continually selects actions to maximize narrative utility. This architecture has been implemented in a narrative planner for *Crystal Island* [26], a narrative-centered learning game in which users play the role of a medical detective solving a science mystery. Further research on the *Crystal Island* interactive storyworld has adopted (and tested) some other approaches, such as the use of supervised machine learning to recognize players' affective states or the use of dynamic Bayesian networks to model their knowledge [27].

The interactive storytelling game engine *NOLIST* [28] utilizes Bayesian networks to determine the culprit of a murder mystery. The Bayesian network changes in response to user actions and observations, so that the engine creates a dynamic storyline attuned to player actions and choices. It utilizes the user's moves and logical inference to determine details of the story (which is not entirely preset), including the identity of the murderer. For example, if the user finds a body and a gun lying beside the body, then the probability that the victim was shot with the gun increases. NOLIST recreates the past as a reaction to player interaction; neither the plot nor the culprit are known by the game engine in the beginning but are determined in the course of the game.

[4] *Murder, Anyone?* (1982). Cincinnati: Vidmax.

Another example of the use of Bayesian networks to help create a mystery narrative is the murder mystery game proposed in [29]; at each game run, a new narrative plot and set of characters are generated. The initial plot is created with the *Dynamic Plot Generating Engine* (DPGE), which creates new mystery plots on demand using a Bayesian network and Proppian functions [30]. This use of Bayesian networks to form a murder mystery plot resembles that of NOLIST, but NOLIST does not fix the mystery plot from the start, but rather develops it continuously through game play. DPGE, on the other hand, fixes the initial plot at the start and then expects characters in the game to use it as background for their future actions.

Fabulator [31] is an interactive storytelling prototype based on the "riddle" master plot [32], which comprises stories wherein a mystery must be solved, under the typical guise of whodunits. This prototype uses a storyworld called *Ugh's Story 2*, telling the story of cavemen whose worshiped statue was stolen. The user plays the role of a detective caveman, whose mission is to discover who committed the theft. This work uses a tension arcs model that assumes that the tension rises when the player acquires more knowledge leading towards the truth. The system also uses non-player characters (NPCs), who can help the player character or not, to dynamically adjust the level of difficulty to the desired level.

As part of this brief related work survey, we should also include a number of stories written by well-known authors (e.g. [33][34][35][36][37]), whose protagonists are talented detectives employing different strategies to solve their criminal cases.

B. *Events in our detective stories*

In our subclass of the detective stories genre, two events correspond to crimes: `kill` and `steal`. Event `attack` is also an aggressive action, whose agent can in principle be any of the characters. For the detective's provisional or final conclusions an `expose` event is provided. The specification of operation `kill`, the major focus of all investigations reported in section 3.4, is shown below. The pre-condition combines some `motivation` clause with the general requirements that the victim should not already be dead and that the killer must be at the same location as the victim. Each fact `F` mentioned in the `Circ` parameter (the circumstances that motivate the crime) of the `motivation` clause is converted into `believes(X,F)`, where `X` is the would-be criminal.

```
operation(kill(X,Y,M)).
added(killed(X,Y),kill(X,Y,M)).
added(motive(X,[kill(X,Y),M]),kill(X,Y,M)).
added(dead(Y,true),kill(X,Y,M)).
precond(kill(X,Y,M),P) :-
    motivation(X,[kill(X,Y,M),Circ]),
    prep_mot(X,Circ,Circ1),
    appc((current_place(X,L),
    /current_place(Y,L),
    /(not dead(Y,true))), Circ1,P).
```

A few typical motivation clauses follow:

```
motivation(A,[kill(A,B,greed),(owns(B,O),
       not (A = B), carries_object(A,O))]).

motivation(A,[kill(A,B,jealousy),(loves(A,B),
       loves(B,C),gender(A,M),gender(C,M),
       not (A = B), not (A = C))]).

motivation(A,[kill(A,B,request),(obeys(A,C),
       not (B = C),
       complied(A,[C,kill(A,B,request)]))]).

motivation(A,[kill(A,B,vengeance), (loves(A,C),
       not (A = B), not (A = C), killed(B,C),
       not (B = C))]).

motivation(A,[kill(A,B,'self-defense'),
      (attacked(B,A), not (A = B))]) :-
       A = 'Marian'.

motivation(A,[kill(A,A,lovesickness),(loves(A,B),
       not loves(B,A), not (A = B))]).
```

A detective is supposed to not only find the identity of the killer but also the motive of the crime. An economical way to formulate an appropriate inference clause, to be applied by detective `D`, is, in words: "`D` will infer that, if `D` believes that `X` killed `Y` and that the circumstances described in `Circ` currently held, then the motive `M` that guided `X` was that indicated in the motivation clause corresponding to those same circumstances". Two clauses were used to express that, the second clause serving to introduce a bias: we anticipate that, in the case of suicide, our detective will always assume a crisis of lovesickness as explanation for the killer/victim's desperate action:

```
inf_rule(D,(killed(X,Y),Circ) =>
     motive(X,[kill(X,Y),M])) :-
     motivation(X,[kill(X,Y,M),Circ]), not (X == Y).

inf_rule(D,killed(X,X) =>
     motive(X,[kill(X,X),lovesickness])).
```

As said before, detective stories make ample use of all the communicative events of section 2. Among the conditioners that provide the necessary flexibility to the operations, those associated with the directive and commissive events deserve special attention. In the context of detective stories, such events mobilize the relationship between instigators and their accomplices. The following `will_comply` conditioners convey a possible arrangement: a character `C1` who `obeys` to `C2` will always comply with whatever `C2` may request; on the other hand, if the obedient `C1` requests that the dominating `C2` shall kill some person `C3`, `C2` will comply only if `C1` in turn complies to also get involved in the criminal aggression against `C3`, specifically by stealing an object owned by the victim (such tricky negotiation occurs in case 6 of section 3.4):

```
will_comply(C1,C2,Act,obeys(C1,C2)).

will_comply(C2,C1,kill(C2,C3,request),
      (obeys(C1,C2), owns(C3,O),
      complied(C1,[C2, steal(C1,O,C3)]))).
```

C. Enter Bertillon – the context of his miniworld

The *dramatis personae* figuring in our example are:

Bertillon - a private French detective initiating his career in England
Marian - lovely red-haired young lady, still unmarried
Robin - Marian's suitor
Patrick - another suitor of Marian, a notorious lecher
Jane - former actress serving as Patrick's accomplice
Cogsworth - British butler, head of Marian's household

At the initial state, all characters are in London, except Jane, who is in Manchester. Cogsworth, Marian's loyal butler, is destined to participate as Bertillon's main witness. He is ever disposed to testify and to volunteer information, always consistent with what he believes to be true. Marian is equally sincere, but a shade too credulous. She accepts whatever is told by Jane, who happens to be a compulsive liar. Marian often plays the role of the victim. Patrick is tempted to kill her, either impelled by jealousy or because he covets a precious jewel that she imprudently wears attached to a necklace. She must also beware of Jane, obsessively averse to red-haired persons and an obedient accessory to Patrick's machinations. On the bright side, she has nothing to fear from her butler, and enjoys a reciprocal love relationship with Robin.

Cases 1 through 5 are rather straightforward. Mainly relying on Cogsworth's testimony – but also on his (un)fair knowledge of the meta data – Bertillon is able to infer who is the culprit and his or her motivation. Cases 6 and 7 are somewhat more involved, compelling the detective to resort to our library of crime patterns. The library item for case 7 is:

```
lib([ ... (U/A/[A,' deceived ',B,',
    possibly with criminal intent']/(loves(B,C),
    told(A,[B, not loves(C,B)]),
    believes(B, not loves(C,B)),
    told(A,[B, loves(C,D)]),
    believes(B, loves(C,D)),
      killed(B,C),killed(B,B))/
      (start => tell(A,B,not loves(C,B)) =>
      agree(B,A, not loves(C,B)) =>
      kill(B,C,jealousy) =>
      kill(B,B,M))/(/told(A,[B,not loves(C,B)]),
      trusts(U,X), asked(U,[X,loves(C,B)]),
      told(X,[U,loves(C,B)]),
      agreed(U,[X,loves(C,B)]))), ...]).
```

D. His seven cases

For each case, we shall provide a brief synopsis. To save space, we give only for the first case a calling sequence to the planner able to generate the plot, and the natural language textual rendering of the plot. Such texts result from the application of still rather crude templates [38], which we intend to improve at a later phase of the project.

case 1: ***In a fit of passion.*** Patrick kills Marian, unaware of the presence of the butler, who is later in a position to communicate the event to Bertillon, together with all information needed to establish that the murderer's motive was jealousy.

calling sequence:

```
ex1 :- plans((
    motive('Patrick',[kill('Patrick','Marian'),
         jealousy]),
    watched('Cogsworth',kill('Patrick','Marian',M)),
    related('Cogsworth',['Bertillon',
         kill('Patrick','Marian',M)]),
    agreed_op('Bertillon',['Cogsworth',
         kill('Patrick','Marian',M)]),
    agreed('Bertillon',['Cogsworth',
         loves('Patrick','Marian')]),
    agreed('Bertillon',['Cogsworth',
         loves('Marian','Robin')]),
    inferred('Bertillon',
         motive(S,[kill(S,'Marian'),M])),
    exposed('Bertillon',[S,'Marian',jealousy,nil])
    ),Plan),narrate(Plan), nl, nl, !.
```

plot in template-driven natural language: Patrick senses that Marian loves Robin. Patrick kills Marian. Cogsworth watches the event: 'Patrick kills Marian'. Cogsworth relates to Bertillon the event: 'Patrick kills Marian'. Bertillon agrees with Cogsworth about the event. Cogsworth tells Bertillon: "*Patrick loves Marian*". Bertillon agrees with Cogsworth. Cogsworth tells Bertillon: "*Marian loves Robin*". Bertillon agrees with Cogsworth. Bertillon assumes that Patrick, in the event 'Patrick kills Marian', was motivated by jealousy. **Bertillon says: "The suspect is Patrick and the motive is jealousy"**.

case 2: ***All that glitters.*** This time Patrick commits two crimes: murder and theft. The first is once again watched by the butler, who later notices that the culprit carries the object of the theft, thus characterizing greed as the primary motivation.

case 3: ***Murder by proxy.*** Patrick procures Marian's death by ordering Jane to do the killing. The butler watches their conversation and Jane's fatal act. Learning of both scenes from Cogsworth, Bertillon establishes Jane's direct involvement and Patrick's role as instigator, not bothering however to disclose his motive (jealousy).

case 4: ***The reluctant victim*****.** But our victim is not necessarily so helpless! Jane, on her own initiative, repelled as she is at the sight of red-haired Marian, attacks her to be promptly killed in reaction. The butler watches the aggression and the counter-aggression events, which, reported to Bertillon, result in a verdict of self-defense.

case 5: ***Avenging fury.*** Patrick kills Marian moved by jealousy. Two persons watch the murder: her butler and her lover. Cogsworth limits himself to give testimony, but Robin's reaction is more effective: he kills the assassin. To many a body of jurors, one might surmise, vengeance in the aftermath of a heinous murder should seem admissible as extenuating circumstance.

case 6: ***Framed!*** Patrick's loose morals lead him to arm a trap for the submissive Jane. Preparing to kill Marian, pressed as before by his jealous impulses, he takes advantage of Jane's own inclinations. To exterminate the detested red-haired young lady, Jane requests the aid of her superior. As a condition to comply, Patrick requires that she should also perform some aggressive act, while making a profit; namely, she must steal Marian's rich jewel. The butler misses Jane's request to Patrick, but watches Patrick imposing the theft of the jewel as a condition for complying, and also the theft itself. However, he reports to Bertillon only the latter event – which in no way incriminates Patrick. Perceiving that Marian is dead, Bertillon, like so many detectives in popular fiction, builds the hypothesis (an instance of abductive reasoning from the only rule he knows to explain death: `killed(X,Y) => dead(Y,true)`) that someone killed her. And, at a loss for anything else, he proceeds from the only clue available, i.e. the theft reported by Cogsworth, to consult his library of crime patterns. A match occurs with a pattern involving requested theft by one person as cover-up for the more drastic act of the instigator. The library item recommends checking whether such a request occurred, leading Bertillon to question Cogsworth, who responds relating the event that he, at first, had failed to report. Once, for a change, Bertillon has not enough data for a firm inference, but he advances the possibility that Patrick murdered Marian and sought to cast all suspicion upon his accomplice, Jane.

case 7: ***Iago syndrome.*** Though consenting to the whims of Patrick, Jane surpasses him by her far richer imagination. Patrick is not even mentioned in this case, the hardest one until now in Bertillon's career. Jane lies to Marian, convincing her that Robin does not love her. Marian is therefore overcome by lovesickness and kills herself. The ill-intentioned conversation and its fatal outcome are watched by Cogsworth, and this time he at once reports both scenes to Bertillon. The verdict of suicide is inescapable and Bertillon clearly states it. Feeling, however, that Jane's talking to the victim may have further implications, he again resorts to the library. What he finds is the same pattern that took his Belgian colleague, Hercule Poirot, to identify Norton as the wonted "X" in his last case [33]. The located library item contains a recommendation, leading him to check with the well-informed Cogsworth: does Robin love Marian, contrary to what Jane had proclaimed? The butler's affirmative reply is an indication that Jane deliberately induced Marian to commit suicide. Bertillon denounces Jane, despite his conviction that no British court of justice would condemn her.

IV. INTERACTIVE COMPOSITION AND DRAMATIZATION

Hopefully the cases narrated in the previous section are sufficient to give an idea of what can arise from the current specification of detective stories, and hence offer some indication of how useful is the package of communicative speech acts described before. However, generating an entire plot via a single call to the plan-generator algorithm is not satisfactory from a digital entertainment viewpoint. An environment to run plot composition interactively is essential. In the course of our **Logtell** project, we developed several interfaces to allow the user to create plots interacting with the plan-generator, which also permit visualization, displaying the generated plots as frame picture sequences in storyboard style [39], or in comic book format [40], or by employing more expressive animation [1] and video-based [41] techniques.

We shall now describe how interactive composition can be achieved, on ordinary computers and on tablets, with the support of the two simpler forms of dramatization mentioned above.

A. *With PlotBoard*

The flow of control of our **PlotBoard** tool [39] is shown below (Figure 1). The tool's plan-based design, as in the other **Logtell** products, keeps playing a fundamental role, since it incorporates a knowledge of the domain that a casual user may not possess, but now the plan-generation algorithm serves in a secondary helper's capacity. Under the user's command, the tool generates the plot in a stepwise fashion, alternating between the `user mode` and the `planner mode`.

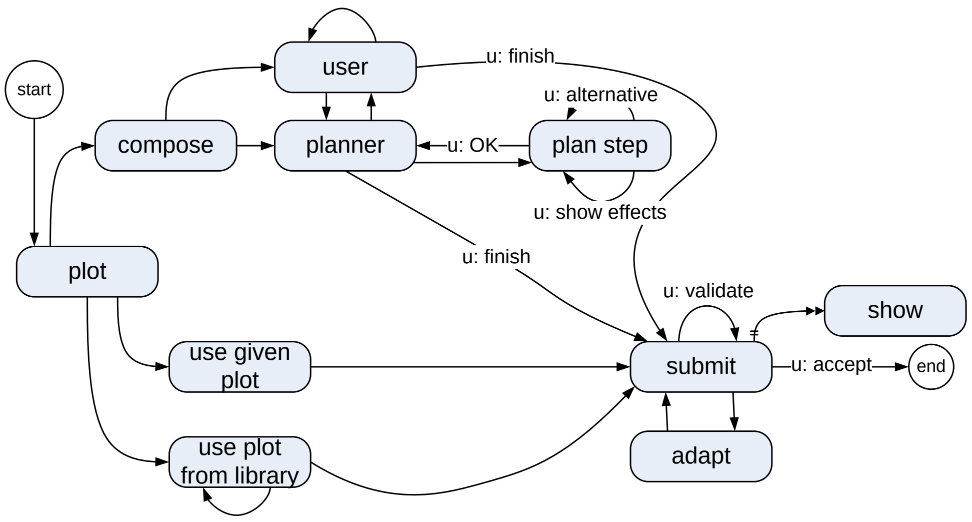


Fig. 1. Flow of control of **PlotBoard**.

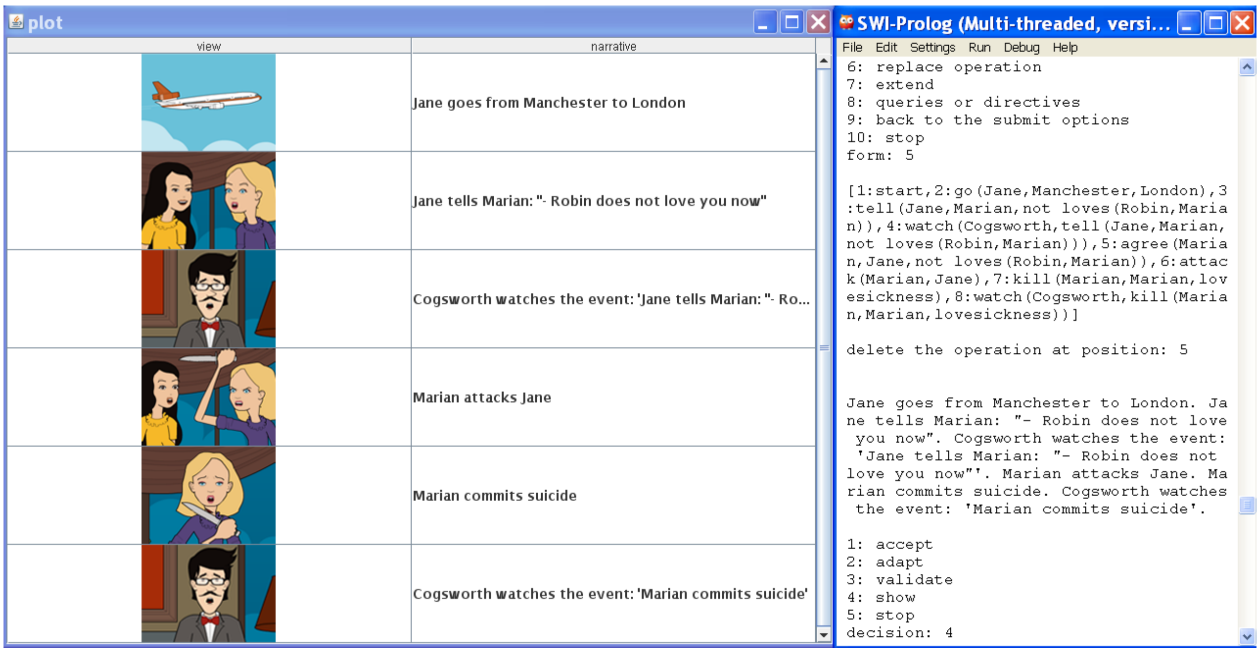
When the `planner mode` is on, the situation-objective rules are activated to find the short-range goals that can, at the current state, be pursued by the planner. Picking one of these goals, the planner then produces an appropriate plan and displays it to the user, who may issue an `ok` or ask for an alternative plan (for the same or for another simultaneously active goal). After the chosen plan step is executed, the user is asked whether the planner may `continue`, in which case the situation-objective rules are again activated for the next step. But the user can prefer to shift to `user mode`, wherein several types of intervention are announced in a menu, such as indicating the goal to be achieved next by the planner or even a specific operation to be executed. And, after signaling `finish`, the user can still decide to perform one or more adaptations of several kinds over the events, such as inserting, deleting, replacing, reordering, summarizing, detailing, etc.

The situation-objective rules adopted in our trial run are listed below, prefixed with numbers for easy reference:

```
(1) sit_obj('Patrick',
   (loves('Patrick',Y),not dead(Y,true), not
  loves(Y,'Patrick'), loves(Y,Z), not ('Patrick' = Z)),
   (motive('Patrick',[kill('Patrick',Y),jealousy]))).

(2) sit_obj('Jane',
   (owns(Y,O),not dead(Y,true),carries_object(Y,O)),
   (sensed('Jane',owns(Y,O)),carries_object('Jane',O),
    motive('Jane',[kill('Jane',Y),greed]))).

(3) sit_obj('Jane',
    (hair_colour(V,red),not dead(V,true),loves(H,V),
    loves(V,H)),(told('Jane',[V,not loves(H,V)]),
    watched('Cogsworth',tell('Jane',V,not loves(H,V))),
    agreed(V,['Jane',not loves(H,V)]))).

(4) sit_obj('Marian',
   (not dead('Marian',true), loves('Marian',H),
  believes('Marian',not loves(H,'Marian'))),
   (loves(M,'Marian'),not (M = H),
  told('Marian',[M,'Let us meet someday!']))).

(5) sit_obj('Marian',
   (not dead('Marian',true), loves('Marian',H),
  believes('Marian',not loves(H,'Marian'))),
   (killed('Marian','Marian'),
    watched('Cogsworth',kill('Marian','Marian',_)),
    believes('Cogsworth',killed('Marian','Marian')))).

(6) sit_obj('Cogsworth',
   (believes('Cogsworth',killed(V,V))),
   (related('Cogsworth',['Bertillon',
    tell(S,V,not loves(H,V))]),agreed_op('Bertillon',
    ['Cogsworth', tell(S,V,not loves(H,V))]),
    related('Cogsworth',['Bertillon',kill(V,V,M)]),
    agreed_op('Bertillon',['Cogsworth',kill(V,V,M)]))).

(7) sit_obj('Bertillon',
   (believes('Bertillon',killed(V,V))),
   (inferred('Bertillon',motive(V,[kill(S,V),M])),
    exposed('Bertillon',[S,S,M,nil]))).

(8) sit_obj('Bertillon',
   (believes('Bertillon',killed(V,V)),
    exposed('Bertillon',[S,S,M,nil])),
   (obs('Bertillon',Obs),recognized('Bertillon',
    'Bertillon'/Ag/Cr_type/Goals/Pl_lib/Q),
    tried('Bertillon',Q),
    exposed('Bertillon',[S,V,lib,Cr_type]))).
```

At the initial state, rules (1), (2) and (3) offer alternative possibilities to the user's choice. If the goals of either (1) or (2) are pursued, Marian is murdered (by Patrick or by Jane, respectively). If (3) is preferred, Jane comes from Manchester and tells a lie to Marian in the butler's presence, coinciding with the first events of case 7, as described in the previous section. The entire plot of case 7 will indeed be generated if one chooses, at each subsequent step, the alternatives offered by rules (5) through (8). But the story does not have to end badly. After applying (3), rules (4) and (5) become active. With (5) Marian yields to depression, but (4) allows her to recover and teach the (allegedly) disloyal Robin a lesson, by giving a chance to his rival, the ill-reputed Patrick.

When we started the trial run shown in figure 2, we took the planner mode and, from the alternative event sequences generated, chose the one produced by rule (3), and then shifted to the `user mode`. In a direct intervention, we added an `attack(Marian,Jane)` event, whereby Marian somehow reacted to Jane's provocation. But, returning next to the `planner mode`, we chose (5) so that the suicide scene ensued, again witnessed by the horrified Cogsworth. At this point, we indicated that the generation phase was finished. Asked by the tool whether the obtained plot should be accepted, we chose instead the `adapt` option, and removed the fifth event (in which Marian expressed her belief in Jane's false assertion). We then selected the `show` option from the menu, thus causing the plot to be displayed via the Prolog/Java interface.


plot
view
narrative
SWI-Prolog (Multi-threaded, versi...
File Edit Settings Run Debug Help

view	narrative
	Jane goes from Manchester to London
	Jane tells Marian: "- Robin does not love you now"
	Cogsworth watches the event: 'Jane tells Marian: "- Ro...
	Marian attacks Jane
	Marian commits suicide
	Cogsworth watches the event: 'Marian commits suicide'

```
6: replace operation
7: extend
8: queries or directives
9: back to the submit options
10: stop
form: 5

[1:start,2:go(Jane,Manchester,London),3
:tell(Jane,Marian,not loves(Robin,Maria
n)),4:watch(Cogsworth,tell(Jane,Marian,
not loves(Robin,Marian))),5:agree(Maria
n,Jane,not loves(Robin,Marian)),6:attac
k(Marian,Jane),7:kill(Marian,Marian,lov
esickness),8:watch(Cogsworth,kill(Maria
n,Marian,lovesickness))]

delete the operation at position: 5


Jane goes from Manchester to London. Ja
ne tells Marian: "- Robin does not love
 you now". Cogsworth watches the event:
 'Jane tells Marian: "- Robin does not
love you now"'. Marian attacks Jane. Ma
rian commits suicide. Cogsworth watches
 the event: 'Marian commits suicide'.

1: accept
2: adapt
3: validate
4: show
5: stop
decision: 4
```

Fig. 2. Interactive composition with **PlotBoard**.

B. *With Comic Books*

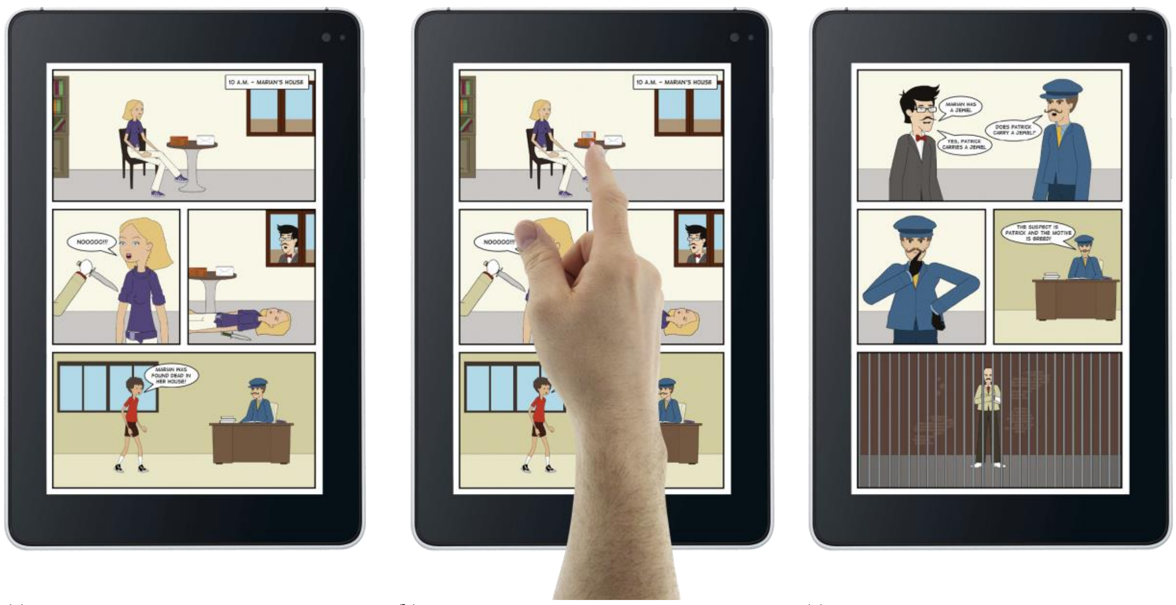
We now turn to an interactive storytelling system capable of representing the generated detective stories in the format of *comic books* [42][43][44] on tablet computers, where users are able to interact with certain objects that can affect the unfolding stories (Figure 3). Comics are a classical form of visual storytelling that combines images and text, and is capable of evoking strong emotional reactions from readers, creating identification, and conveying a story in a very appealing manner [43].

The system is based on a client-server architecture: the server hosts the planner responsible for generating stories, and the client contains the visualization and interaction interface that presents the narrative in the format of comic books. The process of representing the generated stories through comics consists of three main phases: plot structuring, panel definition, and panel compositing. In the *plot structuring* phase, the plot is organized into two storylines: crime and investigation. The story of the crime is presented first, but without revealing the criminal. More details about the crime are revealed during the investigation, with the detective's intervention.

The *panel definition* phase comprises the process of dynamically assigning the story events to their corresponding panels, computing the size required for each panel, and defining the layout of each page. The size of a panel is generally proportional to the amount of narrative content presented, and its position is relative to the chronological order of the events. In order to dynamically calculate the size of the panels, the system estimates the importance of a panel based on weights associated with the events and on the location where they take place. For example, a `go` event (a character goes from one place to another) may have less importance to the narrative than a `kill` event (a character kills someone); also, some places are more important than others.

Panels can be composed of four types of objects: background layers, characters, interactive objects, and text balloons. The *panel compositing* process consists of gathering all these elements together to form the final image of the panel. Background layers are a representation of the environment where the events occur. Every available location of the story is associated with a set of static or dynamic image layers that are used to create the scenarios of the story. Characters are composed of a set of behaviours representing the actions they can perform during the story. Each behaviour comprises a set of static images representing the action from different angles. During the compositing process, the behaviour is selected according to the action performed by the characters, and its position and angle are defined by waypoints positioned in the scenarios. Interactive objects are composed of two images, shown before and after the user interaction with them. During the compositing process, the objects are added to the panels as part of the scenarios. Text balloons are dynamically generated and inserted in the panels according to the chronological order of the events.

Once a plot is generated by the planning algorithm, the corresponding panels representing the story events are created, and users can read the story as a traditional comic book. Moreover, some scenarios include interactive objects that can be activated by tapping on them. Whenever such objects are present, the logical context of the story can be modified at that point of the narrative, according to the effects associated with the activated object. Reacting to the user's touch, the system requests a new plot from the plan-generator algorithm to create an alternative story consistent with the changes caused by the user interaction. As a result, the effects of the user's intervention are propagated to the next story events, and the comic panels are updated to reflect the new storyline. The user interaction, whereby Bertillon's case 1 (murder motivated by jealousy) was converted into case 2 (murder motivated by greed) is illustrated in Figure 3b, which shows the user's finger pointing to Jane's precious jewel.


10 A.M. – MARIAN'S HOUSE
NOOOOO!!!
MARIAN WAS FOUND DEAD IN HER HOUSE!
10 A.M. – MARIAN'S HOUSE
NOOOOO!!!
MARIAN HAS A JEWEL
DOES PATRICK CARRY A JEWEL?
YES, PATRICK CARRIES A JEWEL
THE SUSPECT IS PATRICK AND THE MOTIVE IS GREED!

(a) (b) (c)

Fig. 3. The interactive comic book: (a) a sample page; (b) the user touching an interactive object; (c) a page containing the modified events resulting from the user intervention.

V. Concluding remarks

Plot-composition methods based on plan-generation have the distinctive advantage of enforcing consistency within the genre specified. To ease the designers' task of specifying new genres, we are working on authoring facilities to help express in Prolog clauses the conventions of the chosen genre. However, even now, end-users do not have to see the clausal definitions when interactively creating plots over a pre-defined genre, thanks to various user interfaces developed in the course of the **Logtell** project.

On the basis of a preliminary study dealing with a different genre [45], we shall endeavour, as the next step in the project, to amplify the scope of criminal action and investigation, in particular to incorporate more subtle detective skills such as the analysis of the psychological profile of the suspects.

Finally, drawing from all kinds of detective works, we propose to continuously expand the library of story patterns. Our intention is to exploit such patterns as resources to produce new plots, by adapting and combining their diverse ways to machinate and solve memorable criminal cases.

Acknowledgment

This work was partially supported by CAPES (Coordination for the Improvement of Higher Education Personnel, linked to the Ministry of Education) under grant RH-TVD 01/2007 No. 133/2008 and by Research Productivity Grants of CNPq (National Council for Scientific and Technological Development, linked to the Ministry of Science and Technology) under grant PQ 304090/2009-3.

References

[1] A.E.M. Ciarlini, C.T. Pozzer, A.L. Furtado, B. Feijó. "A logic-based tool for interactive generation and dramatization of stories". Proc. Adv. in Comp. Entertain.Technology, 2005.

[2] A.E.M. Ciarlini, M.A. Casanova, A.L. Furtado, P.A.S. Veloso. "Modeling interactive storytelling genres as application domains". Journal of Intel. Info. Systems, v. 35, n. 3, 2010.

[3] T. Todorov. The Poetics of Prose. Cornell U. Press, 1977.

[4] F.A.G. da Silva, A.L. Furtado, A.E.M. Ciarlini, C.T. Pozzer, B. Feijó, E.S. de Lima. "Information-gathering events in story plots". Proc. 11th Int'l Conf. on Entert. Comp., 2012.

[5] C. Batini, S. Ceri, S. Navathe. Conceptual Design - an Entity-Relationship Approach. Benjamin Cummings, 1992.

[6] R.E. Fikes, N.J. Nilsson. "STRIPS: a new approach to the application of theorem proving to problem solving". Artif. Intelligence, 2 (3-4).

[7] A.S. Rao, M.P. Georgeff. “Modeling rational agents within a BDI-architecture”. Proc. of the Int’l Conf. on Principles of Knowledge Representation and Reasoning, 1991.

[8] D. Batista, C.T. Pozzer, E.W.G. Clua, E. B. Passos, “A Perception Simulation Architecture for Plot Generation of Emergent Storytelling,”. Proceedings of the Annual International Conference on Computer Games, Multimedia and Allied Technology, Bali, pp. 6-11, 2012.

[9] FIPA Communicative Act Library Specification, 2002, at: www.fipa.org/specs/fipa00037

[10] M.D. Sadek. “Logical task modelling for man-machine dialogue”. Proc. Eighth National Conference on Artificial Intelligence, 1990.

[11] H.P. Grice. “Logic and conversation”. In P. Cole, J.L. Morgan (eds.), Syntax and Semantics, Speech Acts, vol. 3, Academic Press, 1975.

[12] J.R. Searle. Expression and Meaning. Cambridge U. Press, 1979.

[13] C.S. Peirce. Writings of Charles S. Peirce: a Chronological Edition. N. Houser (ed.). Indiana University Press, 1998.

[14] R. Schank, R. Abelson. Scripts, Plans Goals, and Understanding. Psychology Press, 1977.

[15] J.L. Kolodner, Case-based Reasoning. Morgan Kaufmann, 1993.

[16] A.E.M. Ciarlini, A.L. Furtado. “Constructing libraries of typical plans”. Proc. 13th Conf. on Advanced Info. Systems Eng., 2001.

[17] J. Meehan, “TALE-SPIN, an interactive program that writes stories,” Proceedings of the Fifth Interactional Joint Conference on Artificial Intelligence, p. 91-98, 1977.

[18] M. Lebowitz, “Creating characters in a story-telling universe,” Poetics, vol. 13 (3), pp. 171-194, 1984.

[19] S. Turner, “MINSTREL: a computer model of creativity and storytelling,” Ph.D. Thesis, University of California, Computer Science Department, USA, 1992.

[20] S. Klein, J.F. Aeschlimann, D.F. Balsiger, S.L. Coverse, C. Court, M. Forster, R. Lao, J.D. Oakley, J. Smith. Automatic Novel Writing:A Status Report. Technical Report 186, Computer Sciences Department, U. of Wisconsin, 1973.

[21] A. Bruckman. The Combinatorics of Storytelling: Mystery Train Interactive. (Interactive Cinema Group). MIT Media Lab, 1990.

[22] Deadline. Infocom, 1982, cf.: http://en.wikipedia.org/wiki/Deadline_(video_game)

[23] A. Gustafsson, J. Bichard, L. Brunnberg, O. Juhlin, M. Combetto. “Believable environments: generating interactive storytelling in vast location-based pervasive games”. Proc. of the ACM SIGCHI Int’l Conf. on Adv. in Computer Entertainment Technology, 2006.

[24] J. Paay, J. Kjeldskov, A. Christensen, A. Ibsen, D. Jensen, G. Nielsen, R. Vutborg. “Location-based storytelling in the urban environment”. Proc. OZCHI, 2008.

[25] B. Mott, J. Lester. “U-Director: A Decision-theoretic narrative planning architecture for storytelling environments”. Proc. of AAMAS, 2006.

[26] J. Rowe, B. Mott, S. McQuiggan, J. Robison, S. Lee, J. Lester, J. “Crystal Island: A Narrative-Centered Learning Environment for Eighth Grade Microbiology”. Proc. of the AIED’09 Workshop on Intelligent Educational Games, 2009.

[27] J. Rowe, J. Lester, "Modeling user knowledge with dynamic Bayesian networks in interactive narrative environments". Proc. 6th Annu. AI Interact. Digital Entertain. Conerence., 2010.

[28] O. Bangsø, O.G. Jensen, F.V. Jensen, P.B. Andersen, T. Kocka, T. “Non-Linear Interactive Storytelling Using Object-Oriented Bayesian Networks”. Proc. of the International Conf. on Computer Games: Artificial Intelligence, Design and Education, 2004.

[29] M. Arinbjarnar. Rational Dialog in Interactive Games. MSc Thesis, School of Computer Science – Reykjavík University, 2007.

[30] V. Propp. Morphology of the Folktale. L. Scott (trans.). University of Texas Press, 1968.

[31] L.M. Barros, S.R. Musse, S. R. “Introducing narrative principles into planning-based interactive storytelling”. Proc. of the Int’l Conf. on Adv. in Computer Entert. Tech., 2005.

[32] R.B. Tobias, 20 Master Plots: And How to Build Them. Writer’s Digest Books, 1993.

[33] A. Christie. Curtain. Pocket Books, 1976.

[34] E.A. Poe. Complete Stories and Poems. Doubleday, 1984.

[35] A.C. Doyle. The Complete Sherlock Holmes: Doubleday, 1986.

[36] G.K. Chesterton. The Secret of Father Brown. The Editorium, 2006.

[37] H. Kenner. Paradox in Chesterton. Sheed & Ward, 1947.

[38] K.V. Deemter, E. Krahmer, M. Theune. “Real versus Template-Based Natural Language Generation: A False Opposition?”. Computational. Linguistics, vol. 31, n. 1, 2005.

[39] A.E.M. Ciarlini, S.D.J. Barbosa, M.A. Casanova, A.L. Furtado. “Event relations in plan-based plot composition”. Computers in Entertainment, 7, 4, 2009.

[40] E.S. Lima, B. Feijó, A.L. Furtado, S.D.J. Barbosa, C.T. Pozzer, A.E.M. Ciarlini, “Non-Branching Interactive Comics,” Proceedings of the International Conference on Advances in Computer Entertainment Technology, 2013.

[41] E.S. Lima, B. Feijó, A.L. Furtado, A.E.M. Ciarlini, C.T. Pozzer. “Automatic Video Editing for Video-Based Interactive Storytelling”. Proceedings of the International Conference on Multimedia & Expo (ICME 2012), pp. pp. 806-811, 2012.

[42] T. Alves, A. Simões, R. Figueiredo, M. Vala, A. Paiva, R. Aylett. “So tell me what happened: Turning agent-based interactive drama into comics”. Proc. AAMAS, 2008.

[43] S. McCloud (1994). Understanding Comics: The Invisible Art. William Morrow 1994.

[44] T. Shuda, R. Thawonmas. “Frame Selection for Automatic Comic Generation from Game Log”. Proc. 7th International Conference on Entertainment Computing, 2008.

[45] S.D.J. Barbosa, A.L. Furtado, M.A. Casanova. “A Decision-making Process for Digital Storytelling”. Proc. of the Brazilian Symp. on Games and Digital Entertainment, 2010.