Information-gathering Events in Story Plots

Fabio A. Guilherme da Silva[1], Antonio L. Furtado[1], Angelo E. M. Ciarlini[2], Cesar Tadeu Pozzer[3], Bruno Feijó[1], and Edirlei Soares de Lima[1]

[1]PUC-Rio, Depto. de Informática, Brasil
{faraujo,furtado,bfeijo,elima}@inf.puc-rio.br
[2]UNIRIO, Depto. de Informática Aplicada, Brasil
angelo.ciarlini@uniriotec.br
[3]UFSM – Departamento de Eletrônica e Computação, Santa Maria
pozzer3@gmail.com

Abstract. Story plots must contain, besides physical action events, a minimal set of information-gathering events, whereby the various characters can form their beliefs on the facts of the mini-world in which the narrative takes place. In this paper, we present an approach to model such events within a plan-based storytelling context. Three kinds of such events are considered here, involving, respectively, inter-character communication, perception and reasoning. Multiple discordant beliefs about the same fact are allowed, making necessary the introduction of higher-level facilities to rank them and to exclude those that violate certain constraints. Other higher-level facilities are also available for pattern-matching against typical-plan libraries or previously composed plots. A prototype logic programming implementation is fully operational. A simple example is used throughout the presentation.

Keywords: Plot Composition, Communicative Acts, Perception, Deduction, Abduction, Plan Recognition, Plan Generation, Logic Programming.

1 Introduction

Story plots typically include action events, but another class of events is also needed for the sake of realism: the *information-gathering* events, which 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, being aware of all facts that currently hold and of how they change as a consequence of the action events. In order to create and analyze more plausible story plots, we propose an approach to model *information-gathering* events.

Here we shall recognize a sharp distinction between the facts themselves and the sets of *beliefs* 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 actual 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 consider three types of information-gathering events, each type associated with a set of operations: *communication events*,

supported by the operations `tell`, `ask` and `agree`; *perception events*, supported by the operations `sense` and `watch`; and *reasoning events*, supported by the operations `infer` and `suppose`.

All operations refer to beliefs on facts, except `watch`, whose object is some action event witnessed by a character. The operations are defined in terms of their pre-conditions and post-conditions, in the same way operations corresponding to action events are defined, i.e. using the STRIPS formalism [1]. The pre-conditions are logical expressions commonly 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. The specification of the operations is deliberately kept at a minimum to be independent from the context. It is however, complemented, both with respect to pre-conditions and post-conditions, by separate *conditioners* that express the peculiarities of the different characters participating in the stories.

The approach described here is going to be integrated with the **Logtell** interactive storytelling system [2, 3]. **Logtell** uses a *plan-generation* algorithm that deals with nondeterministic and partially-ordered events to compose plots. Since the present work focuses on the construction of an information-gathering package, to be later integrated to the design of full-fledged narrative genres, we decided to initially use a simpler planner that deals only with deterministic events. In addition, a single action event will be mentioned in our example. This event, associated with the `go` operation, consists of the displacement of a character from a place to another, which is needed because presential verbal interaction is the only form of communication that we currently cover.

All features discussed were implemented in a logic-programming prototype, and a simple running example is used as illustration. Section 2 explains how the example was formulated so as to run in a plan-based context. Section 3 describes the information-gathering events. Section 4 adds some higher-level facilities, which help to analyze the resulting beliefs and to make comparisons by means of *plan-recognition*. Section 5 reviews related work, and section 6 contains concluding remarks.

2 Example in a plan-based context

2.1 Conceptual specification

Our conceptual design method involves three schemas: static, dynamic and behavioural. The *static schema* specifies, in terms of the *Entity-Relationship* model [4], the entity classes, attributes and binary-relationships. The information-gathering package requires the Prolog clauses below to describe entities, attributes and relationships:

```
entity(person,name).
attribute(person,gender).
relationship(current_place,[person,Place]) :-  taken_as_place(Place).
attribute(person,believes).
relationship(trusts,[person,person]).
```

Similarly to what happens to `believes`, we have defined `sensed`, `watched`, `inferred`, `supposed` and `asked` as attributes of the entity `person`. Also, notice that the specification of the `current_place` relationship associates the entity `person` with a still unde-

termined entity, represented by the variable `Place`. Putting the package together with a story context, other clauses can be added. In our example, we use the following:

```
entity(country,country_name).
entity(city,city_name).
attribute(person,hair_colour).
attribute(person,daltonic).
relationship(born,[person,country]).
relationship(home,[person,country]).
relationship(citizen,[person,country]).
taken_as_place(city).
```

The *dynamic schema* defines a fixed repertoire of operations for consistently performing the state changes corresponding to the events that can happen in the mini-world of the application. The *STRIPS* [1] model is used. Each operation is defined in terms of pre-conditions, which consist of conjunctions of positive and/or negative literals, and any number of post-conditions, consisting of facts to be asserted or retracted as the effect of executing the operation. The operations that constitute the core of the information-gathering package will be described in section 3.

Currently our *behavioural schema* specifications mainly consist of goal-inference (a.k.a. situation-objective) rules. Since our present running example does not employ such rules, we shall not discuss them here (cf. [2]).

2.2 States of the stories

States of the story are sets of ground clauses denoting valid instances of the specified static schema. These clauses are facts (positive literals) specifying the entity instances, their attributes and relationships. Whenever a fact is not part of a state, it is assumed to be false. Beliefs and facts that describe what is told, asked, sensed, watched, inferred and supposed by characters are attributes of the characters and can also be part of a state and mentioned in pre- and post-conditions of the events. These facts about facts can speak about both positive and negative literals (e.g. a character can believe that a certain fact is not true).

To generate a plot, it is necessary to populate the initial database state. Informally speaking, the mini-world of our example comprises four characters, John, Peter, Mary and Laura, three countries, UK, USA and Canada, and two cities, both in the UK, London and Manchester. The recorded information does not provide a uniform coverage. It registers where Mary, Peter and Laura were born but does not indicate John's birth-place. About Mary it adds that her domicile (home) is also in the UK and that she has red hair, whereas Laura ─ who, in spite of having been born in the USA, is a Canadian citizen ─ is blond. Peter is said to be daltonic. John, Peter and Mary are currently in London, and Laura in Manchester. Contrary to the other characters, whose beliefs are initially confined to their explicitly recorded properties, John is aware of all registered facts.

2.3 Main Features of the Plan-generator

The plan generator follows a backward chaining strategy. For a fact `F` (or `not F`) that is part of a given goal, it checks whether it is already true (or false) at the current

state. If this is not the case, it looks for an operation `Op` declared to add (or delete) the fact as part of its effects. Having found such operation, it then checks whether the pre-condition `Pr` of `Op` currently `holds` – if not, it tries, recursively, to satisfy `Pr`. Moreover, the plan generator must consider the so-called frame problem [5].

In view of the needs of the information-gathering package, we specified `pre_state(Op,S)` as one more effect of every operation `Op`, which allows to capture in `S` the *prefix* of operation `Op`, i.e. the entire plan sequence, starting at the initial state, to which `Op` will be appended. Indeed, sequence `S` supplies a convenient operational denotation of the state immediately before the event denoted by `Op`, to which we may, in particular, apply the `holds` predicate to find out who was present at some place associated with the occurrence of the event. This is important, in particular, because (at least for the time being) we assume that, to watch an event, a character should be at the place where the event occurs.

Like goals, pre-conditions are denoted by conjunctions of literals. We distinguish, and treat differently, three cases for the positive or negative facts involved:

1. facts which, in case of failure, should be treated as goals to be tried recursively by the plan generator;
2. facts to be tested immediately before the execution of the operation, but which will not be treated as goals: if they fail the operation simply cannot be applied;
3. facts that are not declared as added or deleted by any of the predefined operations.

Note that the general format of a pre-condition clause for operation `Op` is `precond(Op, Pr) :- B`. In cases (1) and (2), a fact `F` (or `not F`) must figure in `Pr`, with the distinction that the barred notation `/F` (or `/(not F)`) will be used in case (2). Case (3) is handled in a particularly efficient way. Since it refers to facts that are invariant with respect to the operations, such facts can be included in the body `B` of the clause, being simply tested against the current state when the clause is selected.

An example is the precondition clause of operation `tell(A,B,F)`, where character `A` tells something to character `B`. We require that the two characters should be together at the same place, and, accordingly, the `Pr` argument shows two terms containing the same variable `L` to express this location requirement, but the term for `B` is barred: `/current_place(B,L)`, which does not happen in `A`'s case. The difference has an intuitive justification: character `A`, who is the agent of the operation, has to either already be present or to go to the place `L` where `B` is, but the latter would just happen to be there for some other reason.

The proper treatment of (1) and (2) is somewhat tricky, because of the backward chaining strategy of the planning algorithm. Suppose the pre-condition `Pr` of operation `Op` is tested at a state S_1. If it fails, the terms belonging to case (1) will cause a recursive call whereby one or more additional operations will be inserted so as to move from S_1 to a state S_2 where `Op` itself can be included. But it is only at S_2, not at S_1, that the barred terms in case (2) ought to be tested, and so the test must be *delayed* until the return from the recursive call, when the plan sequence reaching S_2 will be fully instantiated. Delayed evaluation is also needed, as one would expect, for instantiating the `pre_state` predicate mentioned before.

Once generated, a plan can be processed via the `execute` command, thus effecting the desired state transition, i.e. adding and/or deleting facts to/from the current data-

base state. As a side effect, the `log(S)` clause (initially set as `log(start)`) is updated by appending to `S` the plan executed. At any time, the entire story thus far composed can be narrated, in pseudo-natural language, by entering "`:- log.`".

To finish this partial review of the plan features, we remark that the planning algorithm `plans(G,P)` can be called in more than one way. More often `G` is given, as the goal, and `P` is a variable to which a generated plan will be assigned as output. However an inverse usage has been provided, wherein `P` is given and `G` is a variable. In this case, the algorithm will check whether `P` is executable in view of the initial state and of the interplay of pre- and post-conditions, and, if so, assign its net effects (a conjunction of `F` and `not F` terms) to `G`.

2.4 Templates for pseudo-natural language generation

Both for facts and events, we resort to *templates* for description and narration in pseudo-natural language. The template device allows, to begin with, to list all properties registered in the initial database state (or in the current state reached by executing a plan), via the `facts` predicate. The templates for operations and facts are combined in a way that favours a fairly readable style. Consider, as an example, operation `tell(A,B,F)`, in which `A` tells fact `F` to `B`. Suppose fact `F` corresponds to a property of character `C`. Concerning the identity of the three characters, we can distinguish three situations:

- They are all distinct: `tell('John','Peter',hair_colour('Mary',red))`
- C is the same as A: `tell('Mary','Peter',hair_colour('Mary',red))`
- C is the same as B: `tell('John','Mary',hair_colour('Mary',red))`

The `def_template` algorithm that drives the application of the templates produces for the above events:

```
John tells Peter: "- Mary has red hair".
Mary tells Peter: "- My hair is red".
John tells Mary: "- Your hair is red".
```

The algorithm duly uses gender information, rendering for example `infer('Mary',citizen('Mary','UK'))` as:

```
Mary infers that she is a citizen of UK.
```

Negative facts in the several operations, and the occurrence of variables in the `ask` operation, are treated as expected by the algorithm. So, again for the `hair_colour` property, one will have, respectively:

```
tell('John','Laura',not hair_colour('Laura',red))
   John tells Laura: "- Your hair is not red".
ask('John','Laura',hair_colour('Laura',X))
   John asks Laura: "- What is the colour of your hair?".
ask('John','Laura',hair_colour(X,red)))
   John asks Laura: "- Who has red hair?".
```

Analogous templates are provided for rendering in pseudo-natural language the information-package facts, such as `told`, `asked`, `sensed`, `watched`, `inferred`, `supposed`, `believes` and `trusts`. These templates are generic and can be used in different story contexts.

3 The information gathering events

Our information gathering events change the current state by adding clauses that represent when facts are believed, told, asked, sensed, watched, inferred and supposed by characters. They were modeled by a generic schema of the corresponding operations, which extends the set of pre-conditions in accordance with context-dependent rules. In this section such schema is described.

3.1 Communication events

In the Computer Science community, communication between characters immediately brings to mind the communication processes executed by software agents in multi-agent environments [6]. In particular, the Agent Communication Language consists of formally specified operations similarly defined by their pre-conditions and post-conditions [7]. However software agents differ from fictional characters (and, ironically, from human beings in general) in that they are supposed to only transmit information on 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 [8]. The bare 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 local `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 (note the "/" sign before `current_place(B,L)`, as showed in section 2.3, to indicate that `B` would not be expected to move). And the only necessary effect of the operation is merely that `F` is `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`.

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` clauses. These clauses specify the conditions for telling or asking about a fact within a certain story context. Such conditions are automatically considered as part of the pre-conditions of an event `tell` or `ask` by the planning algorithm.

Example 1: Mary is willing to ask John about his current whereabouts. She asks, he replies and, since she trusts him, adopts the belief that he is in London.

```
Goal: believes(Mary, current_place(John, A))
Mary asks John: "- Where are you?". John tells Mary: "- I am now in
London". Mary agrees with John.
```

3.2 Perception events

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` operation to apprehend any sort of fact, 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 is accompanied by a side-remark on the true fact. In any case, besides the effect that `F` was `sensed` by the character, a `belief` clause is immediately added, since direct perception does not depend on a third party who might not be trusted.

The `watch(W,O)` operation was harder to implement, requiring the inclusion of the already mentioned `pre_state(O,S)` clause as an extra feature, where `O` is the operation witnessed by `W`, and `S` denotes the state previous to the application of `O` (i.e. the sub-sequence of the generated plan that precedes `O`). It becomes possible then to check the location of character `W` at the time when `O` happens.

As before, the definitions are left to be 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 an entity instance `E` currently at place `L`, a character `W` must be at `L`, either originally or as the result of pursuing `current_place(W,L)` as a sub-goal. For `watch`, normally applicable to action events only, the `current_place` requirements depend on what is being watched, which justifies their being left to the special `watch_rule` clauses, to which the `current_place(W,L)` information, checked as described before, is passed. For instance, `go(A,L1,L2)` can be watched partly by persons at `L1` (origin) and partly by those present at `L2` (destination). Naturally the agent (character `A`) is able to watch the action in its entirety.

Example 2: Peter obtains three different indications concerning the colour of Mary's hair. Only the first, supplied by trustworthy John, was correct. Laura was lying, and Peter himself, being daltonic, failed to perceive the true colour.

```
Goal: believes(Peter, hair_colour(Mary, A))
John tells Peter: "- Mary has red hair". Peter agrees with John.
Laura goes from Manchester to London. Laura tells Peter: "- Mary has
blond hair". Peter agrees with Laura.
Peter wrongly senses that Mary has green hair -- in fact Mary has red
hair.
```

3.3 Reasoning events

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 (fundamental to the natural sciences), the systematic occurrence of B whenever A occurs may justify the adoption of rule $A \rightarrow B$. Abduction is a non-guaranteed but nevertheless most 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, who try to diagnose an illness in view of observed symptoms. The trouble is, of course, that it is often the case that more than one illness may provoke the same symptom — in other words: there may exist other applicable rules $A^1 \rightarrow B$, $A^2 \rightarrow B$, ..., $A^n \rightarrow B$, suggesting different justifications for the occurrence of B. In abduction, 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. The conditioners for both can be the same rules of inference (`inf_rules`) to be trav-

ersed forward in the former case or backward in the latter. In the case of `infer` over a rule `P=>F` adopted 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 an `added` effect (another addition being an `inferred` clause) upon a successful evaluation of `P`. Conversely, in the case of `suppose`, the belief on the consequent will motivate the addition of a belief in some fact present in the logical expression of the antecedent.

We must stress that the inference rules adopted by the characters in a given story do not have to be scientifically established rules. Informally, in our example, the rules are:

1. if person A was born in a country B, and A's domicile is also in B, then A is a citizen of B.
2. if person A is daltonic, and says that the colour of B's hair is C1, but a daltonic when looking at an object coloured C2 would mistakenly perceive that colour as C1, then the colour of B's hair is C2.
3. if person A was observed departing from location L to some other location, then A is no longer at L.
4. if person A was observed arriving at location L, coming from some other location, then A is currently at L.

Rules **3** and **4** are trivial, representing a more general case: watching an event from an appropriate place allows the observer to conclude that the direct and indirect effects of the event should hold. We point that rule **1** does not really cover the legal citizenship requirements prevailing in most countries, and rule **2** represents a naive understanding (taking red for green and vice-versa) of one variety of colour blindness (cf. [9]).

Example 3: As a goal that, so we thought, should fail, we enquired how red-haired Mary could come to believe that her hair was not red. The planner found a solution using inference in a most devious way: John gives Peter the correct information, which he transmits to Mary. However, having first noticed that Peter is daltonic, Mary is led to apply our (naive) inference rule dealing with red-green colour blindness.

```
Goal: believes(Mary, hair_colour(Mary, A)), not A=red
Mary senses that Peter is daltonic. John tells Peter: "- Mary has red
hair". Peter agrees with John. Mary asks Peter: "- What is the colour of
my hair?". Peter tells Mary: "- Your hair is red". Mary infers that she
has green hair.
```

4 Higher-level facilities

In order to analyze composed plots and the resulting beliefs, we have implemented a set of higher-level facilities. In the current prototype, the predicates implementing the higher-level facilities are, so to speak, external to the narrative, to be applied as an instrument of analysis after plot composition. Future work may promote their inclusion in the repertoire of events, especially in the context of detective stories, where the critical examination of past facts and events plays a fundamental role. In this way the analysis of the plot so far might become part of the narrative (more on that in section 6). To run the higher-level facilities with a specific application, a number of clauses must be specified. The facilities and the corresponding additional information necessary to use them are described in the sequel.

4.1 Surveying and ranking multiple beliefs

Since multiple beliefs about the same fact are allowed, one needs a device to rank them on the basis of their provenance. The survey predicate collects all beliefs of a character about an indicated fact, together with their provenance (operation whereby they were acquired) and puts them in decreasing order with respect to the appropriate weights. These must have been declared by a conditioner; for the example below, we shall assume the following one: `weights('Peter', hair_colour(X,Y), [1:sensed('Peter',_), 2:told('John',['Peter',_])])`.

Example 4: Peter collects all possible inputs on the colour of Mary's hair. He then ranks the results, according to his pre-defined list of weights based on provenance. As far as hair-colour is concerned, what he hears from John is ranked (weight 2) above the evidence of his own defective eyesight (weight 1). He trusts Laura, but never thought of assigning a weight to her opinion (weight 0 is the default). By using our facility to survey and rank Peter's beliefs towards Mary's hair color, the following output can be obtained.

```
Surveying: hair_colour(Mary, A)
sensed(Peter, hair_colour(Mary, green))
told(John, [Peter, hair_colour(Mary, red)])
told(Laura, [Peter, hair_colour(Mary, blond)])
Ranking the results:
2:hair_colour(Mary, red)
1:hair_colour(Mary, green)
0:hair_colour(Mary, blond)
```

Another stricter surveying facility, when collecting the various inputs rejects those that cause the activation of any `violate_rule`. Such rules play a central role in the higher-level facility described in the next section.

4.2 Validating a belief

Certain beliefs may not make sense in that they violate some natural law, or legal norm or even some convention of the chosen story genre. An elementary kind of violation refers to the schema definition itself: e.g. instances of a relationship are not acceptable if not declared between existing instances of the entity classes over which the relationship was defined. Also, similarly to naive inferences, a `violate_rule` established for a story genre may reflect its conventions, rather than the real world.

The `violations` predicate, illustrated below, checks a given expression in view of the established `violate_rules`. If a rule is violated more than once, the rule identifier will be repeated an equal number of times in the resulting list.

In our example, both `violate_rules` are about the `citizen` relationship. According to rule r1, any instance thereof is considered not valid if the first parameter is not a `person` or the second is not a `country`, whereas rule r2 excludes the possibility of plural citizenship (although such cases are often encountered in practice).

Example 5: When checking a belief corresponding to the conjunction of the facts `citizen('Mary','UK'), citizen('Mary','London'), citizen('Mickey','USA')`, rule r1 is activated twice: London is not a `country` and Mickey is not a `person`. A violation related to rule r2 is also detected, since there should be no more than one clause declaring Mary's citizenship .

4.3 Recognizing a library plan from events observed

Typical plans can be extracted from previously existing plots, and, after their parameters are consistently replaced by variables, be stored under this plan-pattern form in a library, for future reference. Our tiny example library comprises two short plans:

```
lib([start=>go(A,L1,L2)=>go(B,L1,L2)=>sense(A,current_place(B,L2)),
     start=>go(A,L1,L2)=>go(A,L2,L1)]).
```

In the former, two characters `A` and `B` follow the same itinerary and, next, `A` senses that they are now together at place `L2`. The latter just shows `A` leaving from and returning to the same place.

One of the uses of a library is to match one or more observed events against each plan-pattern. If all the observations supplied, ideally in a small number, unify with plan events that must lie in the same sequence but do not have to be contiguous, one gains the following complementary benefits: (1) anticipating what the characters are trying to achieve in the long run and (2) extending the few events to a larger plot, consisting of the matching plan pattern, with some (or all) variables instantiated as a consequence of unification.

To obtain further intuitive understanding of what (1) means, take the commonplace example of a person being observed to hail a taxi and go to an airport. These observations would match a plan-pattern with events such as buying an air ticket, hailing a taxi, loading a number of bags on the taxi, going to the airport, etc., etc., checking-in, boarding the plane, etc. But they might also match a similar plan in which the person would be going to the airport not to embark but to meet another person in an arriving flight. The fact that the same observations can match alternative plans shows that recognition can, in general, be hypothetical.

On the other hand, a prospective author would have, in view of (2), one or more possible plots obtained by extending an initial fragmentary sketch. So, curiously, both plan-generation (as shown in the previous sections) and plan-recognition provide useful story composition strategies. Indeed plan-recognition brings to mind the notion of reuse, and in the literary domain is in consonance with the remark in [10] that "any text is a new tissue of past citations".

Example 6: Two observed `go(X,Y,Z)` events are matched against the given library of typical plans. The first event is fully instantiated, whereas the second seems to result from a vague observation: the only clue is that the agent was either John himself or a woman. With the first option, the library plan wherein the same character travels forward and then backward is recognized; with the second, the recognized plan is that two different characters embark on the same trip, and the former notices the presence of the latter when they have both reached their destination. Calling the plan-generator to validate the plans has the effect of restricting the choice of the female character to Mary, who happens to be initially in London as required. By using our facility to recognize plans the following output can be obtained.

```
Observed: [go(John, London, Manchester), go(A, B, C)]assuming that A was
either John himself or some person of the female genderLibrary plans
recognized - and executable:
John goes from London to Manchester. John goes from Manchester to Lon-
don.
John goes from London to Manchester. Mary goes from London to Manches-
ter. John senses that Mary is now in Manchester.
```

4.4 Recognizing a pattern in a generated plan

Pattern-matching can also take an opposite direction, working on an existing plot and checking whether it contains some not necessarily contiguous subsequence to which a pattern may be matched. Both verifying that the match succeeds and, if so, extracting the matching subsequence are relevant to the analysis of plots.

Example 7: A pattern expressing a going and returning trip performed by some character is matched using our facility against an existing plot, which, among its events, contains an instance of the pattern ─ which is duly found. The following output can then be displayed.

```
Applying the pattern: [go(A, B, C), go(A, C, B)]
to the given sequence: start=>go(John, London, Manchester)=>
go(Laura, Manchester, London)=>go(John, Manchester, London)
one finds: [go(John, London, Manchester), go(John, Manchester, London)]
```

5 Related work

Within the Interactive Storytelling area, the problem of controlling the diversity and coherence of the stories has been tackled by means of the specification of reactive behaviour, such as in Façade [11], or by means of deliberative planning, such as in [12-14], where techniques such as hierarchical task networks-HTN [15] and partial-order planning are applied. In our **Logtell** tool [3], partial-order planning and HTN are combined to conciliate flexibility and efficiency and to treat nondeterministic events. In this paper, we propose a generic approach to deal with information gathering events that could be adapted to different story contexts. In order to simplify the reasoning process about beliefs, we have resorted to a simpler backward chaining linear planner, and used logic programming to infer pre-conditions and post-conditions of the events. As we intend to incorporate this approach to **Logtell**, it will be adapted to deal with partially-ordered nondeterministic events.

Reasoning about beliefs in order to perform actions is a major characteristic of the Belief–Desire–Intention (BDI) software model [16]. The model's architecture purports to implement the principal aspects of the theory of human practical reasoning originally proposed by Bratman [17]. Efforts to formulate logical models to define and reason about BDI agents have led to formal logical descriptions such as BDICTL [18] and, more recently, LORA (the Logic Of Rational Agents) [19]. In [20], beliefs, and their respective strengths, are recognized from the surface form of utterances, from discourse acts, and from the explicit and implicit acceptance of previous utterances. Communicative operations can be performed with the aid of the Knowledge Query and Manipulation Language (KQML) [21]. ACL is a proposed standard language for agent communications in multi-agent systems, whose semantics are based on the BDI model of agency [7]. KQML and ACL define a set of *performatives* (operations performed by the agents) and, more fundamentally, rely on speech act theory [22-23]. In our approach, we explicitly reason about beliefs in order to achieve goals of the characters, and such goals can be generated by means of goal-inference rules, which can represent each character's desires. In spite of the similarities with BDI models, there is a fundamental difference: instead of being concerned about the actual

communication between software agents, we are focused on creating coherent plots in which characters' actions can be justified by their beliefs.

Regarding the use of templates, one important issue in applied Natural Language Generation (NLG) is deciding between the usage of complete NLG systems, or the option – which we favoured in our current work – for template-based approaches [24]. Although some scholars have stressed the disadvantages of template-based systems in comparison to full-fledged NLG systems [25-26], some more recent studies do not concur with this point of view [27]. In fact, they are considered Turing-equivalent computational systems [25]. Reiter [28] compares the two techniques, showing the advantages in each case. Indeed, template-based approaches are easier to implement, and generate texts more quickly than traditional approaches. Yet, on the other hand, rigid templates are inflexible and difficult to reuse. An Augmented Template-Based approach is proposed in [29] as a means of yielding templates that should prove more flexible and reusable.

There surely exists a good deal of interesting research about the automatic generation of texts for different purposes, in some cases with the use of templates (cf., for example, [30-31]). More specifically, NLG methods have been applied to interactive storytelling systems (e.g. [32-34]). However we still find that the automatic rendering of a given plot in a high quality text is more often than not a complex task [35].

6 Concluding remarks

Besides the simple-minded example used as illustration, we have already applied some of the information-gathering events to enhance the Swords-and-Dragons genre that runs in **Logtell**, by dropping the unrealistic omniscience assumption. Now a damsel watches the villainous Draco kidnapping Princess Marian, runs to `tell` Sir Brian about the mischief, and he `infers` that, as a kidnapped victim, she should have been carried to the villain's dwelling, whereto he promptly rides to rescue his beloved.

But the availability of information-gathering events will, probably after further elaboration, open the way to more sophisticated genres. In particular, we have been examining the requirements of detective stories. It has been convincingly argued [36] that such narratives actually contain two stories: one covers the crime and the other the investigation. The first story ends before the second begins. And the characters of the story of the investigation do not predominantly act ─ they *learn* ─, which is well within the scope of the package discussed here.

Future work should extend the repertoire of events to contemplate other speech acts, for example to allow a character C1 to solicit or to order another character C2 to execute an action of C1's interest, which C2, but not C1, is empowered and in a position to perform. Moreover, in stories of even moderate complexity, behaviour should be characterized as a decision-making process affecting the participation of each character in every kind of event, either involving physical action or the information-gathering activities of the present study ─ and this process hinges on both cognitive and emotional considerations [37- 42], further influenced by the goals and plans of the other characters [43]. We have done some initial work on drives, attitudes, emotions, and mutual interferences among agents [44], but a full integration within the **Logtell** system still remains to be achieved.

References

1. Fikes, R.E.; Nilsson, N.J. (1971). "STRIPS: A new approach to the application of theorem proving to problem solving". *Artificial Intelligence* , 2(3-4).
2. Camanho, M.M; Ciarlini, A.E.M.; Furtado, A.L; Pozzer, C.T.; Feijó, B. (2008). "Conciliating coherence and high responsiveness in interactive storytelling". Proc. of the *3rd International conference on Digital Interactive Media in Entertainment and Arts*, pp. 427-434.
3. Silva, F.A.G.; Ciarlini, A.E.M.; Siqueira, S.W.M. (2010). "Nondeterministic Planning for Generating Interactive Plots". Proc. *Ibero-American Conference on Artificial Intelligence*, 162, Bahía Blanca, Argentina, 1-5 November.
4. Batini, C.; Ceri, S.; Navathe, S. (1992). *Conceptual Design – an Entity-Relationship Approach*. Benjamin Cummings.
5. Lloyd, W. (1987). *Foundations of Logic Programming*. Springer.
6. Sadek, M.D. (1990). "Logical task modelling for man-machine dialogue". In: *Proceedings of the eighth AAAI conference*, Boston, MA.
7. FIPA Review of FIPA Specifications (2006). At http://www.fipa.org/subgroups/ROFS-SG-docs/ROFS-Doc.pdf.
8. Grice,H.P. (1975). "Logic and conversation". In: P. Cole, J.L. Morgan (Eds.), *Syntax and Semantics, Speech Acts*, vol. 3, Academic Press, New York.
9. Dalton, J. (1798). "Extraordinary facts relating to the vision of colours: with observations". *Memoirs of the Literary and Philosophical Society of Manchester* **5**: 28–45.
10. Barthes, R. (1981). "The Theory of the Text". In *Untying the Text: A Post-Structural Reader*. R. Young (ed.). Boston: Routledge, 31-47.
11. Mateas, M.; Stern, A. (2005). "Structuring content in the Facade interactive drama architecture". In: *Proc. Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*.
12. Cavazza, M.; Charles, F.; Mead, S. (2002). "Character-based interactive storytelling". *IEEE Intelligent Systems, special issue on AI in Interactive Entertainment*, 17(4):17-24.
13. Young, R. (2001). "An overview of the mimesis architecture: Integrating narrative control into a gaming environment". In: *Working notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, pp. 78-81, Stanford, CA. AAAI Press.
14. Riedl, M.; Young, R.M. (2006). "From Linear Story Generation to Branching Story Graphs". *IEEE Computer Graphics and Applications* 26(3), 23–31.
15. Erol, K.; Hendler, J.; Nau, D. S. (1994). "UMCP: A sound and complete procedure for hierarchical task-network planning". In: *Proceedings of the International Conference on AI Planning Systems (AIPS)*, pp. 249-254.
16. Rao, A.S.; Georgeff, M.P. (1995). "BDI-agents: From Theory to Practice". In *Proceedings of the First International Conference on Multiagent Systems (ICMAS'95)*, San Francisco.
17. Bratman, M.E. (1999). *Intention, Plans, and Practical Reason*. CSLI Publications.
18. *Dastani, M., van der Torre, L. (2002). "An extension of BDICTL* with functional dependencies and components". In: *Procs. of LPAR'02*. LNCS 2514, Springer.
19. Wooldridge, M. (2000). *Reasoning About Rational Agents*. The MIT Press. ISBN 0-262-23213-8.
20. Carberry, S.; Lambert, L. (1999). "A Process Model for Recognizing Communicative Acts and Modeling Negotiation Subdialogues". *Computational Linguistics* 25(1):1--53.
21. Finin, T.; Weber, J.; Wiederhold, G.; Gensereth, M.; Fritzzon, R.; McKay, D.; McGuire, J.; Pelavin, R.; Shapiro, S.; Beck, C. (1993). *DRAFT Specification of the KQML Agent-Communication Language* (PostScript), June 15. At http://www.cs.umbc.edu/KQML/kqmlspec.ps.
22. Searle, J.R. (1969). *Speech Acts An Essay in the Philosophy of Language*, Cambridge University Press, Cambridge.

23. Winograd, T.; Flores, F. (1986). *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishing Corp, (Norwood).
24. Reiter, E.; Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge.
25. Reiter, E.; Dale, R. (1997). "Building applied natural language generation systems". *Nat. Lang. Eng.*, Vol. 3, pp. 57-87.
26. Busemann, S. and Horacek, H. (1998). "A flexible shallow approach to text generation". In: *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 238–247: Niagara-on-the-Lake, Ontario, Canada.
27. van Deemter, K.; Krahmer, E.; Theune, M. (2005). "Real versus Template-Based Natural Language Generation: A False Opposition?". *Comput. Linguist.*, Vol. 31, pp. 15-24.
28. Reiter, E. (1995) NLG vs. Templates. CoRR, Vol. cmp-lg/9504013.
29. McRoy, S.W.; Channarukul, S.; Ali, S.S. (2003). "An augmented template-based approach to text realization". *Nat. Lang. Eng.*, Vol. 9, pp. 381-420.
30. Stenzhorn, H. (2002). " XtraGen: A Natural Language Generation System Using XML-And Java-Technologies". In: *Proceedings of the 2nd workshop on NLP and XML* - Volume 17, pp. 1-8.
31. Piwek, P. (2003). "A flexible pragmatics-driven language generator for animated agents". In: *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics* - Volume 2, pp. 151-154.
32. Gervás, P.; Díaz-Agudo, B.; Peinado, F.; Hervás, R. (2005). "Story plot generation based on CBR". *Knowledge-Based Systems*, Vol. 18(4-5), pp. 235 – 242.
33. Szilas, N. (2007). "A Computational Model of an Intelligent Narrator for Interactive Narratives". *Applied Artificial Intelligence*, Vol. 21(8), pp. 753-801.
34. Montfort, N. (2007). *Generating narrative variation in interactive fiction*. PhD dissertation, Computer and Information Science - University of Pennsylvania.
35. Callaway, C.B. and Lester, J.C. (2002). "Narrative prose generation". *Artificial Intelligence* 139 (2) 213–252.
36. Todorov, T. (1977). *The Poetics of Prose*. Cornell University Press.
37. Brave, S.; Nass, C. (2008) "Emotion in Human-Computer Interaction". In: A. Sears & J.Jacko (eds.) *The Human-Computer Interaction Handbook*, pp. 77–92.
38. Loewenstein, G.; Lerner, J.S. (2003). "The role of affect in decision making". In *Handbook of Affective Sciences*. Davidson, R.J.; Scherer, K.R.; Goldsmith, H.H. (eds.). Oxford University Press, pp. 619-642.
39. McCrae, R.R.; Costa, P.T. (1987). "Validation of a five-factor model of personality across instruments and observers". *J. Pers. Soc. Psychol.*, 52, pp. 81-90.
40. Goldberg, L.R. (1992). "The Development of Markers for the Big-Five Factor Structure". *Psychological Assessment*, v4, n1 pp. 26-42.
41. O'Rorke, P.; Ortony, A. (1994). "Explaining Emotions". *Cognitive Science*, 18, 2, pp. 283-323.
42. Ortony, A. (2003). "On making believable emotional agents believable". In *Emotions in Humans and Artifacts*. Trappl, R.; Petta, P.; Payr, S. (eds). The MIT Press, pp. 189-211.
43. Willensky, R. (1983). *Planning and Understanding - a Computational Approach to Human Reasoning*. Addison-Wesley.
44. Barbosa, S.D.J.; Furtado, A.L.; Casanova, M.A.C. (2010). "A Decision-making Process for Digital Storytelling". Proc. *IX Symposium on Computer Games and Digital Entertainment - Track: Computing*.