Table of Contents

Attempto Controlled English: A Knowledge Representation Language Readable by Humans and Machines

Norbert E. Fuchs, Stefan Höfler, Kaarel Kaljurand, Fabio Rinaldi, Gerold Schneider

Department of Informatics & Institute of Computational Linguistics
University of Zurich, Switzerland
{fuchs,hoefler,kalju,gschneid,rinaldi}@ifi.unizh.ch
WWW home page: http://www.ifi.unizh.ch/attempto/

Abstract. Attempto Controlled English (ACE) is a knowledge representation language with an English syntax. Thus ACE can be used by anyone, even without being familiar with formal notations. The Attempto Parsing Engine translates ACE texts into discourse representation structures, a variant of first-order logic. Hence, ACE turns out to be a logic language equivalent to full first-order logic. The two views of ACE — natural language and logic language — complement each other, and render ACE both human- and machine-readable. This paper covers both views of ACE. In the first part we present the language ACE in a nutshell, and in the second part we give an overview of the discourse representation structures derived from ACE texts.

1 Introduction

Attempto Controlled English (ACE) is a controlled natural language, i.e. a precisely defined subset of full English that can automatically and unambiguously be translated into full first-order logic. One could say that ACE is a first-order logic language with the syntax of a subset of English. Thus ACE is readable by humans and machines. ACE seems completely natural, but is in fact a formal language that must be learned. Experience shows that one or two days suffice to learn ACE's small number of construction and interpretation rules. More time, though, will be needed to become fluent in ACE.

ACE is based on Discourse Representation Theory [5] whose central concern is to assign meaning to natural language texts and discourses, and to account for the context dependence of meaning. While in general the context of a natural language text is only vaguely defined and can vary, the context of an ACE text is completely fixed. Concretely, an ACE text consists of a sequence of interrelated sentences where each sentence can anaphorically refer to noun phrases occurring in previous sentences. Thus, each sentence is interpreted in the context of the preceding sentences. No further context exists.

Furthermore, the Attempto system is not associated with any specific application domain, or with any particular formal method. By itself it does not contain any knowledge of application domains, of formal methods, or of the world in general. Thus users must explicitly define domain knowledge — definitions, constraints, ontologies — through ACE texts. Words occurring in ACE texts are processed by the Attempto system as uninterpreted syntactic elements, i.e. any interpretation of these words is solely performed by the human writer or reader.

The Attempto Parsing Engine (APE) translates ACE texts unambiguously into discourse representation structures (DRS) the representation language of Discourse Representation Theory. DRSs use a variant of first-order logic, and can be easily translated into any formal language equivalent to first-order logic. For the current version 4 of ACE we developed an extended form of discourse representation structures that allows us to express complex linguistic features, for instance plurals, in first-order logic, and that furthermore supports logical deductions on ACE texts.

A DRS can get a model-theoretic semantics [5], and we can assign the same semantics, i.e. unique meaning, to the ACE text from which the DRS was derived. Thus, the Attempto system treats every ACE sentence as unambiguous, even if people may perceive the same sentence as ambiguous in full English.

2 ACE in a Nutshell

This section is a brief introduction into ACE 4. For a full account readers should consult the ACE documentation found at the Attempto website (see [1]).

Sections 2.1 to 2.6 describe the syntax of ACE 4, sections 2.7 to 2.9 summarize the handling of ambiguity, and section 2.10 explains anaphoric references.

2.1 Vocabulary

The vocabulary of ACE comprises

– predefined function words (e.g. determiners, conjunctions, prepositions),
– content words (nouns, verbs, adjectives, and adverbs).

The Attempto system provides a basic lexicon of content words. Users can define additional, e.g. domain specific, content words with the help of a lexical editor, or can import existing lexica. User-defined words take precedence over words found in the basic lexicon.

2.2 Grammar

The grammar of ACE defines and constrains the form and the meaning of ACE sentences and texts. ACE's grammar is expressed as a small set of construction rules.

2.3 ACE Texts

An ACE text is a sequence of anaphorically interrelated sentences. There are

- simple sentences, and
- composite sentences.

Furthermore, there are query sentences that allow users to interrogate the contents of an ACE text.

2.4 Simple Sentences

A simple sentence describes a situation that can be an event or a state.

A customer inserts 2 cards.
A card is valid.

Simple ACE sentences have the following general structure:

subject + verb + complements + adjuncts

Every sentence has a subject and a verb. Complements (direct and indirect objects) are necessary for transitive verbs (*insert something*) and ditransitive verbs (*give something to somebody*), whereas adjuncts (adverbs, prepositional phrases) are optional.

All elements of a simple sentence can be elaborated upon to describe the situation in more detail. To further specify the nouns *customer* and *card*, we could add adjectives:

A new customer inserts 2 valid cards.

possessive nouns and of-prepositional phrases

John's customer inserts a card of Mary.

or proper nouns and variables as appositions

The customer Mr Miller inserts a card A.

Other modifications of nouns are possible through relative sentences

A customer who is new inserts a card that he owns.

which are described below since they make a sentence composite. We can also detail the insert-event, e.g. by adding an adverb

A customer inserts some cards manually.

or equivalently

A customer manually inserts some cards.

or by adding prepositional phrases, e.g.

A customer inserts some cards into a slot.

We can combine enhancements to arrive at

John's customer who is new inserts a valid card of Mary manually into a slot A.

2.5 Composite Sentences

Composite sentences are recursively built from simpler sentences through coordination, subordination, quantification, and negation.

Coordination by *and* is possible between sentences and between phrases of the same syntactic type.

A customer inserts a card and the machine checks the code.
A customer inserts a card and enters a code.
An old and trusted customer enters a card and a code.

Note that the coordination of the noun phrases *a card and a code* represents a plural object.

Coordination by *or* is possible between sentences, relative clauses and verb phrases.

A customer inserts a card or enters a code.

Coordination by *and* and *or* is governed by the standard binding order of logic, i.e. *and* binds stronger than *or*. Commas can be used to override the standard binding order. Thus the sentence

A customer inserts a VisaCard or inserts a MasterCard, and inserts a code.

means that the customer inserts a VisaCard and a code or, alternatively a MasterCard and a code.

There are two forms of subordination: relative sentences and *if-then* sentences.

Relative sentences starting with *who*, *which*, and *that* allow to add detail to nouns, e.g.

A customer who is new inserts a card that he owns.

With the help of if-then sentences we can specify conditional or hypothetical situations, e.g.

If a card is valid then a customer inserts it.

Note the anaphoric reference via the pronoun *it* in the then-part to the noun phrase *a card* in the if-part.

Quantification allows us to speak about all objects of a certain class, or to denote explicitly the existence of at least one object of this class. The textual occurrence of a universal or existential quantifier opens its scope that extends to the end of the sentence, or in coordinations to the end of the respective coordinated sentence.

To express that all involved customers insert cards we can write

Every customer inserts a card.

This sentence means that each customer inserts a card that may, or may not, be the same as the one inserted by another customer. To specify that all customers insert the same card — however unrealistic that situation seems — we can write

There is a card that every customer inserts.

ACE does not know the passive voice. To state that every card is inserted by a customer we write somewhat indirectly

For every card there is a customer who inserts it.

Negation allows us to express that something is not the case, e.g.

A customer does not insert a card.
A card is not valid.

To negate something for all objects of a certain class one uses *no*

No customer inserts more than 2 cards.

or, equivalently, *there is no*

There is no customer who inserts a card.

To negate a complete statement one uses sentence negation

It is not the case that a customer inserts a card.

2.6 Query Sentences

Query sentences permit us to interrogate the contents of an ACE text. There are yes-no queries and wh-queries.

Yes/no-queries establish the existence or non-existence of a specified situation. If we specified

A customer inserts a card.

then we can ask

Does a customer insert a card?

to get a positive answer.

With the help of wh-queries, i.e. queries with query words, we can interrogate a text for details of the specified situation. If we specified

A new customer inserts a valid card manually.

we can ask for each element of the sentence, e.g.

Who inserts a card?
Which customer inserts a card?
What does the customer insert?
How does the customer insert a card?

Note, however, that we cannot ask for the verb itself.

Questions can also be constructed by a sequence of declarative sentences followed by one query sentence. This can be used to temporarily add information to an already existing ACE text before one asks the question. Here is an example.

There is John and there is a card that John enters. Does John enter the card?

2.7 Constraining Ambiguity

To constrain the ambiguity of full natural language ACE employs three simple means

– some ambiguous constructs are not part of the language; unambiguous alternatives are available in their place,
– all remaining ambiguous constructs are interpreted deterministically on the basis of a small number of interpretation rules,
– users can either accept the assigned interpretation, or they must rephrase the input to obtain another one.

2.8 Avoidance of Ambiguity

Here is an example how ACE replaces ambiguous constructs by unambiguous constructs.

In full natural language relative sentences combined with coordinations can introduce ambiguity, e.g.

A customer inserts a card that is valid and opens an account.

In ACE the sentence has the unequivocal meaning that the customer opens an account. This is reflected by

A customer inserts {a card that is valid} and opens an account.

To express the alternative — though not very realistic — meaning that the card opens an account the relative pronoun *that* must be repeated, thus yielding a coordination of relative sentences.

A customer inserts a card that is valid and that opens an account.

with the interpretation

A customer inserts {a card that is valid and that opens an account}.

2.9 Interpretation Rules

However, not all ambiguities can be safely removed from ACE without rendering it artificial. To deterministically interpret otherwise syntactically correct ACE sentences we use about 20 interpretation rules. Here are some examples.

If we write

The customer inserts a card with a code.

we get the interpretation

The customer {inserts a card with a code}.

that reflects ACE's interpretation rule that a prepositional phrase always modifies the verb.

However, this is probably not what we meant to say. To express that the code is associated with the card we can employ the interpretation rule that a relative sentence always modifies the immediately preceding noun phrase, and rephrase the input as

The customer inserts a card that carries a code.

yielding the interpretation

The customer inserts {*a card that carries a code*}.

or — to specify that the customer inserts a card and a code — as

The customer inserts a card and a code.

Adverbs can precede or follow the verb. To disambiguate the sentence

The customer who inserts a card manually enters a code.

we employ the interpretation rule that the postverbal position has priority.

The customer who {*inserts a card manually*} *enters a code.*

2.10 Anaphoric References

Usually ACE texts consist of more than one sentence, e.g.

A customer enters a card and a code. If a code is valid then SimpleMat accepts a card. If a code is not valid then SimpleMat rejects a card.

To express that all occurrences of *card* and *code* should mean the same card and the same code, ACE provides anaphoric references via the definite article, i.e.

A customer enters a card and a code. If the code is valid then SimpleMat accepts the card. If the code is not valid then SimpleMat rejects the card.

During the processing of the ACE text all anaphoric references are replaced by the most recent and most specific accessible noun phrase that agrees in gender and number, yielding

A customer enters a card and a code. If [the code] is valid then SimpleMat accepts [the card]. If [the code] is not valid then SimpleMat rejects [the card].

What does"most recent and most specific" mean? Given the sentence

A customer enters a red card and a blue card.

then

> *The card is correct.*

yields

> *[The blue card] is correct.*

while

> *The red card is correct.*

yields

> *[The red card] is correct.*

What does "accessible" mean? According to Discourse Representation Theory noun phrases introduced in if-then sentences, universally quantified sentences or negations cannot be used anaphorically in subsequent sentences. Thus *the card* in

> *A customer does not enter a card. The card is correct.*

cannot refer to *a card*.

Anaphoric references are also possible via personal pronouns

> *A customer enters a card and a code. If it is valid then SimpleMat accepts the card. If it is not valid then SimpleMat rejects the card.*

or via variables

> *A customer enters a card CARD and a code CODE. If CODE is valid then SimpleMat accepts CARD. If CODE is not valid then SimpleMat rejects CARD.*

Anaphoric references via definite articles and variables can be combined.

> *A customer enters a card CARD and a code CODE. If the code CODE is valid then SimpleMat accepts the card CARD. If the code CODE is not valid then SimpleMat rejects the card CARD.*

Note that proper nouns like *SimpleMat* always refer to the same object.

3 Extended Discourse Representation Structures in Attempto Controlled English

3.1 Introductory Notes

The Attempto Parsing Engine (APE) translates ACE texts unambiguously into extended discourse representation structures (DRS) that have the following characteristics:

- they use only a small number of predefined predicates,
- they represent information derived from words as arguments of the predefined predicates,
- they have eventuality types,
- they use a lattice-theoretic representation of objects that allows us to encode plurals in first-order language,
- they contain quantity information.

In the following we will explain extended discourse representation structures by means of illustrative examples. Readers are referred to [2] for a practical introduction to Discourse Representation Theory.

Section 3.2 introduces the notation used in this report. Sections 3.3 to 3.11 describe discourse representation structures derived from declarative ACE sentences, and section 3.12 those derived from ACE query sentences.

3.2 Notation

APE translates an ACE text unambiguously into an internal representation using Prolog notation

```
paragraph(DRS,Text)
```

where `Text` stands for the predicate `text/1`, the only argument of which is a list of the input sentences represented as character strings. The example text

```
John enters a card. Every card is green.
```

would thus be represented as

```
text(['John enters a card.', 'Every card is green.'])
```

The discourse representation structure derived from the ACE text is stored in the first argument DRS of `paragraph/2` as

```
drs(Domain,Conditions)
```

The first argument of `drs/2` is a list of discourse referents, i.e. quantified variables naming objects of the domain of discourse. The second argument of `drs/2` is a list of simple and complex conditions for the discourse referents. The list separator ',' stands for logical conjunction. Simple conditions are logical atoms, while complex conditions are built from other discourse representation structures with the help of the logical connectors negation '`-`', disjunction '`v`', and implication '`=>`'.

A DRS like

```
drs([A,B],[condition(A),condition(B)])
```

is usually pretty-printed as

$$\begin{array}{|l|}\hline A\ \ B \\ \hline \mathit{condition(A)} \\ \mathit{condition(B)} \\ \hline \end{array}$$

The above DRS corresponds to the standard first-order logic (FOL) representation

$$\exists AB : condition(A) \land condition(B)$$

Accordingly, a negated DRS like

$$\neg \begin{array}{|l|}\hline A\ \ B \\ \hline \mathit{condition(A)} \\ \mathit{condition(B)} \\ \hline \end{array}$$

corresponds to the standard FOL representation

$$\neg\exists AB : condition(A) \land condition(B)$$

and is internally represented as

```
-drs([A,B],[condition(A),condition(B)])
```

in the Attempto system. We have defined `-/1` as a prefix operator which stands for the logical '¬'.

In a DRS, all variables are thus existentially quantified unless they stand in the restrictor of an implication. The implication

$$\begin{array}{|l|}\hline A \\ \hline \mathit{condition(A)} \\ \hline \end{array} \Rightarrow \begin{array}{|l|}\hline B \\ \hline \mathit{condition(B)} \\ \hline \end{array}$$

corresponding to the standard FOL representation

$$\forall A : condition(A) \to \exists B : condition(B)$$

is internally represented as

```
drs([A],[condition(A)]) => drs([B],[condition(B)])
```

The disjunction

$$\begin{array}{|l|}\hline A \\ \hline \textit{condition(A)} \\ \hline \end{array} \vee \begin{array}{|l|}\hline B \\ \hline \textit{condition(B)} \\ \hline \end{array}$$

corresponding to the standard FOL notation

$$\exists A : condtion(A) \vee \exists B : condition(B)$$

is likewise internally represented as

```
drs([A],[condition(A)]) v drs([B],[condition(B)])
```

The predicates `=>/2` and `v/2` are defined as infix operators.

In nested discourse representation structures, a DRS can occur as an element of the conditions list of another DRS. Therefore

$$\begin{array}{|l|}\hline A\ B \\ \hline \textit{condition(A)} \\ \neg \begin{array}{|l|}\hline \\ \hline \textit{condition(B)} \\ \hline \end{array} \\ \hline \end{array}$$

is represented as

```
drs([A,B],[condition(A),-drs([],[condition(B)])])
```

The discourse representation structure uses a reified, or ‘flat’ notation for logical atoms (see [4]).

For example, the noun *card* that customarily would be represented as

$$\exists A : card(A)$$

is represented here as

$$\exists A : object(A, card, object), ...$$

relegating the predicate ‘card’ to the constant ‘card’ used as an argument in the predefined predicate ‘object’.

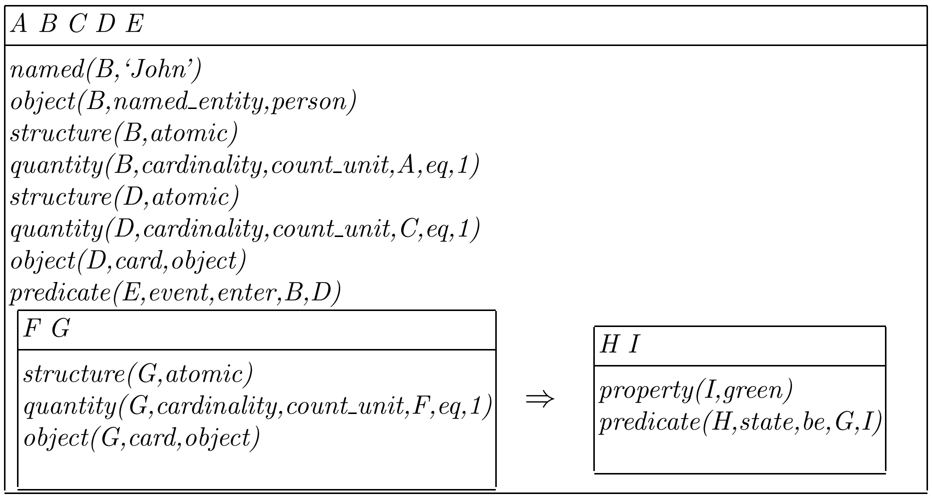
As a consequence, the large number of predicates in the customary representation is replaced by a small number of predefined predicates. This allows us to conveniently formulate axioms for the predefined predicates within the Attempto Reasoner RACE (see [3]).

Logical atoms occurring in `drs/2` are actually written as `Atom-I` (using an infix operator `-/2`) where the index `I` refers to the `I`-th element of the list in `text/1`, i.e. to the sentence from which `Atom` was derived.

The example text

```
John enters a card. Every card is green.
```

the DRS of which is


A B C D E
named(B,'John')
object(B,named_entity,person)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,card,object)
predicate(E,event,enter,B,D)
F G
structure(G,atomic)
quantity(G,cardinality,count_unit,F,eq,1)
object(G,card,object)
⇒
H I
property(I,green)
predicate(H,state,be,G,I)

will thus internally be represented as

```
paragraph(drs([A,B,C,D,E],[named(B,'John')-1,
object(B,named_entity,person)-1,structure(B,atomic)-1,
quantity(B,cardinality,count_unit,A,eq,1)-1,structure(D,atomic)-1,
quantity(D,cardinality,count_unit,C,eq,1)-1,object(D,card,object)-1,
predicate(E,event,enter,B,D)-1,drs([F,G],[structure(G,atomic)-2,
quantity(G,cardinality,count_unit,F,eq,1)-2,
object(G,card,object)-2])=>drs([H,I],[property(I,green)-2,
predicate(H,state,be,G,I)-2])]),text(['John enters a card.',
'Every card is green.']))
```

The following sections provide the discourse representation structures for a selected number of ACE 4 sentences in the form they will be output by APE.

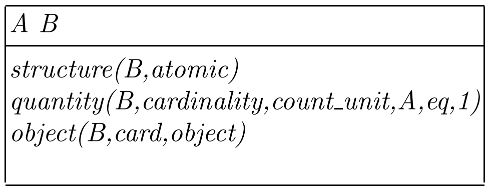
Using illustrative ACE 4 examples this paper describes the language of extended DRSs derived from ACE texts. For a complete description of the ACE 4 language itself please refer to the ACE 4 Language Manual found on the Attempto web site [1].

3.3 Noun Phrases

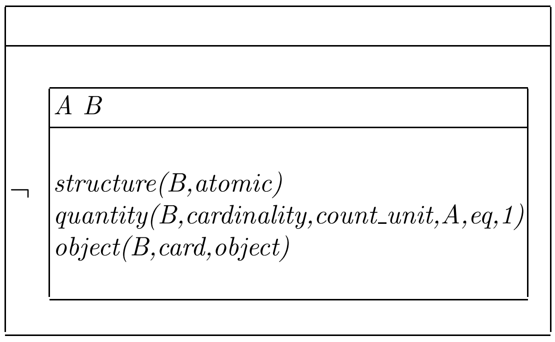
ACE noun phrases (NP) are headed by a countable noun such as *card*, a mass noun such as *bread*, or they are a proper names such as *Mary* or pronouns such as *she*. All ACE NPs except proper names and pronouns are introduced by a determiner. We also introduce special determiners called generalized quantifiers, NP conjunctions and measurement NPs.

Singular Countable Noun Phrases Singular countable NPs are typically introduced by an existential quantifier such as *a* or a universal quantifier such as *every*. Both quantifiers can also be negated. Existentially quantified NPs are typically introduced with an indefinite article *a* if they are new discourse participants and with a definite article *the* if they refer to a referent that has been previously introduced. A noun phrase with a definite article that does not anaphorically refer to a previously introduced noun phrase is treated as if having an indefinite article, i.e. as a new discourse participant.

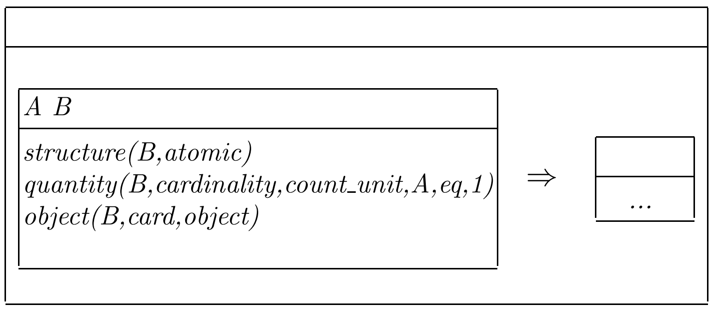
a card



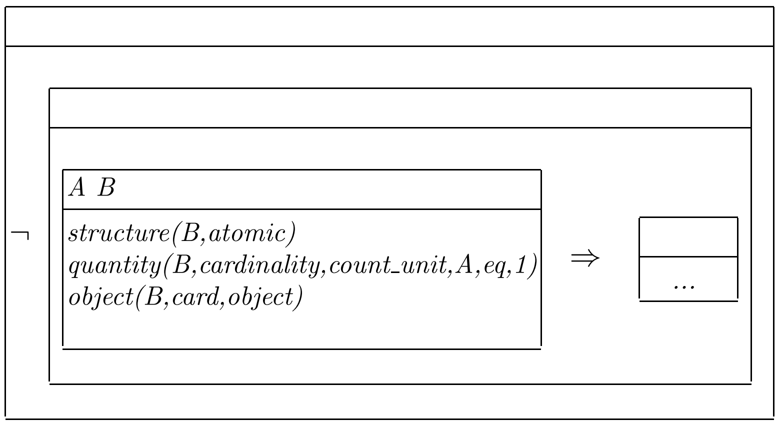
no card



every card

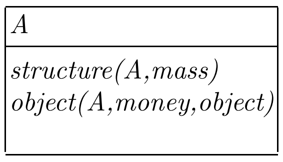


not every card

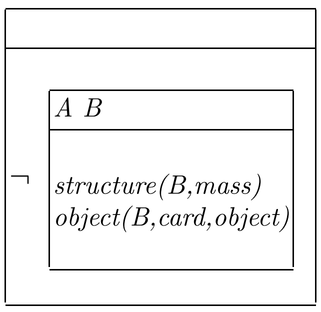


Mass Nouns Non-countable nouns, also called mass nouns, are introduced by *some* in their existential and *all* in their universal affirmative version. Both quantifiers can also be negated.

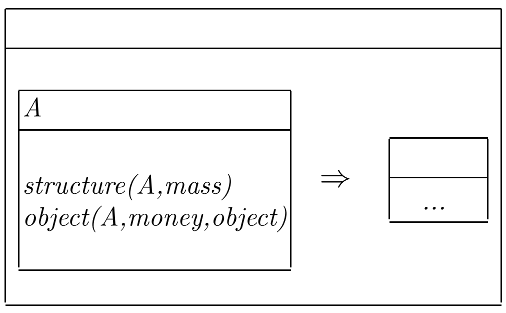
some money



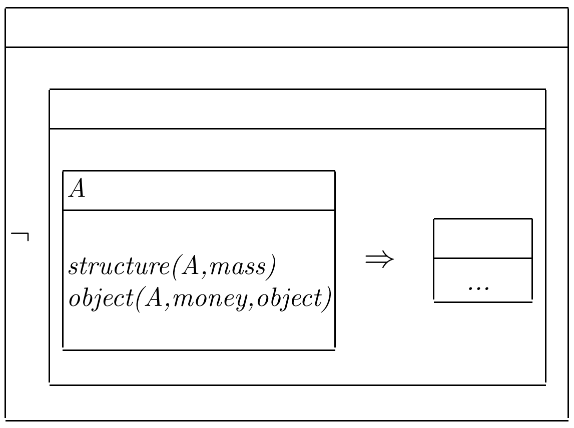
no money



all money

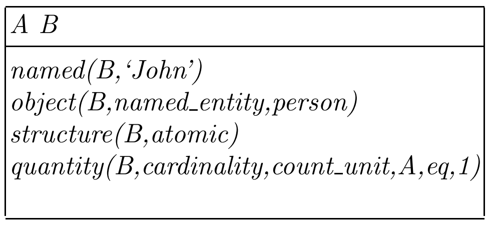


not all money



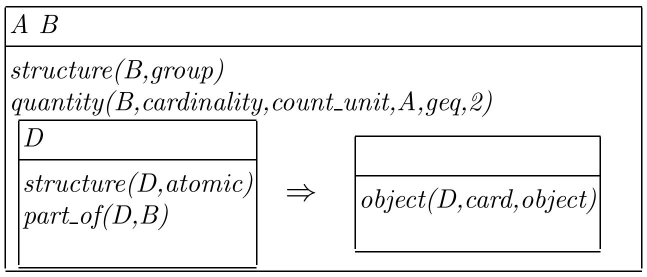
Proper Names Proper names denote a unique object. They can be singular or plural.

John

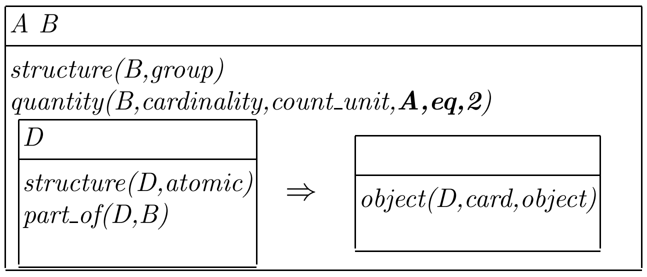

A B
named(B,'John')
object(B,named_entity,person)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)

Plural Noun Phrases Plural NPs are of known or unknown quantity. If the quantity is unknown but restricted, a generalized quantifier (*at least*, *at most*, *more than*, *less than*) can be used. Plurals introduce group objects of which the individual constituents form parts.

some cards



2 cards



at least *2 cards*

```
+-----------------------------------------------------------------+
| A B                                                             |
+-----------------------------------------------------------------+
| structure(B,group)                                              |
| quantity(B,cardinality,count_unit,A,geq,2)                      |
| +-----------------------+        +-------------------------+    |
| | D                     |        |                         |    |
| +-----------------------+        +-------------------------+    |
| | structure(D,atomic)   |   =>   | object(D,card,object)   |    |
| | part_of(D,B)          |        |                         |    |
| |                       |        +-------------------------+    |
| +-----------------------+                                       |
+-----------------------------------------------------------------+
```

Plural Interpretations In ACE, a plural noun phrase has a default collective reading. In order to express a distributive reading, a noun phrase has to be preceded by the marker *each of.* Since the relative scope of a quantifier corresponds to its surface position, we use *there is/are* and *for each of* to move a quantifier to the front of a sentence and thus widen its scope.

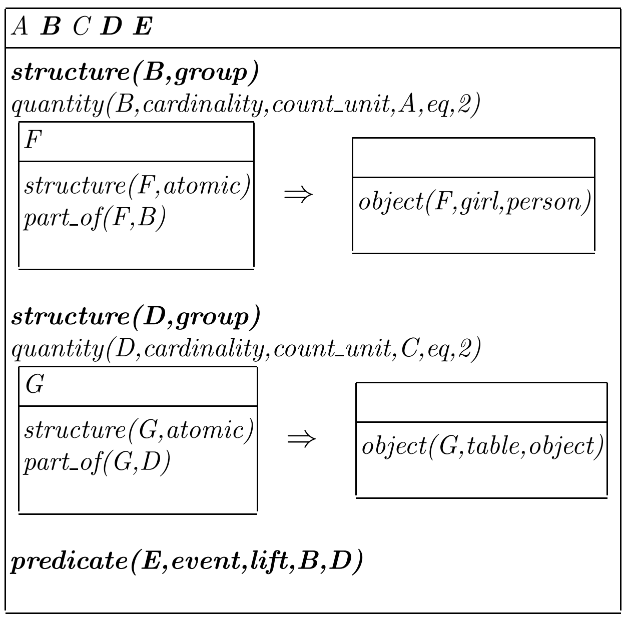
The natural English sentence

2 girls lift 2 tables.

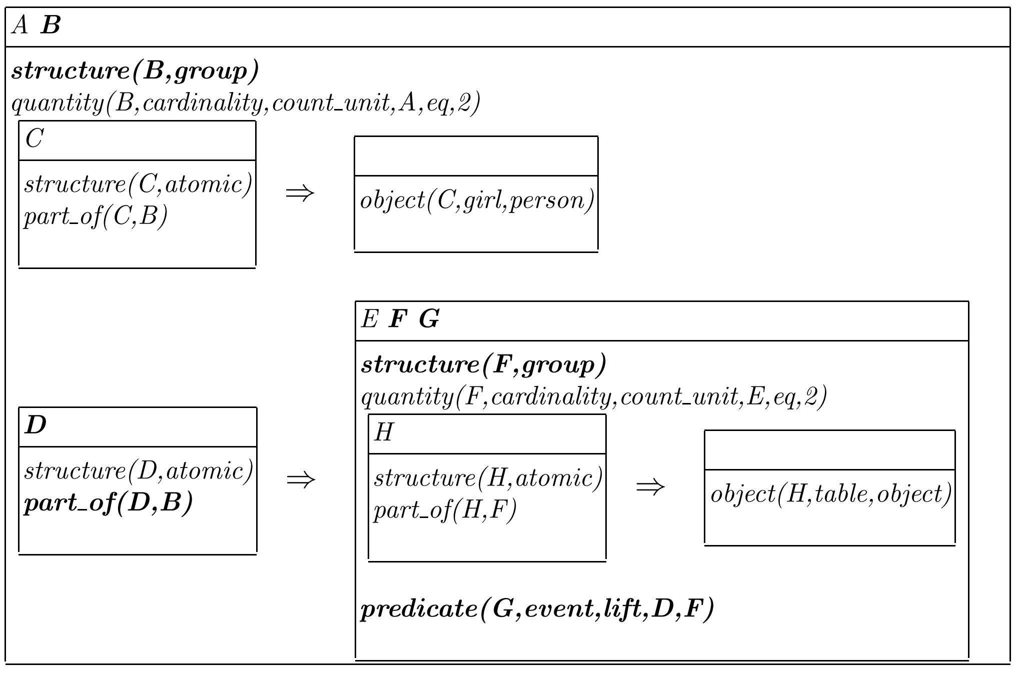
has a multitude of readings (see [6]), eight of which can be expressed in ACE. Here we present two of these eight readings.

The first one shows the default collective reading of both *2 girls* and *2 tables*, while the second shows the distributive reading of *2 girls* and the collective reading of *2 tables.* The other six readings can be expressed analogously using *each of* and *there is/are.*

2 girls lift 2 tables.

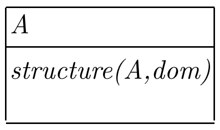


Each of 2 girls lifts 2 tables.

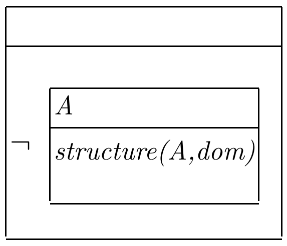


Non-anaphoric Pronouns Anonymous objects can be introduced by non-anaphoric pronouns. They offer a natural way to express a passive voice situation in ACE.

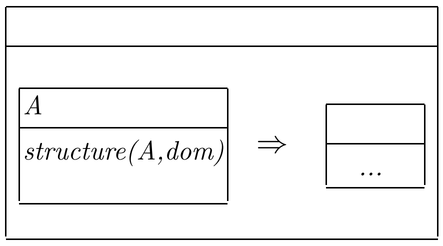
someone / somebody / something



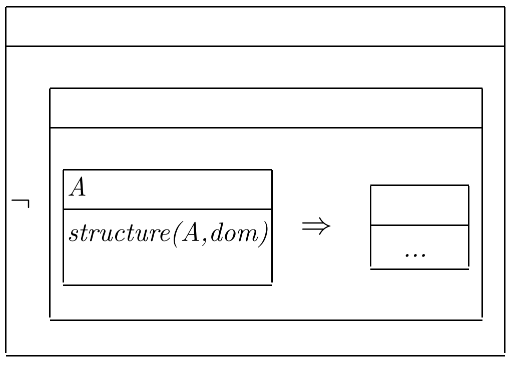
no one / nobody / nothing



everyone / everybody / everything



not everyone / not everybody / not everything



Noun Phrase Conjunction NPs can be conjoined, introducing a plural object into the discourse. The default interpretation of conjoined plurals (e.g. *a customer and a clerk*) is that the individuals act together. Both the conjoined plural object and the individuals can be anaphorically referred to.

a customer and a clerk

```
+--------------------------------------------------+
| A B C D E F                                      |
+--------------------------------------------------+
| structure(B,group)                               |
| quantity(B,cardinality,count_unit,A,eq,2)        |
| sum_of(B,[D,F])                                  |
| structure(D,atomic)                              |
| quantity(D,cardinality,count_unit,C,eq,1)        |
| object(D,customer,person)                        |
| proper_part_of(D,B)                              |
| structure(F,atomic)                              |
| quantity(F,cardinality,count_unit,E,eq,1)        |
| object(F,clerk,person)                           |
| proper_part_of(F,B)                              |
|                                                  |
+--------------------------------------------------+
```

Measurement Noun Phrases Mass NPs cannot be counted but often come in defined amounts. This can be expressed by using measurement NPs. Also plural object quantities can be expressed in this way.

1 kg of *gold*

```
+--------------------------------------+
| A B                                  |
+--------------------------------------+
| structure(B,mass)                    |
| quantity(B,weight,kg,A,eq,1)         |
| object(B,gold,object)                |
|                                      |
+--------------------------------------+
```

2 kg of *apples*

```
+------------------------------------------------------------------+
| A B                                                              |
+------------------------------------------------------------------+
| structure(B,group)                                               |
| quantity(B,weight,kg,A,eq,2)                                     |
| +-----------------------+       +--------------------------+     |
| | C                     |       |                          |     |
| +-----------------------+       +--------------------------+     |
| | structure(C,atomic)   |  =>   | object(C,apple,object)   |     |
| | part_of(C,B)          |       |                          |     |
| |                       |       +--------------------------+     |
| +-----------------------+                                        |
+------------------------------------------------------------------+
```

3.4 Verb Phrases

Verbs fall into classical subcategories known as intransitive (e.g. *wait*), transitive (e.g. *enter something*), and ditransitive (e.g. *give something to somebody*). ACE also knows the copula *be*. The copula can be followed by a (simple, transitive or comparative) adjective, noun phrase or a prepositional phrase.

The customer ***waits****.*

A B **C**
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,customer,person)
predicate(C,state,wait,B)

John ***enters*** *a card.*

A B C D **E**
named(B,'John')
object(B,named_entity,person)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,card,object)
predicate(E,event,enter,B,D)

A clerk ***gives*** *a password* ***to*** *a customer.*

A B C D E F **G**
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,clerk,person)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,password,object)
structure(F,atomic)
quantity(F,cardinality,count_unit,E,eq,1)
object(F,customer,person)
predicate(G,event,give_to,B,D,F)

A card ***is valid****.*

A B ***C D***
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,card,object)
predicate(C,state,be,B,D)
property(D,valid)

A card ***is valid and correct****.*

A B ***C D***
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,card,object)
predicate(C,state,be,B,D)
property(D,valid)
property(D,correct)

2 codes ***are valid****.*

A B ***C D***		
structure(B,group)		
quantity(B,cardinality,count_unit,A,eq,2)		
D	$\Rightarrow$	
structure(D,atomic) *part_of(D,B)*		*object(D,code,object)*
predicate(C,state,be,B,D)		
property(D,valid)		

Each of 2 codes ***is valid***.

```
A B
----------------------------------------------------
structure(B,group)
quantity(B,cardinality,count_unit,A,eq,2)
  C                                   
  -------------------         -------------------------
  structure(C,atomic)    =>   object(C,code,object)
  part_of(C,B)

  D                           E F
  -------------------         -------------------------
  structure(D,atomic)    =>   predicate(E,state,be,D,F)
  part_of(D,B)                property(F,valid)
```

John ***is*** *a rich customer.*

```
A B C D E
------------------------------------------
named(B,'John')
object(B,named_entity,person)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,customer,person)
property(D,rich)
predicate(E,state,be,B,D)
```

A customer is ***richer than*** *John.*

```
A B C D E F
------------------------------------------
named(B,'John')
object(B,named_entity,person)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,customer,person)
property(F,richer_than,B)
predicate(E,state,be,D,F)
```

John ***is*** *in the bank.*

A B ***C*** *D E*
named(B,'John')
object(B,named_entity,person)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
predicate(C,state,be,B)
structure(D,atomic)
quantity(D,cardinality,count_unit,E,eq,1)
object(D,bank,object)
modifier(C,location,in,D)

3.5 Verb Phrase Coordination

Verb phrases can be conjoined (*and*) and disjoined (*or*).

A screen flashes ***and*** *blinks.*

A B C D
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,screen,object)
predicate(C,event,flash,B)
predicate(D,state,blink,B)

A screen flashes ***or*** *blinks.*

```
┌──────────────────────────────────────────────────────────────┐
│ A B                                                          │
├──────────────────────────────────────────────────────────────┤
│ structure(B,atomic)                                          │
│ quantity(B,cardinality,count_unit,A,eq,1)                    │
│ object(B,screen,object)                                      │
│  ┌──────────────────────────┐      ┌──────────────────────────┐│
│  │ C                        │      │ D                        ││
│  ├──────────────────────────┤  ∨   ├──────────────────────────┤│
│  │ predicate(C,event,flash,B)│     │ predicate(D,state,blink,B)││
│  └──────────────────────────┘      └──────────────────────────┘│
└──────────────────────────────────────────────────────────────┘
```

3.6 Modifying Verb Phrases

Facultative additional information detailing, for instance, how or where an action is performed is expressed by modifying the verb by an adverb or a prepositional phrase.

Adverbs can precede or follow the verb they modify. In case of ambiguity, attachment to following adverbs is preferred. Adverbs fall into semantic classes such as manner, time, location, direction.

A customer enters a card ***quickly****.*

A B C D E
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,customer,person)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,card,object)
predicate(E,event,enter,B,D)
modifier(E,manner,none,quickly)

Prepositional phrases (PPs) follow the verb they modify. The semantic class of a PP depends on the preposition of the PP as well as on the type of the noun occurring in the PP.

John enters a card ***in a bank****.*

A B C D E ***G F***
named(B,'John')
object(B,named_entity,person)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,card,object)
predicate(E,event,enter,B,D)
structure(G,atomic)
quantity(G,cardinality,count_unit,F,eq,1)
object(G,bank,object)
modifier(E,location,in,G)

John enters a card ***in the morning***.

A B C D E ***G F***
named(B,'John')
object(B,named_entity,person)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,card,object)
predicate(E,event,enter,B,D)
structure(G,atomic)
quantity(G,cardinality,count_unit,F,eq,1)
object(G,morning,time)
modifier(E,time,in,G)

3.7 Modifying Nouns and Noun Phrases

ACE offers a wide range of NP modifications: adjectives, relative clauses, *of*-PPs, Saxon genitives, possessive pronouns, and appositions.

Adjectives An adjective or a conjunction of adjectives precede a noun.

A ***rich*** *customer waits.*

A B C
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
property(B,rich)
object(B,customer,person)
predicate(C,state,wait,B)

The ***rich and old*** *customer waits.*

A B C
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
property(B,rich)
property(B,old)
object(B,customer,person)
predicate(C,state,wait,B)

Relative Sentences Relative sentences are an important natural language option to express complex NP modification.

A customer enters a card ***which is valid****.*

A B C D E ***F*** *G* ***H***
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,customer,person)
structure(E,atomic)
quantity(E,cardinality,count_unit,D,eq,1)
object(E,card,object)
property(H,valid)
predicate(F,state,be,E,H)
predicate(G,event,enter,B,E)

A customer enters a card ***which is green and which is valid****.*

A B C D ***E F*** *G* ***H I***
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,customer,person)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,card,object)
property(H,green)
predicate(E,state,be,D,H)
property(I,valid)
predicate(F,state,be,D,I)
predicate(G,event,enter,B,D)

A customer enters a card ***which is green or which is red****.*

A B C D G

structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,customer,person)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,card,object)

E H

property(H,green)
predicate(E,state,be,D,H)

∨

F I

property(I,red)
predicate(F,state,be,D,I)

predicate(G,event,enter,B,D)

***of*-Prepositional Phrases** NPs can be modified by *of*-PPs. Other PP modification of NPs is not possible but can be rephrased using relative sentences.

The surface ***of the card*** *has a green color.*

A B C D E ***F G***

structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,surface,object)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
property(D,green)
object(D,color,object)
predicate(E,state,have,B,D)
structure(F,atomic)
quantity(F,cardinality,count_unit,G,eq,1)
object(F,card,object)
relation(B,surface,of,F)

Possessive Nouns Possessive noun phrases are either introduced by a Saxon genitive (e.g. *Peter's*) or a possessive pronoun (e.g. *his*).

The customer's *card is valid.*

A B C ***D E*** *F*
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,card,object)
property(F,valid)
predicate(C,state,be,B,F)
structure(D,atomic)
quantity(D,cardinality,count_unit,E,eq,1)
object(D,customer,object)
relation(B,card,of,D)

Appositions Appositions of noun phrases can be proper names, quoted strings or variables.

The customer ***Mr Miller*** *enters a card.*

A B C D E
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,customer,person)
named(B,'Mr Miller')
object(B,named_entity,person)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,card,object)
predicate(E,event,enter,B,D)

*A customer **X** enters the password **"Jabberwocky"**.*

A B C D E
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,customer,person)
variable(B,'X')
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,password,object)
quoted_string(D,'Jabberwocky')
predicate(E,state,enter,B,D)

3.8 Conditional Sentences

Conditional sentences combine two sentences by an if-then construction.

If the password is valid then the machine accepts the request.

A B C D
property(A,valid)
predicate(B,state,be,D,A)
object(D,password,object)
quantity(D,cardinality,count_unit,C,eq,1)
structure(D,atomic)

$\Rightarrow$

E F G H I
predicate(E,unspecified,accept,G,I)
object(G,machine,object)
quantity(G,cardinality,count_unit,F,eq,1)
structure(G,atomic)
object(I,request,object)
quantity(I,cardinality,count_unit,H,eq,1)
structure(I,atomic)

3.9 Coordinated Sentences

Coordinated sentences combine simpler sentences by *and* and *or*.

The screen blinks and John waits.

```
A B C D E F
-------------------------------------------------
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,screen,object)
predicate(C,state,blink,B)
named(E, 'John')
object(E,named_entity,person)
structure(E,atomic)
quantity(E,cardinality,count_unit,D,eq,1)
predicate(F,state,wait,E)
```

A screen blinks or John waits.

```
D E
------------------------------------------------------------------------------------
  A B C
  -------------------------------------------
  structure(B,atomic)                                  F
  quantity(B,cardinality,count_unit,A,eq,1)     ∨      ------------------------
  object(B,screen,object)                              predicate(F,state,wait,E)
  predicate(C,state,blink,B)
named(E, 'John')
object(E,named_entity,person)
structure(E,atomic)
quantity(E,cardinality,count_unit,D,eq,1)
```

3.10 Quantified Sentences

Quantified sentences allow users to express existential quantification and universal quantification. Furthermore, a construct *there is* followed by a noun phrase introduces an existentially quantified singular object. Similarly, *there are* introduces a plural object.

a card ⇔ *There is a card.*

A B

structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,card,object)

John enters a card.

A B C D E

named(B,'John')
object(B,named_entity,person)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,card,object)
predicate(E,event,enter,B,D)

John enters every code. (= If there is a code then John enters it.)

A B

named(B,'John')
object(B,named_entity,person)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)

C D

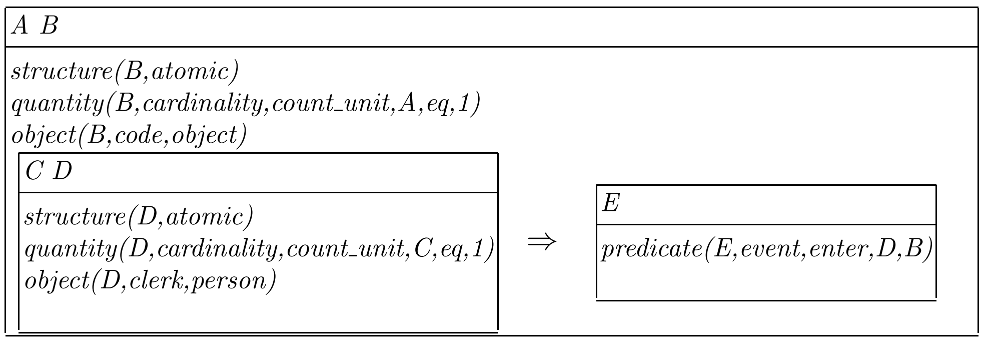
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,code,object)

⇒

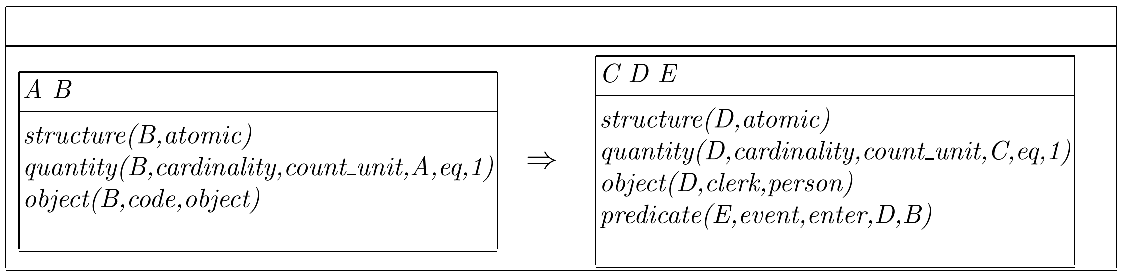
E

predicate(E,event,enter,B,D)

There is *a code* ***such that*** *every clerk enters it.*


A B
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,code,object)
C D
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,clerk,person)
⇒
E
predicate(E,event,enter,D,B)

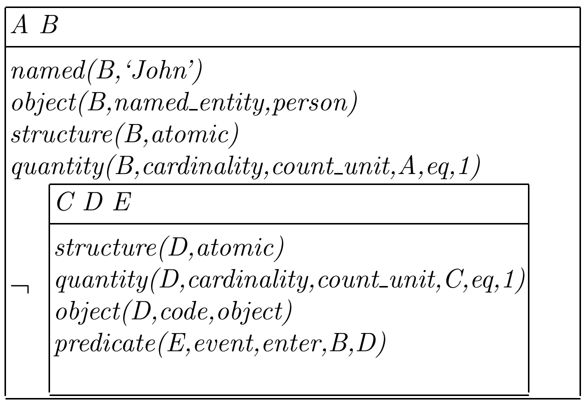
For every *code (there is) a clerk (such that he) enters it.*


A B
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,code,object)
⇒
C D E
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,clerk,person)
predicate(E,event,enter,D,B)

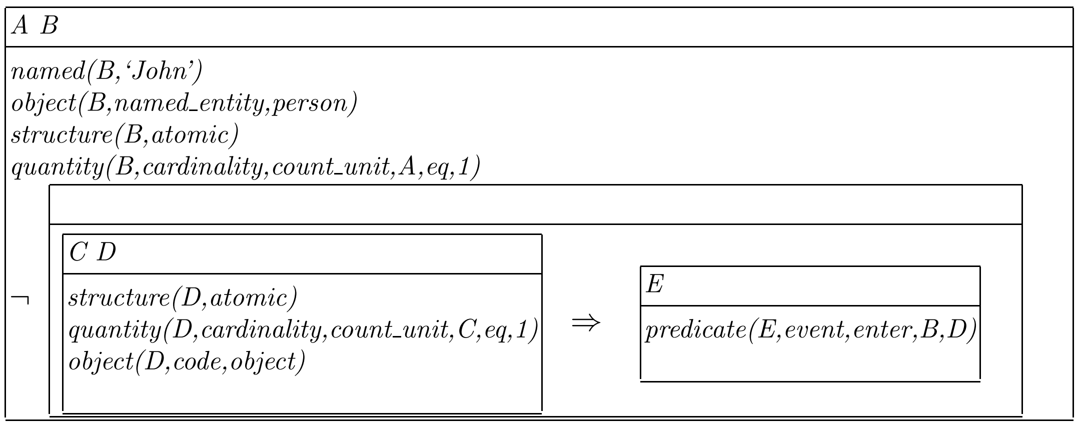
3.11 Negation

ACE offers many ways to negate noun phrases, quantified noun phrases, verb phrases and complete sentences.

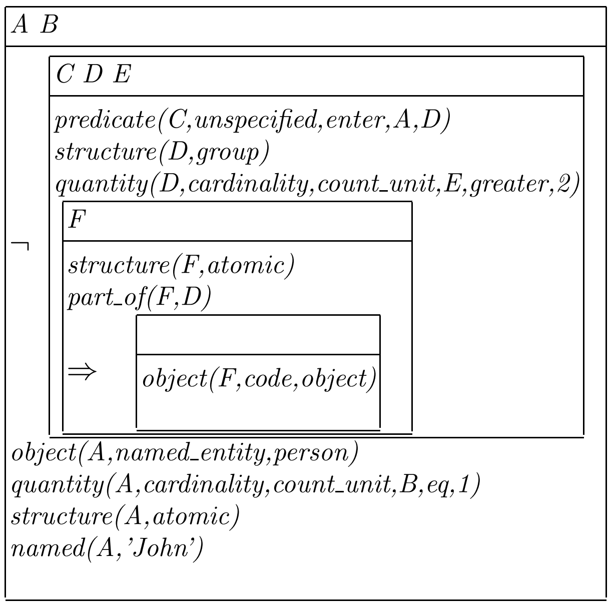
John enters no code.


A B
named(B,'John')
object(B,named_entity,person)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
¬
C D E
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,code,object)
predicate(E,event,enter,B,D)

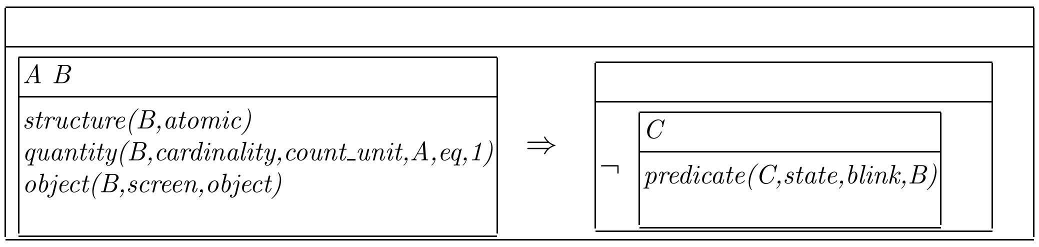
John enters not every code.


A B
named(B,'John')
object(B,named_entity,person)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
¬
C D
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,code,object)
⇒
E
predicate(E,event,enter,B,D)

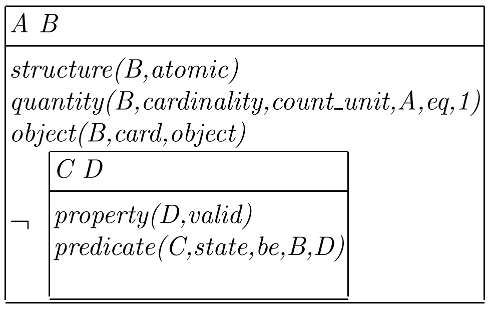
*John enters **not more than 2** codes.*


A B
¬
C D E
predicate(C,unspecified,enter,A,D)
structure(D,group)
quantity(D,cardinality,count_unit,E,greater,2)
F
structure(F,atomic)
part_of(F,D)
⇒
object(F,code,object)
object(A,named_entity,person)
quantity(A,cardinality,count_unit,B,eq,1)
structure(A,atomic)
named(A,'John')

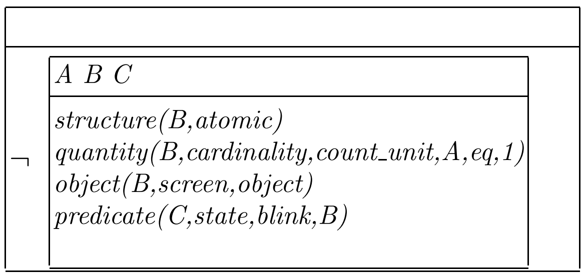
Every screen does not blink.


A B
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,screen,object)
⇒
¬
C
predicate(C,state,blink,B)

A card is not valid.


A B
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,card,object)
¬
C D
property(D,valid)
predicate(C,state,be,B,D)

It is not the case that *a screen blinks.*


¬
A B C
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,screen,object)
predicate(C,state,blink,B)

3.12 Query Sentences

Yes/no-questions ask for the existence of a state of affairs. These questions are translated exactly as their declarative counterparts.

Does John enter a card?

A B C D E
object(C,card,object)
quantity(C,cardinality,count_unit,B,eq,1)
structure(C,atomic)
predicate(D,event,enter,A,C)
object(A,named_entity,person)
quantity(A,cardinality,count_unit,E,eq,1)
structure(A,atomic)
named(A,'John')

Is the card valid?

A B C D
object(B,card,object)
quantity(B,cardinality,count_unit,A,eq,1)
structure(B,atomic)
property(C,valid)
predicate(D,state,be,B,C)

Who/What/Which-Questions Who/what/which-questions ask for the subjects or the objects of sentences. These questions are translated as their declarative counterparts but contain additional conditions for the query words.

Who *enters what?*

A B C
query(A,who)
structure(A,dom)
query(B,what)
structure(B,dom)
predicate(C,event,enter,A,B)

Which customer *enters a card?*

A B C D E
query(B,which)
structure(B,atomic)
quantity(B,cardinality,count_unit,A,eq,1)
object(B,customer,person)
structure(D,atomic)
quantity(D,cardinality,count_unit,C,eq,1)
object(D,card,object)
predicate(E,event,enter,B,D)

Where/When/How/...-Questions Where/when/how/...-questions ask for details of an action. These questions are translated as their declarative counterparts but contain additional conditions for the query words.

Where *does John enter a card?*

A B C D E F G
named(G,'John')
structure(G,atomic)
quantity(G,cardinality,count_unit,A,eq,1)
object(G,named_entity,person)
query(F,E,where)
modifier(B,location,F,E)
predicate(B,event,enter,G,C)
structure(C,atomic)
quantity(C,cardinality,count_unit,D,eq,1)
object(C,card,object)

When *does John enter a card?*

A B C D E F G
named(G,‘John’)
structure(G,atomic)
quantity(G,cardinality,count_unit,A,eq,1)
object(G,named_entity,person)
query(F,E,when)
modifier(B,time,F,E)
predicate(B,event,enter,G,C)
structure(C,atomic)
quantity(C,cardinality,count_unit,D,eq,1)
object(C,card,object)

How *does John enter a card?*

A B C D E F G
named(G,‘John’)
structure(G,atomic)
quantity(G,cardinality,count_unit,A,eq,1)
object(G,named_entity,person)
query(F,E,how)
modifier(B,manner,F,E)
predicate(B,event,enter,G,C)
structure(C,atomic)
quantity(C,cardinality,count_unit,D,eq,1)
object(C,card,object)

3.13 Predicate Declarations

$\boxed{\text{modifier(X,K,Preposition,Y/Adverb)}}$

X	discourse referent of the event or state that is modified
K ∈	{location, origin, direction, time, start, end, duration, instrument, comitative, manner, ...}
Y	discourse referent of an object, i.e. the NP of the modifying PP

$\boxed{\text{named(X,ProperName)}}$

X discourse referent of the object that is named

$\boxed{\text{object(X,Noun,K)}}$

X	discourse referent of the object that is denoted by the noun
K ∈	{person, object, time}

$\boxed{\text{part\_of(X,Y)}}$

X	discourse referent of an (atomic) object
Y	discourse referent of a (group) object

$\boxed{\text{predicate(E,D,Verb,X)}}$

E	discourse referent of the event or state that is denoted by the verb
D ∈	{event, state}
X	discourse referent of the subject

$\boxed{\text{predicate(E,D,Verb,X,Y)}}$

E	discourse referent of the event or state that is denoted by the verb
D ∈	{event, state}
X	discourse referent of the subject
Y	discourse referent of the direct object

$\boxed{\text{predicate(E,D,Verb,X,Y,Z)}}$

E	discourse referent of the event or state that is denoted by the verb
D ∈	{event, state}
X	discourse referent of the subject
Y	discourse referent of the direct object
Z	discourse referent of the indirect object

$\boxed{\text{proper\_part\_of(X,Y)}}$

X	discourse referent of an (atomic) object
Y	discourse referent of a (group) object

$\boxed{\text{property(X,IntransitiveAdjective)}}$

X discourse referent of the object a property of which is described by the adjective

$\boxed{\text{property(X,Comparative/TransitiveAdjective,Y)}}$

X	discourse referent of the object that is described
Y	discourse referent of the object with which X is compared or the object of the adjective

$\boxed{\text{property(X,TransitiveComparative,Y,Z)}}$

X	discourse referent of the object that is described
Y	discourse referent of the object of the adjective
Z	discourse referent of the object with which X is compared

$\boxed{\text{quantity(X,K,I,Q,J,N)}}$

K	∈ {cardinality, weight, size, length, volume, ...}
I	∈ {count_unit, kg, cm, liter, ...}
X	discourse referent of the object the quantity of which is indicated
Q	discourse referent of the (reified) quantity of X
J	∈ {eq, leq, geq, greater, less}
N	a number

$\boxed{\text{query(X,Q)}}$

X	discourse referent of the object that is asked for
Q	∈ {who, what, which}

$\boxed{\text{query(P,Y,Q)}}$

P	preposition
Y	discourse referent of an object, i.e. the NP of the modifying PP or an adverb
Q	∈ {where, when, how, ...}

$\boxed{\text{quoted\_string(X,QuotedString)}}$

X	discourse referent of the object that is denoted by the quoted string

$\boxed{\text{relation(X,Relation,of,Y)}}$

X	discourse referent of the object that is related to Y
Y	discourse referent of the object that is related to X

$\boxed{\text{structure(X, D)}}$

X	discourse referent of the object the structure of which is indicated
D	∈ {atomic, group, mass, dom}

$\boxed{\text{sum\_of(X,L)}}$

X	discourse referent of a (group) object
L	list of discourse referents of objects that are a proper part of X

$\boxed{\text{variable(X,Variable)}}$

X discourse referent of an object that is denoted by the variable

4 Conclusions

Attempto Controlled English (ACE) is a knowledge representation language with a dual face — humans can read ACE texts and machines can process them. ACE has already been used as specification language, as knowledge representation language, and as interface language to formal systems. We believe, that the attributes of ACE — specifically its ability to express business and policy rules — make it a prime candidate for the knowledge representation and query tasks of the semantic web.

5 Acknowledgment

The authors would like to thank Uta Schwertel, a former collaborator of the project Attempto, for her important contributions to the project.

References

1. Attempto Website. http://www.ifi.unizh.ch/attempto.
2. Patrick Blackburn and Johan Bos. *Working with Discourse Representation Structures*, volume 2nd of *Representation and Inference for Natural Language: A First Course in Computational Linguistics.* September 1999.
3. Norbert E. Fuchs and Uta Schwertel. Reasoning in Attempto Controlled English. In *Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2003)*, Lecture Notes in Computer Science, Hannover, 2003. Springer.
4. Jerry R. Hobbs. Ontological Promiscuity. In *Proceedings of the 23rd Annual Meeting of the ACL.* University of Chicago, 1985.
5. Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory.* Kluwer, Dordrecht/Boston/London, 1993.
6. Uta Schwertel. *Plural Semantics for Natural Language Understanding — A Computational Proof-Theoretic Approach.* PhD thesis, University of Zurich, Zurich, 2003.