Abstract
Computational humor is a subdiscipline of computational linguistics with applications in human computer interfaces, edutainment, affective computing, intelligent agents, and other areas. Based on ontological semantics, we develop the resources and algorithms the computer needs to understand and produce humor, in principle and on a detailed example. Our ultimate output will not be that of a toy system that fills in templates, previously the only available approach, but rather true natural language generation, based on the computer approximation of the human understanding of the joke. This paper shows how ontological semantics provides for and computes the full computer understanding of humor, a sine qua non of humor generation.

Introduction
When Apple, still only a computer manufacturer, still in those good old pre-iPod times, introduced OS 9 in 1999, it included many pioneering features, among them a speech recognition system that could tell jokes. A child of its time, it is a very rudimentary system that reacts to the recognition of the spoken command “computer, tell me a joke.” Whenever it does recognize the command, it starts a punning knock-knock joke, guiding you through a simple dialogue, for example:

(1) You: Computer, tell me a joke.
Computer: Knock, knock.
You: Who’s there.
Computer: Thelma.
You: Thelma who?
Computer: Thelma your soul. [Sell me your soul.]

Apple knew why they invested in this feature: First and foremost, it gave their speech recognition system a human touch. Because when humans interact, they frequently use humor for a variety of important functions. Second, humor is a more narrowly and easily circumscribable function than human language use at large, thus providing a more tractable engineering task, as a step towards full speech-based human-computer interaction. Finally, an additional benefit of studying computational humor can be reaped for those interested in humor, as it requires formalization of the key components in order to make them palatable to the dumb machine.

Before and since Apple’s joke teller, little progress has been made in computational humor, both in humor analysis and humor generation (HG), and we predict that extant template-based systems are not going to lead to breakthroughs in natural language engineering, and much less in forcing us to create theories that could tell us about the mechanisms involved in human humor use. The approach presented here is different in nature: It will outline and formalize the computer understanding of humor required by a complete HG system within a natural language generation (NLG) system, with a focus on a full-scale proof-of-concept example. The aim is to facilitate on the fly generation of more appropriate and sophisticated, thus funnier humor.

In the following, we will first outline the motivation for computational humor and provide a short overview over existing systems. Next, we will propose an improved system based on ontological semantics and integrated into a full NLG system by outlining the requirements for such a system, introducing ontological semantics, and developing the humor-relevant resources in detail. Finally, in the last section, we will discuss one full example each for computational humor analysis and for computational humor generation in detail.

Computational Humor

Applications for Computational Humor
The general rationale for and usefulness of the introduction of humor into NLP in general and into human-computer interface (HCI) design in particular has been shown in Binsted (1995), Mulder and Nijholt (2002: 15-16), Nijholt (2002: 102), Raskin (1996: 12-14), Raskin (2002: 33-34), Raskin and Attardo (1994), and Stock and Strapparava (2002). Some of these applications will be very briefly

---

1 By that time the underlying technology, PlainTalk, had been under development for almost ten years (see MacWEEK 4-28: 82. 8/14/1990).
surveyed here with a view to our contribution of a full-
fledged NLG system.

Binsted argues, typically, that humor can help “make
clarification queries [...] less repetitive, statements of
ignorance more acceptable, and error messages less
patronising” (Binsted 1995: n.p.), and, overall, make a
computational agent seem more human. General
‘humanization’ of NL interfaces through adding humor
capabilities to the computer side have been identified as
the main field of application for computational humor.
Morkes et al. show that users consider computer systems
with humor as “more likable and competent” (1999: 215),
which leads to an enhancement of customer acceptance for
such systems, for example in information assurance and
security systems (Raskin 2002: 33-34). Specific purposes
for humor in HCI have been addressed by McDonough’s
(2001) system for easier memorization of random
passwords by associating them with a funny jingle, its use
in electronic commerce (Stock and Strapparava 2002), as
well as Raskin’s (2002) suggestion for the detection of
unintended, harmful humor.

From a completely different theoretical angle comes the
equally relevant benefit that computational humor can help
verify the humor theory (if any) that underlies the
computational humor system (Raskin 1996: 14), just as the
verification of any theory lies in the application of the
methods and tools developed on its basis. That is, if the
system based on the theory produces text that a human
considers funny, the theory is valid.

The two most advanced toy systems of computational
humor generation are LIBJOG (Raskin and Attardo 1994)
and JAPE (Binsted and Ritchie 1994, 1997), implemented
by Loehr (1996). JAPE’s joke analysis and production
engine is merely a punning riddle generator, as it is not
based on a theory that would provide a basis for the
meaning of generation in the mathematical sense intended
by Chomsky (1965), neutral to and possibly forming the
basis for both perception and production. It provides a
good example of a limited-range application based largely
on ad-hoc decisions during its creation.

LIBJOG is a light-bulb generator based on a template that
explicitly associates a target group with a stereotypic trait
and selects the appropriate modification of the same light-
bulb-changing situation. LIBJOG was the first ever toy
system of computational humor, but its authors were much
more aware of its zero intelligence. The following is a
template on which LIBJOG’s pseudogenerative power is
based (the joke itself is, of course, the first ever light-bulb
joke, “How many Polaks does it take to change a light
bulb? Five. One to hold on to the bulb and four to turn the
table he is standing on.”):

(2) Polish Americans DUMB
(activity_1 hold light bulb) 
(number_1 5)
(activity_2 turn table)
(number_2 4)

Raskin’s assessment that “each such lexicon entry is
already a ready-made joke” (1996: 14) is a criticism that
holds just as much for JAPE whose components are
hardwired into “templates” and “schemas” so that the
“generator” has no freedom or intelligence to make any
choices, because, as Ritchie himself correctly observes,
“[t]he JAPE program has very little in the way of a theory
underpinning it” (2001: 126).

In fact, the main thrust of LIBJOG was to expose the
inadequacy of such systems and to emphasize the need to
integrate fully formalized components, like the GTVH and
the SMEARR lexical database, in a usable model of
computational humor. (The subsequent widespread
emulation of LIBJOG’s lack of intelligence or insight with
similar systems, such as JAPE, developed by computer
scientists without any expertise or interest in either NLP or
humor research, was a totally unexpected and unintended
effect.) The present study should be understood as an
attempt in this direction, using the current evolution of
knowledge-based meaning representation tools in
ontological semantics to create a useful tool that is
theoretically based, applicable, and modular.

Further recent developments in computational humor have
aimed to improve humor analysis, not generation, and are
limited-range implementations of general stochastic
algorithms (Mihalcea and Strapparava 2005), partially
improved by humor-theoretic underpinnings (Taylor and
Mazlack 2004).

Humor Theory

The humor theory used in this paper is based on the formal
and machine-tractable linguistic theory of humor
developed in Raskin’s Semantic Script Theory of Humor
(SSTH) and its revision, the General Theory of Verbal
Humor (GTJVH) by Attardo and Raskin. Although some
aspects of these will have to be outlined in more detail, the
interested reader is referred to Raskin (1985), Attardo and

Attardo and Raskin’s (1991) revision of the SSTH, the
main hypothesis of which is given in Fig. 1, encompassed
six knowledge resources (KR)s ordered hierarchically:
- script opposition (SO), logical mechanism (LM), situation
  (SI), target (TA), narrative strategy (NS), and language

2 Auteur (Nack 1996), a system to generate humor for film,
claims to integrate humor theory, but ultimately relies on
insufficiently motivated, ad-hoc templates (“strategies”) in
the vein of Berger (1997).
Ontological semantics has developed the following resources, all of which are currently expanded:

- a 5,500-concept language independent ontology
- several ontology-based lexicons, including a 20,000-entry English lexicon, and a couple of dozen of smaller lexicons for other languages
- a bunch of onomastics, dictionaries of proper names for a number of languages
- a text-meaning representation (TMR) language, an ontology-based knowledge representation language for natural language meaning
- a fact repository, containing the growing number of implemented and remembered TMRs
- a preprocessor analyzing the pre-semantic (ecological, morphological, and syntactic) information
- an analyzer transforming text into TMRs
- a generator translating TMRs into text, data, potentially images.

An ontological semantic NLP system represents input text as a complex TMR—initially, one for each sentence. Thus, starting to analyze a sentence, the system uses morphological information, syntactic information, and lexical entries based on ontological concepts to arrive finally at a (much simplified) TMR (see Fig. 2 below). Meaning representation in TMRs is sufficiently rich for the purposes of computational humor (see Nirenburg and Raskin 2004: ch. 6). For lack of space, the reader is referred to the cited sources for further discussion of the theory and applications of ontological semantics.

For the purpose of humor analysis and generation, the ontology centrally has to be augmented by lexicon enhancement to include humorous stereotypes as used in Attardo and Raskin (1994) and suggested by Raskin (1996). A complementary approach is the effort to develop the possibility to include complex concepts into the ontology (cf. Raskin et al. 2003), in order to finally be able to make full use of the semantic theory of humor based on scripts, as described in Raskin (1985). In the following subsection, we will explain on a full example how this integration is achieved. The necessary components of the integrated system will be described and it will be pointed out, which ones have already been developed and which are desiderata. On the basis of the humor theory adopted, the focus here will be the role of scripts and the oppositeness relations between them.

Integration of Computational Humor into a Full-Fledged NLP System

The general semantic/pragmatic framework for a computational humor system, including its status as part of a general NLP system able to detect humor and switch to its appropriate non-bona fide mode of communication, and accounting for humor analysis as well as generation have been formulated by Raskin and Attardo (1994). Raskin (2002), a follow-up of Raskin (1996), reports the progress
in this direction. The rationale is still “that only the most complex linguistic structures can serve any formal and/or computational treatment of humor well” (Raskin 1996: 17).

We are currently bolstering a full-blown ontological semantic system based on the vast extension of the legacy resources described in Nirenburg and Raskin (2004). While the present paper describes the addition of humor-relevant components, applications that guide the current development include information security and internet-search technology.

**Ontological Semantic Enablement of Computer Humor Understanding**

**Script Opposition Detected**
The legacy implementation of ontological semantics automatically produces the following TMR for the joke in Fig. 1:

request-info-1
agent value human-1
gender value male
has-social-role value patient
beneficiary value human-2
gender value female
age value <.5
attraction-attribute value >.5
marital-status value married
beneficiary value human-3
theme value location-of
theme value human-3
gender value male
marital-status value married
beneficiary value human-2
has-social-role value doctor
instrument value natural-language
loudness value <.3
time-begin unknown
time-end <deny-1.time-begin
deny-1
agent value human-2
beneficiary value human-1
theme value location-of
theme value human-3
time-begin >request-info-1.time-end
time-end <invite-1.time-begin

invite-1
agent value human-2
beneficiary value human-1
location value dwelling-1
owned-by value set-1
time-begin >deny-1.time-end
time-end unknown

Figure 2: Text Meaning Representation of the Joke Text

The current implementations of ontological semantics no longer ignore the property of effect, which was largely redundant in machine translations. It will, therefore, note that the patient’s cue has the effect given in (3), while the doctor’s wife’s cue will not.

(3) examine
agent doctor
beneficiary patient

Thus, the first half of the joke, the setup, puts forward a DOCTOR script, specifying the typical events and object involved in the training and career of a medical professional, while the second part, the punchline, disconfirms it. This will alert the system to the need to search for an alternative script that will, like the first script, embrace part of the text and will have some compatibility with the other part. The second script will be ADULTERY given in (4):

(4) adultery
is-a value sex-event
agent value set-1
has-parts value sex-event
agent value human-1
marital-status value married
beneficiary not human-2
human-2
marital-status value unmarried
married
beneficiary not human-1

which includes the subscript SEX, and a sex/no-sex opposition will be recorded.

This opposition is recognized as part of the set of oppositions with humorous potential, first proposed by Raskin as the “few binary categories which are essential to human life” (1985: 113f) and included into the ontology as relations under the property grouping:

(5) real vs. unreal
good vs. bad
live vs. death

---

3 For initial research into the influence of these opposition types on the perception of humor, see Hempelmann and Ruch (2005: 361-365) and Ruch and Hempelmann (forthcoming).
sex vs. no sex  
money vs. no money  
high stature vs. low stature

These oppositions have as daughter nodes a number of more specific relations, e.g., under good/bad we find feces/no feces, while high stature/lower stature subsumes religion/no religion (see below) and authority/no authority.

**Script Opposition Generated**

A previous implementation (Hempelmann 2004a) focused on the integration of a phonological punning component into an ontological semantic humor system. Taken from this approach, the following reverse-engineered example (6) illustrates the further integration of these components towards a humor generation system.

(6) What did the egg say in the monastery?  
Out of the frying pan, into the friar.

As we have shown above, the two central elements of a joke are the script opposition (SO) and the related logical mechanism (LM), masking the tenuousness of the necessary script opposition’s false logic (Hempelmann 2004b, Hempelmann and Attardo, forthcoming). To generate a text with these necessary and sufficient qualities for it to be a joke, we have to describe how those two elements are arrived at by the computational humor system in the way described above.

The script-switch trigger in our example of an imperfect pun is “friar” and the similar sounding target “fire.” Beyond the sound similarity of these two, the recovery of the target is, of course, aided by the proverb “out of the frying pan, into the fire.” The identification of this similarity will be the task of a phonological component (“Ymperfect Pun Selector,” YPS) described in Hempelmann’s (2004a). The SO of this text is that between one script MONASTERY involving the concept MONK that is selected as a in a high-stature—low stature (religion—no religion) relation to the other script FOOD-PREPARATION, including the concept FIRE.

If we assume the system has detected the target word “fire” in an input text produced by a human, it is able to produce the output in example (6). Following the outlined mechanism it will have to work as shown in Fig. 3.

First, the target “fire” will be identified as the lexical entry fire that is mapped onto the concept labeled FIRE. Among other scripts, FIRE will be found to be part of the script FOOD-PREPARATION, or, even simpler, one of its possible INSTRUMENTS. From its set of humorous opposition relations, the system will choose, inter alia, high/lower stature for which it finds that FOOD-PREPARATION is in this relation to MONASTERY, a relation that both concepts have inherited from parent nodes. For the latter the system will select all its slot-filler concepts, including PRAY, MONK, PREACH.

As the final task for the ontological semantic component of the system, all words in the lexicon of the target language that are mapped onto all the concepts of all scripts that are marked to be in one of the relations described in the previous section. This is the candidate set P that is passed on to the phonological module. This will now evaluate the sound similarity of the phonological representation of all candidates from P against the phonological representation of the target “fire.” The selected optimal candidate will be the output of the system, given as the lexical entry “friar.” This will form the basis of the full joke text generation.

**Conclusion**

We have shown how ontological semantics computes TMRs, full and close to human understanding, and thus vastly superior to what other approaches have been able to achieve. This understanding is directly usable in humor comprehension. Independently of computational humor, ontological semantics has moved to keeping tab of effects and goals as well as to the use of complex events, or scripts. Detecting a script opposition is also necessary for various current implementations, including semantic forensics (Raskin et al. 2004). So, just as the SSTH predicted back in 1985, the only uniquely humor-related extension of ontological semantics is the addition of a tiny new resource—the list of standard script oppositions. Further improvements of generative and analytical power will be achieved by integrating the current research on the more complex issue of LMs besides the straightforward cratystical analogy of punning described here.

With this fully integratable, knowledge-based approach, we are in a position to analyze and generate humor, one example of this also having been outlined in the previous section, not just as built into a limited number of templates, but on the basis of the vast resources that ontological
semantics has accumulated and to offer at this point. This enables us to create humor that is not only intended, but also appropriate to the current topic of human-computer interaction, more sophisticated, and thus perceived to be funnier than that attainable by previous systems. It certainly brings us closer to modeling the human capacity for generating situational humor by detecting any two of the three necessary elements, viz., Script 1, Script 2, which have to be in an opposition relation, and the trigger (punchline) and providing the third (Raskin 1985: 114).

References


