


Computing in Formal Knowledge Base Contexts for Natural Language Ambiguity Resolution

Udo Hahn and Martin Romacker

 Text Knowledge Engineering Lab, Freiburg University
Werthmannplatz 1, D-79085 Freiburg, Germany
<http://www.coling.uni-freiburg.de>

Abstract

We introduce a formal context mechanism, embedded into a description logics framework, which is able to uniformly represent and manage different forms of ambiguities as they occur in the course of text understanding. Alternative lexical, syntactic and semantic interpretations are clearly separated by assigning each reading a single context space for local reasoning.

Introduction

The vast potential to create ambiguities at all levels of natural language analysis – lexical, syntactic, semantic, referential and pragmatic – constitutes one of the great challenges of natural language processing (NLP). Although formal analyses of ambiguity phenomena in isolation yield truly discouraging results (e.g., Church & Patil (1982) give evidence that syntactic ambiguity grows exponentially with the number of genitive or prepositional phrase attachments, noun-noun modifiers, stack relative clauses, etc.), NLP researchers build trustfully on the disambiguating power of the *linguistic context* in which utterances occur. The phrasal and clausal level of language description is then taken to eliminate many of the lexical, syntactic and semantic ambiguities (Tabossi 1989), whereas the level of discourse context is believed to help resolve many referential and higher-level pragmatic ambiguities (Wiebe, Hirst, & Horton 1996; Ferrari 1997).

While a considerable amount of work has been done on disambiguating effects due to the linguistic context, only few proposals have been made with respect to a *general* computational framework how to manage ambiguous language and knowledge structures in NLP systems. One of the most promising approaches to deal with contexts as *formal objects* at the level of knowledge representation and reasoning proper is due to John McCarthy and his co-workers (McCarthy 1993; Buvač, Buvač, & Mason 1995). Despite the fact that language theorists have already tried to incorporate a more rigid representational notion of context into their language models (for a survey, cf. Sowa (1995)), a comprehensive attempt to integrate *all* levels of language analysis into a uniform formal framework is still lacking.

In this paper, we aim to combine the representational and reasoning perspective with the needs of ambiguity management for NLP. Given the application framework of the text understanding system SYNDIKATE (Hahn & Romacker 2000), we propose an extension of its knowledge representation backbone by a formal context mechanism. The way we will use contexts leads to the creation of alternative hypothesis spaces which account for the different levels of ambiguity mentioned above. In order to keep the number of context spaces manageable, we make direct use of constraints for disambiguation purposes that are inherent to the particular discourse context provided by the input text.

Generally, we consider an interpretation to be invalid, if adding an axiom (originating from the semantic interpretation of the input text) to a formal context leads to a set of axioms that are no longer satisfiable. The context in which the analysis of an input text evolves with respect to the satisfiability of a set of logical axioms consists of the *static* context, as given by the a priori domain knowledge. It is further augmented by the *dynamic* context as resulting from the incremental processing of a text, whose new information (in terms of interpretation constraints) is made continuously available.

Overview of the System Architecture

Grammatical knowledge for syntactic analysis is based on a fully lexicalized dependency grammar. Such a grammar captures binary valency constraints between a syntactic head (e.g., a noun) and possible modifiers (e.g., a determiner or an adjective). These include restrictions on word order, compatibility of morphosyntactic features and semantic integrity conditions. For a dependency relation $\delta \in \mathcal{D} := \{\text{specifier, subject, dir-object, ...}\}$ to be established between a head and a modifier, all valency constraints must be fulfilled. Figure 1 depicts a sample dependency graph in which word nodes are given in bold face and dependency relations are indicated by labelled edges.

At the parsing level, these constraint checking tasks are performed by lexicalized processes, so-called *word actors*. Word actors are encapsulated by *phrase actors* which enclose partial parsing results in terms of a dependency subgraph (e.g., for phrases). Syntactic ambiguities, i.e., several phrase actors keeping alternative dependency structures for the same text segment, are packaged in a single *container actor* (cf. Hahn, Bröker, & Neuhaus (2000) for details).

¹Copyright © 2001, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

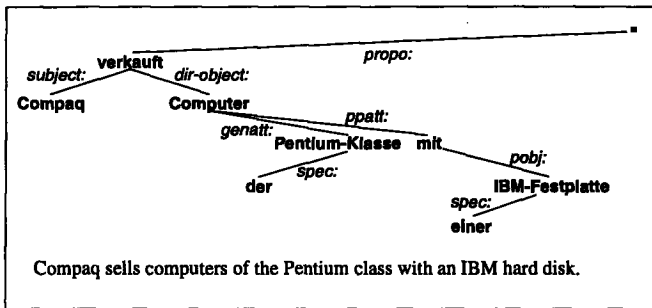


Figure 1: A Sample Dependency Graph

Domain Knowledge is expressed in terms of a concept description language (*CDL*), which has several constructors combining *atomic* concepts, roles and individuals to define the terminological theory of a domain (see Table 1; cf. Woods & Schmolze (1992) for a survey of languages based on such a description logics framework). *Concepts* are unary predicates, *roles* are binary predicates over a domain Δ , with *individuals* being the elements of Δ . We assume a common set-theoretical semantics for this language — an interpretation \mathcal{I} is a function that assigns to each concept symbol (from the set \mathcal{F}) a subset of the domain Δ , $\mathcal{I} : \mathcal{F} \rightarrow 2^\Delta$, to each role symbol (from the set \mathcal{R}) a binary relation of Δ , $\mathcal{I} : \mathcal{R} \rightarrow 2^{\Delta \times \Delta}$, and to each individual symbol (from the set \mathcal{I}) an element of Δ , $\mathcal{I} : \mathcal{I} \rightarrow \Delta$.

Concept terms and *role terms* are defined inductively. Table 1 states corresponding constructors for concepts and roles, together with their semantics. C and D denote concept terms, while R and S denote role terms. $R^{\mathcal{I}}(d)$ represents the set of *role fillers* of the individual d , i.e., the set of individuals e with $(d, e) \in R^{\mathcal{I}}$. By means of *terminological axioms* (cf. Table 2, upper part) a symbolic name can be defined for each concept and role term. We may supply necessary and sufficient constraints (using “ \doteq ”) or only necessary constraints (using “ \sqsubseteq ”) for concepts and roles. A finite set of such axioms, \mathcal{T} , is called the *terminology* or *TBox*. Concepts and roles are associated with concrete individuals by *assertional axioms* (see Table 2, lower part — a, b denote individuals). A finite set of such axioms, \mathcal{A} , is called the *world description* or *ABox*. An interpretation \mathcal{I} is a model of an ABox with regard to a TBox, iff \mathcal{I} satisfies the assertional and terminological axioms.

Syntax	Semantics
C	$\{d \in \Delta^{\mathcal{I}} \mid \mathcal{I}(C) = d\}$
$C \cap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \cup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$\forall R.C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$
R	$\{(d, e) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \mathcal{I}(R) = (d, e)\}$
$R \cap S$	$R^{\mathcal{I}} \cap S^{\mathcal{I}}$

Table 1: Syntax and Semantics for a Subset of *CDL*

Terminological Axioms	
Axiom	Semantics
$A \doteq C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$
$A \sqsubseteq C$	$A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
$Q \doteq R$	$Q^{\mathcal{I}} = R^{\mathcal{I}}$
$Q \sqsubseteq R$	$Q^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
Assertional Axioms	
Axiom	Semantics
$a : C$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
$a R b$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Table 2: *CDL* Axioms

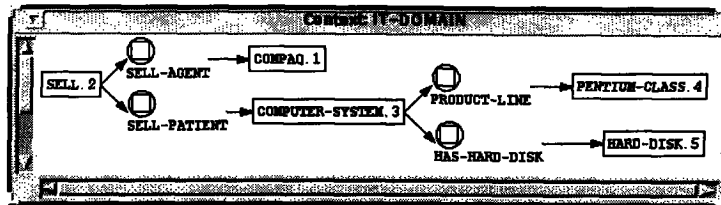


Figure 2: A Sample Semantic Interpretation

Semantic knowledge accounts for conceptual linkages between instances of concept types according to those dependency relations that are established between their corresponding lexical items (Romacker, Markert, & Hahn 1999). When the first word in our sample sentence, “*Compaq*”, is read, its conceptual correlate, COMPAQ.1, is instantiated. The next word, “*verkauft*” (*sells*), also leads to the creation of an associated instance (SELL.2) (cf. Figure 1 and 2). Syntactic constraints of the transitive verb “*verkauft*” (*sells*) lead to checking the *subject* dependency relation for “*Compaq*”. At the conceptual level, *subject* always translates into AGENT or PATIENT (sub)roles, since we statically link each dependency relation to a set of conceptual relations by a function $i : \mathcal{D} \mapsto 2^{\mathcal{R}}$ (e.g., $i(\text{subject}) = \{\text{AGENT}, \text{PATIENT}\}$). To infer a valid semantic relation we incorporate knowledge about the concept types of COMPAQ.1 and SELL.2, viz. COMPANY and SELL, respectively. Semantic interpretation then boils down to a search of the knowledge base, checking whether a COMPANY can be interpreted in terms of an AGENT or a PATIENT of a SELL event. For SELL, only SELL-AGENT and SELL-PATIENT are allowed for interpretation as they are subroles of AGENT and PATIENT. Checking sortal restrictions (e.g., SELL-AGENT requires a PERSON, while SELL-PATIENT requires a PRODUCT) succeeds only for SELL-AGENT (cf. Figure 2).

Description Logics with Contexts

Lexical, syntactic and semantic ambiguities of an utterance translate into different conceptual interpretations at the ABox level. So we need a uniform representation device *within* description logics to deal with different readings locally. This is achieved by reformulating the formal notion of context within description logics. We, first, introduce the set of context symbols \mathcal{H} . Syntactically, assertional axioms internal to a context $h \in \mathcal{H}$ are enclosed by brackets and are subscripted by the corresponding context identifier. For example, $(a : C)_h$ means that in a context h the individual a is asserted to be an instance of the concept C . We then define the set-theoretical semantics of the interpretation \mathcal{I}_h relative to a context h for assertional axioms as summarized in Table 3. The TBox \mathcal{T} and the ABox \mathcal{A} for a context $h \in \mathcal{H}$ is given by \mathcal{T}_h and \mathcal{A}_h , respectively.

Syntax	Semantics
$(a : C)_h$	$a^{\mathcal{I}_h} \in C^{\mathcal{I}_h}$
$(a R b)_h$	$(a^{\mathcal{I}_h}, b^{\mathcal{I}_h}) \in R^{\mathcal{I}_h}$

Table 3: Context-Embedded Assertional Axioms

$$\boxed{\begin{array}{l} \text{subcontextOf}(h_1, h_2): \Leftrightarrow \\ \forall h_1, h_2 \in \mathcal{H}: \\ \mathcal{T}_{h_1} \supseteq \mathcal{T}_{h_2} \wedge \mathcal{A}_{h_1} \supseteq \mathcal{A}_{h_2} \end{array}}$$

Table 4: Hierarchy of Contexts: The *Subcontext* Relation

We then define the transitive and reflexive relation $\text{subcontextOf} \subseteq \mathcal{H} \times \mathcal{H}$ (cf. Table 4) to account for property inheritance in a context hierarchy. We require the TBox and the ABox of a parent context to be inherited by all of its child contexts. Since multiple inheritance may occur, a (directed, acyclic) ‘context graph’ emerges. We also allow for incremental, context-specific extensions of the TBox or ABox. However, some restrictions apply:

1. Extensions of contexts by additional terminological or assertional axioms have to be monotonic, i.e., neither are redefinitions of concepts or relations, nor are retractions of assertions allowed.
2. Context-specific assertions which assign an individual to a concept type or to conceptual relations are permitted, while context-specific concept definitions are prohibited. If a concept occurs in two different contexts, it must have the same definition.
3. The discourse universe Δ is identical for all contexts. We use the top concept \top , the interpretation of which covers all individuals of the domain Δ , $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, and assert every individual to be an instance of \top in the uppermost context.

Provided these extensions to standard description logics, the basic idea for the application of the *formal context* mechanism is to use a separate context for each assertion added to the text knowledge base during text analysis. Such an assertion contains a statement about the meaning of a word or an utterance. Previous assertions (which constitute the formal counterpart of the *discourse context*) are made accessible by inheritance between contexts. Since, under ambiguity, alternative assertion sets have the status of hypotheses, they may or may not be true. Whenever an assertion in a particular context turns out to be unsatisfiable for a particular TBox¹ and a dynamically extended ABox, the corresponding reading is treated as erroneous and will be excluded from further consideration. An ABox is unsatisfiable if there exists an individual a for which the TBox and ABox imply that its interpretation is empty (formally: $\mathcal{T} \cup \mathcal{A} \models a^{\mathcal{I}} = \emptyset$). In this view, contexts provide the representational foundation for managing ambiguities and for computations aimed at their disambiguation.

The linkage between the syntactic dependency level and the evolving text knowledge base, consisting of a context graph (contexts related by *subcontextOf*), is made by assigning these contexts to phrase actors. Let \mathcal{P} be the set of phrase actors. Every instance of a phrase actor $p \in \mathcal{P}$ is linked to a (possibly empty) set of contexts $\text{cont}_p \subseteq 2^{\mathcal{H}}$ that hold all of p ’s alternative semantic interpretations.

¹We assume this TBox \mathcal{T} to be consistent, i.e., there exist no concept C for which \mathcal{T} implies an empty extension.

Computing in Contexts for Disambiguation

Contexts account for ambiguities at all levels of language interpretation. We here focus on two forms of lexical ambiguity – due to multiple part-of-speech assignments and polysemy —, as well as sentential semantic ambiguity due to compositional interpretation. Three phases have to be considered as the text analysis proceeds:

- *Instantiation*: Different conceptual instances for lexical items contained in the input text are created, each one of them in a separate context.
- *Semantic Interpretation*: Whenever a semantic interpretation relating several conceptual instances is performed, different readings (if they exist) are encapsulated in corresponding alternative contexts.
- *Selection*: The interpretation results of sentence analysis, which are contained in alternative contexts, are finally ranked to select the most plausible reading(s).²

Instantiation. Text analysis starts with an empty text knowledge base, one that contains no assertions at all. Nevertheless, we let the empty text knowledge base be a subcontext of the a priori given domain knowledge base, thus preserving the entire information it encodes for subsequent interpretations. By convention, the initial text knowledge base is called NEWHYPO. All interpretation contexts created during text analysis are subcontexts of NEWHYPO. As the system incrementally reads the words from an input text, instances are created in the text knowledge base for each content word associated with a concept identifier.

During the instantiation phase, we have to cope with two different sources of lexical ambiguity. First, a lexical item in a text may refer to different word classes (*part-of-speech ambiguity*) and, therefore, requires the creation of different contexts for semantic interpretation. Second, a lexical item in a text may relate to more than one conceptual correlate (*polysemy*) for each of which a separate context has to be created and maintained.³

Consider Figure 3 where the instantiation of contexts for the German lexical item “*entwickelt*” (*develop*) is depicted. In the lexicon, it is linked to three different word classes, viz. VERBFINITE, VERBPARTPASSIVE, VERBPARTPERFECT. Though different parts of speech are to be considered, they, nevertheless, refer to the same (sense of the associated) lexeme (“*entwickeln*”). For each of the three categorial readings a corresponding word actor is created. As the base lexeme is associated with only one concept identifier in the domain knowledge (DEVELOP), each word actor triggers the creation of the same single assertion in a separate context. In Figure 3, e.g., the uppermost word actor initializes the creation of the assertion (*Develop.1-01 : Develop*)_{LexAmbigHypo1.1}. Note that the instance symbol is also introduced in NEWHYPO,

²We currently extend our context-based ambiguity management system to deal with referential ambiguities within the framework of the centering model (Strube & Hahn 1999), too.

³By convention, these contexts are named LEXHYPO, if no ambiguities occur, and LEXAMBIGHYPO, otherwise.

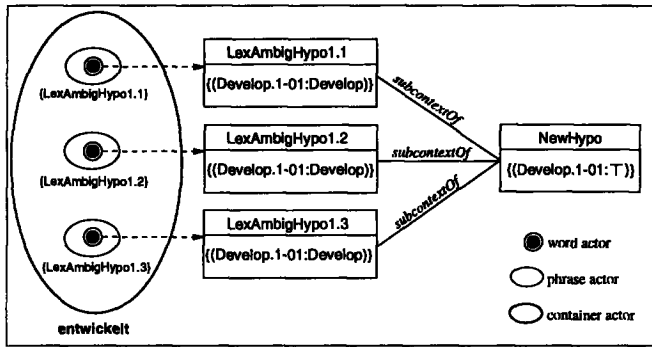


Figure 3: Context Management for Part-of-Speech Ambiguity

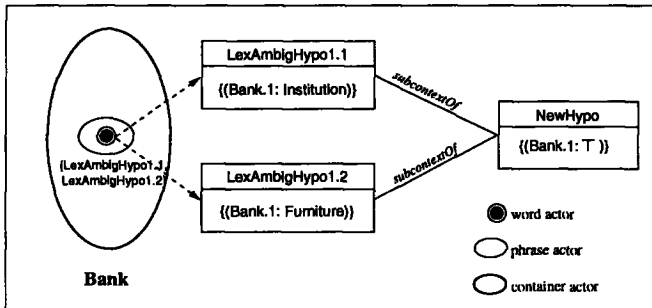


Figure 4: Context Management for Polysemy

the uppermost context, by the assertion $(Develop.1-01 : T)_{NewHypo}$. The contexts are then linked to the phrase actors enveloping the three word actors.

In case of polysemy, the lexical entry is linked to more than one conceptual correlate. Consider the German noun "Bank" which may refer to 'financial institution' or 'kind of furniture'. Both meanings are linked to a single lexical item belonging to the same word class. The corresponding word actor (cf. Figure 4) causes the creation of two lexical contexts, with the assertions $(Bank.1 : Institution)_{LexAmbigHypo1.1}$ and $(Bank.1 : Furniture)_{LexAmbigHypo1.2}$. The same instance symbol, BANK.1, receives different interpretations depending on its context. The interpretation alternatives are administrated by the phrase actor which embeds the word actor in terms of a set of contexts.

Semantic Interpretation. Semantic interpretation is basically a search for a (composed) conceptual relation in the domain knowledge base, holding between the conceptual correlates of the two content words spanning a semantically interpretable dependency subgraph. If the search succeeds, a corresponding assertional axiom is added in a dedicated context. If more than one conceptual relation is computed, a case of *semantic ambiguity* is encountered. Each of these representational alternatives is kept in a separate context, and each of the resulting contexts is defined as a subcontext of the contexts containing the two content words. Thus, they inherit the assertions of their parent contexts and contain the interpretation of the dependency graph that emerges after syntactically linking the two content words.

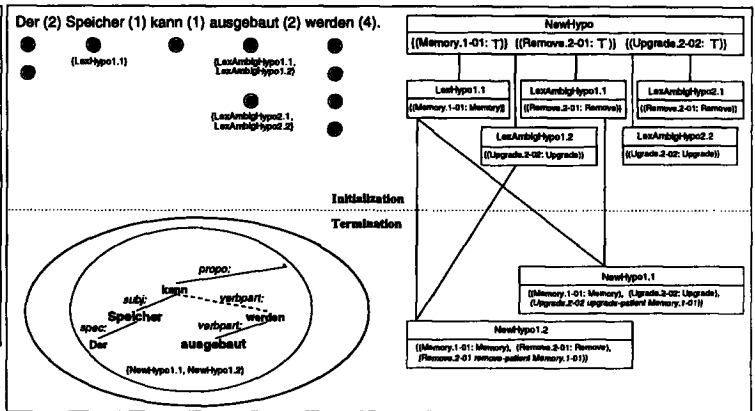


Figure 5: Context Management for Semantic Ambiguity

Let p_1 and p_2 be the phrase actors that contain the word actors for the two content words, which negotiate a dependency relation, and let all constraints except the semantic one be fulfilled. Let $cont_{p_1}$ and $cont_{p_2}$ be the sets of contexts attached to p_1 and p_2 , respectively. Semantic interpretation applies to all context tuples contained in $cont_{p_1} \times cont_{p_2}$. Note that an instance identifier – the conceptual correlate of a lexical item involved – may belong to different concept types in different contexts (cf. Figure 4), and that the interpretation space holding the assertional axioms necessarily differs for all tuples. All new contexts resulting from semantic interpretation are finally included in the set of contexts acquainted with the phrase actor p_3 , which is created after the dependency relation has been established and, thus, encompasses p_1 as well as p_2 .

Consider an ambiguous sentence such as "Der Speicher kann ausgebaut werden." Due to the lexical ambiguity of the German word "ausbauen", one reading is given by "The storage can be upgraded", while the second reading can be phrased as "The storage can be removed". The left side of Figure 5 depicts the syntactic level, its right side contains the (semantic) context graph. Horizontally, Figure 5 is divided into two layers, initialization and termination.

The number of word actors (reflecting lexical ambiguities) instantiated at the syntactic level is given in brackets behind each word in the sentence. A corresponding number of word actor symbols is depicted beneath each lexical item. Since "Speicher" (storage) and "ausgebaut" (upgrade, remove), the sole content words, are linked to a conceptual correlate, each associated word actor initiates the creation of instances in separate contexts. "ausgebaut", e.g., belongs to the word classes VERBPARTPERFECT and VERBPARTPASSIVE. Hence, two word actors are created with two meanings contained in two independent lexical contexts, viz. $LexAmbigHypo1.1$ (remove) and $LexAmbigHypo1.2$ (upgrade) for the word class VERBPARTPASSIVE, as well as $LexAmbigHypo2.1$ (remove) and $LexAmbigHypo2.2$ (upgrade) for the word class VERBPARTPERFECT.

Semantic interpretation starts as soon as the word actor for "kann" (can) tries to govern its modifier "werden" (be) – which itself already governs the VERBPARTPASSIVE "ausgebaut" (upgrade/remove) – by the dependency relation *verbpart*. Identifying the phrase actor of "kann" with