On the Evolving Transformation System Model Representation of Fairy Tales

by

Sean Falconer
BCS, University of New Brunswick, Canada, 2003

A Thesis , Dissertion or Report Submitted in Partial Fulfilment of
the Requirements for the Degree of

Master of Computer Science

in the Graduate Academic Unit of Computer Science

Supervisor(s):	Goldfarb, Lev, Diploma Math/CS (Leningrad), PhD (Waterloo), Computer Science
Examining Board:	Boley, Harold, MSc, PhD (Hamburg), Adjunct Professor, UNB, NRC, Chair Fritz, Jane, BSc (McGill), MScCS (UNB), PhD (York, UK), Computer Science

This dissertion, thesis or report is accepted by the

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

May, 2005

© Sean Falconer, 2005

Abstract

This thesis presents an initial application of the Evolving Transformation System (ETS) formalism to the structural representation of Russian fairy tales. This work was motivated by the needs of information retrieval (IR) in general, but specifically concentrates on proposing a new type of document representation. The new representation is based on the temporal or constructive process of events that occur in the mind of the "generic" fairy tale listener/reader.

Representing documents in this way yields several advantages to using ETS over traditional IR techniques, such as being able to retrieve partial information from a document (i.e. a single sentence or paragraph), fast searching by traversing the multi-level hierarchial class structure generated by the ETS *intelligent process*, and locating similar documents based on structural similarity rather than key words. Moreover, the ETS formalism suggests an explanation for the mechanism behind plot anticipation by an experienced reader as well as various levels of fairy tale conceptualization.

Using the ETS formalism and in-depth analysis of many fairy tales, we propose approximately 40 ETS primitives that make up the underlying structure of events for

any Russian fairy tale. Using these primitive events, we develop the structural representation of several story segments belonging to the same class, which we then use to identify ETS *transformations*. Using these transformations, we develop the *class representation* and demonstrate the ETS multi-level representation for this class of fairy tale segments.

Acknowledgements

Firstly, I thank my supervisor, Dr. Lev Goldfarb, for all his support and guidance throughout the development of my thesis work. I also thank him for taking such an interest in my education and in my ideas.

I would also like to thank the entire ETS Group, in particular, David Gay, Oleg Golubitsky, Muhammad Al-Digeil, and Alexander Gutkin. I especially thank Oleg Golubitsky for taking the time to introduce me to his work and the work being done by the ETS Group. Also, I especially thank David Gay for his help with understanding the ETS concepts and his help during various brainstorming sessions.

Thank you to the UNB Computer Science faculty for their support. Also thank you to all the professors that have instructed me through both my undergraduate and graduate degrees.

Thanks to the members of my examining committee, Dr. Jane Fritz, Dr. Harold Boley, and Dr. Yevgen Biletskiy for their helpful comments and suggestions.

Finally, I'd like to thank my wife Theresa for her unyielding support and encouragement throughout my academic career.

Contents

List of Figures

Chapter 1

Introduction

Since the mass introduction of computers to the public, people have been waiting for the day when a computer will "know" enough, so that it becomes easy to use. Specifically, in information retrieval (IR), we want a smart computer to be able to really know what it is searching for when it searches its memory [39]. It should be able to search its memory using natural language, that is, the language we use in everyday conversations. It should be able to understand the difference, based on context, when we are searching for apple the fruit versus Apple the company. Finally, and perhaps most importantly, it should be able to understand our search criteria well enough so that it can retrieve what we want, rather than just what we say. However, after many years of research, we still have not been able to fully address these issues or answer many fundamental questions about language: How do we represent language? How did language evolve? What parts of language are innate? How do children acquire language? and How do we represent real world knowledge?

Most researchers in IR have concentrated on document retrieval, rather than the

problems mentioned above [39]. This is somewhat understandable, as questions about the nature of language representation are some of the most challenging questions facing IR and language researchers today. Because of this difficulty, most researchers have avoided addressing the issue by using ready-made statistical approaches based on key word schemes. The emphasis of their work is on the classification of queries to retrieve relevant documents, rather than on the representation of the data they are searching, therefore, information is stored independent of meaning. However, without a meaningful representation of the data a user wishes to search, it is not possible to intelligently address the query. Moreover, Mandler states that [29, p. 20] "... meaning does not exist until some structure is achieved and the case can be made that the deeper the understanding of a domain the more abstract the structure that has been uncovered or imposed." It is this point that sets this thesis work apart from much of the work currently being done in IR. In this thesis we are concerned with the data representation, that is, how can we best organize and represent our data such that the meaning of the data is structured in such a way that we can easily communicate with it?

1.1 Motivation

This thesis introduces a preliminary IR application based on the Evolving Transformation System (ETS) formalism [16]. For this preliminary application we used Russian fairy tales as our data set. We attempted to address all issues surrounding the problem of how one retrieves all relevant fairy tale segments based on input as some query; however, as ETS is a new formalism, and as we are attempting to address

the fundamental issue of data representation in IR, the concentration of the thesis is on the representation of the fairy tales.

We chose the domain of fairy tales because it appears that both the structure and semantics of fairy tales as "documents"[1] are more pristine than those of many modern documents (e.g. newspaper articles, web pages, and other documents). One of the reasons why fairy tales are more enduring and more universal than many present day documents is that their structure is less contaminated by various non-essential expository elements so prevalent in modern documents. For example, a typical North American newspaper article is full of insignificant details appearing almost randomly in the text. In fairy tales, every event, character, and object is important to the development of the plot. For instance, if the hero of the story picks up a rock while travelling, the hero will participate in an event involving that rock later in the story. Since all structural elements of fairy tales are semantically meaningful, they provide a very good database for the investigation of various formalisms for information retrieval (IR). Similar observations have been made by other researchers [18, 29].

1.2 Background

Currently, by far the dominant representational formalism in IR is the vector-space-based formalism, that is, it is the choice formalism when it comes to the classification of documents. What are the main deficiencies of this classification formalism? They have to do with the inability of the vector-space formalism to deal with the fluent nature of most documents. Most documents can only be understood if the sequence of

[1]The term "document" is used in this paper as a generic name for a record in a generic information retrieval database.

interconnected (often implicit) "events"—which are encoded in these documents—is properly perceived: e.g. for a typical professor's webpage, these events could be the awarding of a PhD, courses taught, grants received, paper's published, conferences chaired, etc. The fluent nature of interconnected events in documents manifests the following two facts.

The first fact has to do with one's inability to enumerate, *in advance*, all possible *interrelationships* that could exist between various events or features. This fact implies, in particular, that one cannot deal reliably with the classification of new documents involving new interrelationships between the events. This results in the brittleness of the classification system: new interrelationships present in the document to be classified often lead to misclassification of this document.

The second fact is related to the first one and has to do with the resulting necessity to deal with very high dimensional space (which still cannot "capture" all *possible* interrelationships). The implication of this fact is that one must now deal with enormous computational complexity of the algorithms involved.

ETS is the first formalism specifically designed to address structural representation in a more universal setting and with emphasis on classes of objects/events as the main underlying concepts [2]. As was mentioned in [16], "[o]ne can develop an initial intuitive understanding of the proposed [ETS] representations by simply generalizing the process of construction/generation of natural numbers: replace the identical structureless 'primitives' out of which numbers are build by various structural ones." Also,

[2]In what follows below, we will be using terminology introduced in [16].

> [t]he concept of class representation—which inspired and directed the development of this formalism—differs radically from the known concepts of class. Indeed, the evolving transformation system ... is the first one developed to support that concept; a class representation is a finite set of weighted and interrelated transformations ("structural segments"), out of which class elements are built.
>
> The formalism [also] allows for a *very natural* introduction of representational levels: a next-level unit [i.e. primitive] corresponds to a class representation at the previous level. [16]

In this thesis, we demonstrate that the ETS representational levels correspond and clarify the levels of comprehension of the fairy tale reader.

In contrast to the vector-space-based formalism, the combinatorial explosion of various structural relationships is handled within ETS in a natural way: it is the class description, i.e. class supertransform [3], that absorbs the combinatorial explosion of the corresponding vector-space "features". The class supertransform is composed of constituent transforms, which are, roughly speaking, *closely related* structural segments in the object representation. In other words, the constituent transformations encapsulate all inductive structural variation that one encounters in the document segments.

One of the most important features of the ETS formalism is that it erases the differences between the syntax and semantics of the objects representation. Thus, for each primitive, its syntax and semantics are indistinguishable and in view of the structure

[3]The class supertransform is constructed on the basis of the supertransform (see Chapter 3, section 4) by abstracting away the supertransform's site labels.

of the formalism, this feature carries over to all of its concepts, i.e. to *structs*, *transforms*, and other levels of representation. We believe this to be the first formalism that possesses this feature.

1.3 Organization of the thesis

In Chapter 2, I discuss previous work done in IR and language representation. I begin by discussing language representation in general (section 2.1), then I review work done on story understanding systems and story representation (section 2.2), and finally discuss the most popular approach to IR and representation, the vector-based-space formalism (section 2.3). In each of the above mentioned sections, I review the current work, and critically evaluate each approach.

The previous work section leads into Chapter 3 where I discuss the basics of the ETS formalism. I first discuss some of the philosphical and motivational issues of the formalism (section 3.1). Then, using non-formal language, I introduce all the fundamental concepts of ETS, i.e. primitives, structs, transformations, supertransformations, and class supertransformations. I discuss the multi-level structure of ETS (section 3.5). Finally, I end the discussion by briefly introducing the intelligent process (section 3.6), which is an unsupervised inductive learning algorithm.

After the introduction of the basic principles of ETS in Chapter 3, I introduce, in Chapter 4, the ETS concepts in the context of this application, that is the representation of fairy tales. First, I discuss some general ideas about story structure, then introduce the initial level ETS fairy tale primitives (section 4.2). I then give

example ETS structs (section 4.3) for several fairy tale segments. Based on the example structs, I extract several transformations and demonstrate their corresponding supertransformations (section 4.4). Then, using the supertransformations, I introduce second level primitives (section 4.5) and give an example of a second level struct representation of a fairy tale segment (section 4.6). I show two example second level transformations based on the second level representations of the fairy tale segments (section 4.7). I then discuss the validity of the proposed document representation (section 4.8). I then compare this representation with previously introduced ideas about story representation from Chapter 2 (section 4.9). Finally, I discuss some limitations of the proposed representation, and how this could possibly be improved.

In Chapter 5, I discuss the fairy tale retrieval system. First, I discuss the preprocessing system to automatically extract the ETS primitives from the fairy tale segments (section 5.1). I begin this discussion by introducing the area of *information extraction*, then I discuss current tools in this area, and finally discuss an experiment where I use one of these tools to extract some primitives from a data set of fairy tales. After this, I discuss the construction of the IR system based on the preprocessor and ETS representation (section 5.2). Finally, I discuss some advantages that this system provides over traditional IR techniques (section 5.3).

Finally, in Chapter 6, I end the thesis with conclusions and a discussion about future directions and work. I also briefly discuss why learning is not covered in the thesis, and what deficiencies of the current working version of ETS were discovered through this application.

Chapter 2

Previous work

2.1 Language representation

Generally, there are two philosophical approaches to linguistics, the *rationalist* approach and *empiricist* approach. Between 1960 to 1985, linguistics was mostly dominated by the rationalist, where a rationalist believes that a significant part of knowledge in the mind is not derived by the senses but is fixed in advance [28, p. 4]. The reason for this dominance was partly due to the acceptance of Chomsky's arguments about an innate language faculty. An Natural Language Processing (NLP) researcher may apply this philosophy by hard coding various knowledge and rules into their system, and defend this by claiming that humans are born with similar knowledge and rules.

Conversely, the empiricist does not believe that humans are born with such knowledge coming hardwired; instead, they believe that we are born with general operations for

association, pattern recognition, and generalization, and that these can be applied to the sensory input available to the child to learn the detailed structure of natural language [28, p. 5]. The empiricist philosophy was most popular from the 1920s to 1960 and is now gaining popularity again. In NLP, the empiricist approach would be to specify a general language model, and then, based on that model and some pattern recognition learning model, the complicated structure of language could be learned.

Linguistics, as a science, has primarily been concerned with the formal syntax of language rather than with the semantics. This is because some linguists felt that semantics or meaning could not be studied with scientific rigour in the same way as linguistic sounds and forms [42, p. 3]. However, today, many linguists are attempting to shift the field of linguistics back towards dealing with semantic issues. These particular linguists feel that "... meaning underlies language, not the other way around." [42, p. 21], therefore, in order to understand language, we must begin with meaning.

One such approach to dealing with meaning is the theory of semantic primitives. These *semantic primitives* correspond to the basic elements in any language that cannot be defined, and all complex meanings can be represented by these primitives. Linguist Anna Wierzbicka and her group have been studying various languages over the past 30 plus years in order to discover these primitives. They have been able to locate, through in-depth analysis, around 55 semantic primitives that they believe match across all human languages. Their hypothesis is that the semantic primitives for each language are just one language-specific manifestation of a universal set of *fundamental human concepts* [42, p. 13]. This hypothesis is based on the assumption or belief that fundamental human concepts are innate, i.e. we are born with some abstract representation of certain concepts, which we eventually map to labels/words

as we acquire language.

This innate representation corresponds to what they term as a *natural semantic metalanguage*. This metalanguage is universal and language-independent. She states, "the shared core of all languages can be seen as a set of isomorphic minilanguages, which can be used as language-specific versions of the same, universal Natural Semantic Metalanguage (NSM)" [42, p. 22]. Moreover, she believes that these semantic primitives can be combined to express meaning:

> ...to say anything meaningful we need more than words: we need sentences in which words are meaningfully put together. Similarly, to think something we need more than "concepts": we need meaningful combinations of concepts. [42, p. 19]

Similarly, Roger Schank believed that the mental representation of language is made up of interconnected concepts, where each concept is dependent on some other concept [35]. Moreover, in order to learn language it is necessary to learn the model of the world that underlies language [36]. He developed Conceptual Dependency (CD) theory based on this philosophy. The goal of CD theory was to enable a computer program to interact with a human in natural language.

Furthermore, CD theory was based on Schank's belief that it is possible to represent a great part of the meanings underlying natural language by use of a conceptual representation schema that includes only fourteen basic actions, an infinite set of objects, and a small number of states, in addition to about sixteen rules governing the combination of these items [36]. These 14 actions made up the CD theory's set of primitives. Schank described his notion of an action as an actual action that can

be performed on some object by an actor. As an example, consider his three primitives based on the abstract notion of transfer: ATRANS, MTRANS, and PTRANS. ATRANS represents an abstract transfer such as possession of an object. MTRANS is mental transfer such as the flow of information from one individual to another. PTRANS is a physical transfer such as an individual changing location. A sentence's representation would then be constructed based on the action primitives and on a semantic network representation (see Figure 2.1).

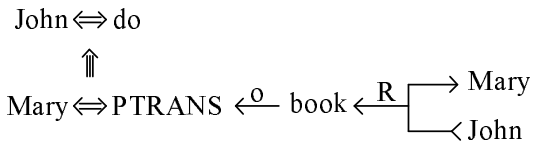


Figure 2.1: Example of CD representation of the sentence "John gave Mary the book." [36] The arrows or "causal arrows", as Schank refers to them, represent two-way dependencies. Different types of arrows represent different types of relationships, such as between actor and action, actor and object, and object-state complexes. The "do" represents an unknown action, and the arrows indicate that this action involves the two actors John and Mary. This action is unknown as we don't know how Mary received the book. The PTRANS primitive represents the physical transfer of the book to Mary. Finally, the complex symbol on the right involving John, Mary, and book represent a change in the object-state of book, that is Mary becomes the recipient of the book.

Schank produced a lot of work in the 1970s based on this theory. He even performed experiments with young children in order to show evidence that humans are born with knowledge of these fourteen basic actions. Also, he and his colleagues developed several computer systems that used the CD theory.

2.2 Story understanding

Story understanding is a classic problem in Artificial Intelligence (AI). This is because it involves several components: parsing, representation, cognitive-modelling, searching, and classification. From an IR perspective, we are primarily concerned with representation, searching, and classification. Ideally, all of these components would be unified by a single model; however, for most of the history of this problem, each component is handled somewhat independently, and there has been no general model introduced. Various structural models such as story schemas, story grammars, and affect states exist for story representation and use in story understanding systems, while the classification of stories or documents has been dominated by statistical vector-based approaches.

Below, I begin by discussing work done by Vladimir Propp on the analysis and classification of fairy tales. His work, primarily done during the early 1900s, is important as it was the inspiration for many researcher's story understanding models. Also, his philosophies about representation and classification coincide with ETS. After this brief introduction to his work, I discuss some of the approaches story understanding systems have used for the representation of stories.

2.2.1 Vladimir Propp

Propp believed that the study and classification of fairy tales could be carried out with the same type of scientific rigor as biological taxonomy. He made several interesting observations and suggestions about fairy tale representation. Firstly, he proposed to view a fairy tale via a sequence of events, where the structure of each

event should be independent as much as possible of a particular actor or the details involved. Second, he insisted on *structural* classification via *structural similarity* of fairy tales (and their segments) as the main framework in which one should think about the "representation" [1] of fairy tales. Third, in modern terminology, he insisted on structural "representation" as compared to other forms of representation and strongly criticized non-structural approaches and their classification. He also insisted on structural "composition" as the basis for classification, i.e. "[d]ifferent plots can have the same composition" [32, p. 41]. In other words, the structural composition can be used as the description of the class of related fairy tales.

Propp, in an effort to approach the classification of fairy tales using the above mentioned guidelines, performed an in-depth study of 100 Russian fairy tales. Based on this study he proposed 31 functions, each denoting "the action of the character from the point of view of its significance for the progress of the narrative" [32, p. 74].[2]. He imposed an order on his 31 functions, meaning that although certain functions may be missing from a particular story, the functions always appeared in a specified order. Propp's work also inspired work by Claude Levi-Strauss in his study of the structure of myth [27].

It is interesting to note two points about Propp's functions and ideas. Firstly, I believe most of his 31 functions are too generic to be used for classification purposes, i.e. the identification of one of his functions appearing in a story can be ambiguous. I think each function can be decomposed into smaller more concrete actions and I believe that the primitives I introduce in Chapter 4 can be combined to represent

[1] Here, we are interpreting him in IR language.

[2] "The content of the task varies, but the presence of a task is something stable. I called such stable elements the functions of the characters." [32, p. 73]

any of Propp's functions. Secondly, although Propp criticized many people's work on classification of fairy tales, he himself was never able to propose classes in which he was satisfied with. I think this is partly due to Propp's lack of formal training; he was not able to properly formulate his ideas about structure and classes into a mathematical model that would guide his selection and representation of classes.

2.2.2 Schema-based approach

The beginning of psychological and computational study of story understanding is generally credited with Bartlett [40]. He conducted a study to show how schemas are used during the recall of a story. The term *schema* comes from psychologists, where many use the term to refer to the basic building block of cognition [29, p. 2]. Many AI researchers have used the schema-based approach as a basis for their own models of representation, for instance, Minsky's frames and Schank and Abelson's scripts.

Minsky introduced frames in [30] as a psychologically valid data structure for storing information about 'things' in the world [40]. For example, a particular frame may represent the situation of going to a convenience store. A frame consists of nodes called "slots", which represent certain default information about the frame; for instance, in our example a slot may be that the convenience store sells gas. However, for a particular instance of the convenience store frame, it could be the case that this default information is overridden because this particular store does not sell gas. Minsky introduced this idea of a frame by hypothesizing it was a general and unified model for intelligence. He was criticizing current attempts in AI and psychology as "too

minute, local, and unstructured to account–either practically or phenomenologically–for the effectiveness of common-sense thought." [30] His paper, "A Framework for Representing Knowledge", raises more questions than answers and attacks existing models of thought representation.

Similar to frames, *scripts* were introduced by Schank and Abelson to characterize knowledge about a familiar event or sequence; thus, scripts are actually an elaboration of the concept of frame [29, p. 75]. Scripts were introduced specifically for dealing with story understanding, and Schank and Abelson developed several story understanding systems in the 1970s based on this idea, along with the CD theory previously discussed.

The first such system Schank and his students developed was called MARGIE (Memory, Analysis, and Response Generation in English) [37]. MARGIE was mostly a toy developed to test some of their theories about language and it was based on the following two assumptions. The first is the *Primitive Decomposition Hypothesis*, which states that for any two sentences of identical meaning, in any language, a single underlying symbolic representation can be assigned, composed of structures encoded in terms of a relatively small set of "primitive elements." The second is the *Understanding as Spontaneous Inference Hypothesis*, which states that the process of understanding is at least partly that of computing the inferences that follow, in an asynchronous, forward-chaining manner, from a conceptual form representing the meaning of a sentence [10, p. 6].

Later, Schank and his students created SAM (Script Applier Mechanism) [38]. Also, around the same time several other story understanding systems based on the schema approach were developed: Ms. Malaprop [7], PAM [43], IPP [25], and CADHELP [9].

Although there have been some interesting attempts at developing story understanding systems based on story schemas, there are several limitations intrinsic in any of the variations of schemas, i.e. frames and scripts. Firstly, story schemas have very little hierarchial level organization. Most schema-based approaches store all information about a story on a single level. However, this contrasts psychological evidence as well as common sense which suggests that stories consist of many sub-structures, where one sub-structure may be subordinate to others [40]. Secondly, schemas are a very general idea about the organization of information that comes from psychology; there is no formal schema model. This is a limitation because there is no model that constrains one's interpretation about how to use a schema, there is no formal language for representing something as a schema, and there are no guiding principles about how to classify information represented as a schema. Moreover, schemas are so general that one could use them in any arbitrary fashion.

Finally, all schema-based approaches suffer from a well-known issue called "The Frame Problem" [19]. The *Frame Problem* is the challenge of representing the effects of actions without explicitly representing a large number of intuitively obvious non-effects. This problem is present in many forms of representation, but specifically, all frame-based approaches assume certain default information being known before-hand, which leads to this problem. How can any frame-based approach fully represent the dynamic and fluent nature of the world? This is related to the *Quantification Problem*, which is, how can anyone choose the right number of things without formalizing everything? Finally, there is the *Ramification Problem*, which states that often formalized models leave out the unexpected, that is, unexpected interactions can take place, and the model cannot represent them. These are all very serious and well studied problems.

Moreover, due to these deficiencies, which have been, for the most part, intrinsic in all symbolic models, many people see statistical models as a way to avoid these issues.

2.2.3 Story grammars

Story grammars are related to story schemas, however, schemas represent mental constructs in a story while story grammars describe structure. "The contention of all story grammars is that stories have an underlying, or base, structure that remains relatively invariant in spite of gross differences in content from story to story." [29, p. 20]

Story grammars were inspired by Propp's work on fairy tales. They use Propp's notion of structure along with traditional grammar notation to encapsulate the structure of a story. For instance, a simple example grammar is shown below:

$$
\begin{array}{lcl}
S & \leftarrow & NP\ VP \\
VP & \leftarrow & VERB\ NP \\
NP & \leftarrow & NAME \\
NP & \leftarrow & ARTICLE\ NOUN
\end{array}
$$

Figure 2.2: An example grammar [3].

Each line from Figure 2.2.3 represents a rule. The first rule, $S \leftarrow NP\ VP$, means that S may consist of an NP (noun phrase) followed by a VP (verb phrase); a VP consists of a $VERB$ followed by an NP; an NP can consist of a $NAME$ or an $ARTICLE$ followed by a $NOUN$. Symbols on the right-hand side of a rule that can be further decomposed (e.g. S, VP, and NP), are called *non-terminals*. Symbols that cannot

be further decomposed (e.g. *VERB*, *NAME*, *ARTICLE*, and *NOUN*), are called *terminals* or more accurately *pre-terminals* [3].

In a story grammar, the terminals are equivalent to story statements and the non-terminals describe the structure of the story, such as events and states, which are sub-structures of episodes and settings [40].

Story grammars have not been implemented as widely in computer understanding systems as schema-based approaches, and have been for the most part largely theoretical. However, there have been many experiments conducted to test the validity of story grammars. Black and Bower demonstrated the validity of *episode constituents*. Episodes are generally considered to consist of a sequence of *events* that lead the episode to a conclusion. An event is considered to be an atomic unit or level in the story; episodes are higher level constituents that contain one or more atomic level event [40]. Black and Bower showed that reading rate is affected by such episode constituents [5]. For instance, around the boundaries of an episode, i.e the most critical part of the episode, reading rate slows down.

The second validity for story grammars concerns the "levels effect". Mandler describes the levels effect as "people are more likely to remember high-level propositions than low-level ones." [29, p. 62] That means that when someone reads a story, they are more likely to remember high-level information about the plot rather than low-level details. This idea comes from psychologists, and is most often credited with work that Kintsch and his colleagues did in the 1970s [24]. Since story grammars describe the structure of a story in a hierarchial fashion, one can say that they encapsulate the levels effect.

Wilensky and others have criticized story grammars, saying that you cannot describe stories with grammatical structures in the same way as sentences [44]. I agree with this due to the following criticism. Since the rules governing the structure of the story must be specified before any processing can be done, I do not believe one can specify all the rules necessary to encapsulate all possible structural variations for all possible stories beforehand. There must be a way to learn new rules as the understanding system reads/processes new stories. However, there is no formal machinery for learning new rules within the grammar model.

2.3 Vector-based representation and classification

As mentioned, the vector-space-based formalism is by far the most popularly used formalism in the area of text categorization/classification. This area is concerned with automatically assigning predefined categories to free text documents [31]. This process typically consists of extracting a set of features from a particular document, i.e. keywords, and applying a machine learning model to determine the text category.

A large number of machine learning techniques have been applied in this area, including nearest neighbor classification, Bayes probabilistic approaches, decision trees, neural networks, symbolic rule learning, and inductive learning algorithms [34]. All of these models rely on input data that has been transformed into a feature vector. Within the vector space, the models attempt to separate the data points by a surface or cluster them in some way. The clusters or surfaces indicate where a particular category's boundaries lie and any data point within that boundary belongs to that particular category. The labels of the categories are assigned by the individual

applying the model or possibly by a domain expert.

Although this approach ignores many fundamental issues about language representation, and many issues in NLP such as generation and summarization, it is very widely used. Why is this technique so widely used? There are several reasons.

Firstly, the approaches used to extract "important" features for a training set of text documents are based on statistical methodologies that have existed for a long time. Relying on these existing methodologies allows one to produce a text categorization application in a reasonably short period of time. This has been important, especially with the introduction of the Internet. The Internet forced the requirement for a ready-made model for searching information efficiently, and statistical based approaches were the logical choice to fulfill this need. Secondly, given a new document, a feature vector representation of the document can be generated quickly and categorized efficiently. In practically all applied scenarios of text categorization, efficiency is a critical factor for the success of the model. Finally, as mentioned in Chapter 1, language representation is an extremely difficult area; by relying on feature vectors and statistical analysis, one can avoid many fundamental issues about language representation. This is because these models generally rely only on the keywords within the documents to classify, thus, both semantic and conceptual modelling are avoided.

Supporters of probabilistic techniques argue that language and cognition can be best explained probabilistically. In [28], Manning and Schütze state that the reason researchers have been skeptic of probabilistic models is because many of the well-known early approaches at applying statistics to language were extremely simple. The simplicity could not adequately explain the complexities of language. Manning and Schütze believe that more complex probabilistic models are needed to adequately

explain the uncertainty found in cognition and language.

2.3.1 Feature selection

The goal of feature selection is to drastically reduce the feature space of a given set of text documents without losing categorization accuracy. Without this selection process, the feature space consists of all unique terms, which can be tens of thousands of terms for a moderate-sized text collection [31]. Many classical machine learning algorithms cannot handle this many features within a reasonable amount of time, so reducing the feature space is essential.

Feature selection methods typically attempt to remove terms deemed to be non-informative (stop-words) according to some statistical measurement. Typical techniques to accomplish this are: document frequency thresholding (DF), information gain (IG), mutual information (MI), χ^2 statistic (CHI), and term strength (TS).

Pederson and Yang conducted an empirical study to determine which of these methods, when combined with a given text categorization algorithm, yielded the best results [31]. They found that there is a correlation between IG, DF, and CHI in how each method scores a term's goodness. Also, they found that IG and CHI have similar performances, and DF has comparable performance with up to 90% term removal. Thus, in this case, the simplest feature selection method (DF) yielded similar performance levels to the most complicated methods (IG and CHI).

Moreover, all of these *term-weighting* feature selection methods are very similar, that is, they each attempt to reduce the feature space by evaluating the goodness of a term and removing those terms which are below some set threshold. How they accomplish

this task varies from technique to technique, but the end result is the same, a feature vector representing a given textual document. Regardless of the technique used, the problem of estimating a huge number of parameters needed for such a model is statistically problematic, and it has been found that choosing the right attribute set or the right set of weights is critical to the success of your classification model [4, 14, 22]. Furthermore, Steven Finch found that when various term-weighting models were used, they all evaluated within 5 points of each other on both precision and recall [14].

As stated, the attributes (i.e. words) selected are the differentiating criteria for any text categorization model based on a vector-space representation. There have been many selection algorithms experimented with, some employing complex statistical methods, while others use extremely simple techniques, and each method appears to only receive marginal gains over its counterparts. There have been some attempts to analyze text in a way that takes into account the semantics, however, most of these techniques still rely on vector-based representation and often perform worse than straight-forward keyword based approaches.

Chapter 3

ETS Model

In this chapter I introduce, non-formally, the central concepts of the ETS model (see [16] for the formal definitions). I begin by discussing the motivation, inspiration, and philosophy of the formalism.

3.1 What is ETS and where did it come from?

ETS began as an attempt to develop a formalism to deal effectively with structural representation. Currently, in natural science, we usually have one basic formalism, which is the numeric or vector-space-based formalism. All science relies on this formalism to describe phenomenon that exists in the universe. AI researchers have been attempting to apply classical mathematical models to the study of the mind and learning with varying degrees of success, however, these classical models appear to be inadequate for talking about the mind [16, 17]. This inadequacy stems from the

inability for a numeric formalism to represent a formative history of an object. We need a representation of objects directly related to the manner in which the object evolved or was constructed. Moreover,

> [i]nformation processing relies on the concept of *class*. Subsequently, areas of information processing (i.e. data mining, information retrieval, pattern recognition, etc.) must depend on a formalism to deal effectively with this concept. However, traditionally, these areas depend on conventional formalisms, and these formalisms were not developed to deal with the concept of *class description*. One has to adapt these conventional formalisms to deal with this concept, which is generally inadequate, as the formalism does not have language for describing a class. [16]

For instance, as mentioned in the last chapter, the vector-space formalism is able to classify objects by decision surfaces. However, these surfaces do not describe anything about what types of objects fit into the class, and they do not describe the class itself; we must name the class, which in itself, can be ambiguous.

The ETS model is a model for object/event representation and was motivated by two considerations: "the fundamental inadequacies of existing formalisms for class description and by the vision of the class description as a set of structural transformations; these transformations are supposed to play the role of structural units, out of which class objects are assembled" [16]. Each object can be represented by the assembly of these structural units where this assembly is known as the object's *constructive history*. Moreover, we can think of the constructive history as the sequence

of events that took place to construct the present object (see Figure 3.1). For instance, consider a water molecule, H_2O. Before the water molecule was formed, there existed, in the universe, two hydrogen atoms and an oxygen atom. Due to some event at some given time in the universe, the two hydrogen atoms and the oxygen atom bonded to form the water molecule. This bond event is part of the constructive or generative history of the molecule.

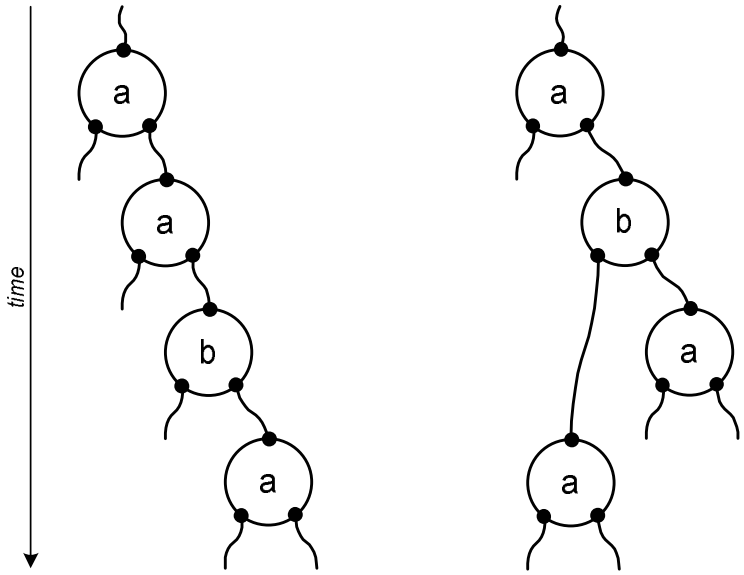


Figure 3.1: Two different constructive histories for the insertion string "aaba". In Figure 3.1a, the string is formed by attaching each character one after the other. In Figure 3.1b, the string is formed by inserting 'a', then attaching 'b', then attaching 'a', and finally inserting 'a' before 'b'.

Viewing objects and classes in this manner, i.e. dynamic and changing, means that "one needs an evolving set of transformations that captures the class description and

also modifies the corresponding (evolving) mathematical structure on the representation 'space'." [17] Moreover, ETS supports dynamic structures, that is, structures being modified on the basis of the inductive experience. The evolving nature of these structures creates the need to introduce *temporal information* into the structural representation of an object. The temporal information is encapsulated by the constructive history, and it is this feature that sets ETS apart from many known mathematical formalisms. The advantage of encapsulating this temporal information is that we now know exactly how the object was constructed, which helps us understand the future of the object, i.e. we can inductively predict what type of events the object may participate in.

ETS proposes a multi-levelled view of reality, where several *primitive transformations* can be combined in some particular way, and at a higher level of representation, be considered as a single primitive. For instance, a water molecule consists of three more primitive objects, two hydrogen atoms and an oxygen atom. The molecule itself is a more complex object than the atoms, but the entire molecule can be considered to be one object at the molecular level, which can then participate in various molecular level events possibly spawning higher-level processes. The "discovery" of these higher-level processes forms the multi-levelled view, where the discovery takes place as the *intelligent process* evolves. The intelligent process is "an actual non-deterministic process operating on structured actual entities by assembling them into larger entities, guided by some 'abstract description'." [16] (see Figure 3.2).

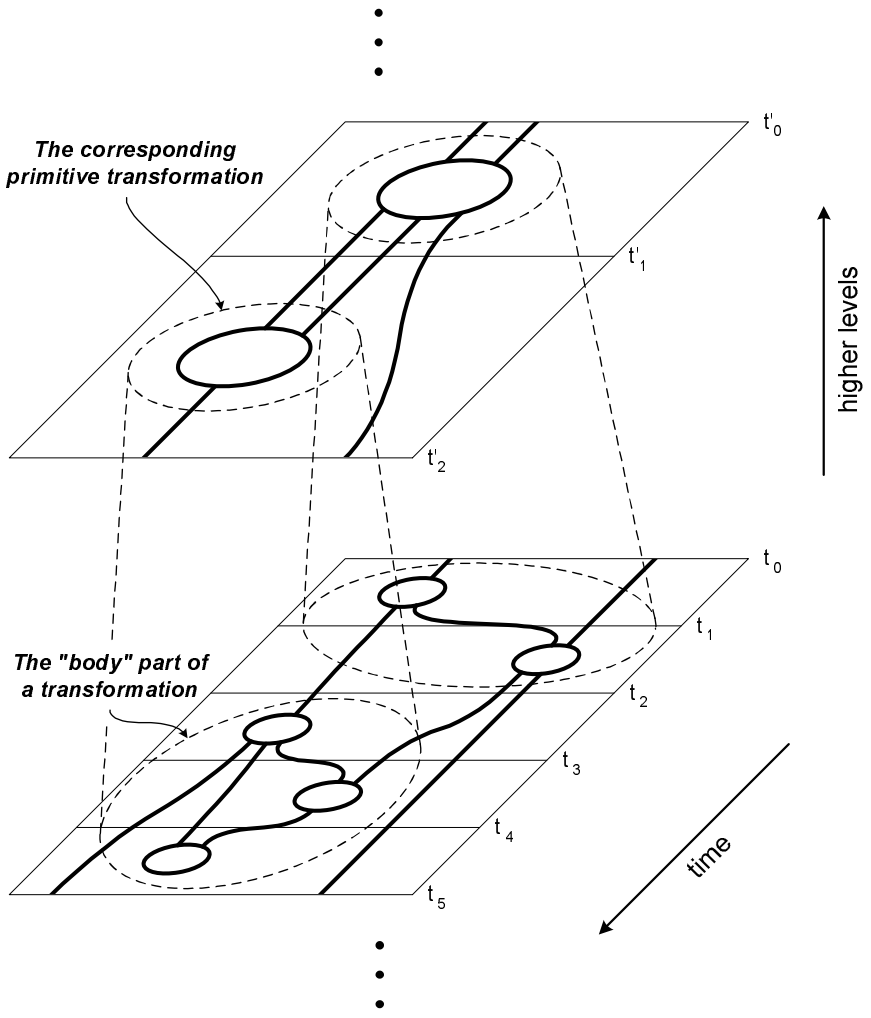


Figure 3.2: Simplified multi-level ETS representation with different time scales for each level. (Two consecutive levels are shown. The time scale for the higher level is measured in coarser units, i.e. $t_0^{'}$ corresponds to t_0, $t_1^{'}$ corresponds to t_2, etc.) The shown *supertransforms* consist of single transformations, and the context parts of the transformations are not identified [16].

Finally, ETS proposes two perspectives of the universe, that is, an observation view and an event view. The observation view is the common scientific view of reality,

which is related to observations in the object environment. The event view corresponds to the generating process view, where transforming events are the focus of study (see Figure 3.3).

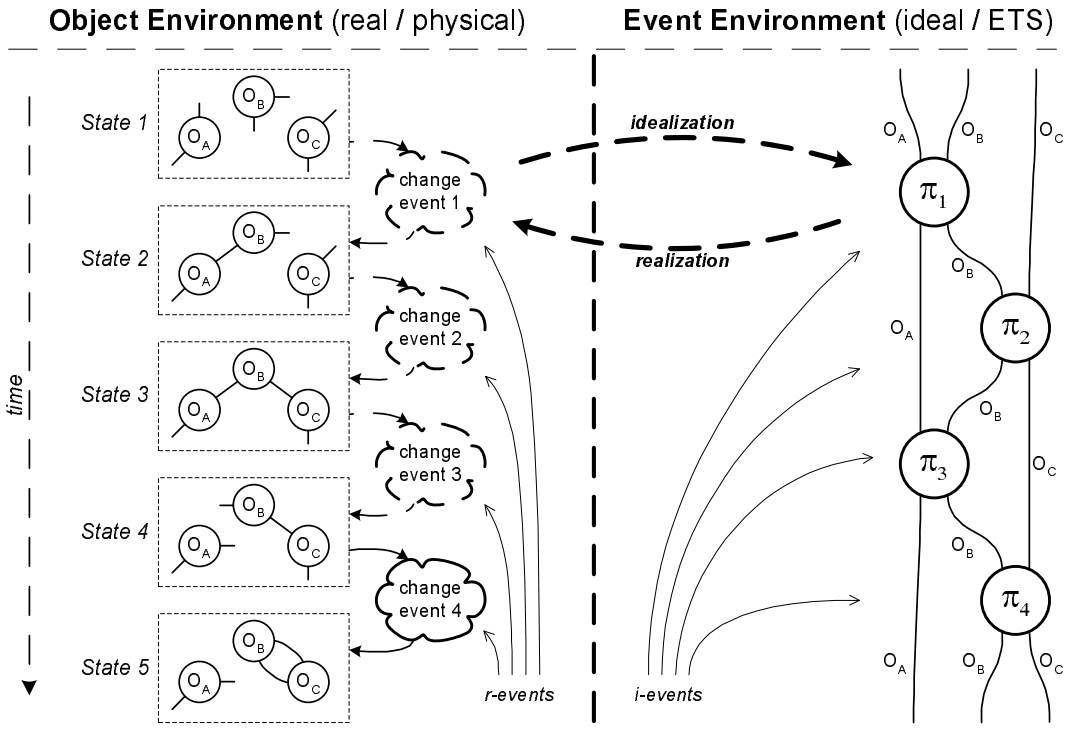


Figure 3.3: Event environment versus object environment. In State 1, three unbonded oxygen atoms are shown. After the first real event has occurred, O_A and O_B become bonded. The corresponding ideal event (primitive π_1) is depicted on the right. Three subsequent state changes are also depicted. [16]

3.2 ETS primitive transformations

Primitives are the basic constructive elements of the ETS model. A **primitive** is an elementary process that transforms the initial objects into terminal ones [17] (see

Figure 3.4). The initial objects and terminal objects are represented by sites. Roughly speaking, a site is some entity involved in the primitive event, which could be thought of as a reference point in the flow of the events.

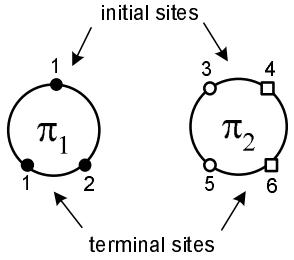


Figure 3.4: Pictorial illustration of two primitives. The solid circle, empty circle, and empty square denote three different site types. The site labels are named using natural numbers; this is for convenience only.

The primitives in Figure 3.4 are particular instances of primitives, however, the site labels may change due to different circumstances in which the primitive event appears. Therefore, we can think of a particular instance of a primitive as belonging to a class of *structurally identical* primitives, that is, the structure of the primitive remains the same, but the site labels are different. The equivalence class of structurally identical primitives is called the **class primitive transformation** or **class primitive** (see Figure 3.5).

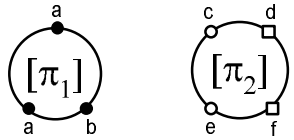


Figure 3.5: Pictorial illustration of two class primitives. $\{a, b\}$, $\{c, e\}$, and $\{d, f\}$ are names for the variables that are allowed to vary over non-overlapping sets of numeric labels.

Finally, it is important to note that the name of a primitive (π_1 and π_2 above) should be chosen in such a way as to specify explicitly what the primitive event represents. In ETS, the syntax and semantics are inseparable, therefore, a good primitive is one that supports this. It supports this by having its structure and name chosen in such a way that it does not need much extra explanation, i.e. one can look at the primitive and understand the "gist" of how it would be used.

3.3 ETS structs (segments of formative history)

An ETS **struct** represents a particular instance of structural history, where the structural history can be thought of as recording the sequence of elementary or primitive events that appear during the running of some process. In Figure 3.6 we see an example struct. The vertical positioning of the primitives corresponds to the actual order in which the events take place (time flows downwards), thus, parallel primitives signify simultaneous events. A primitive event can be attached to an existing primitive if at least one initial site of the new primitive matches a terminal site of the existing primitive. The *initial sites of a struct* are the initial sites from primitives within the struct that are not attached to any primitives (e.g. the initial site with label 4 in the example). Similarly, the *terminal sites of a struct* are the terminal sites from primitives within the struct that are not attached to any primitives (e.g. the terminal sites with label 1 and 4 in the example).

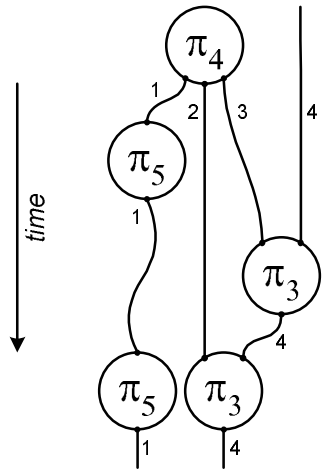


Figure 3.6: Pictorial illustration of a struct.

Analogous to primitives, structs also form classes, where a particular struct is an instance from a class of *structurally identical structs*. The class of a particular struct can be represented by changing the concrete labels to generic labels, thus, the structure remains the same for all instances of the class, but the labels vary.

3.4 ETS transformations and supertransformations

Transformations, *supertransformations*, and *class supertransformations* are the central concepts in the ETS formalism. A **transformation** (or transform) consists of two parts: a context and a body. As the term suggests, the **context** of a transform consists of those primitives that embody the "preconditions", i.e. events, necessary (but not sufficient) for the appearance of a body in a struct. The **body** of a transform captures a segment of the struct that can now be thought of as *segmented into such bodies*, where each body corresponds to a next level event and becomes a next level

primitive [16].

The context of a transformation is specified by an *extruct*, while the body is specified by a struct. An **extruct** can be thought of as a fragment of recent history, where this fragment identifies where the transform may originate [16]. Figure 3.7 shows three example exstructs. The ×'s at the end of some site lines signify that these are **detached sites**. The term detached sites comes from the procedure that constructs the context by excising/detaching it from the enveloping struct (at these sites). The bold lines are the **interface sites** (Iface) that connect the context to the body.

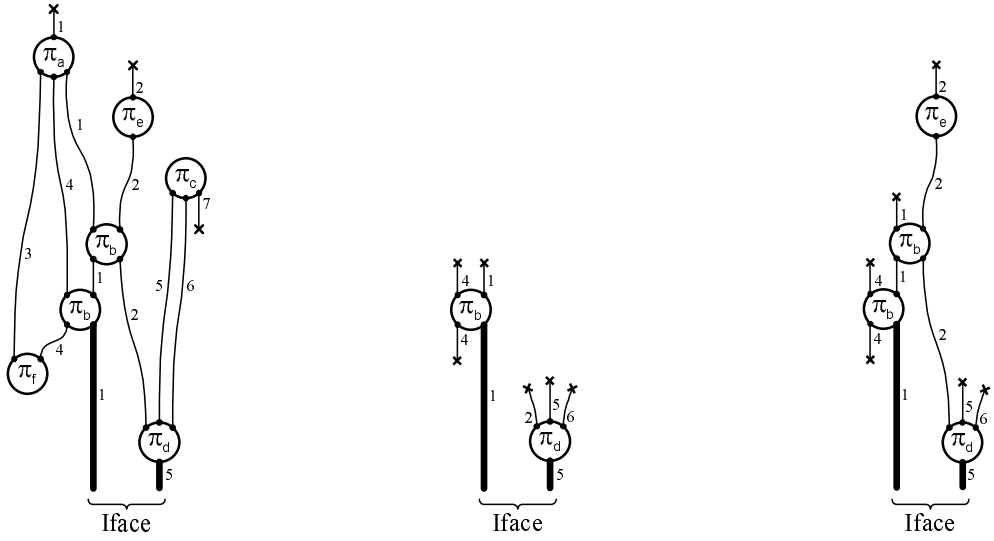


Figure 3.7: Pictorial illustration of an exstruct [16]

.

A transformation can be thought of as a "representational module", where a struct is "formed" by a series of transformations (see Figure 3.8). The transformation embodies a sequence of recurring events, where the context specifies the preconditions for such events, and the body is what is recurring. If we consider the body as a whole,

we should be able to think of it as a single, semantically and structurally meaningful unit.

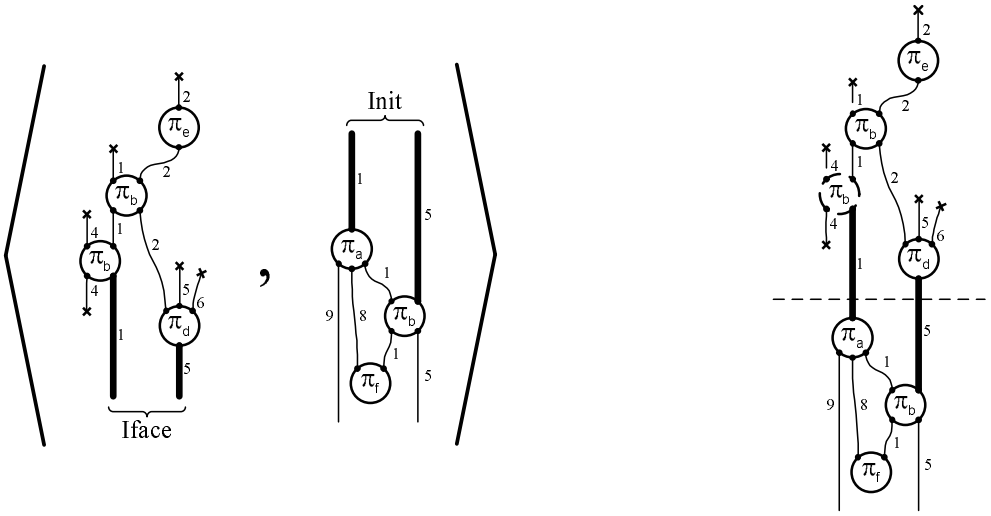


Figure 3.8: Pictorial illustration of a transformation [16]. The left-hand side represents the transform as a pair: context and body. The right-hand side depicts the "assembled" transform corresponding to a more appropriate interpretation/understanding of the transform.

The concept of *supertransformation* (supertransform) is a generalization of the concept of a transformation; it can be thought of as an abstraction of the set of several closely related transforms acquired during one's inductive experience. This concept is the central one in the ETS formalism since it encapsulates the concept of *class description*. A **supertransform** is defined, basically, as a set of (related) transformations, called *constituent* transformations, with similar bodies and common interface sites [16]. The contexts are suppose to capture all the necessary "preconditions" for the appearances of the corresponding bodies and are typically more varied than the bodies themselves. The supertransform's bodies are suppose to capture various instances

of the corresponding next level event, including those events that include "noise" (see Figure 3.9).

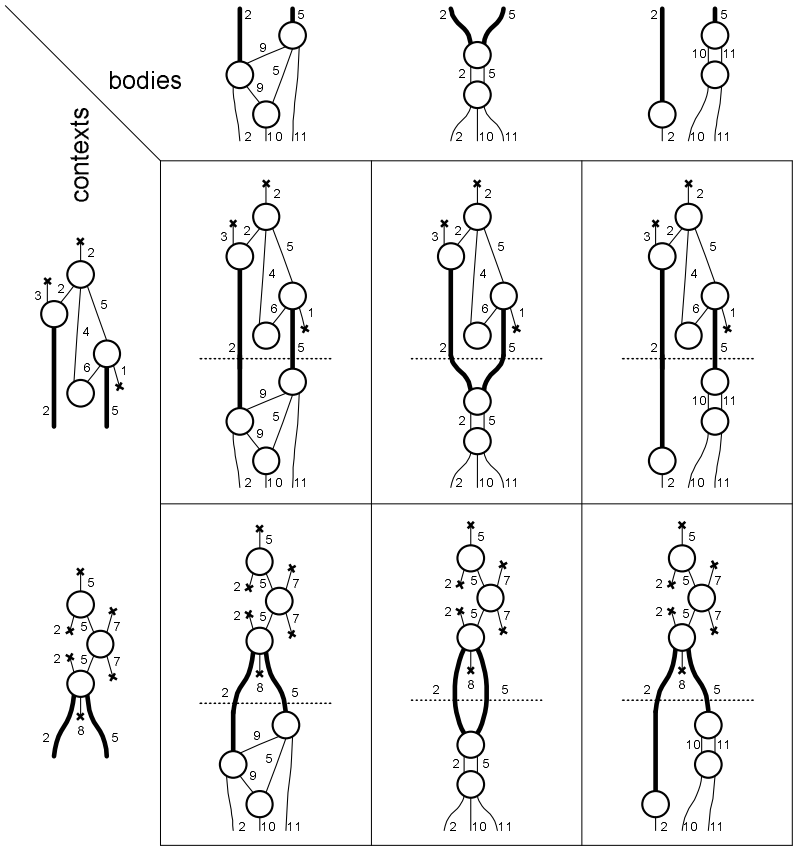


Figure 3.9: Pictorial illustration of a supertransformation [16]. All the contexts have the same interface sites and all the bodies have the same initial and terminal sites.

Finally, as with primitives and structs, a particular supertransformation belongs to a class of *structural identical* supertransformations, where the labels can vary. This class of structural identical supertransformations is called the **class supertransform**, where the class supertransform is obtained on the basis of the supertransforms, by abstracting away the supertransform's labels (see Figure 3.10).

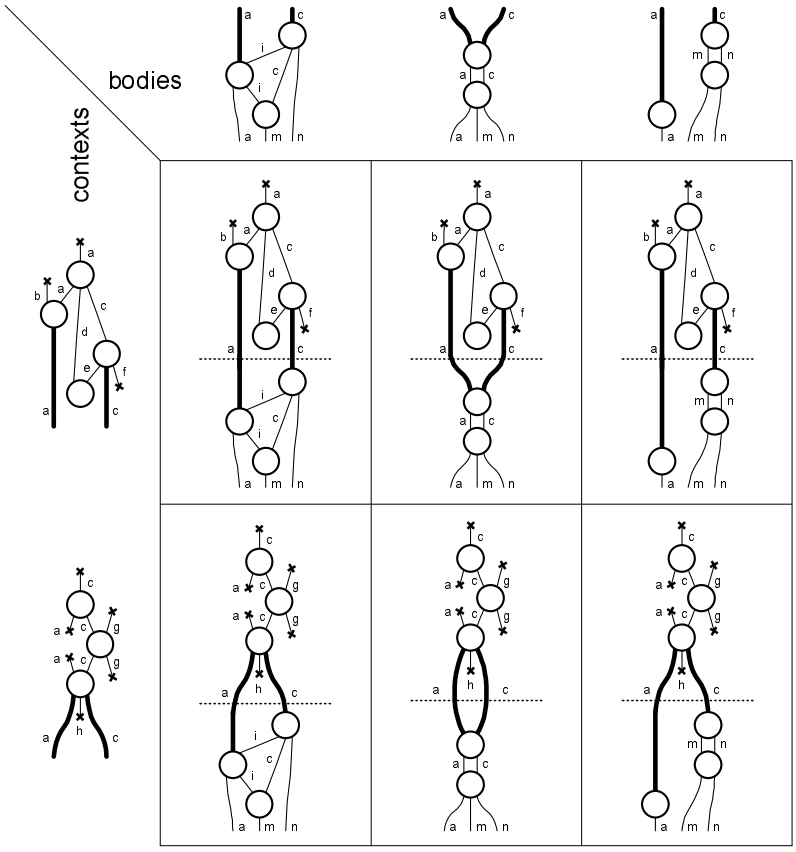


Figure 3.10: Pictorial illustration of a class supertransform induced by the supertransform depicted in Figure 3.9. Each letter is the name of a variable that is allowed to vary over numeric labels of the same type. [16]

3.5 The next level representation

ETS moves to the next level of representation through a form of "chunking". This chunking allows us to deal more effectively with the complexity of event representation, i.e. noise is absorbed by moving to higher-levels and the size of the representation

is reduced [16]. How is a new level of representation introduced in the ETS formalism? Figure 3.11 explains the basic idea behind the ETS *level ascension postulate*, which shows how to convert a supertransform into the next level primitive transform [16]. Thus, the supertransform's interface sites (i.e. sites connecting the context to the body) become the primitive's initial sites while the body's terminal sites become the primitive's terminal sites.

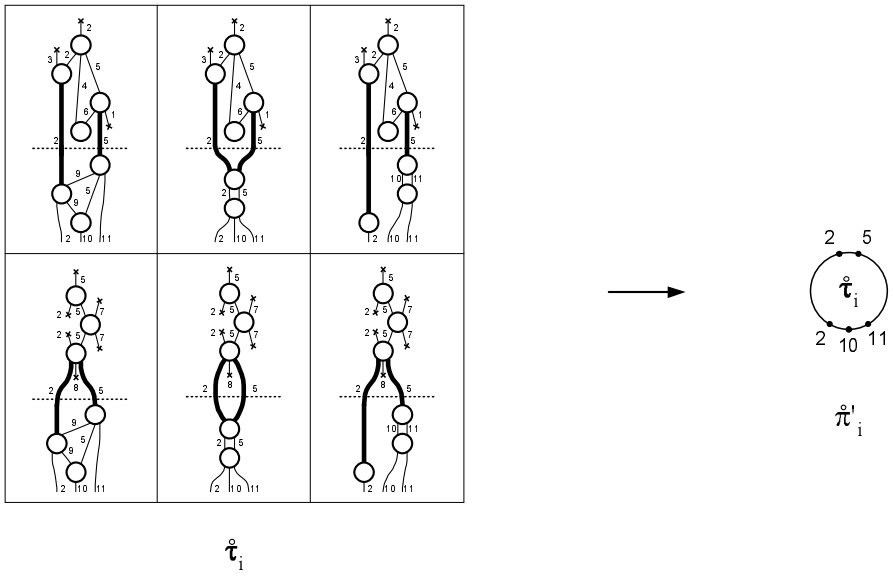


Figure 3.11: A *canonical supertransform* and the corresponding next level original primitive [16].

The next level primitives can be used similarly as the initial level primitives (i.e. in structs, extructs, transformation, supertransformations, etc.). Each level is inductively constructed based on the construction of the previous level (see Figure 3.12).

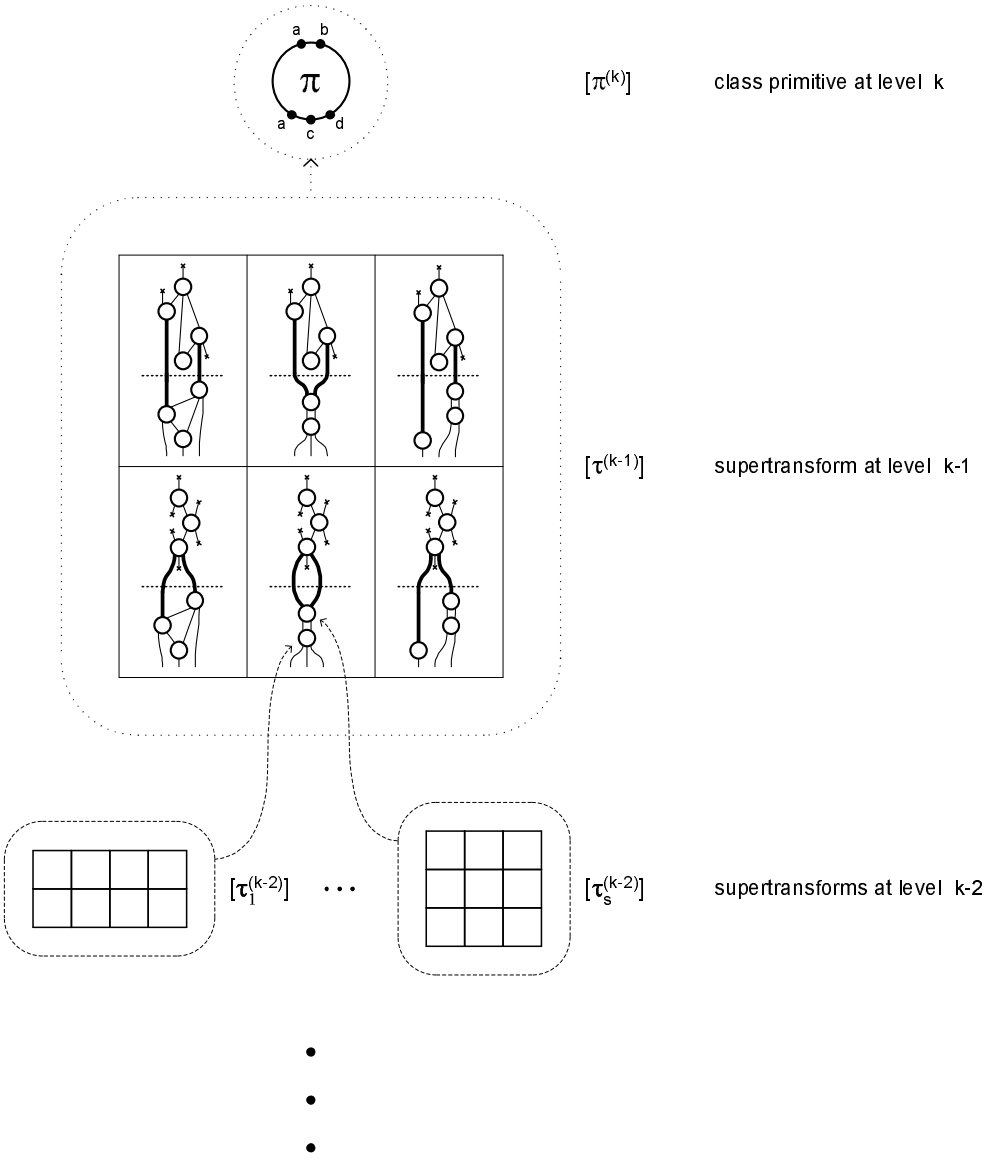


Figure 3.12: Pyramid view (partial) of a k-th level class supertransform: the pyramid should be thought of as being formed by the subordinate class supertransforms [16].

3.6 The intelligent process

An **intelligent process** is a process which "optimally" captures/represents the development of some universe by expanding and refining (in a discrete mode) its multi-level inductive structure, including the number of its levels. It accomplishes this mainly by the creation and modification (but never deletion) of relevant class supertransforms at the appropriate levels [16]. That is, as the process evolves, it creates new levels by discovering new supertransforms and climbing levels as shown in Figure 3.11 of the previous section. The discovery of new supertransforms subsequently expands the set of primitive transforms at the next level. The only input required of the process is the initial level primitives as they appear in the working struct.

In Part III of [16], a provisional algorithmic sketch of this process is introduced. The algorithm introduced is an unsupervised (could be supervised) inductive learning algorithm for the ETS formalism. It is important to note that this is an area of ongoing research, and the algorithm is still considered to be much more tentative than the rest of the model. Here, I provide a very high level overview of the steps involved in the algorithm. I encourage the reader to look to [16] for implementation details.

The intelligent process algorithm consists of three steps: learning, recognition, and facilitation. The learning step involves the modification or the possible discovery of class supertransforms. The sub-step chooses a transform from those available and identifies that either it has already learned the transform (in this case it is done) or that part or all of the transform has not been learned yet. If the context is known, then it adds the body to the appropriate class supertransform. If the body is known,

then it adds the context to the appropriate class supertransform. Otherwise a new class supertransform is created along with a corresponding next level primitive.

The recognition sub-step identifies the set of all available transforms based on the latest primitive added to the working struct. Then it randomly chooses a transform from the available set, and finally performs various updates to the working memory and association memory (auxiliary weight matrices used by the algorithm).

Finally, the facilitation sub-step increases the status of learned transforms in order to increase the likelihood of their reappearance. This is so that learned transforms are easier to recall or recognize later as the working struct progresses.

Figure 3.13 shows an example of the expected behavior of the process. Transforms are identified, beginning at the initial level, and the results are propagated to the next level. The initial level process is operating at a much faster time scale as there are continually small changes at this level, and as we climb levels, these small changes have less effect, thus, the time between salient events becomes longer.

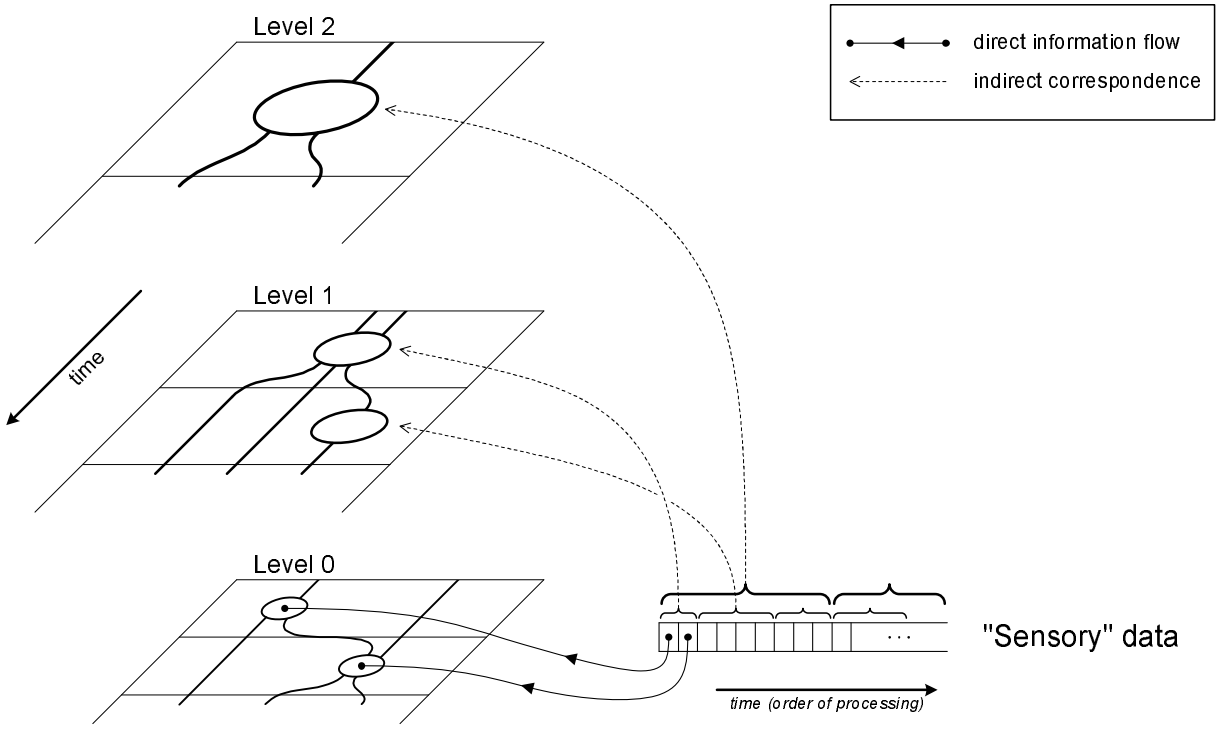


Figure 3.13: A multi-level representational tower with a single-level sensor at level 0. [16]

Finally, it is interesting to note that the intelligent process is continually evolving and continually guiding the process of learning. Of course, in an implementation, the algorithm can always be stopped by some termination flag once it has reached a desired level of learning. I feel that this intelligent process goes beyond traditional machine learning algorithms by attempting to explain the fundamental process of inductive learning in the universe.

Chapter 4

ETS fairy tale representation

In this chapter we propose a small number of primitive events based on the atomic cognitive events that should occur in the mind of the "generic" ten-year old fairy tale listener/reader. For example, as one reads a story, certain events take place that we "recognize", such as an actor picking up an object, we are then able to *associate* or *connect* each of these events with other events in the story. As mentioned in Chapter 1, fairy tales make a particulary interesting domain as every event, character, object, etc. is important to the development of the plot. Therefore, it appears that one can characterize or classify fairy tale plots based on story structure. This observation has also been noted by several other researchers [29, 32].

Below, I give a condensed version of the fairy tale "Salt" from [1]. This is a typical Russian fairy tale involving three brothers, a princess, and a quest. I give this example in order to "aid" with your understanding of the primitives introduced in section 4.2.

In a certain city there lived a merchant who had three sons; Fyoder, Vasily, and

Ivan the Fool. The merchant sent the two older sons out on ships. He didn't trust Ivan because Ivan was a drunk. Ivan convinced his father to give him a ship, but his father filled it with cheap cargo so that Ivan wouldn't spend it all on drink.

Ivan sailed away and arrived at an island. On the island he found pure Russian salt. He had his crew remove the cheap cargo and replace it with the salt.

Later, he arrived at a wealthy city and showed the King the salt. At first the King did not want the salt, but Ivan convinced him and received gold and silver in exchange.

The King's daughter wanted to see Ivan's ship, so she came down to the harbor to take a look. While Ivan was showing her the ship, he had his crew cast away. At first the Princess was upset, but since Ivan was handsome, she was soon smiling and ceased grieving.

While at sea, Ivans brothers found him, learned of what he had done and came onto his ship, threw him into the sea and took the princess and loot for themselves.

Ivan, with the help of a Giant, was able to return to his father's land before his eldest brother married the Princess. The Princess and Ivan told his father what had happened, and the two older brothers were driven out. After this, they celebrated with a feast.

It is important to note that as this is one of the first applications of the ETS model, and as the model is very new, much of the representation, ideas, and conclusions should be considered tentative. Moreover, many of the following sections are based on work proposed in [13].

4.1 Fairy tale structure

Before introducing my initial level primitives, it is important to understand some basic ideas about fairy tale and story structure. It appears that all stories have an underlying generic structure, with certain types of structure being more common to certain cultures than others [18]. For instance, in North America, a typical story consists of a setting, several episodes, each having a beginning and end, and finally a resolution. This is a very high-level view of the structure of a story, but generally speaking, this high-level structure is common to most North American stories. As one investigates specific genres of stories within specific cultures, such as Russian fairy tales, the structure becomes less general and more rigid in its structure.

For instance, within the genre of fairy tales, there's generally a hero, and he is assigned to complete an impossible task. The impossible task is usually only completed at the last moment or barely completed. Characters are usually extreme or caricatures, and they generally do not exist outside their function in the story. For example, the hero is simply the hero of the story because he's supposed to be the hero, just as the villain is the villain for no particular reason other than that's how the story is told. Furthermore, there's a commitment to the plot, and the story doesn't waver from that commitment [1, 2].

If we localize a particular genre to a particular culture, the structures become even more specific and apparent. For example, most eastern European fairy tales have patterns of threes, i.e. three brothers, three tries to complete task, etc., while Asian fairy tales have patterns of four [1, 2]. In European tales, the hero is always the underdog, and the villains are always the rich or privileged. A lot of the cultural variation has

to do with the social conditions of where and when the story was generated. For instance, in Europe, the listeners of the stories were generally commoners, and they wanted to hear stories about unlikely heroes prevailing over the rich and powerful.

Finally, moving beyond than genre and culture, we can look at classes of stories. It appears that different classes of stories have different underlying structure. The structure of a story where an unlikely hero performs a task to win the princess is different than the structure of a story where the hero completes a task to win the princess but is then betrayed by his brothers. This variation on the first story is a sub-class of a larger class of stories where the unlikely hero wins the princess. Since the second story belongs to a sub-class, it should share some structural similarities to the first story, and having a representation that is able to capture all this information is essential to differentiate the classes. It is this consideration that makes ETS such an appropriate formalism for the representation of fairy tales.

4.2 Initial level primitives

In this section we present our initial level primitive events/transformations. As was mentioned above, we tried to select as primitives those events that are "atomic" mental events of the listener/reader of fairy tales. In other words, a primitive transformation encapsulates the *structure* of the corresponding mental event. When an ETS primitive models/stands for an atomic mental event, it means that the corresponding event transforms input (or "initial") sites into output (or "terminal") sites. Within the fairy tale domain, a site is some conceptual entity involved in the perception of the story, which participates as a reference point in the flow of the events.

In Figure 4.1, we show the site types used, including a generic site type that is used for convenience to denote any site.

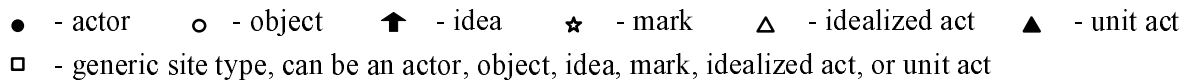


Figure 4.1: Pictorial representation of the site types used.

Similar to Propp's functions (discussed in section 2.2.2), our primitive events were chosen through in-depth analysis of many Russian fairy tales. However, due to the ETS methodology, we were able to discover a better set of primitives (in my opinion) in a short amount of time. In contrast to Propp's functions, our primitives are much more implementable since they are more consistent with the initial level of processing. The problem with Propp's functions is that they assume a certain "knowledge" about fairy tales and about the fairy tale one is reading. For instance, consider Propp's function "Hero and villain join in direct combat". In order to know that this particular event takes place, one must know who the "hero" and "villain" are, but without first reading the story, one can only know who to associate with these terms through induction. Therefore, we must begin at a lower level of representation, and at a higher level, through transformations, perform the induction necessary to identify who the "hero" is.

The information about our initial primitives is summarized in Figure 4.2 and 4.3. They were chosen on the basis of the fairy tales contained in [1, 2].

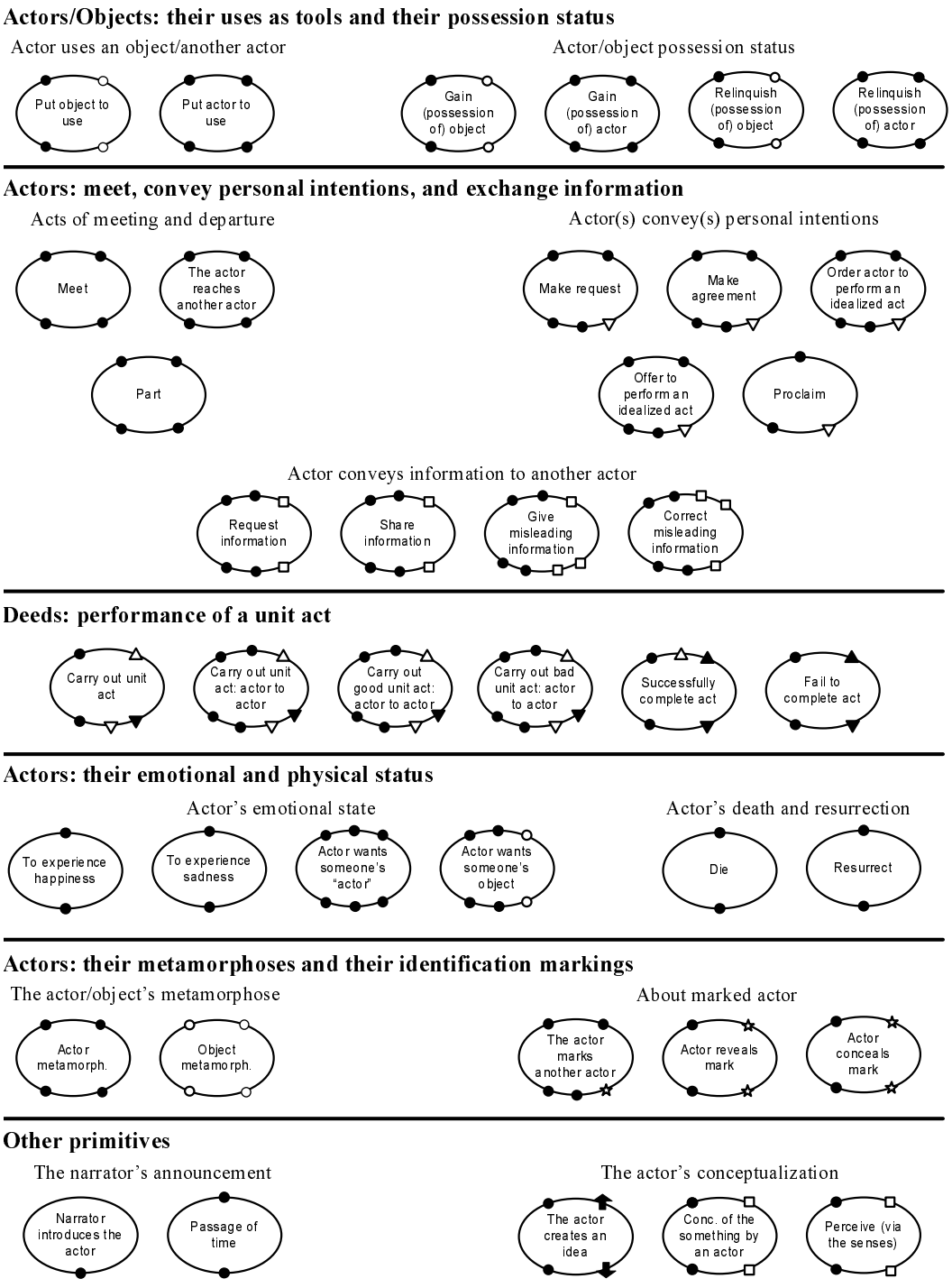
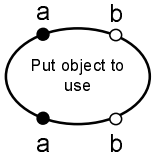
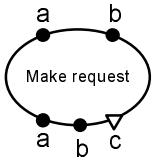


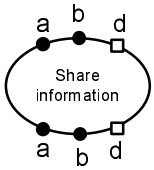
Figure 4.2: Initial level primitives subdivided into their related groups.



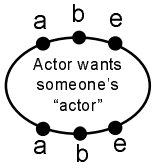
Actor ***a*** uses object ***b***.
Prince Ivan first let his mother down on the linen cloth



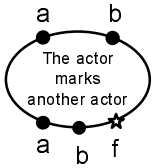
Actor ***a*** asks actor ***b*** to perform previously not mentioned act ***c***.
She wanted to go down to the Russian ship and asked her father's permission.



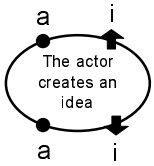
Actor ***a*** shares information with actor ***b*** about actor/object/idea/act/mark ***d***.
"Very well, boys," cried Ivan to crew, make for shore!"



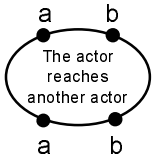
Actor ***a*** finds out actor ***e***'s possession status (possessed by ***b***) and conceives a desire for ***e***.
"I shall take Elena the Fair for myself", said Prince Piotr.



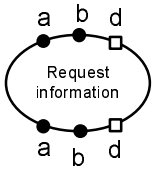
Actor ***a*** brands actor ***b*** with a distinguishing mark ***f***.
And the Tsarevna struck his brow with her signet-ring and left her seal on him.



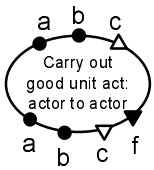
Idea ***i*** is the result of an actor's ***a*** creative process.
He went on farther, saw a snake, pierced it with his spear, and thought: "Here is another riddle!"



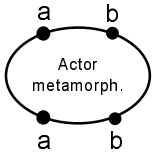
Actor ***a*** attains (geographic) proximity of actor ***b***.
Then his elder brothers overtook him...



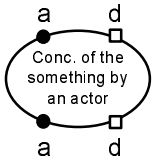
Actor ***a*** asks actor ***b*** information about actor/object/idea/act/mark ***d***.
"What do you want here?" asked the Giant.



Actor *a* carries out unit act *f* (real act version of *c*) for actor *b*.
The king gave her permission.



Actor ***a*** metamorphoses into actor ***b***.
And lo! he turned into a youth as fair as the sky at dawn.



Actor ***a*** conceptualizes/abstracts in his mind actor/object/idea/act/mark ***b***.
"This is my seal!" she cried, "Here is my betrothed!"

Figure 4.3: A more detailed description of some initial level primitives with examples.

4.3 Examples of (initial level) structs

In this section, we show how the segments of the fairy tales are represented, i.e. we show the structs corresponding to such segments. Thus, such structs represent the sequence of the basic mental events generated during the reading/listening of the original fairy tale segments.

Figure 4.4 demonstrates a simple example of how to construct an ETS struct for a fairy tale segment. The segment consists of three sentences from the fairy tale "Salt". Figure 4.4a is the struct representation of sentence one, Figure 4.4b is the struct representation of sentence one and two, and Figure 4.4c is the struct representation of the entire segment.

1. The king had a daughter, a beautiful princess.
2. She wanted to see the Russian ship and asked her father's permission.
3. The king gave her permission.

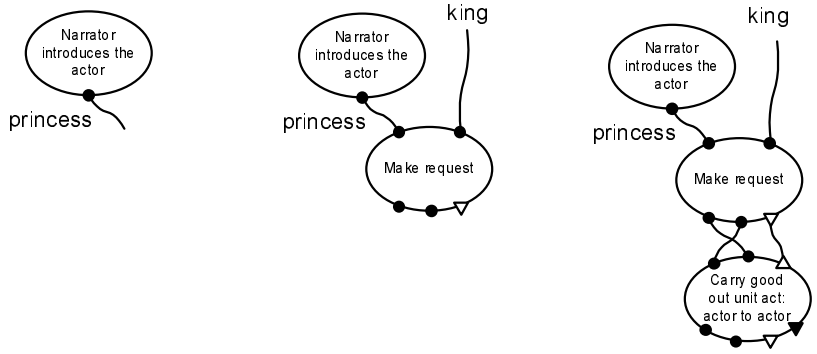


Figure 4.4: ETS representation of a small fairy tale segment from the story "Salt". Figure 4.4a corresponds to sentence one, Figure 4.4b corresponds to sentence one and two, and Figure 4.4c corresponds to sentences one, two, and three.

The structs shown in Figures 4.5, 4.6, 4.7, and 4.8 represent segments from four typical Russian fairy tales taken from [1, 2], "Salt", "The Three Kingdoms, Copper, Silver, and Golden", "The Three Kingdoms", and "Tsarevich Ivan and the Grey Wolf", respectively. These segments were chosen as belonging to the same *class of segments*. At a high level, this class could be described as follows: a hero is returning from the main quest and is intercepted by his brothers, who rob him of all the spoils that he has obtained during the quest.

The sites of the structs acquire now the names of the concrete actors/objects/ideas/idealized act/unit act/marks mentioned in the corresponding fairy tale segment. The "through" sites, i.e. when the initial and terminal sites coincide, are not always labelled to improve the readability. The vertical positioning of the primitives corresponds to the actual order of the events in the fairy tale, thus, the parallel primitives signify simultaneous events. The thick vertical lines are used as space saving devices: the struct on the right is the continuation of the struct on the left. The lines at the top of each figure depict the *initial sites* of the struct [16]. They "lead" to other primitives/events that are not part of the corresponding segment. The same applies to the bottom lines, which depict the *terminal sites* of the struct.

Note that the size of the structs directly correlates with the size of original story segments. Moreover, it is important to observe the explicitness/transparency of the ETS representation: "what you see is what you get". This is a critical feature of the ETS representation, since it captures completely and faithfully the content of the fairy tale.

It is important to note the similarity of the structs. Certain patterns of events occur in each struct. This is most apparent in the fairy tales "The Three Kingdoms, Copper,

Silver, and Golden" and "The Three Kingdoms", which are almost identical even though the overall stories are quite different.

For some time, a long time or a short time, Ivan sailed on the sea with the princess. Then his elder brothers overtook him, learned of his audacity and good fortune, and greatly envied him. They came on board his ship, seized him by his arms, and threw him into the sea; then they cast lots between them and divided the booty: the eldest brother took the princess, and the second brother took the ship full of silver and gold.

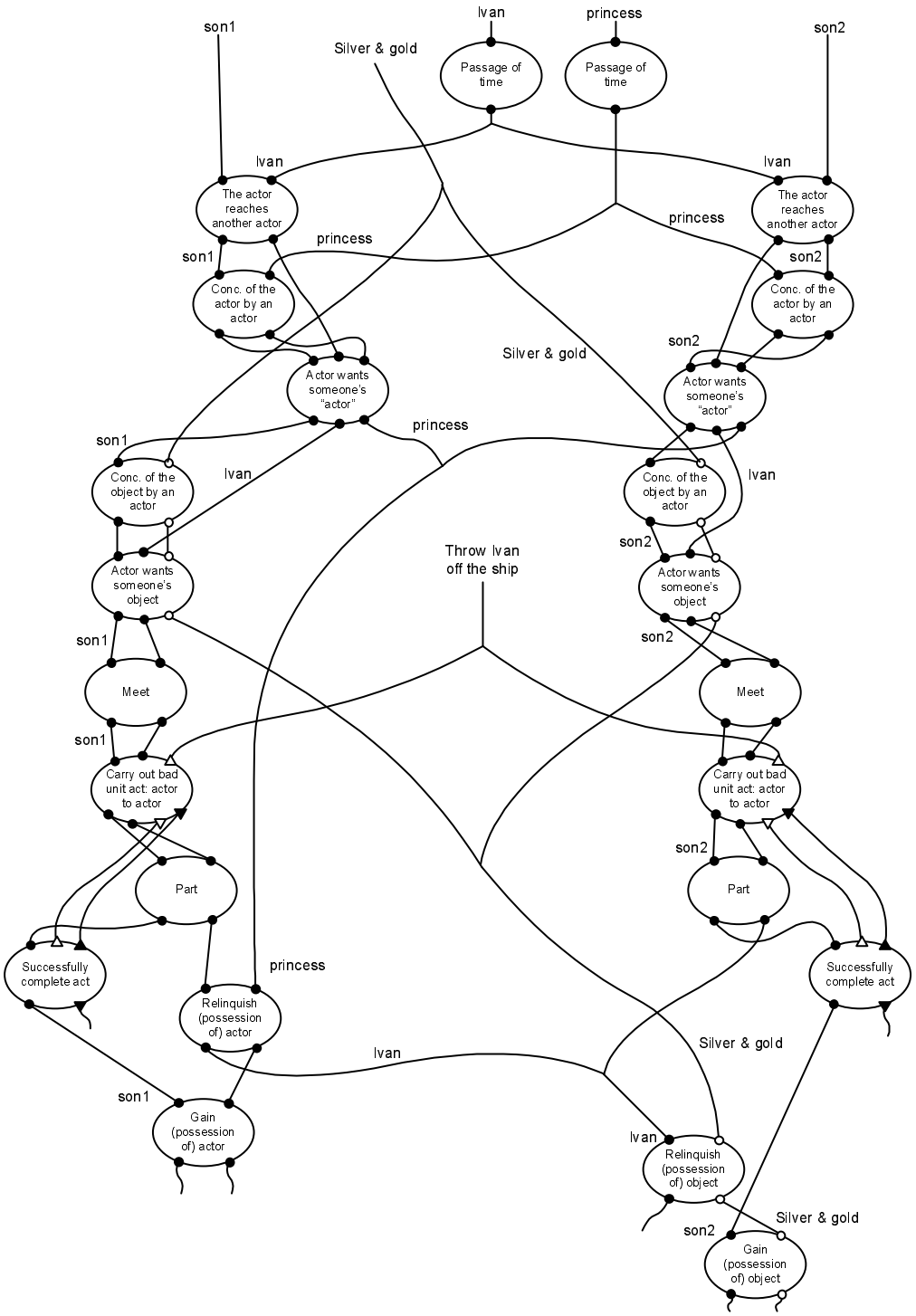


Figure 4.5: ETS representation of a fairy tale segment from the story "Salt" shown at the beginning of this figure.

... and in a short time they [Prince Ivan, Nastasya (Ivan's mother), Elena the Fair, and her two sisters] came to the place where they had to climb down the mountain. Prince Ivan let his mother down first on a linen cloth, then Elena the Fair and her two sisters. The [Ivan's] brothers stood below waiting and thought to themselves: ``We will leave Prince Ivan up there, and will take our mother and the queens to our father and tell him that we found them.'' ``I shall take Elena the Fair for myself,'' said Prince Piotr. ``You, Vasily, will take the queen of the silver kingdom, and the queen of the copper kingdom we will marry to some general.''

When Prince Ivan's turn came to descend the mountain, the older brothers seized the cloth, pulled it, and ripped it off. Prince Ivan remained on the mountain. What could he do? He wept bitterly and turned back ...

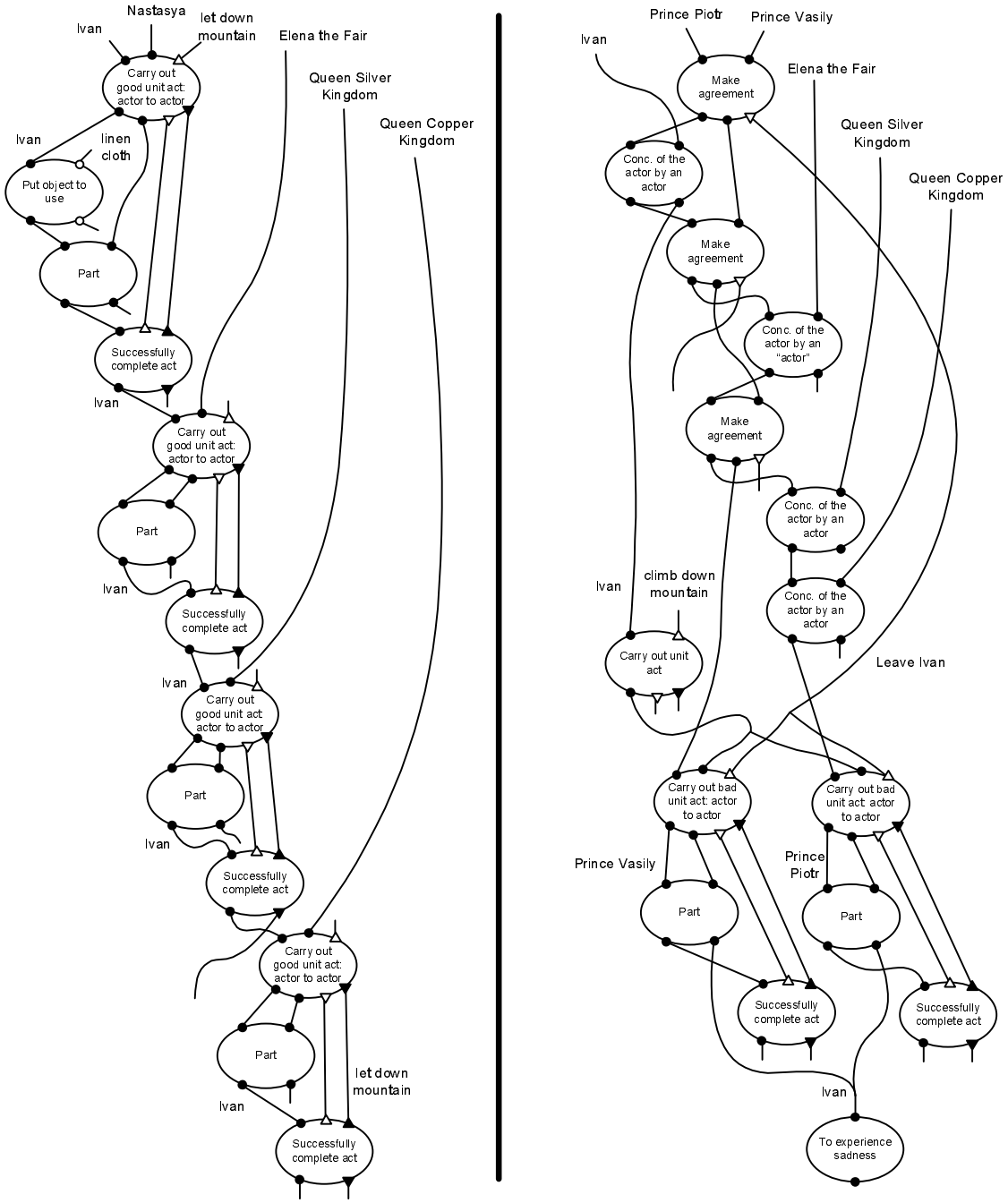


Figure 4.6: ETS representation of the fairy tale segment from the story "The Three Kingdoms, Copper, Silver, and Golden" shown at the beginning of this figure.

Then all of them [Ivashko, maiden from copper kingdom, maiden from silver kingdom, and maiden from golden kingdom] came to the hole from which they climbed out, and they found the thongs hanging there. And the elder brothers [Ivashko's] were already standing by the hole, about to climb into it to find Ivashko.

Now Ivashko seated the maiden from the copper kingdom on the thongs and shook them; the brothers pulled and lifted out the maiden, and lowered the thongs again. Then Ivashko seated the maiden from the silver kingdom, and the brothers pulled her out, and sent down the thongs. Finally he seated the maiden from the golden kingdom, and the brothers pulled her out too, and dropped back the thongs. Ivashko now seated himself on them; his brothers pulled him too, pulled and pulled, but when they saw that it was Ivashko they thought: "If we pull him out he might refuse to give us a maiden." And they cut the thongs, and Ivashko fell down. Well, there was nothing he could do; he wept and wept, and then went on.

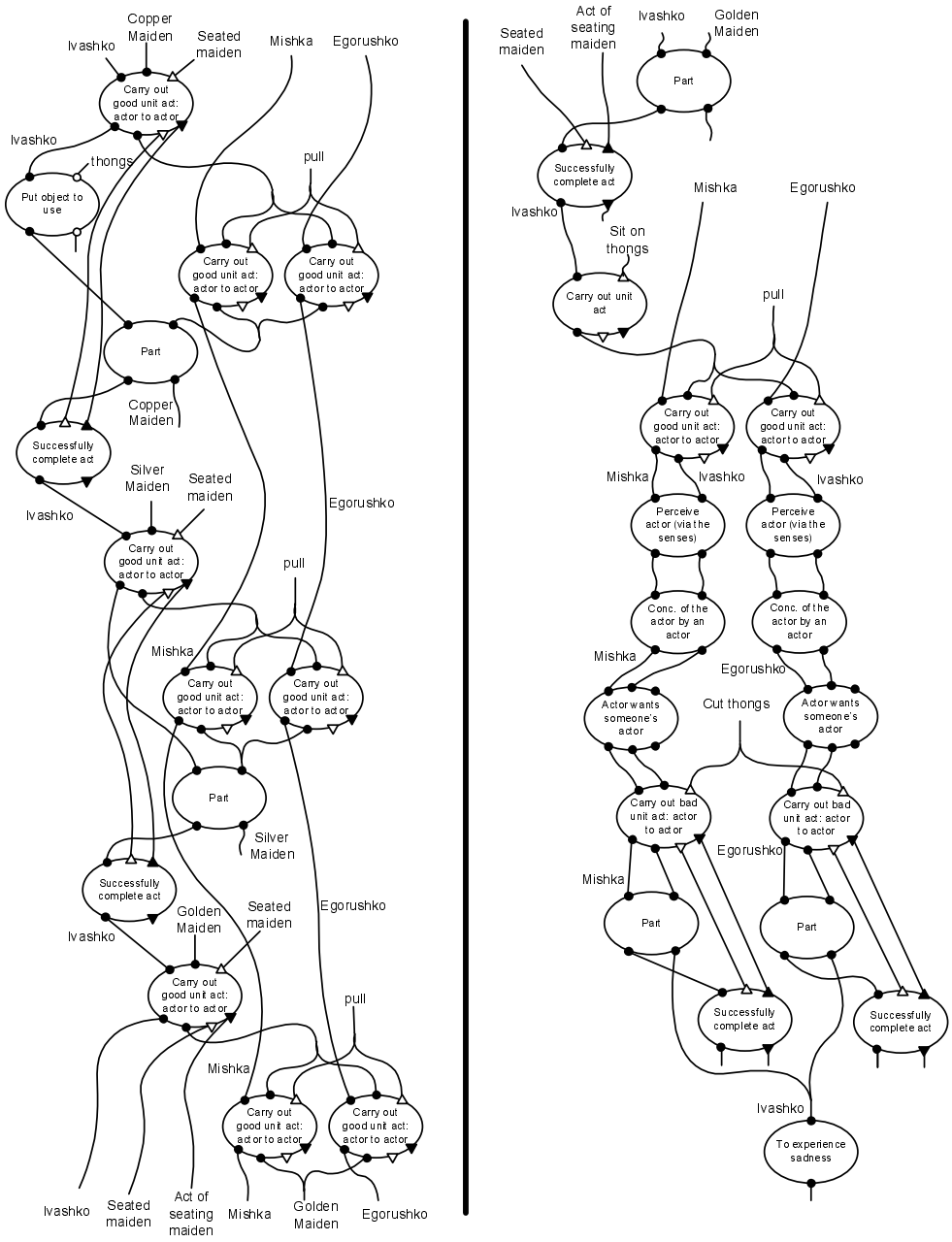


Figure 4.7: ETS representation of the fairy tale segment from the story "The Three Kingdoms" shown at the beginning of this figure.

By and by they [Tsarevich Ivan and Yelena the Fair] reached his native land, and Tsarevich Ivan decided to stop for a bite to eat. He had a little bread with him, so they ate the bread and drank fresh water from the spring, and then lay down to rest.

No sooner had Tsarevich Ivan fallen asleep than his brothers came riding up. They had been to other lands in search of the Fire-Bird, and were now coming home empty-handed.

When they saw that Tsarevich Ivan had got everything, they said:

``Let us kill our brother Ivan, for then all his spoils will be ours."

And with that they killed Tsarevich Ivan. Then they got on the horse with the golden mane, took the Fire-Bird, seated Yelena the Fair on another horse and said:

``See that you say not a word about this at home!"

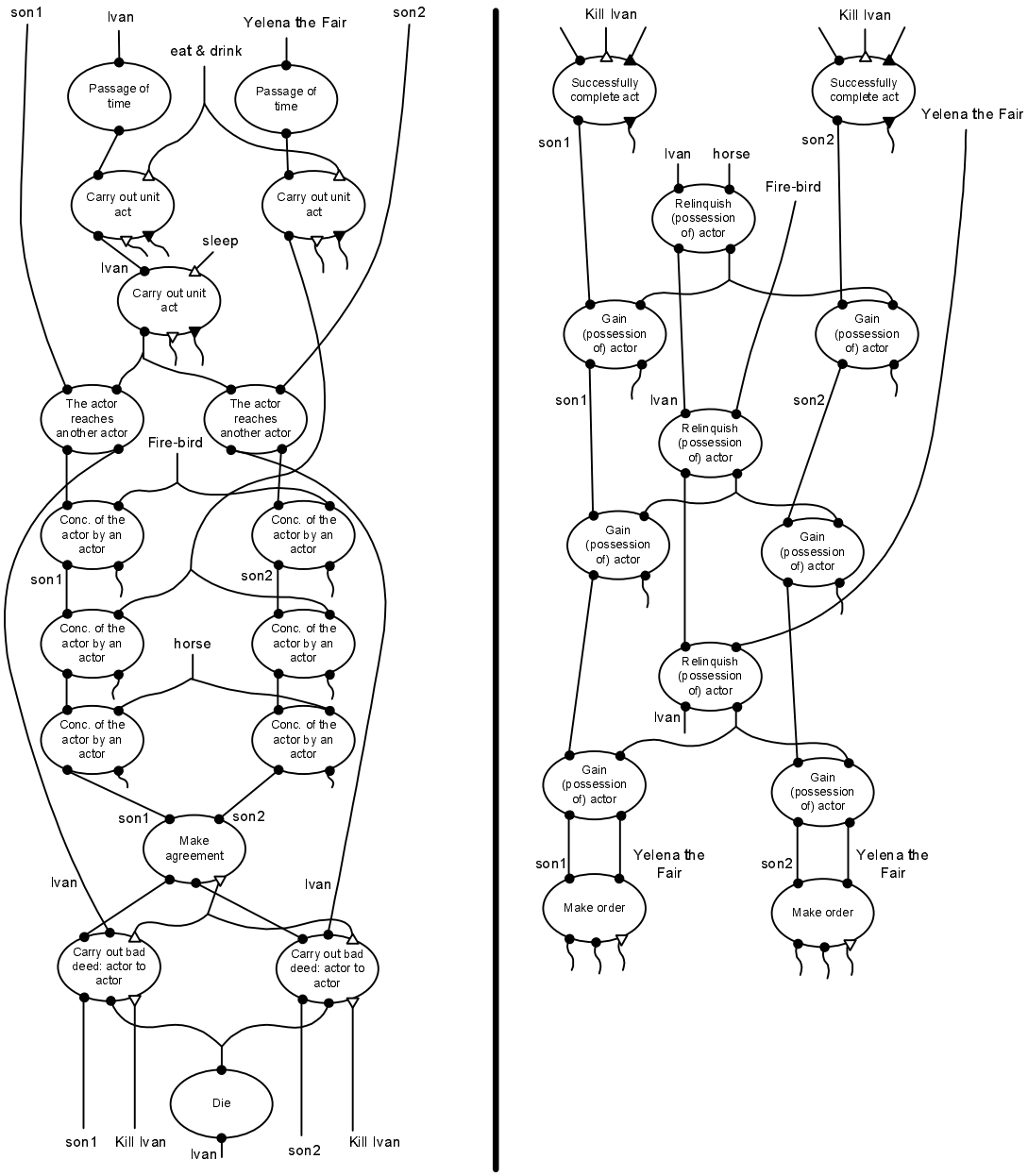


Figure 4.8: ETS representation of the fairy tale segment from the story "Tsarevich Ivan and the Grey Wolf" shown at the beginning of this figure.

4.4 Examples of (initial level) transformations

In this central section, we show several examples of the initial level transformations, or simply transforms, common to several stories. Each of these transforms consists of two parts, the context and the body, where the context are those primitives that embody the preconditions for the appearance of the body. Obviously, a context of a transform must be contained in the struct segments preceding its body. Thus, it is quite natural to associate with the transform the same "name" as that of its body.

It is important to note that single primitives allow me to represent low level plot details, but combinations of primitives allow me to represent larger plot units. The primitives correspond to the basic building blocks that can be combined in a particular way to construct a more complex plot representation. These meaningful combinations of primitives are represented as ETS transformations, where a transformation encapsulates the representation of a commonly occurring plot pattern in the story.

In Figure 4.9, we show a single transformation, whose body is a particular instantiation of a typical (for the chosen fairy tales) small story segment that can be described at a higher level as "good act and its immediate consequences". Note that the specification of this transformation corresponds to the discussion of transformations in the previous chapter, except for the concept of a "sub-context". This concept has not been formalized, but in this application it is used to reduce the number of transformations that must be specified for the supertransformations. The sub-context allows for variability of the context, where the sub-context corresponds to a list of *valid* existing contexts (specified by other transformations) that may exist between this

particular transform's context and its corresponding body. For instance, if we consider the transform below, once the "Make offer" event has taken place, the context has appeared, however there could be an instance of a different transformation before the appearance of the corresponding body. Only certain transformations after the "Make offer" event are applicable..

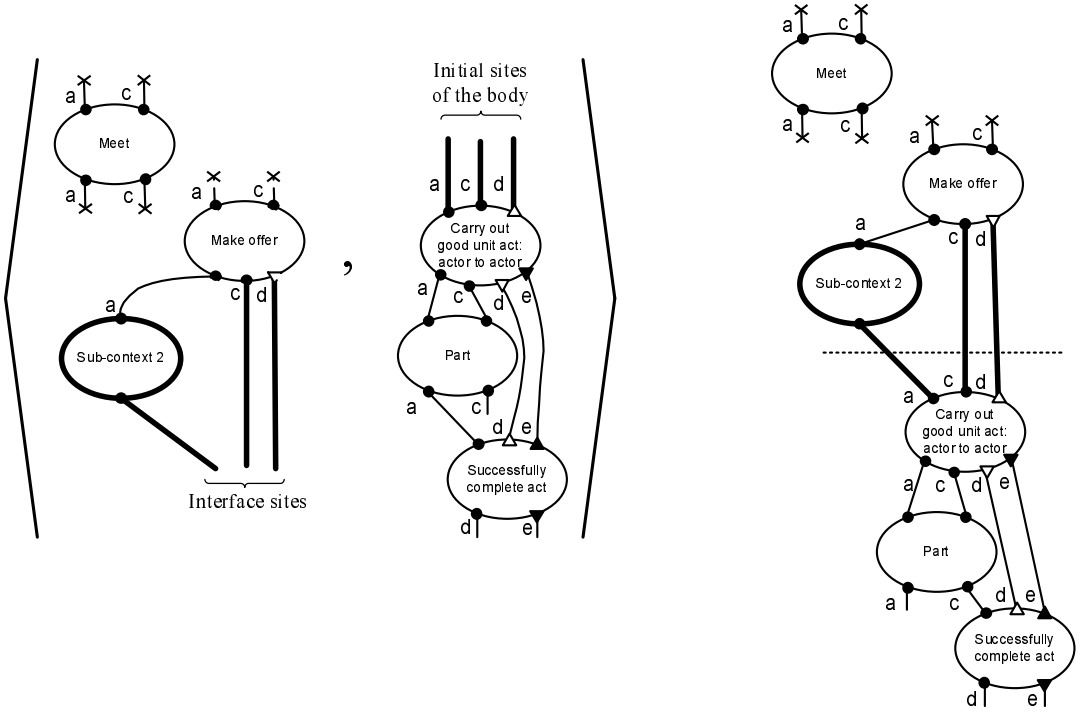


Figure 4.9: An example of a transform. The right hand side of the figure depicts the "assembled" transform corresponding to a more appropriate interpretation/understanding of the transform.

It is interesting to note that Propp, in his "Transformations of the Wondertale", speaks of "reduction" and "expansion" as associated with the corresponding modifications that could occur in various retelling of the same sequence of events [32, p. 86]. In other words, he allows for a certain variability range for a particular fairy tale

segment. The concept of the supertransform formally accounts for such, as well as other, inductively encountered variability. It does this by allowing for a great deal of variability in the constituent transform's contexts and bodies, where one context or body may consist of more details about the sequence of events than another, but still have the same semantic meaning.

The supertransforms shown in Figures 4.10, 4.11, 4.12, and 4.13 encapsulate the following four events: "discovery of an actor, its possession status, and the desire to get that actor", "good act and its immediate consequences", "premeditated bad act and its immediate consequences", and "premeditated taking away an object". They were constructed on the basis of the four example structs from the previous section. Upon closer examination of the corresponding structs, it is easy to see that the constituent transforms indeed appear more or less regularly in the structs. Note that to improve the readability of the above mentioned figures not all constituent transforms are depicted for each supertransform.

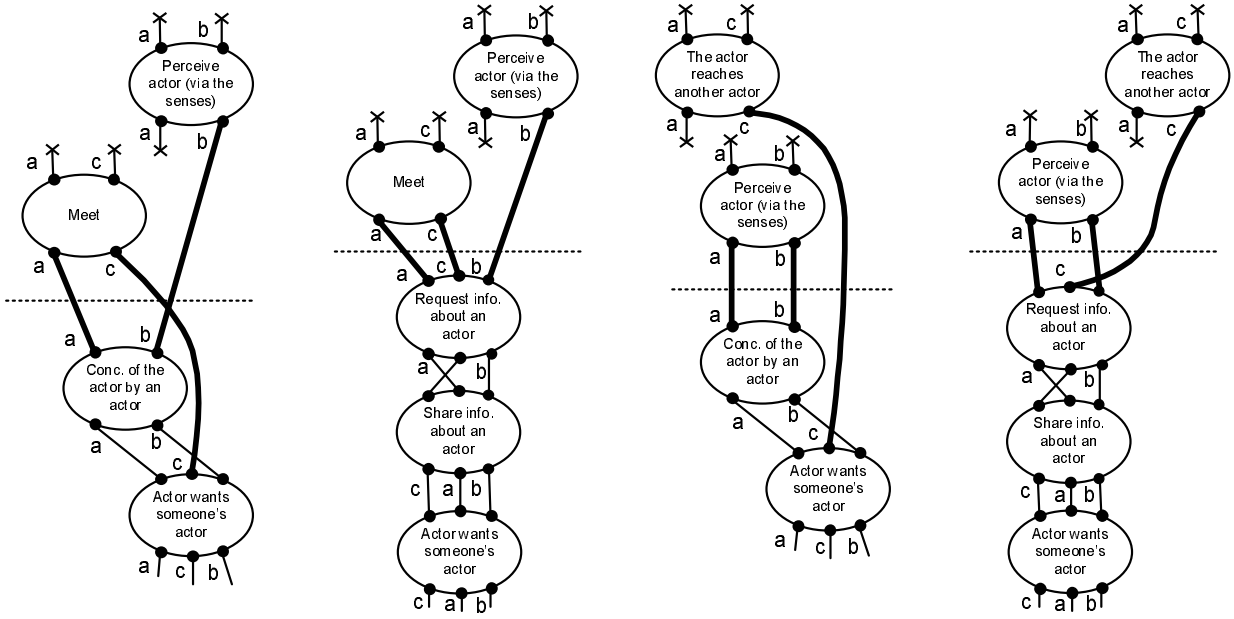


Figure 4.10: Some of the constituent transforms from the supertransform named "discovery of an actor, its possession status, and the desire to get that actor".

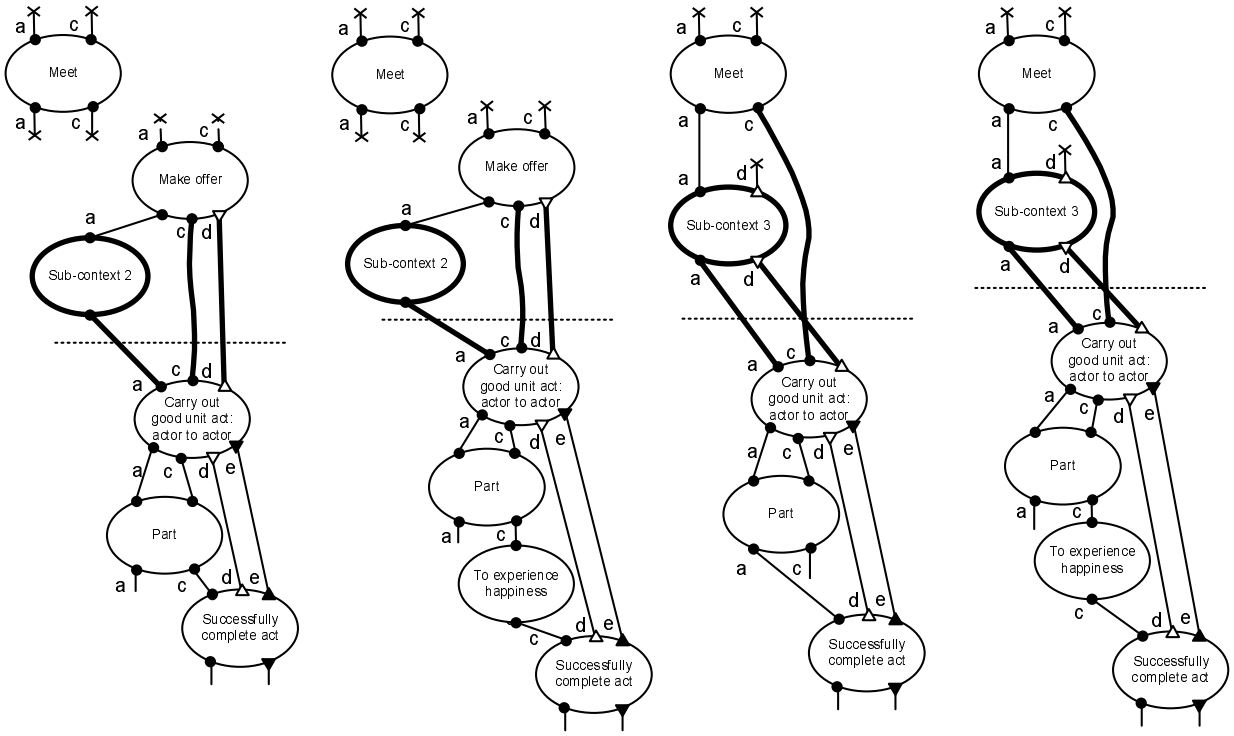


Figure 4.11: Some of the constituent transforms from the supertransform named "good act and its immediate consequences".

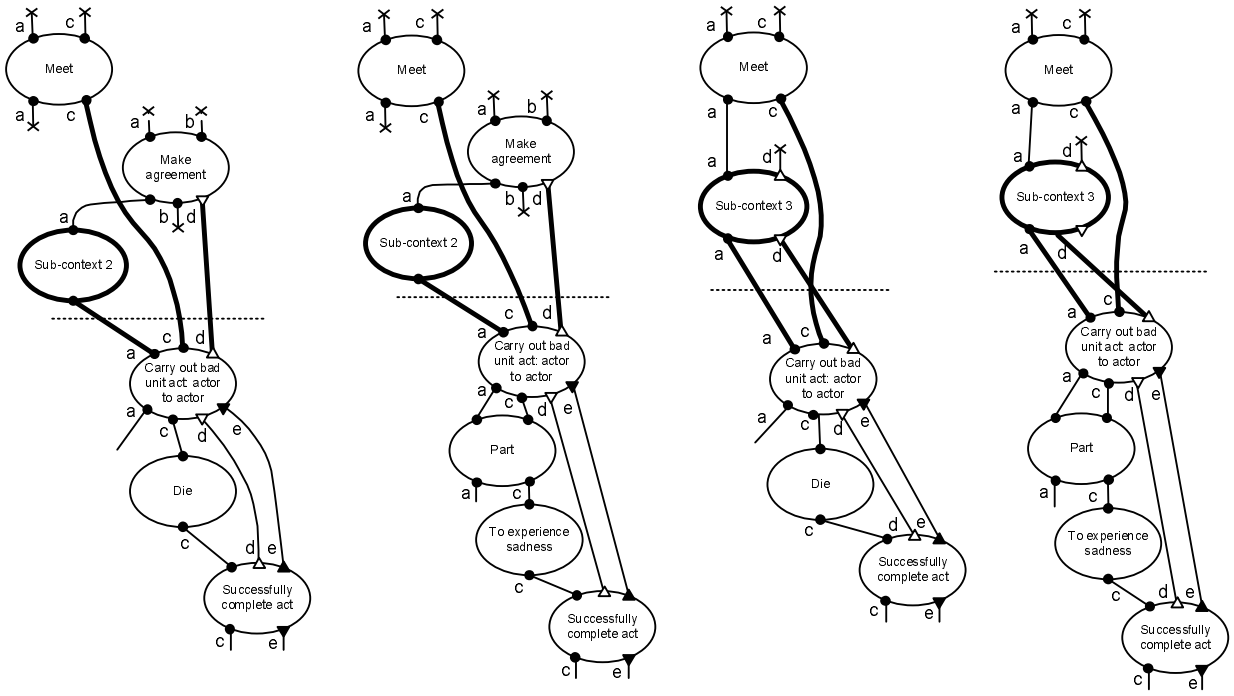


Figure 4.12: Some of the constituent transforms from the supertransform named "premeditated bad act and its immediate consequences".

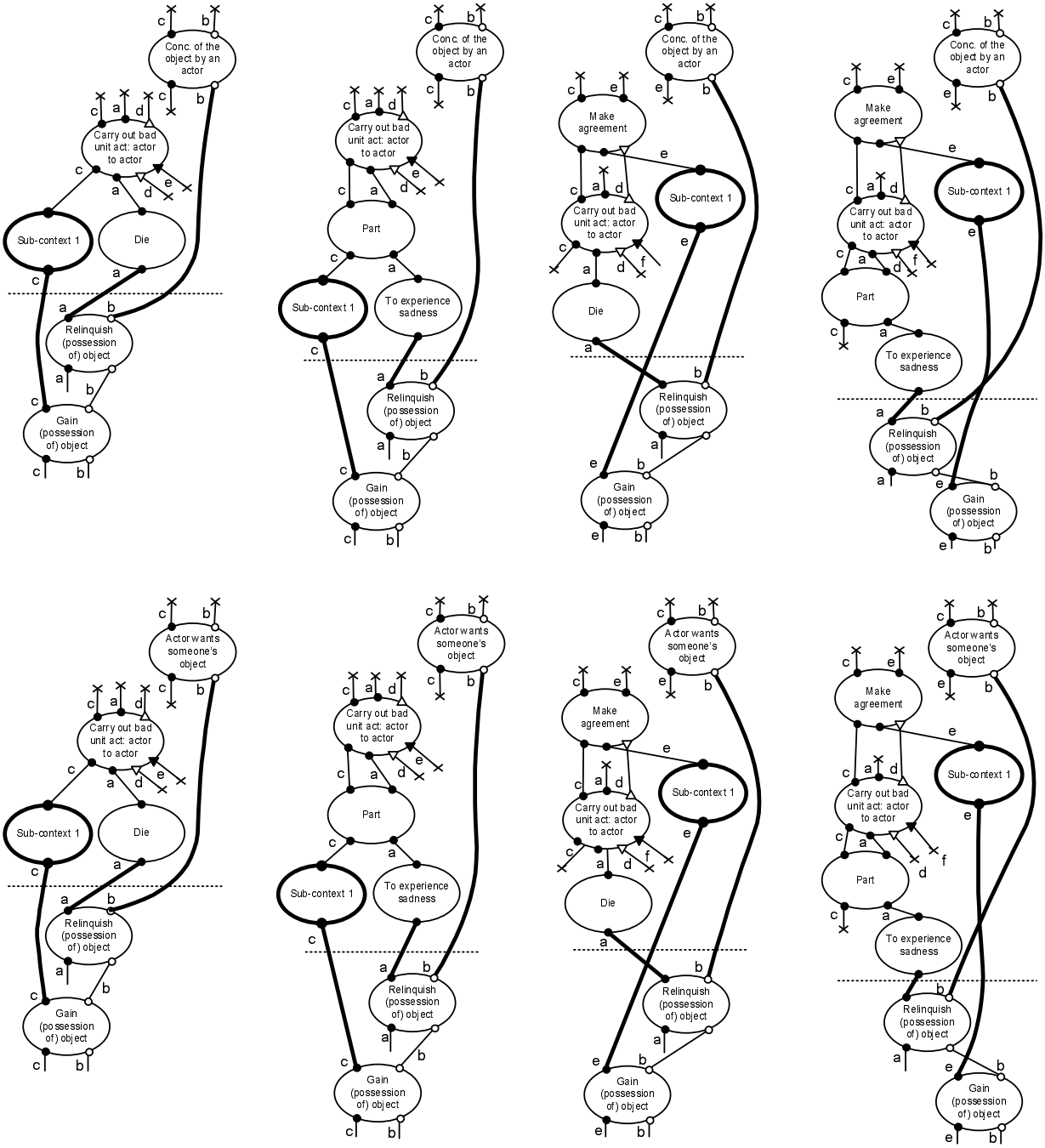


Figure 4.13: Some of the constituent transforms from the supertransform named "premeditated taking away an object".

We believe that any reader of fairy tales accumulates his/her cognitive experience in the form of (inductively learned) supertransforms. In particular, the appearance of a context (from some previously learned supertransform) in a new fairy tale may trigger the anticipation of the corresponding body, thus, transforms become a "working memory" for anticipation in a story. Certain transforms get associated with specific character types, such as a hero, villain, or magical helper. Moreover, as readers become more and more experienced with fairy tales, they build up more relevant supertransforms, and are able to "recognize" the appearance of various contexts more regularly, consequently, allowing them to predict the stories with greater accuracy.

Finally, one should note the fundamental differences between Chomky's formal grammar model and the ETS formalism. The differences are manifold and it is sufficient to mention here just the most obvious but important one. It is related to the way in which the ETS representation is constructed. In ETS, transformations are directly present in the corresponding structs, while in the formal grammar model, the production rules that were used to construct a given string representation are not stored within the representation. This leads to the problem of having a combinatorial explosion of possible histories for a given string.

4.5 The second level primitives

We believe that the formalism's capability to address effectively various levels of fairy tale comprehension is decisive feature in favor of the formalism. Moreover, the levels effect (as mentioned in section 2.2.2) can be explained by the ETS multi-level representation. At higher levels, the details of the story are hidden while generalizations

about the story's plot become more transparent.

The next level primitives corresponding to the chosen initial level supertransforms are shown in Figure 4.14.

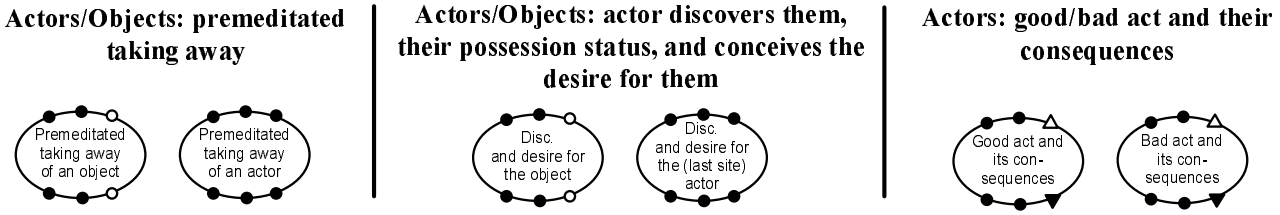


Figure 4.14: Next level primitives corresponding to the above initial level transforms obtained in the manner shown in Figure 3.11.

In light of the above, it is not difficult to see how the compression of information in ETS is accomplished. For example, the primitive "bad act and its consequences" now encapsulates several events that include the bad act itself as well as its immediate consequences, e.g. carrying out a bad unit act, parting of the main actors, and the experience of sadness by the actor on the receiving end. The concept of the supertransform allows one to realize this compression by accumulating the transform's variations encountered in the read fairy tales.

It is important to note that the quality of a fairy tale's comprehension by a reader is directly affected by his/her reading experience and ability to generalize from this experience. According to the ETS formalism, the latter ability is directly related to one's aptitude to form the relevant supertransforms or, the next level primitives.

4.6 An example of a second level struct

In Figure 4.15, we show the next level struct for the first fairy tale segment from section 4.3. Observe that there are less details present but, at the same time, the higher level plot becomes more recognizable.

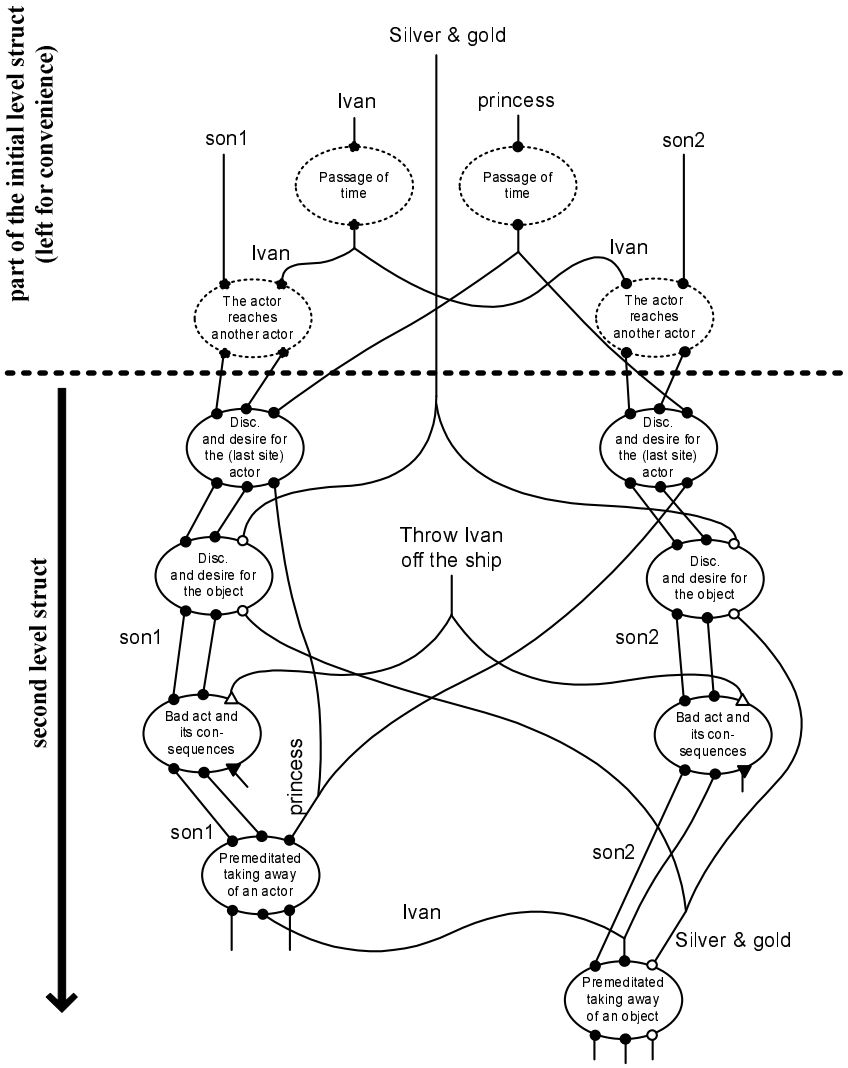


Figure 4.15: Second level representation of a segment from the fairy tale Salt given in Figure 4.5.

4.7 Examples of two second level transformations

Figure 4.16 shows two related second level transforms/events: "premeditated bad act followed by its goal of taking possession of the desired object" and "premeditated bad act followed by its goal of taking possession of the desired actor". They were constructed on the basis of the corresponding four *second level* structs: one of them is shown in Figure 4.15 and the other two (not shown) are the representations of the remaining above two story segments. Again, as was mentioned in section 4.4, these constituent transforms indeed appear more or less regularly in the four structs. It is also easy to see that, indeed, at this higher level, the overall plot becomes more transparent as compared to the previous level, and this should precisely be the goal of a good representational formalism for document classification and retrieval, since, ideally, one wants to allow the document query to relate to *any* semantic level.

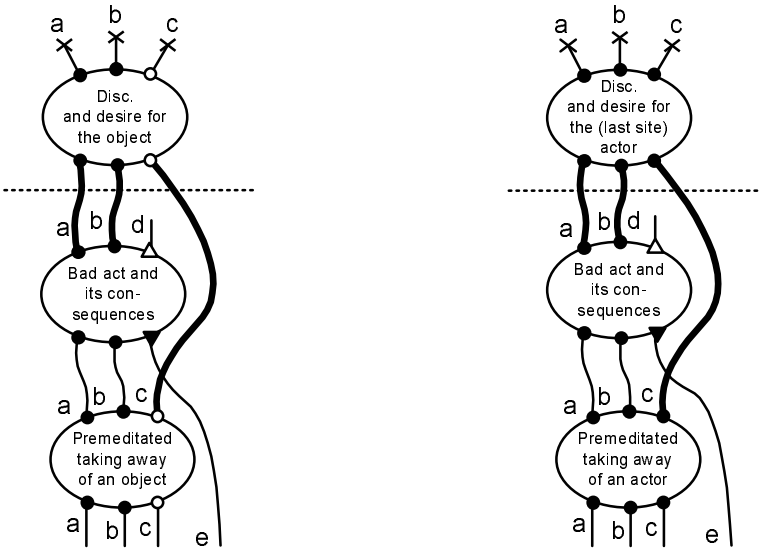


Figure 4.16: Examples of just two (of several) constituent transforms belonging to different supertransforms: "premeditated bad act, followed by its goal of taking possession of the desired object" and "premeditated bad act, followed by its goal of taking possession of the desired actor".

Finally, Figure 4.17 summarizes the hierarchial ETS representation of the corresponding third level primitive.

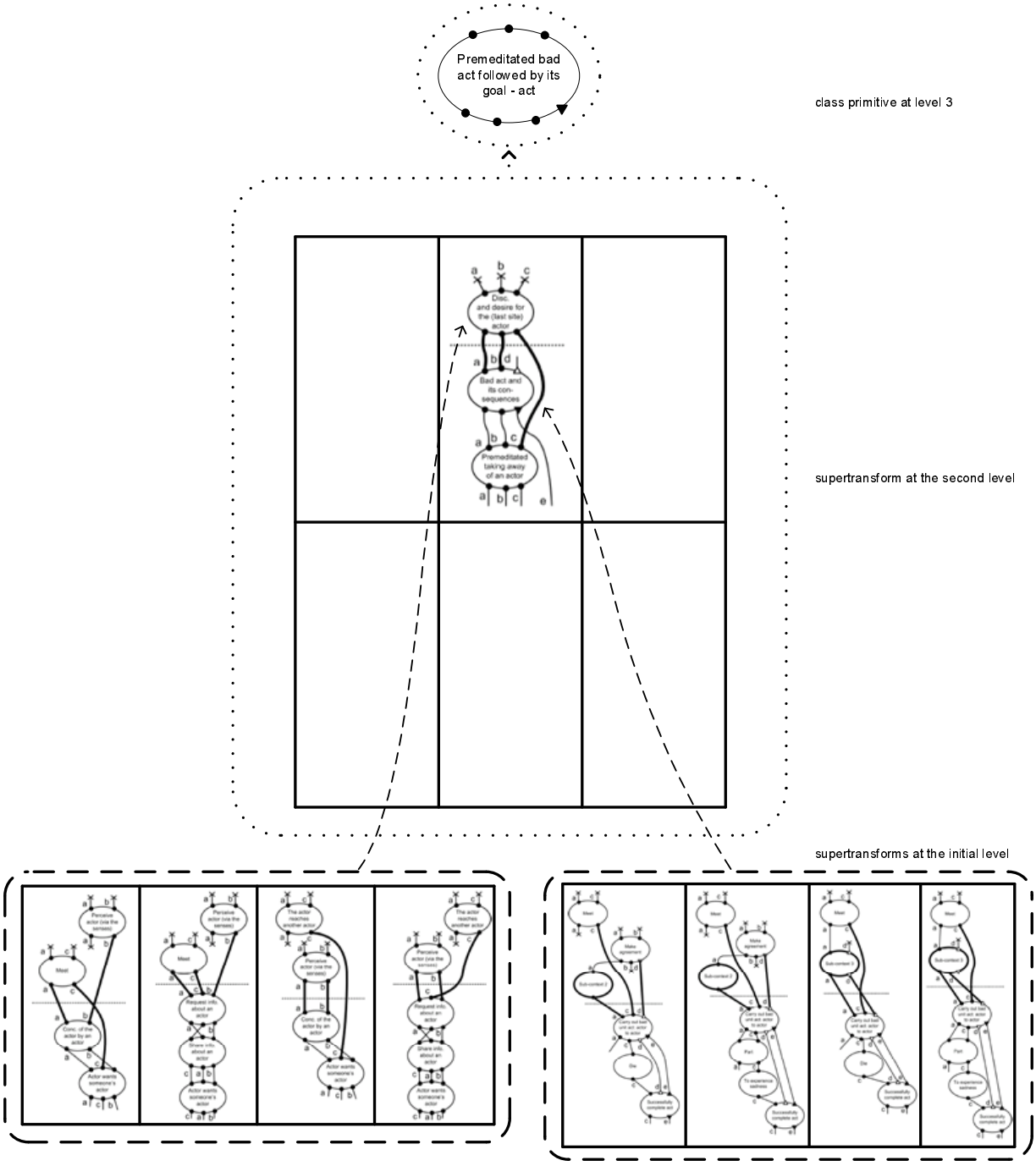


Figure 4.17: Pyramid view of a third level primitive: the pyramid should be thought of as being formed by the subordinate class supertransforms.

4.8 Psychological validity

As a validation criteria for the proposed document representation, we chose psychological validity, which has been used by various other researchers to guide the development of their models at the pre-implementation stage (see [26, 40]).

4.8.1 Levels effect

The levels effect has already been briefly discussed in section 4.5 and in Chapter 2. The levels effect is the observed phenomenon that people are more likely to remember high-level information about a story rather than specific details [29]. In ETS, this is modelled by the multi-level class hierarchy, where higher level classes encapsulate more general ideas about the story, and lower-level classes correspond to specific chains of events. At the lower levels, changes are continually occurring, but at a higher level, such as the class of Russian fairy tales, the time between salient events or structurally meaningful changes, is much greater.

4.8.2 Anticipation

Within this representation, ETS models anticipation through the inductively learned supertransformations [13]. The context corresponds to certain preconditions that one recognizes in the story; once these preconditions have been recognized, the reader anticipates the corresponding event bodies. As readers read more stories, they learn more of these supertransformations, and become more accurate in their anticipation or predictions about what will happen further in the story.

4.8.3 Generation

The inductively learned supertransformations allow a story reader to generate new stories corresponding to particular classes. Since, in ETS, the class representation is a set of structurally similar supertransformations, in order to create a new element of this class, a particular person must simply apply the events for the context and body of a transform. This element (some sequence of sentences), now becomes a specific instance of a transform within the supertransformation.

4.8.4 Summarization

The multi-level class hierarchy models the process of summarization. Since the hierarchy corresponds to different levels of representation for the story, a reader is able to jump to a specific level and generate information about the story corresponding to that level. This allows the learner to generate a summary about the story with varying degrees of detailed information.

4.8.5 Memory retrieval

Remembering a story involves a compression of the information or details of the story [26]. Generally, humans do not have perfect recall of a story, they only remember higher-level information (corresponding to the levels-effect), therefore we need a representation that models this compression [13, 26, 29, 40]. In ETS, the multi-level hierarchy corresponds to this compression of information. With each level climb, the representation for a specific story becomes smaller and more condensed, lower-level

details are lost, but higher-level information about the plot becomes more transparent.

4.8.6 Reminding

Reminding is closely related to memory retrieval as well as anticipation. Schank explored how different experiences may remind us about other experiences, and how one story may remind us of another story. He suggested that it's the story's structure that is responsible for this reminding effect, where, once we recognize a pattern, other stories that also used that pattern or structure will be brought to mind [26].

In ETS, reminding is modelled through the inductively learned supertransformations. Once a new transformation is learned from a particular fairy tale, a story reader may see the appearance of that transformation's context in a similar story. This may trigger a reminder about the story where that context first appeared, bringing to mind the stories or experiences where that context appeared before.

4.8.7 The Frame Problem

In Chapter 2, I briefly discussed the Frame Problem and some related problems intrinsic to symbolic models. One of the reasons traditional symbolic models suffer from these issues is because they need to assume a large amount of default information about given situations [19]. By assuming this default information, they are unable to accurately represent the unexpected, which is a very common feature of fairy tales. However, in ETS, the only assumptions that are made are with respect to the initial

level primitives. Provided the initial level primitives are chosen well, meaning that they are general enough to represent most underlying events that may take place, the unexpected can be inductively learned through supertransformations provided the unexpected is regularly occurring and not noise.

4.9 Comparison to other representations

It is interesting to note some similarities between ideas about fairy tale representation based on the ETS methodology and other, previously discussed, story representations. As mentioned in Chapter 2, Schank believed that one could discover and define sets of primitives that would form the basis for language representation. Also, he believed that the representation of similar sentences, regardless of the language, should be the same. These ideas are very similar to the ones proposed by the ETS formalism. Specifically, using the representation of fairy tales discussed in this chapter, similar fairy tales should have a similar representation and the representation does not depend on the language of the story. The main difference between this work and Schank's is that Schank was not working from a general and formal model of object representation. He chose to localize his ideas specifically to language. Without a formal model, it was difficult for researchers to apply his ideas to other areas outside of language or to evaluate his work on language. Moreover, his representations were not multi-levelled and although he represented concepts through the semantic interconnections of more primitive concepts, he did not formally introduce any ideas about capturing temporal information.

Secondly, Propp also believed that stories, specifically fairy tales, had an underlying

structure that could be characterized through a set of primitives or functions (as he called them). His work, even more so than Schank's, is similar to the ideas within this thesis. However, as mentioned in section 4.2, his functions relied on a fairy tale understanding too high level for the initial level of representation and he was never able to formalize his ideas about representation in order to apply them to fairy tale classification.

Thirdly, story grammars, which were partly based on Propp's work, share similar ideas to this thesis and ETS. Story grammars describe structure, and the hypothesis of its practitioners is that all stories have an underlying structure that is relatively invariant. Also, story grammars support a multi-level view of stories, which is also supported in the ETS framework. Again, the main difference between story grammars and ETS is that ETS is a general model for object representation, specifically designed to handle the fundamental issues of representation and class description. Conversely, story grammars apply Propp's ideas about structure with traditional grammar notation. Thus, they are constrained by the limitations of the formal grammar model, meaning that they must hard-code all "transformation" information in the form of production rules, which means there is no discovery or learning of new features of a class.

In Chapter 2 I mentioned that some researchers see statistical approaches, which rely on vector-based representation, as a way around issues such as the Frame Problem. This maybe true, however, vector-based approaches introduce their own problems with respect to text or story representation. For instance, in stories the sequence of events, i.e. the causal or temporal order of events, is very important to the understanding and plot representation of a story. In [19], Haplin attempted to use statistical approaches to analyze how accurately students were able to rewrite a story they had

read. He found that the lack of a temporal order in the vector-based representation led to inaccurate evaluation of the student's recall. For instance, in his original story, the main character turns himself into an elf in order to be small enough to ride on the back of a goose. Recalling this particular order is critical to the understanding of the plot, however, if a student rewrote this as the main character getting on the back of the goose, and then turning himself into an elf, a vector-based approach would treat this the same as the original story.

Finally, I should mention a few important benefits of the ETS representation. Firstly, an ETS struct provides a "snapshot" of the real object, and with this snapshot we can see everything about the object, i.e. the events that took place to construct it. Secondly, the ETS notion of class representation allows one to see how object's within a class are constructed. With a class representation we should be able to generate new objects belonging to this class. This is a very powerful feature of ETS, one which is not possible within the vector-space-based formalism.

4.10 Limitations of current representation

There are several limitations of the current set of primitives, which lead to limitations in the representation for fairy tales. Firstly, most fairy tales begin by explaining the setting of the story and some of the main characters. For instance, the fairy tale Salt begins with: "In a certain city there lived a merchant who had three sons: the first was Fyodor, the second Vasily, and the third Ivan the Fool." The first paragraph then goes on to explain that the merchant lives richly, and that he does not trust his youngest son. The current set of initial level primitives are not designed well

for handling this setting information; this is partly due to the first paragraph of a story consisting of so much condensed information. Also, the setting is a narration, which sometimes explains information from the past, which is also currently difficult to represent.

Secondly, there appear to be two issues with the sites of the primitives. The first one is that we may be loading too much information into a site. That is, a site currently corresponds to a conceptual entity involved in particular events, and the site itself is unstructured. If we depend on loading too much information into a site, we'll lose some structural meaning in the story. The second issue is that sites remain the same throughout all levels. This could lead to strange representations as we ascend higher and higher levels, creating more and more abstract primitives. The same sites used at the initial level, will still be the same conceptual entities used in these very abstract levels, however, it may not make sense to "think" about these abstract primitives involving very concrete sites such as an actor.

The final issue is with the initial level primitives. We chose to keep the primitives as simple as possible, however, it may be interesting or even necessary to expand the set of primitives. For instance, we chose to only represent two changes in emotion, that is, "Experience sadness" and "Experience happiness", however, the plot representation may benefit from expanding this set to include a larger range of emotional changes, such as envy and fear. Moreover, in this initial application, we tried to represent various actions, such as fight, steal, and insult as classes, which decompose to initial level primitives, typically involving the primitives "Carry out unit act", "Carry out good unit act", and "Carry out bad unit act". However, there appears to only be a small set of low level actions that occur in fairy tales, therefore, it is worth exploring

using these low level actions directly as primitives and eliminating the three classes of unit act primitives.

Chapter 5

Fairy tale retrieval

The focus of this thesis has been primarily on data representation, specifically, the representation of fairy tale segments. In this chapter, I tie together the representation construction procedure (preprocessing step) with the classification of an IR query to a database of fairy tales.

5.1 Preprocessing fairy tale segments

In the previous chapter the ETS structs were constructed assuming that a preprocessor had already performed the event recognition in order to identify the appearance of the fairy tale primitives. In this section I give a brief overview of what this preprocessor needs to do in order to perform this event recognition and discuss some of the current techniques and tools in NLP that can be used in order to accomplish this recognition step. Finally, I discuss and show the results of an experiment using an

information extraction tool called AutoSlog-TS [33]. This experiment was conducted to demonstrate how to use these existing tools, and to also give a starting point for the future preprocessor development.

5.1.1 Information Extraction

Information extraction (IE) is the process of automatically extracting pre-specified sorts of information from short, natural language text [48]. IE is distinctly different than IR, however, they are complementary. IR is concerned with locating relevant documents based on a query, while IE is concerned with analyzing text to recover pre-specified information that the user may be interested in. Moreover, IE is interested in the structure of the texts, whereas, traditionally in IR, texts are treated as bags of words [45]. Very recently, some NLP research groups have been interested in improving the performance of traditional IR techniques by coupling them with IE in order to achieve more meaningful representations of the searched documents. With the ETS representation, I need an IE layer to extract the primitives as they appear in the text, allowing me to construct the struct representation of the story. Provided the ETS primitives are simple enough, existing approaches to IE should be able to preprocess the documents in this way. Similar work is discussed in [20], where Haplin et. al used existing NLP components to build a preprocessor to extract events from stories. The event sequences were compared to student's rewritten versions of the story, in order to evaluate their understanding of the plot (measured by their recall about the flow of events).

It is very important that the primitives are chosen so as to minimize ambiguity as

much as possible, meaning that they must be sufficiently low-level enough for current IE techniques to preprocess effectively. Original IE systems were constructed using finite state machines (FSM) and hand-crafted pattern extraction rules. These systems required the developer to not only be an expert in NLP, but also be a domain expert, as the rules had to be specifically tailored based on the document domain. This type of IE system performed very well, but could not be used on more general document corpuses or recognize complex events. Provided my initial level primitives are simple enough, a FSM may be a sophisticated enough approach for the preprocessor.

Since the original FSM approaches, there has been a lot of work in NLP in the area of information extraction. IE is now considered to consist of five tasks, *named entity recognition*, *coreference resolution*, *template element construction*, and *scenario template production* [11].

Named entity recognition (NE) identifies different entities that appear in the text, such as "Ivan" or an "axe", and classifies them as corresponding to a specific class, such as an actor or an object. Coreference resolution (CO) attempts to discover which references should be associated with a named entity, such as "him" referring to "Ivan" and "it" referring to "Ivan's axe". This type of recognition is essential to identifying the site types involved with each appearance of a primitive event in the story. NE is much simpler than CO and NE systems perform comparable to human level accuracy, while CO is still domain dependent; and the best systems perform around 51% recall and 71% precision [11].

Template element construction (TE) and template relation construction (TR) have a similar relations as NE and CO. TE adds descriptive information to NE results, and TR finds relations between TE entities. For instance, TE may discover that Ivan is

short and wears a red hat, while TR may discover that Ivan works as a mason. The best IE systems are able to perform these tasks at 80% accuracy, while humans have a performance level of around 93% [11].

Finally, the most important and difficult task is scenario template production (ST). ST recognizes when events take place, and what entities participate in the events. Thus, this part of the preprocessor would recognize when the event "Gain (possession of) an object" has taken place, and associate with that event the actor and object participating in the gain. The best IE systems perform at 56% accuracy while human level accuracy is around 81% [11].

It is important to note that the levels of accuracy for each IE task given above are based on results from MUC-7 (Message Understanding Conference). Both the type of events and documents were much more complex than the ETS primitives given in Chapter 4 and the fairy tales making up our document domain. Thus, it is difficult to gage the performance of the hypothetical preprocessor at this time. However, one of the advantages of ETS is that as the preprocessor is developed, the primitives can be modified or re-worked by the application designer to correspond to an appropriate level in which the preprocessor can perform the event recognition accurately.

5.1.2 Existing tools

NLP researchers have been working towards developing IE tools that are not domain specific, or can be trained for a particular domain. The training data sometimes consists of manually tagged text, annotated text, or pre-classification of the relevant sentences in which you wish the IE tool to extract a pattern from. There are many

existing IE tools; here I only briefly discuss some of the most important ones.

AutoSlog [33] builds a dictionary of extraction patterns, each extraction pattern consisting of a *conceptual anchor* that activates it and a *linguistic pattern*. The conceptual anchor is chosen from the training set based on a set of heuristics, and the linguistic pattern represents a phrase or a set of phrases that are likely to be good for concept activation. The original AutoSlog system needed as training data a set of annotated and tagged text, however, a new system has now been developed called AutoSlog-TS. The AutoSlog-TS system extracts patterns from free text, without any annotation or tagging. The training data consists of a set of sentences representing the same event, then the AutoSlog-TS system creates a large set of extraction patterns based on the training data. The original AutoSlog system performed at 98% accuracy as compared to hand-crafted rules, and it has been shown that AutoSlog-TS has similar performance.

PALKA [23] learns extraction rules that are similar in form to the AutoSlog system. Instead of the conceptual nodes that AutoSlog uses, PALKA learns extraction rules that are expressed as *frame-phrasal pattern structures* (FP-structures). Also, instead of using a set of linguistic patterns in order to extract the patterns, PALKA uses a concept hierarchy, which is a set of predefined keywords that can be used to trigger a pattern. Moreover, PALKA can use its concept hierarchy to match FP-structures via relationships within the hierarchy, thus allowing it to extract events that are not necessarily exact matches.

Finally, CRYSTAL [41], generates *multi-slot concept nodes*. CRYSTAL's extraction patterns are considerably more sophisticated than both AutoSlog and PALKA. CRYSTAL begins by creating very specific extraction rules or concept nodes based on the

phrases in the training data. Afterwards, it traverses each concept node; and for each node it finds the most similar existing node, and relaxes the constraints of each such that both rules become merged into one rule. The new rule is tested against the training data, and provided its error rate is less than some set threshold, the new rule is added to the set of concept nodes, and the original two rules are removed. This process continues for every concept node until all rules are made as general as possible without significant error.

5.1.3 The experiment

The following experiment was conducted using the AutoSlog-TS system. These experiments were conducted to demonstrate that existing tools can be used to recognize the base level primitives. A full preprocessor could use these existing tools, with modification, to build the fairy tale representations. Also, these existing tools could be used to validate the selection of base primitives, as easier to recognize primitives generally make better base level events.

In the first experiment, I trained AutoSlog-TS to extract the event "Meet" from a set of unseen fairy tale paragraphs. To train the system, I used 69 relevant pieces of text and 63 irrelevant pieces of text that I collected from 11 different stories found in [1, 2]. Based on this training data, AutoSlog-TS created 687 case frames (see Figure 5.1 below). Most of the generated case frames are not very useful as they only occur occasionally or are not appropriate, thus, it was necessary to filter the case frames so that only the best are kept. A small C++ program (see Appendix A) was then used to filter the case frames based on the statistics AutoSlog-TS generated for each case

frame. After the filter program was run, only 31 case frames remained.

```
CF:
Name: infinitive_verb_<dobj>__MEET_14
Anchor: VP1(MEET)
Act_Fcns: infinitive_verb_p(VP1(MEET) )
Slot:  dobj
Stats: frequency = 3.000
       relativeFreq = 3.000
       cond_prob = 1.000
       rlog_score = 1.585
```

Figure 5.1: Example case frame generated by AutoSlog-TS, which represents an extraction pattern corresponding the event "Meet".

Using these 31 case frames, the AutoSlog-TS information extraction tool was used on 20 different paragraphs coming from four different fairy tales (5 from each) that were not part of the training data. The results are shown below in Table 5.1. The first column represents the story number, the second column represents the total number of "Meet" events existing in that story's test data [1], the third column represents the correctly identified "Meet" events by AutoSlog-TS, the fourth column is the total number of "Meet" events found by AutoSlog-TS, the fifth column is the percentage of correctly identified events by AutoSlog-TS (CI/ME), and finally, the last column represents the percentage of events recognized by AutoSlog-TS that were correct (CI/TI).

[1]These "Meet" events were counted by hand, for each of the 20 paragraphs, before the test was run.

Table 5.1: Meet event extraction results.

Story	ME	CI	TI	CI/ME (%)	CI/TI (%)
1	5	4	9	80.0	44.4
2	5	5	11	100	45.5
3	6	5	11	83.3	45.5
4	10	7	16	70.0	43.8
Total	26	21	44	80.8	47.7

ME - Number of Meet Events

CI - Correctly Identified Events

TI - Total Events Identified

We see that AutoSlog-TS correctly identified 80.8% of the "Meet" events, which is very promising. However, AutoSlog-TS incorrectly recognized some "Meet" events, that is, it "thought" some "Meet" events occurred in some places of the text where in fact there was no such event occurring. Thus, out of all the "Meet" events recognized, only 47.7% actually corresponded to correctly identified events. There are several reasons why AutoSlog-TS performed badly in this respect, however the main reason was due to confusion over sentences similar to the following: "Ivan arrived in the forest." The system recognized this as a "Meet" event, because structurally it is very similar to sentences such as: "Ivan arrived in front of the King." To deal with this issue, the full preprocessor could potentially use a more sophisticated approach to verifying event instances. For instance, a "Meet" event must involve two actors, however, the first sentence given above only involves Ivan and an object, which is the

forest. Due to this, no "Meet" event should occur, and this would be easy to filter for using named entity recognition and coreference resolution.

In the first experiment, a reasonably small set of relevant and irrelevant text was used to train the data. According to Dr. Ellen Riloff, one of the creators of AutoSlog-TS, typically the training set consists of several hundred examples. A second experiment was conducted, using the "Meet" event again, but this time with a slightly larger training set (89 relevant, 84 irrelevant). Also, the new training data was added with the consideration of attempting to improve the results from the first experiment, that is, cut down on the number of misidentified events. Since the training data was modified based on the learned information from experiment one, the test data had to be expanded to include data from four new stories in order to not bias the result. This time, AutoSlog-TS generated 793 case frames, which was then filtered to only 36, based on the same filtering program as before. The results are displayed below in Table 5.2. The columns represent the same results as the previous experiment.

Table 5.2: Meet event extraction results based on second training data.

Story	ME	CI	TI	CI/ME (%)	CI/TI (%)
1	5	4	8	80.0	50.0
2	5	5	7	100	71.4
3	6	5	9	83.3	55.6
4	10	9	14	90.0	64.3
5	9	9	15	100.0	60.0
6	10	9	17	100.0	52.9
7	6	6	10	100.0	60.0
8	3	3	10	100.0	30.0
Total (1 to 4)	26	23	38	88.5	60.5
Total (5 to 8)	28	27	52	96.4	51.1
Total (1 to 8)	54	50	90	92.6	55.6

ME - Number of Meet Events

CI - Correctly Identified Events

TI - Total Events Identified

Once again we see that AutoSlog-TS performed very well in correctly identifying existing "Meet" events. The original test data, stories 1 through 4, had an increased accuracy, and overall, all 8 stories had an accuracy of 92.6%. Also, not surprisingly, the percentage of events recognized to be correct versus the total number of events recognized in the original training data improved as well. Overall, the proportion of correctly identified events to total events recognized is 55.6%.

The final experiment conducted was to use AutoSlog-TS to extract a different primitive from the fairy tale data set. In this experiment, the event "Metamorphose" was used. This event was chosen as it is a reasonably rare event compared to "Meet", and should provide further indication of whether AutoSlog-TS can correctly identify the primitive events. Results are given below in Table 5.3. AutoSlog-TS was trained using 37 relevant and 103 irrelevant examples. The system generated 673 case frames based on this data, and using the same filter program as before, plus hand removing some case frames, a total of 13 case frames were used against a data set consisting of 44 fairy tale paragraphs taken from 15 different stories. Within these 44 paragraphs, there were 18 instances of the event "Metamorphose". In the table below, the first column represents the total number of "Metamorphose" events that actually existed in the data set. The second column represents the total number of correctly identified events. The third, represents the total number of events AutoSlog-TS thought were real events. The fourth column is the percentage of correctly identified events by AutoSlog-TS, and finally, the last column represents the percentage of events recognized by AutoSlog-TS that were actually correct.

Table 5.3: Metamorphose event extraction results.

MME	CI	TI	CI/MME (%)	CI/TI (%)
18	14	22	77.8	63.6

MME - Number of Metamorphose Events

CI - Correctly Identified Events

TI - Total Events Identified

All three experiments give encouraging results that a preprocessor can be developed in the future, using existing NLP tools and methodologies, to extract the ETS fairy tale primitives. Results for AutoSlog-TS could be improved by hand-crafting some case frames, as well as using a fairy tale dictionary rather than a generic dictionary. Finally, results may also be improved by increasing the size of the training sets and also by gaining more experience with the AutoSlog-TS system.

5.2 Retrieving relevant fairy tales

Now that the preprocessing step has been discussed, we can discuss the process of receiving a query from a user, and retrieving the relevant fairy tale segments. The first step in this process is to preprocess all fairy tale segments (offline), storing their ETS struct representation in a database. Once the struct representations are available, a user can query the system by typing in a natural language query. The preprocessor will convert this query into an ETS struct (online) and retrieve all relevant documents based on the structure of the set of classes present in the database (see Figure 5.2). In other words, the retrieval should be class based: the system should display ordered class elements from the relevant classes. Although the details of such ETS algorithms have not been developed yet, a preliminary outline is presented in Part III of [16] (also discussed briefly in Chapter 3), and is a topic for future research.

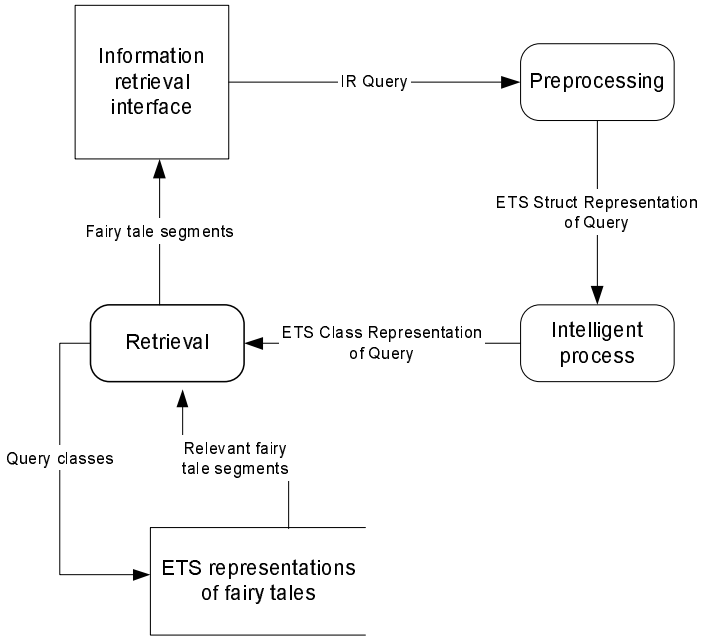


Figure 5.2: Data flow diagram of prototype information retrieval system for fairy tales.

In a prototype system of this process, the user could enter their query directly as an ETS struct, similar to the system being developed by InORB Technologies Inc. where queries are entered as graphs [47]. Moreover, a structural similarity measurement could be used between the document structs and the query struct, in order to simplify the retrieving of relevant documents. These two simplifications, along with the development of a preliminary preprocessor, would be enough to begin evaluation of the ETS IR system compared to traditional approaches.

5.3 Advantages

Throughout this thesis I have attempted to highlight representational advantages of the ETS formalism as compared to other existing formalisms. Here, I will briefly highlight some specific advantages the representation would provide in a fully implemented system over existing IR tools.

Firstly, due to the nature of the representation of the documents, it is relatively straight-forward to perform partial matching between a query and a fairy tale segment. This allows one to retrieve the relevant *piece* of a document corresponding to the query, such as a single sentence or paragraph, rather than only being able to retrieve an entire document. Secondly, the hierarchial class structure, and identification of transformations, could potentially allow the system to generate summaries of the retrieved segments, thus allowing a user to quickly locate the documents of interest. Also, because of the hierarchial class structure, we can quickly narrow down which documents the query corresponds to, by traversing the classes down through the hierarchy, and pruning on irrelevant sub-classes. Finally, since the representation is structural, users will be able to locate similar documents or parts of documents by structural comparison, rather than collections of keywords. I feel that all of these advantages will allow us to breath new life into IR, providing users with a much more powerful and sophisticated search tool.

Chapter 6

Conclusion and future directions

6.1 Conclusion

In this thesis, I have presented an initial application of the ETS model to the representation of Russian fairy tales. This work was motivated by my interest in information retrieval and what I felt was a need for a more powerful representation of IR documents. Using the ETS formalism and in-depth analysis of many fairy tales, I discovered around 40 ETS primitives that make up the underlying structure of events for any Russian fairy tale. These primitives correspond to the events that take place in the mind of the generic fairy tale listener/reader. With these primitives, I developed several example structs representing story segments belonging to the same class. Using these structs, I proposed several supertransformations corresponding to next level events in the conceptual representation of the stories. We believe that any reader of fairy tales accumulates his/her cognitive experience in the form of (inductively learned) supertransforms. Moreover, the next level events encapsulated by the

supertransforms correspond to the next level primitives, thus, allowing the representation to climb levels. With each level climb, the representation is compressed, but the overall plot becomes more transparent in the representation.

At this, still early stage, it appears to us that the ETS (event-based) representation is eminently suitable for representing fairy tales. This is mainly due to a good match between the ETS primitives and the fairy tale's elementary events, as well as due to the resulting explicitness of the representation. The good match is assured by the fact that the ETS formalism is *event-based* and allows one to capture temporal and structural relationships between events. Moreover, the concepts of transformation and supertransformation allow for a very natural introduction of levels of fairy tale representation, without which, it appears, no intelligent information system is possible. The structure of ETS, including the presence of levels, makes the advantages of this formalism over other conventional information retrieval formalisms quite apparent: the query can now be directed towards the right level and the right class.

Information retrieval has always been a result driven area, however, performance levels have not increased as much as one would expect in the past few decades [46]. Some feel that we need more sophisticated statistical techniques for classification and feature selection as well as more computational power, but I feel that it's a question of representation, not computational power or statistics. Many researchers are now looking to techniques in NLP to improve the performance of their IR algorithms, essentially attempting to create a richer representation of the documents within the corpus. I think this is interesting, as the unification of models is always interesting in science. However, we need a representation that will maintain the structure of the documents after preprocessing because this temporal history is essential to a powerful

representation of the corresponding class.

The central idea of ETS is that objects can be described by their formative history. Any application of ETS must begin by postulating a basic/initial level of representation (as done in Chapter 4). All higher levels can be constructed by the inductive learning algorithm, where classes are specified by the description of their representational levels (i.e. supertransformations). This class description is very powerful, as it provides a transparent view of how objects within the class can be constructed. This inductive representation allowed the formal specification of ETS (see [16]) to unify several pattern recognition (PR) concepts such as classification, generation, and structural representation. The unification of so many central ideas in PR makes the model worth exploring for many applications even though all of the proposed ideas within the formalism are still very new.

In [15], Golubitsky states that "The 'proof of concept' for the proposed model [ETS], of course, depends on the success of its applications." The application of ETS discussed in this thesis, is one of the first such applications of the ETS model. Unfortunately, it has not reached a completeness such that it can be compared via performance levels to other existing tools in this area. However, although the ETS model is still in its infancy, it was the right time to proceed with an application, as it is through applying the model that we are able to evaluate and critique the formal specifications. For instance, due to this application and a few other initial applications of ETS, we were able to discover several deficiencies with the current formalism.

The deficiencies have to do with the idea of a "site". Formally, the term *site* was never defined, thus, it is unclear in the current formalism what a site really is. In

this thesis, I used sites as conceptual entities involved in the perception of the story, however, without any formal definition of site I could potentially use very strange sites. Also, any representation problems I experienced may be due to the misuse of sites, rather than the formalism. Moreover, in the current formalism, as shown in Chapter 4, sites do not change as we ascend levels. This leads to a somewhat strange representation at higher levels, as higher level concepts will still involve the same initial level conceptual entities.

Work has begun on a revised version of the formalism, one that defines the concept of a site. It is important to note that due to the new definition of site (which I will not discuss here) the model has been simplified. Also, the new definitions further constrain the initial level selection of primitives, making it easier to determine "good" primitives. Finally, learning appears to be simpler in the newer version, which is very important, as ETS has always claimed that powerful representation makes learning easy. All of these new ideas arose by observing the behavior of the initial applications of the model.

Finally, although a newer version of the model is being researched, it appears that much of the work discussed in this thesis is still applicable in the new version. I feel that with some modification, the representation proposed here can be used with the new version. Also, I feel that a fully working implementation will be possible within the near future and hopefully the work I presented in this thesis is enough for other researchers to see that this is indeed true. Also, hopefully, I have presented enough arguments for the ETS model to convince other researchers (working in computational linguistics, IR, or other areas of PR) that this formalism is worth investigating and applying to their own research areas. As more people research and apply the

model, it will be easier and faster to apply to new domains. Also, the model's formal specification, inductive learning algorithm, and proposed concepts can be improved through the research and critique of other scientists.

6.2 Future directions

The concentration of this thesis has been on the representation of documents for the purpose of IR. The thesis has proposed many new ideas and approaches to IR based on the ETS formalism, however, due to how new the formalism is, a prototype of the proposed system has not been developed yet. In this section, I list some of the future work and directions that I feel this research could go in. I think there are enough new ideas here and within ETS to keep any researcher occupied for most of his/her career.

- Revise representation for the new version of the ETS model.
- Investigate how to represent the association between events/words occurring in a story and real-world knowledge.
- Develop IE layer/preprocessor to extract ETS primitives from document corpus, possibly adjusting ETS primitives to improve accuracy of the preprocessor.
- Create fully working IR prototype using the intelligent process to perform classification of queries and document structs, or using a simplified structural similarity measure.
- Evaluate prototype against statistical based IR tools.

- Investigate applying ETS to general document and language representation.
- Investigate methods of improving IE systems by using the ETS formalism to decompose higher-level event structures into lower-level, more easily recognizable, primitive structures.
- Explore other areas of NLP using ETS, such as summarization, generation, and translation.

Bibliography

[1] Aleksandr Afanas'ev, *Russian Fairy Tales*, Pantheon Books, New York, New York, 1945.

[2] Alex E. Alexander, *Russian Folklore*, Nordand Publishing Company, Belmont, Massachusetts, 1975.

[3] James Allen, *Natural Language Understanding*, The Benjamin/Cummings Publishing Company, Menlo Park, California, 1987.

[4] C. Apte, F. Damerau, S. M. Weiss, Automated Learning of Decision Rules for Text Categorization, *ACM Transactions on Information Systems (TOIS)*, Volume 12, Issue 3, pages 233 - 251, 1994.

[5] J. B. Black, G. H. Bower, Episodes as Chunks in Narrative Memory, *Journal of Verbal Learning and Verbal Behaviour*, no. 20, pages 267 - 275, 1979.

[6] J. G. Carbonell, *Subjective Understanding: Computer Models of Belief Revision*, Ph.D. Thesis, Technical Report TR-150, Department of Computer Science, Yale University, 1979.

[7] Eugene Charniak, Ms. Malaprop, a language comprehension program, *Proceedings of the National Conference on Artificial Intelligence*, AAAI-77, Cambridge, Massachusetts, 1977.

[8] Jim Cowie, Yorick Wilks, Information Extraction, *In R. Dale, H. Moisl and H. Somers (eds.) Handbook of Natural Language Processing*, New York, 2000.

[9] Richard E. Cullingford, M. W. Krueger, M. G. Selfridge, M. A. Bienkowski, Automated Explanations as a Component of a Computer-Aided Design System, *IEEE Transactions on Systems, Man and Cybernetics*, SMC-12, no. 2, pages 168 - 182, 1982.

[10] Richard E. Cullingford, *Natural Language Processing*, Rowman & Littefield, Totowa, New Jersey, 1988.

[11] Hamish Cunningham, *Information Extraction - a User Guide*, Research Memo CS-99-07, Department of Computer Science, University of Sheffield, 1999.

[12] Jerry Everard, *Introduction to Vladimir Propp*, http://mural.uv.es/vifresal/Propp.htm, Last Visit: October 25, 2004.

[13] Sean M. Falconer, David Gay, Lev Goldfarb, ETS Representation of Fairy Tales, *Proceedings of the ICPR 2004 Satellite Workshop*, "Pattern representation and the future of pattern recognition: A program for action", Cambridge, UK, August 2004.

[14] Steven Finch, Exploiting Sophisticated Representations for Document Retrieval, *Proceedings of the 4th Conference on Applied Natural Language Processing*, Stuttgart, Germany, pages 65 - 71, 1994.

[15] Oleg Golubitsky, *On the Formalization of the Evoling Transformation System*, Ph.D. Thesis, Faculty of Computer Science, University of New Brunswick, 2004.

[16] Lev Goldfarb, David Gay, Oleg Golubitsky, Dmitry Korkin, *What is Structural Representation? Second version*, Technical Report TR00-165, Faculty of Computer Science, U.N.B., March 2004.

[17] Lev Goldfarb, Introductory talk at the ICPR 2004 workshop, *Proceedings of the ICPR 2004 Satellite Workshop*, "Pattern representation and the future of pattern recognition: A program for action", Cambridge, UK, August 22, 2004.

[18] Dieter Grabson, Norbert Braun, A Morphological Approach to Interactive Storytelling, *Proceedings of the Conference on artistic, cultural and scientific aspects of experimental media spaces*, Bonn, Germany, 2001.

[19] Harry Haplin, *The Plots of Children and Machines: The Statistical and Symbolic Semantic Analysis of Narratives*, Masters Thesis, School of Informatics, University of Edinburgh, 2003.

[20] Harry Haplin, Johanna D. Moore, Judy Robertson, Automatic Analysis of Plot for Story Rewriting, *Proceedings of the Empiricial Methods in Natural Language Processing*, pages 127 - 133, 2004.

[21] Jerry R. Hobbs, The Generic Information Extraction System, *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, Morgan Kaufman, pages 87 - 91, 1993.

[22] Thorsten Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features, *Proceedings of the ECML-98, 10th European Conference on Machine Learning*, Springer Verlag, Heidelberg, DE, pages 137 - 142, 1998.

[23] J. Kim, D. Moldovan, Acquisition of linguistic patterns for knowledge-based information extraction, *IEEE Transactiops on Knowledge and Data Engineering*, vol. 7 number 5, pages 713724, 1995.

[24] W. Kintsch, T. van Dijk, Toward a Model of Text Comprehension and Production, *Psychological Review*, no. 85(5), pages 363 - 394, 1978.

[25] Michael Lebowitz, *Generalization and memory in an integrated understanding system*, Ph.D. Thesis, Technical Report TR-186, Department of Computer Science, Yale University, 1980.

[26] Wendy G. Lehnert, Cynthia L. Loiselle, *semantic structures*, Chapter 4 - An Inroduction to Plot Units, Lawrence Erlbaum Associates, Hillsdale, New Jersey, Massachusetts, 1989.

[27] Claude Levi-Strauss, *The Structural Study of Myth*, Tavistock Publications, London, 1967.

[28] Christopher D. Manning, Hinrich Schütze, *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, Massachusetts, 2002.

[29] Jean Matter Mandler, *Stories, Scripts, and Scenes: Aspects of Schema Theory*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1984.

[30] Marvin Minsky, *A Framework for Representing Knowledge*, Technical Report AIM-306, Massachusetts Institute of Technology, 1974.

[31] Jan O. Pederson, Yiming Yang, A Comparative Study on Feature Selection in Text Categorization, *Internationals Conference on Machine Learning*, pages 412-426, 1997.

[32] Vladimir Propp, *Theory and History of Folklore*, University of Minnesota Press, Minneapolis, Minnesota, 1984.

[33] Ellen Riloff, Automatically constructing a dictionary for information extraction tasks, *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-93)*, pages 811816, 1993.

[34] Monica Rogati, Yiming Yang, High-Performance Feature Selection for Text Classification, *CIKM'02*, ACM, Virginia, 2002.

[35] Roger C. Schank, *A Conceptual Dependency Representation for a Computer-Oriented Semantics*, Stanford A.I. Memo 83, Computer Science Department, Stanford University, March 1969.

[36] Roger C. Schank, *The Development of Conceptual Structures in Children*, Stanford A.I. Memo 203, Computer Science Department, Stanford University, May 1973.

[37] Roger C. Schank, *Conceptual Information Processing*, Elsevier, North Holland, New York, 1975.

[38] Roger C. Schank, *SAM–A Story Understanding*, Technical Report TR-43, Department of Computer Science, Yale University, New Haven, Connecticut, August 1975.

[39] Roger C. Schank, Janet L. Kolodner, Gerald DeJong, Conceptual information retrieval, *Proceedings of the 3rd annual ACM conference on Research and development in information retrieval*, Cambridge, England, pages 94 - 116, 1980.

[40] Elliot Smith, A Computational Model of On-line Story Understanding, Six Month Progress Report for Ph.D. Thesis, Faculty of Science, School of Computer Science, University of Birmingham, 1997.

[41] S. Soderland, D. Fisher, J. Aseltine, W. Lehnert, Crystal: Inducing a conceptual dictionary, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 13141319, 1995.

[42] Anna Wierzbicka, *Semantics: Primes and Universals*, Oxford University Press, Oxford, New York, 1996.

[43] Robert Wilensky, *Understanding Goal-Based Stories*, Ph.D. Thesis, Technical Report TR-140, Department of Computer Science, Yale University, 1978.

[44] Robert Wilensky, Story Grammar Versus Story Points, *The Behavioural and Brain Sciences*, no. 6, pages 579 - 623, 1983.

[45] Yorick Wilks, *Information Extraction as a core language technology*, M-T. Pazienza (ed.), Information Extraction, Springer, Berlin, 1997.

[46] Yorick Wilks, *IR and AI: traditions of representation and anti-representation in information processing*, Memoranda in Computer and Cognitive Science, CS-00-01, University of Sheffield, 2000.

[47] http://www.ontologystream.com/beads/InORB/three.htm, A First Tutorial on the InOrb Technology Subject Matter, InORB Technologies Inc., December 2003, Last Visit: October 30, 2004.

[48] http://nlp.shef.ac.uk/research/areas/ie.html, Natural Language Processing Research Group, Definition of Information Extraction, University of Sheffield, Last Visit: October 25, 2004.

Appendix A

AutoSlog-TS case frame filtering program

```
#include <iostream>
#include <string>
#include <list>
#include <vector>
#include <fstream>
#include <iomanip>
using namespace std;

// determine if character is a number
bool isNum(char c) {
    return c >= '0' && c <= '9';
}

// parses a string into a double
double getDouble(string s) {
    double d = 0.0, dec = 0.0, fact = 0.1;
    bool dpoint = false;

    for(int i = 0; i < s.length(); i++) {
```

```
        if(isNum(s[i]) && !dpoint) {
            d *= 10;
            d += (s[i] - '0');
        }
        else if(isNum(s[i]) && dpoint) {
            double tmp = (s[i] - '0') * fact;
            fact *= 0.1;
            dec += tmp;
        }
        else {
            dpoint = true;
        }
    }
    return d + dec;
}

// caseframe data structure
struct CaseFrame {
    string CF, Name, Anchor, Slot;
    vector<string> ActFcns;
    double freq, relfreq, condprob, rlogscore;

    CaseFrame() {}
    void parseCF(string & s) {
        CF = s.substr(4, s.length() - 4);
    }
    void parseName(string & s) {
        Name = s.substr(6, s.length() - 6);
    }
    void parseAnchor(string & s) {
        Anchor = s.substr(8, s.length() - 8);
    }
    void parseActFcns(string & s, ifstream & in) {
        char line[80];
        string tmp = s.substr(10, s.length() - 10);
        ActFcns.push_back(tmp);
        while(in.peek() != 'S') {
            in.getline(line, 80);
            s = string(line);
            tmp = s.substr(10, s.length() - 10);
            ActFcns.push_back(tmp);
```

```
        }
    }
    void parseSlot(string & s) {
        Slot = s.substr(5, s.length() - 5);
    }
    void parseFreq(string & s) {
        freq = getDouble(s.substr(19, s.length() - 19));
    }
    void parseRelFreq(string & s) {
        relfreq = getDouble(s.substr(23, s.length() - 23));
    }
    void parseCondProb(string & s) {
        condprob = getDouble(s.substr(19, s.length() - 19));
    }
    void parseRLogScore(string & s) {
        rlogscore = getDouble(s.substr(20, s.length() - 20));
    }
    void print(ostream & out) {
        out << setiosflags(ios::fixed) << setprecision(3);
        out << "CF: " << CF << endl;
        out << "Name: " << Name << endl;
        out << "Anchor: " << Anchor << endl;
        out << "Act_Fcns: ";
        for(int i = 0; i < ActFcns.size(); i++) out << ActFcns[i] << endl;
        out << "Slot: " << Slot << endl;
        out << "Stats: frequency = " << freq << endl;
        out << "       relativeFreq = " << relfreq << endl;
        out << "       cond_prob = " << condprob << endl;
        out << "       rlog_score = " << rlogscore << endl;
        out << endl;
    }
};

// definition of a CaseFrame iterator
typedef list<CaseFrame>::iterator CFIter;

// linked list used to store caseframes
list<CaseFrame> caseFrames;

// function to parse caseframe input into data structure
void parseCaseFrame(CaseFrame & cf, ifstream & in) {
```

```
    char line[80];
    string s;

    for(int i = 0; i < 9; i++) {
        in.getline(line, 80);
        s = string(line);
        switch(i) {
        case 0:
            cf.parseCF(s);
            break;
        case 1:
            cf.parseName(s);
            break;
        case 2:
            cf.parseAnchor(s);
            break;
        case 3:
            cf.parseActFcns(s, in);
            break;
        case 4:
            cf.parseSlot(s);
            break;
        case 5:
            cf.parseFreq(s);
            break;
        case 6:
            cf.parseRelFreq(s);
            break;
        case 7:
            cf.parseCondProb(s);
            break;
        case 8:
            cf.parseRLogScore(s);
            break;
        }
    }
}

// prints a filtering menu
void printMenu() {
    cout << "Menu:" << endl;
```

```
    cout << "1. Filter on frequency." << endl;
    cout << "2. Filter on relative frequency." << endl;
    cout << "3. Filter on conditional probability." << endl;
    cout << "4. Filter on rlog score." << endl;
    cout << "5. Print case frames to file." << endl;
    cout << "6. Display number of case frames." << endl;
    cout << "7. Quit." << endl;
}

// filters case frames based on frequency
void freqFilter() {
    cout << "Choose the smallest acceptable frequency: ";
    double freq;
    cin >> freq;

    for(CFIter iter = caseFrames.begin(); iter != caseFrames.end(); ++iter) {
        if(iter->freq < freq) {
            iter = caseFrames.erase(iter);
            iter--;
        }
    }
}

// filters case frames based on relative frequency
void relFreqFilter() {
    cout << "Choose the smallest acceptable relative frequency: ";
    double relfreq;
    cin >> relfreq;

    for(CFIter iter = caseFrames.begin(); iter != caseFrames.end(); ++iter) {
        if(iter->relfreq < relfreq) {
            iter = caseFrames.erase(iter);
            iter--;
        }
    }
}

// filters case frames based on conditional probability
void condProbFilter() {
    cout << "Choose the smallest acceptable conditional probability: ";
    double condprob;
```

```
    cin >> condprob;

    for(CFIter iter = caseFrames.begin(); iter != caseFrames.end(); ++iter) {
        if(iter->condprob < condprob) {
            iter = caseFrames.erase(iter);
            iter--;
        }
    }
}

// filters case frames based on rlog score
void rlogFilter() {
    cout << "Choose the smallest acceptable frequency: ";
    double rlogscore;
    cin >> rlogscore;

    for(CFIter iter = caseFrames.begin(); iter != caseFrames.end(); ++iter) {
        if(iter->rlogscore < rlogscore) {
            iter = caseFrames.erase(iter);
            iter--;
        }
    }
}

// prints case frames to a file
void printToFile() {
    cout << "Enter file name: ";
    string s;
    cin >> s;
    ofstream out(s.c_str());

    for(CFIter iter = caseFrames.begin(); iter != caseFrames.end(); ++iter) {
        iter->print(out);
    }
    out.close();
}

// allows user to select a menu item
void pickItem() {
    printMenu();
```

```
    int sel = 0;

    cin >> sel;

    switch(sel) {
    case 1:
        freqFilter();
        pickItem();
        break;
    case 2:
        relFreqFilter();
        pickItem();
        break;
    case 3:
        condProbFilter();
        pickItem();
        break;
    case 4:
        rlogFilter();
        pickItem();
        break;
    case 5:
        printToFile();
        pickItem();
        break;
    case 6:
        cout << caseFrames.size() << endl;
        pickItem();
        break;
    case 7:
        break;
    }
}

int main() {
    string f;
    cout << "Enter the case frame file: ";
    cin >> f;
    ifstream in(f.c_str());

    cout << "Reading case frames file." << endl;
```

```
    while(!in.eof()) {
        caseFrames.push_back(CaseFrame());
        parseCaseFrame(caseFrames.back(), in);

        // eat whitespace after a case frame entry
        while(in.peek() == '\n' || in.peek() == '\r'
            || in.peek() == ' ') in.get();
    }
    cout << "Case frames read." << endl;

    pickItem();

    return 0;
}
```

Vita

Candidate's full name: Sean Michael Falconer

University attended: BCS (Computer Science) 2003
University of New Brunswick
New Brunswick, Canada

Journal publications:

Sean M. Falconer, Bradford G. Nickerson, On multi-level k-ranges for range search, *International Journal of Computational Geometry and Applications*, Accepted for Publication, Feb. 2005.

O. Golubitsky, **S. Falconer**, Infinite strings generated by insertions, *Programming and Computer Software*, 31:110-114, Mar-Apr 2004.

Refereed conference proceedings:

Sean M. Falconer, David Gay, Lev Goldfarb, ETS Representation of Fairy Tales, *Proceedings of ICPR 2004 Satellite Workshop*, "Pattern representation and the future

of pattern recognition: A program for action", Cambridge, UK, August 2004.

Sean M. Falconer, Bradford G. Nickerson, On multi-level k-ranges for range search, *Proceedings of the 16th Canadian Conference on Computational Geometry (CCCG04)*, 158-161, August 2004.

Technical report:

S. M. Falconer, B.G. Nickerson, An investigation of multi-level k-ranges, Technical report TR04-163, Faculty of Computer Science, U.N.B., April 2004.

Conference presentations and poster sessions:

Sean M. Falconer, String topology and infinite strings, Presented at APICS 2003, September 2003.