6 Plot Manipulation Algebra

6.1.Chapter Preface

In this paper, co-authored with Simone Barbosa, Antonio Furtado, and Marco Casanova we describe the foundations of PMA, a Plot Manipulation Algebra to be used in mediating between the conceptual design of a given genre and the system that provides its physical implementation.

In this paper the reasoning behind PMA is described, along with its set of operators. PMA is then used directly as a tool to interactively create stories. This paper was published in Entertainment Computing, v. 5709/2009, Lecture Notes in Computer Science, Springer; and it was presented at the 8th IFIP International Conference on Entertainment Computing in September 2009.

More details on the algebra and further examples can be found in [Karlsson and Furtado 2009].

6.2.Introduction

Narratology studies distinguish three levels in literary composition: fabula, story and text [Bal 1997]. Here, we stay at the fabula level, where the acting characters are introduced, as well as the narrative plot, consisting of a partially-ordered set of events.

At least four concerns are involved in plot composition:

- the plot must be formed by a coherent sequence of events;
- for each position in the sequence, several alternative choices may apply;
- non-trivial interesting sequences must permit unexpected shifts along the way;
- one may need to go down to details to better visualize the events.

These concerns led us to identify, drawing on linguistic and semiotic work [Saussure 1967, Burke 1969, Chandler 2002], four relations between events, namely syntagmatic, paradigmatic, antithetic, and meronymic relations [Ciarlini et al. 2009]. It turns out that such relations hold as a consequence of the conventions regulating the chosen narrative genre. Therefore, a necessary preliminary step to plot composition is to provide a conceptual specification of the genre.

In our conceptual modelling approach, we focus on events that correspond to the execution of predefined operations, deliberately performed by the characters. Each operation is defined in terms of its pre-conditions and post-conditions, and

their interplay is what induces the partial order requirements for the plots, and constitutes the basis for characterizing the presence of the four kinds of relations between events.

However, a conceptual specification is still too far removed from a concrete computerized system to support plot composition. An analogous problem was successfully faced by database researchers, and, as we shall indicate, their three-stage solution can be conveniently adapted for our purposes. The key idea is to provide a logic design stage, mediating between conceptual design and physical implementation.

This was very effectively achieved by the Relational Model proposal, which introduced a relational algebra [Codd 1972] to model table manipulation at the logic level.

Codd's relational model is widely recognized today as providing effective guidance for database design at the logical level. The abstract data type on which it is based is the n-ary relation, also known as (relational) table, defined as a subset of the Cartesian product of n data domains: $R \subseteq D1 \times D2 \times ... \times Dn$. To handle database structures, Codd [1972] proposed a relational algebra sublanguage, whose operator set can be reduced, as can be easily demonstrated [Ullman and Widom 2008], to five primitives: product, projection, union, selection and difference. All operators have tables as operands and yield tables as their result. Product, union and difference are called binary operations (since they involve two operands), whereas projection and selection are unary operations (one operand). Codd's operators (with two additions to accommodate the meronymic dimension) will serve our purposes if we substitute our plot abstract data type for tuples and, consequently, sets of plots (to be called libraries) for the relational tables.

Accordingly, we propose here a logic design stage for plot composition, involving an abstract structure for plots and a Plot-Manipulation Algebra (PMA) to handle the structure, taking into due account the relations between narrative events implied by the conceptual level specification of the genre. To illustrate the discussion, as well as the design and use of a logic programming prototype tool, we employ an example involving a small number of events, which, in different combinations, have been treated repeatedly in literary works.

A very simple PMA prototype was implemented to experiment with the notions discussed here. It serves to compose plots by applying the repertoire of algebraic operations, optionally resorting to pre-defined plots and plot libraries to supply useful clues. The entire system was written in SWI-Prolog. The examples illustrated throughout the paper were all executed in the PMA prototype.

The paper is organized as follows. Sections 6.3 and 6.4 cover the background for our work. Section 6.5 presents the basic algebraic operators, illustrated by examples, a few extensions being added in section 6.6. Section 6.7 contains concluding remarks.

6.3.Basic Notions

6.3.1.Relations Between Events

To illustrate the event relations, we shall employ a simple example to be referenced along the paper. Consider four types of events, all having one woman and two men as protagonists: abduction, elopement, rescue, and capture. As demonstrated in folktale studies [Propp 1968], many plots mainly consist of an act of villainy, i.e. of a violent action that breaks the initially stable and peaceful state of affairs, followed ultimately by an action of retaliation, which may or may not lead to a happy outcome.

Syntagmatic Relations. To declare that it is legitimate to continue a plot containing abduction by placing rescue next to it, we say that these two events are connected by a syntagmatic relation. More precisely, the occurrence of the first leaves the world in a state wherein the occurrence of the second is coherent. Similarly, a plot involving elopement followed by capture looks natural, and hence these two events are likewise related. The syntagmatic relation between events induces a weak form of causality or enablement, which justifies their sequential ordering inside the plot.

Paradigmatic Relations. The events of abduction and elopement can be seen as alternative ways to accomplish a similar kind of villainy, and therefore there is a paradigmatic relation between them. Both achieve approximately the same effect: one man takes away a woman from where she is and starts to live in her company at some other place. There are differences, of course, since the woman's behaviour is normally said to be coerced in the case of abduction, but quite voluntary in the case of elopement.

The same type of relation is perceived to hold between the events of rescue and capture, which are alternative forms of retaliation.

As the present example suggests, the so-called syntagmatic and the paradigmatic axes [Saussure 1967] are really not orthogonal in that the two relations cannot be considered independently when composing a plot. Thus, in principle, the two pairs abduction-rescue and elopement-capture are the only normal ones, as illustrated in several literary works (for specific examples, cf. [Karlsson and Furtado 2009]). Yet the next type of relation shows that such limitations can, and even should, be waived occasionally.

Antithetic Relations. While normal plots can be composed exclusively on the basis of the two preceding relations, the possibility to introduce unexpected turns is often desirable in order to make the plots more attractive (cf. the notions of recognition and reversal in [Aristotle 2000]) – and this requires the construct that we chose to call antithetic relation.

A context where a woman suffers abduction by a ravisher whom she does not love would seem incompatible with a capture event. So, in this sense, abduction and capture are in antithetic relation. However, if the woman has a change of heart and falls in love with the villain (perhaps as a case of Stockholm syndrome), she will refuse to be rescued, and will only return to her previous lover if captured.

Generally speaking, if some binary opposition (e.g. "to love or not to love" dilemma) is allowed to be manipulated via some agency external to the predefined events, then one can have plots that no longer look conventional. A sort of discontinuity is produced by such radical shifts in the context. Intervening between abduction and capture, or between elopement and rescue, a sudden change of feelings can give rise to these surprising sequences. Also, a change of beliefs may cause a reversal in the course of actions, usually in a totally opposite direction. Figure 6.1 shows the relations thus far discussed.

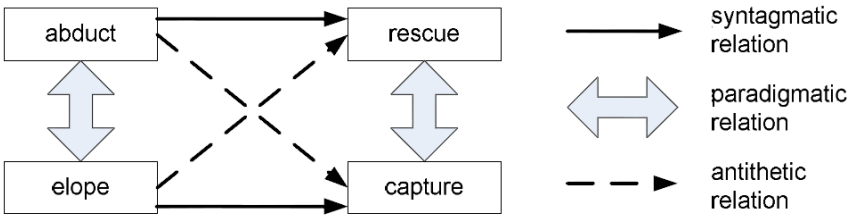

Figure 6.1: Syntagmatic, paradigmatic, and antithetic relations.

Meronymic Relations. Meronymy is a word of Greek origin, used in linguistics to refer to the decomposition of a whole into its constituent parts. Forming an adjective from this noun, we call meronymic relations those that hold between an event and a lower-level set of events, with whose help it is possible to provide a more detailed account of the action on hand.

Thus, we could describe the abduction of a woman called Sita by a man called Ravana (characters taken from the Ramayana [Valmiki 1999]) as: "Ravana rides from Lanka to forest.

Ravana seizes Sita. Ravana carries Sita to Lanka." And her rescue by Rama could take the form: "Rama rides from palace to Lanka. Rama defeats Ravana. Rama entreats Sita. Rama carries Sita to palace." (Figure 6.2).

Detailing is most useful to pass from a somewhat abstract view of the plot to one, at a more concrete physical level [Schank and Colby 1973], that is amenable

(possibly after further decomposition stages) to the production of a computer graphics animation [Ciarlini et al. 2005]. Mixed plots, combining events of different levels, also make sense, satisfying the option to represent some events more compactly while showing others in detail.

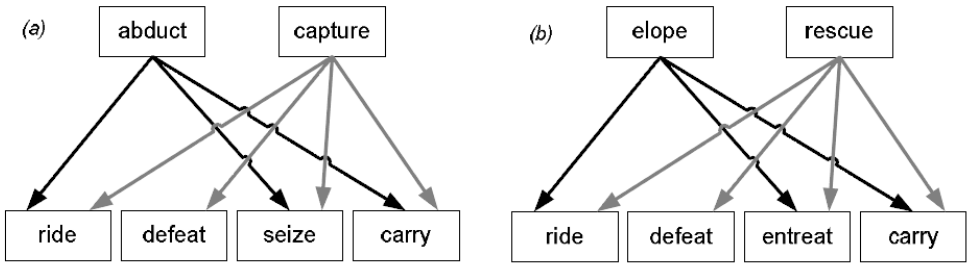

Figure 6.2: Meronymic relations: (a) the forceful actions and (b) the gentle actions.

6.4.Conceptual Specification of Genres

In order to model a chosen genre, to which the plots to be composed should belong, one must specify at least: (a) what can exist at some state of the underlying mini-world; (b) how states can be changed; and (c) the factors driving the characters to act. Accordingly, we start with a conceptual design method involving three schemas – static, dynamic and behavioural – which has been developed for modelling literary genres encompassing narratives with a high degree of regularity, such as fairy tales, and application domains of business information systems, such as banking, which are constrained by a basically inflexible set of operations and, generally, by following strict and explicitly formulated rules [Furtado et al. 2008]. Indeed, in our model, we equate the notion of event with the state change resulting from the execution of a predefined operation.

The static schema specifies, in terms of the Entity-Relationship model [Batini et al 1992], the entity and relationship classes and their attributes (cf. the notion of story databases in [Oinonen et al. 2006]). In our simple example, character and place are entities. The attributes of characters are name, which serves as identifier, and gender. Places have only one identifying attribute, `pname`. Characters are pair-wise related by relationships loves, `held_by` and `consents_with`. The last two can only hold between a female and a male character; thus `held_by(Sita,Ravana)` is a fact meaning that Sita is forcefully constrained by Ravana, whereas `consents_with(Sita,Ravana)` would indicate that Sita has voluntarily accepted Ravana's proposals. Two relationships associate characters with places: `home` and `current_place`. A state of the world consists of all facts about the existing entity instances and their properties holding at some instant.

The dynamic schema defines a fixed repertoire of operations for consistently performing state changes. The STRIPS [Fikes and Nilsson 1971] model is used. Each operation is defined in terms of pre-conditions, which consist of conjunctions of positive and negative literals, and any number of post-conditions, consisting of facts to be asserted or retracted as the effect of executing the operation. Instances of facts such as home and gender, are fixed, not being affected by any operation. Of special interest are the user-controlled facts which, though immune to operations, could be, as suggested in [Ciarlini et al. 2009], manipulated through arbitrary directives. In our example, loves is user controlled.

For the present example, we have provided operations at only two levels. The four main events are performed by level-1 operations: abduct, elope, rescue and capture.

Operations at level-2 are actions of smaller granularity, in terms of which the level-1 operations can be detailed: ride, entreat, seize, defeat, and carry. Our provisional version of the behavioural schema consists of goal-inference (a.k.a. situation-objective) rules, belief rules, and emotional condition rules. For the example, three goal-inference rules are supplied. The first one refers to the ravisher: when the princess is not at her home and the hero is not in her company – and hence she is unprotected – the ravisher will want to do whatever is adequate to bring her to his home. The other goal-inference rules refer to the hero when the ravisher has the woman in his home: either the hero believes that she does not love the other man, or he believes that she does. In both situations, he will want to bring her back, freely in the first case (rescued) and constrained in the second (captured).

Informally, beliefs correspond to the partial view, not necessarily correct, that a character currently forms about the factual context (for a formal characterization, cf. the BDI model [Cohen and Levesque 1990, Rao and Georgeff 1991]). The belief rules that we formulated for our example look rational, but notice that they are treated as defaults, which could be overruled by a directive. A man (the hero or the ravisher) believes that the woman does not love his rival if the latter has her confined, but if she has ever been observed in his company and in no occasion (state) was physically constrained, the conclusion will be that she is consenting (an attitude seemingly too subjective to be ascertained directly in a real context). The emotional condition rules refer to the three characters. A man (or woman) is happy if currently in the beloved's company, and bored otherwise.

6.4.1.The Plot Abstract Data Type

A plot P is a pair [S,D], where: S, the event-set, is a set of tagged events, and D, the dependency-set, is a set of order dependencies, expressed as tag-pairs. Tags

are terms of the form fi, where i is a positive integer. By convention, the tags in an event-set S are numbered consecutively, starting with f1, and the tagged events in S are required to be placed in some sequence compatible with the partial order requirements expressed in D. The order dependencies are determined exclusively on the basis of the satisfaction of post-conditions by pre-conditions, and any dependencies deducible by transitivity are omitted. Such conventions lead in general to simpler and more efficient algorithms to handle the data structure. As an example, consider:

```
P = [[f1: ride(Ravana, Lanka, forest),
      f2: entreat(Ravana, Sita),
      f3: seize(Ravana, Sita))], [f1-f2,f1-f3]]
```

We say that plots Pi and Pj are similar if, even with events with different parameter values and placed in a different sequence in the event-set, they have the same:

- number and type of events
- order dependencies
- co-designation/ non-co-designation schemes

Co-designation (or, respectively, non-co-designation) allows (forbids) the occurrence of the same value at different parameter positions. Note, for instance, that in the example above Sita occurs in the second position of `entreat` and of `seize`; a plot with Guinevere in both places would meet the same co-designation requirement. To check if the order dependencies are the same, one looks for a renaming of the tags of one of the plots that renders the sets of order dependencies in both plots identical. Similar plots are equal if the values at the corresponding parameter positions are identical.

As any abstract data type, plots are equipped with *operators* to handle plots along the dimensions induced by the event relations: 1. syntagmatic: product and projection; 2. paradigmatic: union and selection; 3. antithetic: difference; 4. meronymic: factoring and combination. Unary operators have higher precedence, and binary operators obey the set-theory norm. Operands are plots or libraries represented explicitly or, if predefined, by their names. As a shorthand, single-event plots admit event syntax, e.g. `[[f1:ride(Ravana,Lanka,forest)],[]]` denoted by `ride(Ravana,Lanka,forest)`

6.5.Basic Algebraic Operators

Product. Given two plots P1 = [S1,D1] and P2 = [S2,D2], their product P := P1 * P2 is a plot P = [S,D], where S contains the S1 and S2 events, and D has the dependency-pairs duly recomputed between the S events, regardless of their origin from S1 or S2.

If one or both operands are (non-empty) libraries rather than plots, the result is a library containing the product of each plot taken from the first operand with each plot from the second, according to the standard Cartesian product definition. If one of the operands is the empty plot, denoted by [], the result of the product operation is the other operand, and thus [] behaves as the neutral element for product. The case of an empty library, rather than an empty plot, demanded an implementation decision; by analogy with the zero element in the algebra of numbers, it would be justifiable to determine that a failure should result whenever an empty library occurred as one or both operands. However we decided to return the other operand as result, i.e. to regard this case as a frustrated attempt to extend plots instead of an error. This option is consistent with our decision to denote both the empty plot and the empty list by [].

Example 1. The product `P1*P2` in plot format and template-driven natural language.

```
:- P1:= carry('Ravana','Sita','Lanka') * ride('Rama', palace,'Lanka')
        *  defeat('Rama', 'Ravana'),
P2:= ride('Ravana', 'Lanka', forest) * seize('Rama','Sita') *
     carry('Rama','Sita',palace),
P := P1 * P2.
P = [[f1:ride(Ravana, Lanka, forest), f2:carry(Ravana, Sita, Lanka),
      f3:ride(Rama, palace, Lanka), f4:defeat(Rama, Ravana),
      f5:seize(Rama, Sita), f6:carry(Rama, Sita, palace)],
     [f1-f2, f2-f3, f3-f4, f4-f5, f5-f6]]
```

Ravana rides from Lanka to forest. Ravana carries Sita to Lanka. Rama rides from palace to Lanka. Rama defeats Ravana. Rama seizes Sita. Rama carries Sita to palace.

Projection. Given a plot P' = [S',D'], its projection P := proj [T] @ P' is a plot P = [S,D], where S only contains the events of S' specified in the projection-template T, ordered according to the position of the terms in T, and D only contains dependency pairs involving events placed in S. The projection-template T is a sequence of terms F:O, where F is a tag and O an event. F and O can be variables; if O is not a variable, some or all of its parameters can be variables. The S events receive new consecutive tags, starting from f1. The main use of projection is to extract passages.

If the operand is a library, the result is a library containing the projection of the plots of the operand library. Note however that, since libraries are sets, they cannot contain duplicates, which may arise as the consequence of a projection that suppresses events distinguishing two or more plots – and such duplicates are accordingly eliminated from the result. If the projection fails for some reason,

e.g. because the projection-template T referred to a tag or event that did not figure in S', the result will be the empty plot (or empty list) [] rather than an error.

Example 2. Projection can be simply used to re-order the events in a plot, in which case care must be taken to ensure that the new sequence be also viable. Given a predefined fake abduction plot, a new plot P can be obtained, preserving the original events but inverting the position of the third and fourth events. Instead of the predefined fake abduction plot, we now have a situation in which the villain initially acts as a seducer but, after bringing the princess to his home, decides to confine her.

```
fake_abduct(V,W,F) :- F := [[f1:ride(V,P1,P2),f2:entreat(V,W),
f3:seize(V,W),f4:carry(V,W,P1)],[f1-f2,f1-f3,f2-f4]].
:- P := proj [f1:_,f2:_,f4:_,f3:_] @ fake_abduct('Ravana','Sita').
P = [[f1:ride(Ravana, Lanka, forest), f2:entreat(Ravana, Sita),
      f3:carry(Ravana, Sita, Lanka), f4:seize(Ravana, Sita)],
     [f1-f2, f2-f3, f3-f4]]
```

Ravana rides from Lanka to forest. Ravana entreats Sita. Ravana carries Sita to Lanka. Ravana seizes Sita.

Union. Given two operands U1 and U2, each of them either a plot or a library, their union U := U1 + U2 will always be a library containing all plots in U1 and U2, no two equal plots being retained. One or both operands can be the empty library, ambiguously denoted as said before by [], naturally functioning as the neutral element for union. If one of the operands is a plot and the other is [], their union is a library consisting of this single plot.

Selection. Given a plot P' = [S',D'], its selection P := sel [T]/E @ P' is the plot P' itself if the matching of the selection-template T against P' succeeds, as well as the subsequent evaluation of the logical expression E (whose presence is optional, except if T is empty), also involving information taken from P'. If the test fails, the result to be assigned to P is the empty library []. Usually the operand of selection will be a library, resulting in a library containing all plots that satisfy the test, or the empty library [] if none does. In order to select one plot at a time from a library L, even if L just contains a single plot, the form P := sel [T]/E @ one(L) should be employed.

Example 3. Consider a plot P that starts with abduction, and ought to continue with an adequate form of retaliation. Choosing this second event can be done by performing a (conditional) union of the two possible alternatives, making the effective presence of each of them depend on the recurring dilemma: the hero can only rescue the woman if she loves him, otherwise he must capture her. Notice that this scheme works as an exclusive-or or as an if-then-else. Assume that Sita currently loves Rama.

```
:- P := abduct(V,W) * ((sel []/loves(W,H) @ rescue(H,W)) +
        (sel []/(not loves(W,H)) @ capture(H,W))).
P = [[[f1:abduct(Ravana, Sita), f2:rescue(Rama, Sita)],
      [f1-f2]]]
```

Ravana abducts Sita. Rama rescues Sita.

Difference. Given two operands U1 and U2, each of them either a plot or a library, their difference U := U1 – U2 will always be a library containing all plots in U1 that are not equal to any plot in U2. As happens with standard set difference, the result of the operation is not affected by plots in U2 that have no equals in U1. On the other hand if all plots in U1 have their equals in U2, the empty list [] is assigned to U.

Example 4. The evaluation of plot P below illustrates the interplay of the syntagmatic, paradigmatic and antithetic event relations. The union operation yields a pair of libraries, consisting of the alternatives, respectively, for villainy and for retaliation. All villainy-retaliation sequences are then formed by the product of the two libraries. Finally, by difference, the transgressive sequence elope-rescue is excluded.

```
:- P := (abduct(V,W) + elope(V,W)) * (rescue(H,W) +
capture(H,W)) - elope(V,W) * rescue(H,W).
P = [[[f1:abduct(Ravana, Sita), f2:rescue(Rama, Sita)], [f1-f2]],
[[f1:abduct(Ravana, Sita), f2:capture(Rama, Sita)], [f1-f2]],
[[f1:elope(Ravana, Sita), f2:capture(Rama, Sita)], [f1-f2]]]
```

Ravana abducts Sita. Rama rescues Sita.

Ravana abducts Sita. Rama captures Sita.

Ravana elopes with Sita. Rama captures Sita.

Factoring. Given a plot P' = [S',D'], its factoring P := fac P' is a plot P = [S,D], where each level-1 event ei' present in S' is replaced by a sequence ei1, ei2,..., ein of level-2 events. Each resulting sequence is obtained from a predefined map declaration map(Ei',[Ei1,Ei2,...,Ein]), such that Ei' matches ei'. In map declarations, all terms in both arguments are events, and may contain variables at the parameter positions. The first argument must be a level-1 event and the second a sequence of level-2 events.

When specifying a map declaration for an event Ei', care should be taken that the indicated sequence of level-2 operations should work as a plan successfully applicable at world situations satisfying the pre-conditions of Ei', and producing at the end the effects expressed by the post-conditions of Ei'. And if the sequence may have other (secondary) effects, these must not contradict those expected from executing Ei'.

As a map declaration for Ei' is found to match an event ei' in the course of factoring, all variables in Ei' will be instantiated with the values contained in the respective parameter positions of ei', with the consequence that several parameters of the level-2 events will also be instantiated by consistent variable substitution. In most cases, however, several level-2 events will not become fully ground terms. In order to further instantiate the level-2 event parameters, we decided to apply a heuristic process based on the pre-condition declarations of these events. As said before, some database facts of the mini-world may be invariant, in the specific sense that none of the events provided may change them, an example being the gender of the acting characters. So, if e.g. there is only one female character, and the pre-condition of an event requires a female at a certain parameter position, it seems natural to assign that character's name to the corresponding variable. If the operand is a library, the result is a library with the factoring of all the operand's plots.

Combination. Given a plot P' = [S',D'], its combination P := comb P' is a plot P = [S,D], where each sequence ei1', ei2',...,ein' of level-2 events present in S' is replaced by a level-1 event obtained by the inverse application of the pre-defined map declaration whose second argument happens to match the sequence. If the operand is a library, the result is a library containing all plots in the operand library with the eventual modifications performed by applying the combination operation.

Example 5. Using selection over the detailed description of the two level-1 villainy events, obtained through the factoring operation, we may determine which one involves the violent `seize` level-2 event. The desired level-1 event is reconstituted from the selected level-2 description by applying the combination operator at the end.

```
:- P := comb sel [_:seize(_,_)] @ (fac (abduct(V,W) +
        elope(V,W))).
P = [[[f1:abduct(Ravana, Sita)], []]]
```

Ravana abducts Sita.

6.6.Extensions

Two features extend the product operator, to achieve repeated plot sequences.

Given a plot P' = [S',D'], the nth power of P', expressed by P := P' ** N, for a nonnegative integer N, is evaluated according to the recursive formula:

- `if N = 0, P = []`
- `else P = P' * (P' ** (N - 1))`

Given a plot P' = [S',D'], the iteration of P', expressed by P := E@P', where E is a logical expression sharing any number of variables with P', is evaluated as follows:

- first, the iterator-template T is obtained, as the set of all possible instantiations of E at the initial state, and then:
- `if T is {}, P = []`
- `else, if T = {t1, t2, ..., tn}, P = P't1 * P'{t2, ..., tn}`

where P'ti denotes P' with its variables instantiated consistently with those in ti, and the subscript in P'{ti+1, ..., tn} refers to the remaining instantiations of T to be used at the next stages. As with product, both features apply to single plots and plot libraries.

Example 6. Sequences of actions with the same objective are often repeatedly played by different characters, especially in folktales [Aarne and Thompson 1961]. Here, iteration is applied to make the two men successively attempt to persuade Sita, taking her feelings with respect to them into consideration when deciding how to act. To a man loved by Sita it is enough to entreat her once, whereas an unloved man might try that twice, finally resorting to seizing the reluctant princess.

```
:- P := iter (gender(M,male),home(M,H),gender(W,female)) @
     ( (sel []/(loves(W,M))@
        (ride(M,H,forest) * entreat(M,W))) +
       (sel []/(not loves(W,M)) @
        (ride(M,H,forest) * (entreat(M,W)**2 * seize(M,W)))) ).
```

Suppose Sita loves the hero, Rama, but not the villain, Ravana. Then the logical expression controlling the iteration operator will evaluate to the iterator-template T as shown below. Note that T is represented as a list with two items indicating, as required, different instantiations for the variables.

```
T = [ (M=Rama, H=palace, W=Sita), (M=Ravana, H=Lanka, W=Sita) ]
P = [[f1:ride(Rama, palace, forest), f2:entreat(Rama, Sita),
      f3:ride(Ravana, Lanka, forest), f4:entreat(Ravana, Sita),
      f5:entreat(Ravana, Sita), f6:seize(Ravana,Sita)],
     [f1-f2, f3-f4, f3-f5, f4-f6, f5-f6]]
```

Rama rides from palace to forest. Rama entreats Sita. Ravana rides from Lanka to forest. Ravana entreats Sita. Ravana entreats Sita. Ravana seizes Sita.

The fact that the current version of PMA is embedded in logic programming facilitated the introduction of plot patterns, allowing selection from libraries whose plots have events pertaining to the genre on hand, but are defined over different parameter values (e.g. names of persons, places, etc.). This is a first step

towards the reuse of repositories of typical narratives collected from diverse sources (for folktale narratives, see for instance [1]), so as to exploit imitation as a composition resource.

Given a plot P = [S,D], the pattern P of P, expressed by P := patt P, is obtained from P by substituting variables for the parameters and tags in both S and D. As always, patterning applies to both single plots and to entire libraries. To take advantage of this feature, we provided an additional version of the selection operator, in which a plot can be used as selection-template. Let P' = [S',D'] be a plot at an early phase of composition, still with very few events, and let L be a library with the characteristics above. If one wishes to extend P' to a set of fuller alternative plots containing all events in S' and preserving the order requirements imposed by D', it is possible to adapt the typical narratives in L by way of a pattern-matching technique.

This is accomplished by evaluating the expression P := sel P'@ (patt L), wherein plot P' guides a selection against the result of converting all plots in L into patterns.

Example 7. Library `lib_2`, displayed below, is used to extend an initial plot Pi.

```
lib_2([ [[f1:ride('Meleagant','Gore',forest),
          f2:seize('Meleagant','Guinevere'),
          f3:carry('Meleagant','Guinevere','Gore'),
          f4:ride('Lancelot','Camelot','Gore'),
          f5:defeat('Lancelot','Meleagant'),
          f6:entreat('Lancelot','Guinevere'),
          f7:carry('Lancelot','Guinevere','Camelot')],
         [f1-f2,f2-f3,f3-f4,f4-f5,f5-f6,f6-f7]],
        [[f1:ride('Tristan',forest,garden),
          f2:entreat('Tristan','Isolde'),
          f3:carry('Tristan','Isolde',forest),
          f4:ride('Mark','Cornwall',forest),
          f5:defeat('Mark','Tristan'),
          f6:seize('Mark','Isolde'),
          f7:carry('Mark','Isolde','Cornwall')],
         [f1-f2,f2-f3,f3-f4,f4-f5,f5-f6,f6-f7]] ]).
```

If Pi is formulated as shown next, the Meleagant plot will serve as model to obtain the extended plot Pe. Notice the later application of combination to Pe for recognizing the kind of narrative obtained, which turns out to be of the abduct-rescue variety.

```
:- Pi = [[f1:carry('Ravana','Sita','Lanka'),f2:entreat('Rama','Sita'),
          f3:carry('Rama','Sita',palace)], [f1-f2,f2-f3]],
```

Ravana carries Sita to Lanka. Rama entreats Sita. Rama carries Sita to palace.

```
:- Pe := sel Pi @ (patt lib_2),
:- Pc := comb Pe.
Pe = [[[f1:ride(Ravana, Lanka, forest), f2:seize(Ravana, Sita),
        f3:carry(Ravana, Sita, Lanka),
        f4:ride(Rama, palace, Lanka),
        f5:defeat(Rama, Ravana), f6:entreat(Rama, Sita),
        f7:carry(Rama, Sita, palace)],
       [f1-f2, f2-f3, f3-f4, f4-f5, f5-f6, f6-f7]]]
```

Ravana rides from Lanka to forest. Ravana seizes Sita. Ravana carries Sita to Lanka. Rama rides from palace to Lanka. Rama defeats Ravana. Rama entreats Sita. Rama carries Sita to palace.

```
Pc = [[[f1:abduct(Ravana, Sita), f2:rescue(Rama, Sita)],
       [f1-f2]]]
```

Ravana abducts Sita. Rama rescues Sita.

A different result would be obtained by applying, to the same library, a different initial plot Pi diverging from the previous formulation with respect to the second event (`seize`, instead of `entreat`). With this the Tristan plot would be taken as model, resulting in a plot of the elope-capture variety.

6.7.Concluding Remarks

The contribution expected from a logic model is to provide reliable guidelines to develop systems that may be regarded as "complete" according to some criterion. We claim that PMA is complete in the specific sense that it covers plot manipulation along the dimensions induced by the syntagmatic, paradigmatic, antithetic and meronymic event relations. These relations encompass some fundamental aspects of plot composition, and are respectively associated, as argued in [Ciarlini et al. 2009], with the *four master tropes* (metonymy, metaphor, irony, and synecdoche) of semiotic research [Burke 1969, Chandler 2002].

Of primary significance is the syntagmatic axis, along which plots are created by the **product** operator, whereby events or event-sequences are chained together to form progressively longer sequences. Inversely, a passage of interest can be extracted from a plot by the **projection** operator, possibly re-ordering the events while preserving the partial order. Notice in particular that the familiar cut-and-paste tactic corresponds to projections followed by product. Along the paradigmatic axis, the **union** operator offers different alternatives for certain

positions in a sequence. Libraries, as sets of optional versions of narratives, are collected and expanded by applying union.

Choosing among alternatives is the objective of the **selection** operator. It allows to check if a given plot has (or which plots in a library have) the desired characteristics.

The antithetic constraints, originating from binary oppositions in the mini-world context [Chandler 2002], require the **difference** operator, whose basic purpose is to exclude unwanted plots from a library collection. Curiously, what is negated once may, on second thought, suggest exciting directions for continuing the narrative, as posited in literary studies on the irony trope [Booth 1974] and on the notion of *deconstruction* [Culler 1983]. Such unexpected turns usually require that current facts and beliefs be changed, so that the next events may seem to *blend* [Fauconnier and Turner 2002] naturally with the sequence so far composed.

Finally, the concern with the meronymic dimension led to the inclusion of the factoring and combination operators, allowing, respectively, to detail or summarize the events. The ability to tune the level of granularity of events is vital, on the other hand, to prepare for the transition to the subsequent stages of story and text [Bal 1997].

Theoretic work is needed to systematically investigate the formal properties of the algebra. Practical work, drawing from our early experiments with the PlotBoard prototype [Ciarlini et al. 2009] and from differently oriented implementations [Aylett et al. 2006, Cavazza et al. 2002, Szilas 2003], will include the design of systems based on PMA, taking maximum advantage of the previously developed plan-generation facilities, but also allowing effective user interaction along a step-wise plot composition and adaptation process through a friendly interface. Any such system should have access to an online representation of the conceptual schemas, using this meta-level information to keep checking the *semantic correction* of the plots being generated. Moreover, access to the behavioural schema in particular should serve to verify what might be called *pragmatic plausibility*, i.e. whether the events caused by each character reflect their expected way of reacting, especially in view of the declared situation-goal rules.