Victor Raskin <sup>1,3</sup>	Sergei Nirenburg <sup>2,3</sup>	<b>Christian F.</b> Hempelmann <sup>1</sup>	Inna Nirenburg <sup>3</sup>	Katrina E. Triezenberg <sup>1</sup>

<sup>1</sup>CERIAS and Linguistics Program Purdue University West Lafayette, IN <sup>2</sup>ILIT University of Maryland, Baltimore County Baltimore, MD

vraskin, hempelma, kattriez@purdue.edu

sergei@cs.umbc.edu

<sup>3</sup>Onyx Consulting, Inc. iniren 99@yahoo.com

### Abstract

This paper describes the creation of a script in the framework of ontological semantics as the formal representation of the complex event BANKRUPTCY. This script for BANKRUPTCY serves as the exemplary basis for a discussion of the general motivations for including scripts in NLP, as well as the discovery process for, and format of, scripts for the purposes of processing coreference and inferencing which are required, for example, in high-end Q&A and IE applications.

## **1** Introduction

A spate of advanced new applications has called for a massive effort in script acquisition. Conceptualized as complex events, they have been provided for in the ontology since its inception (see Carlson and Nirenburg, 1990) and their format has always been reasonably welldefined as well as constantly adjusted to the consecutive releases (see Nirenburg and Raskin, 2003, Section 7.1.5; cf. Moreno Ortiz et al. 2002). Throughout the early and mid-1990s, however, lower-end NLP applications, such as knowledge- and meaning-based MT, did not necessitate a heavy use of scripts. The new generation of higher-end Q&A and similar IE applications make it necessary to recognize individual events and their effects as part of scripts, both because humans do and because such recognition is necessary for establishing (co)reference relations. Thus, in the following text, only the availability of the BANKRUPTCY script can relate (i) and (ii) (and thus determine whose bankruptcy it is in the latter), which may be immediately adjacent in a text: (i) ACME, Inc., was actually doomed the moment Jorge Jimenez and 52 other employees were laid off without a warning.

(ii) That bankruptcy was not, however, the last blow.

As an example, we will sketch out the creation process of the BANKRUPTCY script. In Section 2, we will describe the status of scripts in ontological semantics, and in Section 3 the format of their representation. Section 4 deals with the heuristics/discovery of the information that goes into a script, a sort of knowledge engineering, if you will. Section 5 presents the resulting script BANKRUPTCY, formatted to a certain grain size of the information discovered in Section 4. Section 6 touches briefly upon just a few of the problems script acquisitions poses.

## 2 Scripts in Ontological Semantics

In order to represent the meaning of connected text, not simply that of a sequence of ostensibly independent sentences, several things must happen. One of the most obvious connections across sentence boundaries is coreference. The TMR in ontological semantics allows for the specification of co-reference, and special procedures exist for treating at least facets of this phenomenon in extant applications of ontological semantics (see Nirenburg and Raskin, 2003, Section 8.6.1). Discourse relations among propositions can also hold across sentence boundaries, and ontological semantics includes facilities for both detecting and representing them as well (*ibid*, Section 8.6.3).

There are, however, additional strong connections among elements of many texts. These have to do with the understanding that individual propositions may hold well-defined places in "routine," "typical" sequences of

# The Genesis of a Script for Bankruptcy in Ontological Semantics

events (often called complex events, scripts or scenarios) that happen in the world, with a well-specified set of object-like entities that appear in different roles throughout that sequence. For example, if the sequence of events describes a state visit, the "actors" may, under various circumstances, include the people who meet (the "principals"), their handlers, security personnel and journalists, possibly, a guard of honor; the "props" may include airplanes, airports, meeting spaces, documents, etc. All these actors and props will fill case roles and other properties in the typical component events of the standard event sequence for a state visit, such as travel, arrival, greetings, discussions, negotiations, press conferences, joint statements, etc. The component events are often optional; alternatively, some component events stand in a disjunctive relation with some others (that is, of several components only one may actually be realized in a particular instantiation of the overall complex event), and their relative temporal ordering may be fuzzy.

Such typical scripts can be expressed in natural language using expository texts or narratives, sets of the above (indeed, one conceptual story can be "gathered" from several textual sources), plus text in tables, pictures, TV and movie captions, etc. The notion of script is clearly recursive, as every component event can itself be considered a script, at a different level of granularity. The notion of script, under a variety of monikers, was popularized in computer science by Minsky (1975), Schank and Abelson (1977), Charniak (1972), and their colleagues in the 1970s. However, at that time, no realistic-size implementation of natural language processing using scripts could be undertaken, in part, because there was no clear idea about the required inventory of knowledge sources, their relations and content. Scriptbased theories of semantics were proposed in theoretical linguistics (Fillmore 1985, Raskin 1986) but were overshadowed by the fashion for formal semantics, which is not descriptive in nature, focusing instead on the elements of semantics capturable by such logical devices as quantifiers (see Raskin 1994). Moreover, the size of the task of creating the ontological semantic knowledge sources, which are a sine qua non of script representation, was at the time underestimated by the practitioners and overestimated by critics. It can be said that ontological semantics is a descendant of the script-oriented approach to natural language processing, especially in the strategic sense of accentuating semantic content, that is, the quantity and quality of stored knowledge required for descriptions and applications. Ontological semantics certainly transcends the purview and the granularity levels of the older approach as well as offering an entirely different take on coverage of world and language knowledge and on its applicability. Ontological semantics has also the advantage of having been implemented, tested, and (constantly) improved in actual applications.

In the script-based approach to processing text inputs, the scripts in the ontology that get instantiated from the text input provide expectations for processing further sentences in a text. Indeed, if a sentence in a text can be seen as instantiating a script in the nascent TMR, the analysis and disambiguation of subsequent sentences can be aided by the expectation that propositions contained in them are instantiations of event types that are listed as components of the activated script. Obviously, the task of activating the appropriate script from the input is far from straightforward. Also, not all sentences and clauses in the input text necessarily fit a given script-there can be deviations and fleeting extraneous meanings that must be recognized as such and connected to other elements of the TMR through regular discourse relations, that is, through a weaker connection than that among the elements of a complex event.

Scripts usually describe situations with multiple agents. Each of these agents can be said, in some sense, to carry out their own plans that are made manifest through the reported component events in a script. Plans are special kinds of scripts that describe the process of attaining a goal by an agent or its proxies. Goals are represented in ontological semantics as postconditions (effects) of events (namely, steps in plans or components of general scripts). For example, if an agent's goal is to own a TV set, this goal would be attained on a successful completion of one of a number of possible plans. In other words, it will be listed in the ontology as the postcondition (effect) of such events as BUY, BORROW, LEASE, STEAL, MANUFACTURE. Note that the plans can be activated only if all the necessary preconditions for their triggering hold. Thus, the ontology, in the precondition property of BUY, for example, will list the requirement that the agent must have enough money (see McDonough 2000).

Manipulating plans and goals is especially important in some applications of ontological semantics, for instance, in advice giving applications where the system is entrusted with recognizing the intentions (goals) of an agent or a group of agents based on processing texts about their behavior. Goal- and plan-directed processing relies on the results of the analysis of textual input, as recorded in the basic TMR, as well as the complementary knowledge about relevant (complex) events and objects and their instances, stored in the ontology and the Fact Database (see Nirenburg and Raskin, 2003, Section 7.2), and instantiated in the extended TMR. It is clear that reasoning based on the entire amount of knowledge in the extended TMR can be much richer than if only those facts mentioned in the input texts were used for inference making. Richer possibilities for reasoning would yield better results for any NLP application, provided it is supplied with the requisite inference making programs, for instance, for resolving translation mismatches. The reason we are making a

distinction among NLP applications is the extent to which an application depends on such capabilities. For example, MT practitioners have typically assumed that this application does not really need machinery for inference making. This belief is clearly based on the perception that acquiring the knowledge necessary to support reasoning is prohibitively expensive or even outright infeasible, and therefore one must make do with simpler approaches. Of course, should MT developers be able to obtain such resources, they would use them. Ontological semantics has among its goals that of supplying application builders with exactly this kind of knowledge.

A good if unusual example for a family of applications for which knowledge representation at the level of complex events is crucial but where its integration has so far been carefully avoided is computational humor (see, for instance, Raskin 1996; Stock and Strapparava 2002). Here, the analysis and generation of ambiguity-generally the key issue for semantics, and, accordingly NLP-is a key requirement for a text to be funny. Computational humor aims to increase the acceptability of natural language interaction between human and machine by injecting relevant humor into natural language interfaces. The most developed theory of humor-not only in the framework of script-based semantics-and the formalized model based on it (Attardo and Raskin 1991) have at their core the notion of incongruity conceptualized as two partially overlapping scripts in a relation of opposition. Earlier attempts at computational humor have simply hardwired two complex events in such a relation into templates (Raskin and Attardo 1994; Binsted and Ritchie 1997) instead of aiming at true generation based on an operational model of humor. And true humor generation is impossible without the identification of scripts in an appropriate relation of opposition, e.g., sexual vs. religious. On this basis, the overlap between the two scripts can be analyzed and generated, for example, in puns, where one lexical item from each of the two scripts are brought to overlap in one surface form through sound similarity (Hempelmann 2003). This is just an example-esoteric, as it may seem to some-of how the set of scriptrequiring NLP applications may expand with time and need.

Obviously, as mentioned above, in addition to the knowledge, efficient reasoning procedures must be developed. Such procedures must conform to a number of constraints, an example of which is the following. It is common knowledge that, unless a limit is imposed on making inferences from knowledge units in rich knowledge bases, the inferencing process can go too far or even not halt at all. In advanced applications, for example, advice giving again, a good candidate for such a limit is deriving the active goals and plans of all relevant agents in the world. However, even applications that involve more or less direct treatment of basic text meaning, such as MT, will benefit from making fewer inferences. There will always be difficult cases, such as the need to understand the causal relation in The soldiers fired at the women and I saw some of them fall to select the correct reference for them-in Hebrew, for example, the choice of the pronoun (the masculine otam or the feminine otan will depend on the gender of the antecedent). Such cases are not overly widespread, and a prudent system would deliberately trigger the necessary inferences when it recognizes that there is a need for them. In general, any event is, in fact, complex, that is, one can almost always find subevents of an event; whether and to what extent it is necessary to represent it as a script is a matter of grain size dictated by whether an application needs this information for reasoning, and that, in turn, is largely determined by the nature of the corpora in the domain served by the application.

# **3** Format of Scripts

Scripts are represented in ontological semantics using the ontological property HAS-PARTS. It has temporal semantics if it appears in events, and spatial semantics if it appears in physical objects, e.g., to indicate that an automobile consists of an engine, wheels, the chassis, etc. The properties PRECONDITION and EFFECT also carry information necessary for various kinds of reasoning and apply to any event, complex or otherwise. Scripts require an extension to the specification format. The reason for that is the need to bind the case roles and other property values in component events to establish co-reference. Also, the HAS-PARTS slot of scripts should allow for the specification of rather advanced combinations of component events. Therefore, the format of the filler of HAS-PARTS in scripts should allow a) Boolean operators and, or and not, as well as IF, THEN/ELSE and b) loop statements. Scripts also need statements about partial temporal ordering of their components. For this purpose, a special new property, COMPONENT-RELATIONS is introduced.

Component events in a script have a peculiar status. They are not regular instances of concepts, as no instantiation occurs in the ontology—instantiation is one of the two main operations in generating TMRs, the other being matching selectional restrictions in order to combine individual concept instances—but their meaning is different from that of the general concepts to which they are related. Thus, asking questions in the context of a class at school is clearly different from the general idea of asking questions. In order to represent this difference, the notion of ontological instance is introduced. In an ontological instance, some properties are constrained further as compared to their "parent" concept. The constraints typically take the form of cross-reference to the filler of another component event in the same script. For reasons of clarity and convenience, instead of describing the component events and component relations directly in the fillers of corresponding slots in the concept specification for the complex event, we use the device of reification by just naming them in a unique way in that location (we identify ontological instances by appending letters, not numbers as in the case of real instances) and describe their content separately, at the same level as the main script. As a result, the format of the ontological description of a script is a set of ontological concept frames.

Reification in ontological semantics is a mechanism for allowing the definition of properties on properties by elevating properties from the status of slots in frames to the level of a free-standing concept frame. It is desirable from the point of view of nonproliferation of elements of metalanguage to avoid introducing a concept of, say DRIVER if it could always be referred to as DRIVE.AGENT. However, this brings about certain difficulties. For example, if we want to state that somebody is a driver of trucks, we would have to say that there is an instance of DRIVE in which the THEME is TRUCK and the AGENT is the person in question. There is no direct relationship between THEME and AGENT, and it would take a longer inference chain to realize that TRUCK is, in fact, the value of a property of DRIVER, too, not only of DRIVE. The more properties one would want to add to DRIVER and not to DRIVE, the more enticing it would be to reify the property DRIVE.AGENT and treat it as a separate concept. In principle, we can use reification on the fly, while building a TMR, when we need to add a property to a property, which is prohibited in the static knowledge sources such as the ontology and the lexicon. As we will see in the example below, reification also facilitates the specification of scripts.

In the example below, we present a simplified view of the script/complex event TEACH. As illustrated, TEACH has as PRECONDITION two events—that the teacher knows the material and the students do not; as EFFECT, it has the event that the students (now) know the material. The process of teaching is presented as follows: the teacher presents the material to the students, the students ask the teacher questions about this material, and the teacher answers these questions. The above is admittedly a gross simplification of the actual state of affairs but will serve well for the purposes of illustration.

The ontological instances introduced in the process are: TEACH-KNOW-A, -B AND -C, TEACH-DESCRIBE, TEACH-REQUEST-INFO, TEACH-ANSWER, TEACH-AFTER-A AND -B. The constraints in these instances are all references to fillers of slots in other components of the script or the script itself. Reference is expressed using the traditional dot notation (m.s[.f] is read as 'the filler of the [facet f of the] slot s of the frame m'). Ontological instances are not indexed in the Fact Repository. They appear in appropriate slots of scripts and their fillers are all references to fillers of other ontological instances within the same script or the script itself. They are PART-OF (inverse of HAS-PARTS) of the script in which they are listed but instance-of their corresponding basic concept, that is, TEACH-DESCRIBE-A is the first ontological instance of DESCRIBE that is at the same time PART-OF TEACH.

teach is-a value communicative-event agent sem human default teacher theme knowledge sem destination sem human default student precondition default (teach-know-a teach-know-b) effect default teach-know-c has-parts value (teach-describe repeat (teach-request-information teach-answer) until teach-know-c) component-relations (teach-after-a teach-after-b) value component-modalities value (teach-modality-a) teach-know-a instance-of value know patient value teach.agent.sem theme teach.theme.sem value teach-know-b instance-of value know patient value teach.destination.sem teach.theme.sem theme value teach-modality-a value epistemic type teach-know-b scope value value value 0 teach-know-c instance-of value know teach destination sem patient value theme value teach.theme.sem teach-describe describe instance-of value teach.agent.sem agent value teach.theme.sem theme value destination teach destination sem value teach-request-information instance-of value request-information agent teach.destination.sem value theme teach.theme.sem value destination teach.agent.sem value teach-answer instance-of value answer agent value teach.agent.sem theme value teach.request-information.theme.sem destination teach.destination.sem value teach-after-a domain value teach-describe range value teach-request-information teach-after-b domain value teach-request-information range value teach-answer

## 4 Heuristics

As massive research on expert systems in the 1980s abundantly demonstrated, knowledge engineering requires human intelligence and stubbornly resists automation. Moreover, human intelligence must be devoted to heuristics, another highly non-trivial intellectual process. Deciding what goes into a script requires knowledge engineering and heuristics. Part of the problem is similar to the task of extending the ontology to a new domain—something the ontological semantics community has had to face a number of times for various applications, most recently for information security applications (Raskin *et al.*, 2002). There are three main sources for obtaining and structuring the required information to chart out a new domain or to fill out a new script:

dictionaries, encyclopedias, thesauri;

textbooks and reference books;

pertinent corpora, most conveniently websites.

General common sense or a small sample of pertinent texts brings up a small number of apparently basic terms. These terms are looked up in the first source, and that leads to the second. A selection of key terms forms the basis of an Internet search that brings up the corpora. Thus, in the case of bankruptcy, the term itself brings up an informative entry from *Barron's Finance and Investment Handbook* (1995).

"BANKRUPTCY State of insolvency of an individual or an organization—in other words, an inability to pay debts. There are two kinds of legal bankruptcy under U S. law: involuntary, when one or more creditors petition to have a debtor judged insolvent by a court; and voluntary, when the debtor brings the petition. In both cases, the objective is an orderly and equitable settlement of obligations.

The 1978 Bankruptcy Reform Act removed some of the rigidities of the old law and permitted more flexibility in procedures. The Bankruptcy Reform, Act of 1984 curtailed some of the more liberal provisions (mainly affecting consumer bankruptcy) of the 1978 act.

Chapter 7 of the 1978 act, dealing with LIQUIDATION, provides for a court appointed interim trustee with broad powers and discretion to make management changes, arrange unsecured financing, and generally operate the debtor business in such a way as to prevent loss. Only by filing an appropriate bond is the debtor able to regain possession from the trustee.

Chapter 11, which deals with REORGANIZATION, provides that, unless the court rules otherwise, the debtor remains in possession of the business and in control of its operation. Debtor and creditors are allowed considerable flexibility in working together. The 1978 law relaxes the old *absolute priority rule*, which

gave creditor claims categorical precedence over ownership claims. It also makes possible the negotiation of payment schedules, the restructuring of debt, and even the granting of loans by the creditor to the debtor."

The entry, while somewhat cryptical, offers a pretty good guide for what to look for in a textbook or in legal sources. It can easily lead to a number of sources of the textbook category, such as Summers (1989), Caplan (1992), Davidson (1992). The pertinent information, thus acquired, helps to identify the corpora, which should be both essential for the domain and crucial for the application(s), such as the various bankruptcyrelated pages at http://www.uslaw.com/. Just as in field linguistics (cf. Samarin 1967), the corpora should be varied, multi-sourced, and as representative/exhaustive as possible. The corpora give us a good sense of the grain size of the information to be included in the script—see more on this in Section 6.

The most important step is to structure this information in the script. Models of the script's series of events obtained at the previous stage and their key concepts need to be checked against the ontology before the scripting takes place, to avoid later, costly adaptation of newly introduced concepts to the existing inventory. The models will also tend to pay too much attention to details from the field to which they belong. These details have to be weeded out and the parts of the models to be united into the script have to be translated into ontological concepts, existing and, if necessary, newly acquired.

The methods for doing this are not easy to formulate as recommendations, let alone rules. A similar situation in lexical and ontological acquisition leads, with experience, to pretty well-established routines and, as a result of adhering to them, quite good uniformity among different acquirers. Our work on routine acquisition of fullfledged scripts has only been going on for slightly over a year and has included only two domains so far, the financial domain and the domain of meetings. We hope to be able to make enough useful generalizations in the style of Chapter 9 on acquisition of ontological concepts and lexical entries in Nirenburg and Raskin (2003) as we acquire more practical experience. The discovery of heuristic rules remains a major challenge, possibly unattainable.

The following are the factors to be identified in the script as concepts:

the candidates for component events;

the concepts involved in/created by the series of events;

the goals of the component events;

their temporal and causal relations leading to their groupings into subscripts;

decision forks, such as whether to file Chapter 7 or Chapter 11 bankruptcies.

### **5** Formatted Script

The results of the operations described in Section 4 are incorporated in the script for BANKRUPTCY below (for legibility the FACET types SEM and VALUE are omitted in this example):

BANKRUPTCY financial-event is-a agent owe.agent owe.beneficiary precondition approach-bankruptcy (IF modality.pay.value = 0has-parts THEN bankrupt-chapter-7 ELSE bankrupt-chapter-11) APPROACH-BANKRUPTCY financial-event is-a corporation-a agent has-parts (IF AND owe agent corporation-a beneficiary human-a employed-by corporation-a lending-institution-a corporation-b theme money pay agent corporation-a beneficiary human-a lending-institution-a corporation-b theme money THEN bankruptcy corporation-a agent beneficiary human-a lending-institution-a corporation-b) PAY.MODALITY potential type scope pay <1 value BANKRUPT.MODALITY epistemic type bankruptcy scope 0.6 value BANKRUPT-CHAPTER-7 is-a financial-event agent owe.agent owe.beneficiary precondition bankruptcy has-parts (AND bankrupt.declare bankrupt.business-activity bankrupt.appoint bankrupt.change-event bankrupt.pay) **BANKRUPT-CHAPTER-11** financial-event is-a

owe.agent agent owe.beneficiary precondition bankruptcy has-parts (AND bankrupt.declare bankrupt.legal-case bankrupt.audit (IF modality.bankrupt.audit.value = 0 THEN bankrupt-appoint) bankrupt.planning-event bankrupt.follow-plan (IF modality.pay.value = 0THEN bankrupt-chapter-7 ELSE modality.pay.value = 1)) BANKRUPT.DECLARE instance-of declare agent owe.agent owe.beneficiary destination owe.agent judicial-branch path BANKRUPT.BUSINESS-ACTIVITY instance-of business-activity beneficiary owe.agent BANKRUPT.BUSINESS-ACTIVITY.MODALITY epistemic type scope bankrupt.business-activity value Ω BANKRUPT.AUDIT.MODALITY epiteuctic type scope bankrupt-audit value BANKRUPT.APPOINT instance-of appoint manager-corporation theme beneficiary owe.agent judicial-branch agent BANKRUPT.CHANGE-EVENT instance-of change-event theme asset destination monev manager-corporation agent beneficiary owe.agent BANKRUPT.PAY instance-of pay theme money agent owe.agent beneficiary owe.beneficiary BANKRUPT.LEGAL-CASE instance-of legal-case theme money agent human beneficiary owe.agent BANKRUPT.LEGAL-CASE.MODALITY type epistemic scope bankrupt.legal-case value 0 BANKRUPT.AUDIT instance-of audit judicial-branch agent beneficiary owe.agent theme cash-flow

```
debt
BANKRUPT.PLANNING-EVENT
instance-of planning-event
agent owe.agent
judicial-branch
BANKRUPT.FOLLOW-PLAN
instance-of follow-plan
agent owe.agent
```

asset

## 6 Grain Size Issues

The script above, even though much more complex than the script for TEACH is presented in its simplest and probably coarsest form. The gain is parsimony, in the sense of minimizing the need to acquire new lexical entries or concepts. Are there losses? A text may mention, for instance, a supplier's refusal to ship stuff to the bankrupt corporation. It does that because the corporation cannot pay it for the supplies. Can we consider it covered in the script? What if a text mentions the inability to meet the payroll? Meeting the payroll may deserve a script of its own. It may be seen to be covered sufficiently in the script above, but laying off employees may not. To owe a loan is actually to owe an installment payment on a certain date, and to be unable to pay the loan means, actually, the inability to pay an installment payment of the loan on a certain date. The script above also omits the entire credit ratings game.

The rationale for having the scripts is, not surprisingly, to do what Schank declared his group would do a quarter of a century ago (Schank, 1975; Schank and Abelson, 1977) and, unlike them, to deliver a workable non-toy product, in which the whole script is evoked when any element of it at any level of the script hierarchy occurs lexically in the text. The simplistic representation above obligates our analyzer to reduce any such pertinent lexical material to the level of owing and paying. Is it possible? The alternative is to develop much more elaborate scripts, involving a great deal more of ontological acquisition and change.

A more complex and more accurate level of representation, with all the intermediate subsidiary scripts embedded in other scripts as well as component simple events enriched with precondition and effect (and, we increasingly believe, goal values), will be much costlier, so the question is whether the gain in analysis makes it worthwhile. We expect this to be dictated by the needs of the current and future applications as manifested in their goals and the nature of the texts in the pertinent corpora. But much more effort will have to be devoted to developing more specific grain-size recommendations, rules of thumb and repair/recovery procedures for cases when the grain size of the script is not sufficient to handle a text.

## **References:**

- Attardo, S., and V. Raskin 1991. Script theory revis(it)ed. Joke similarity and joke representation model. *HUMOR* 4:3-4, 293-347.
- *Barron's Finance and Investment Handbook, 1995. 4th ed.*, ed. by J. Downes and J. E. Goodman. New York: Barron's.
- Binsted, K. and G. Ritchie 1997. Computational rules for generating punning riddles. HUMOR 10:1, 25-76.
- Caplan, S. 1992. Saving Your Business: How to Survive Chapter 11 Bankruptcy and Successfully Reorganize Your Company. Englewood Cliffs, NJ: Prentice Hall
- Carlson, L., and S. Nirenburg 1990. World Modeling for NLP. Technical Report CMU-CMT-90-121, Center for Machine Translation, Carnegie Mellon University, Pittsburgh, PA. A short version appeared in: Proceedings of the 3rd Conference on Applied Natural Language Processing, Trento, Italy, April.
- Charniak, E. 1972. Toward a Model of Children's Story Comprehension. Artificial Intelligence Technical Report Number 266, Department of Computer Science, Massachusetts Institute of Technology, Cambridge, MA, December.
- Davidson, R. L. 1992. *The Small Business Bankruptcy Kit.* New York: Wiley.
- Fillmore, C. J. 1985. Frames and the Semantics of Understanding. In: V. Raskin (ed.), Round Table Discussion on Frame/Script Semantics, Part I, *Quaderni di Semantica* VI: 2, 222-254.
- Hempelmann, C. F. 2003. Paronomasic Puns: Target Recoverability towards Automatic Generation. Unpublished Ph.D. thesis, Interdepartmental Program in Linguistics, Purdue University, West Lafayette, IN.
- McDonough, C. J. 2000. Complex Events in an Ontologic-Semantic Natural Language Processing System. Unpublished Ph.D. thesis, Department of English, Purdue University, West Lafayette, IN.
- Minsky, M. 1975. A Framework for Representing Knowledge. In: P. H. Winston (ed.), *The Psychology* of Computer Vision. New York: McGraw Hill, 211-77.
- Moreno Ortiz, A., V. Raskin, and S. Nirenburg (2002) New Developments in Ontological Semantics. In: Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002). Las Palmas de Gran Canaria, Spain. May 29-31, pp. 1196-1202.
- Nirenburg, S., and V. Raskin 2003. *Ontological Semantics*. Cambridge, MA: MIT Press (forthcoming).

- Raskin, V. 1986. Script-Based Semantic Theory. In: D. G. Ellis and W. A. Donohue (eds.), *Contemporary Is*sues in Language and Discourse Processes, Hillsdale, NJ: Erlbaum, 23-61.
- Raskin, V. 1994. Frawley: Linguistic Semantics. A Review Article. Language 70: 3, 552-556.
- Raskin, V. 1996. Computer Implementation of the General Theory of Humor. In: J. Hulstijn and A, Nijholt (eds.), Automatic Interpretation and Generation of Verbal Humor. Twente Workshop on Language Technology TWLT 12, The Hague: CIP, 9-19.
- Raskin, V., and S. Attardo 1994. Non-Literalness and Non-Bona-Fide in Language: Approaches to Formal and Computational Treatments of Humor. *Cognition* and Pragmatics 2:1.
- Raskin, V., C. F. Hempelmann, K. E. Triezenberg, and S. Nirenburg 2002. Ontology in Information Security: A Useful Theoretical Foundation and Methodological Tool. In: V. Raskin and C. F. Hempelmann (eds.), Proceedings. New Security Paradigms Workshop 2001. September 10th-13th, Cloudcroft, NM, USA, New York: ACM Press, 53-59.
- Samarin, W. J. 1967. *Field Linguistics*. New York: Holt, Rinehart and Winston.
- Schank, R. 1975. Conceptual Information Processing. Amsterdam: North-Holland.
- Schank, R., and R. Abelson 1977. Scripts, Plans, Goals, and Understanding. Hillsdale, NJ: Erlbaum.
- Stock, O., and C. Strapparava 2002. Humorous Agents for Humorous Acronyms: The HAHAcronym Project. In: O. Stock, C. Strapparava, and A. Nijholt (eds.) 2002. The April Fools' Day Workshop on Computational Humor. April 2002, ITC-irst, Trento. Twente Workshop on Language Technology TWLT 20. Trento: ITC-irst, 125-135.
- Summers, M. 1989. Bankruptcy Explained, a Guide for Businesses. New York: Wiley.