Discourse Representation Structures for ACE 5

Technical Report ifi-2006.10

Norbert E. Fuchs[1], Stefan Hoefler[2], Kaarel Kaljurand[1]
Tobias Kuhn[1], Gerold Schneider[1], Uta Schwertel[3]

[1] Department of Informatics & Institute of Computational Linguistics, University of Zurich
Email: {fuchs,kalju,tkuhn,gschneid}@ifi.unizh.ch
[2] Language Evolution and Computation Research Unit, University of Edinburgh
Email: stefan@ling.ed.ac.uk
[3] Institut für Informatik, Ludwig-Maximilians-Universität München
Email: uta.schwertel@ifi.lmu.de

Abstract

This technical report describes the discourse representation structures (DRS) derived from texts written in version 5 of Attempto Controlled English (ACE 5).

Among other things, ACE 5 supports modal statements, negation as failure, and sentence subordination. These features require an extended form of discourse representation structures.

The discourse representation structure itself uses a reified, or 'flat' notation, meaning that its atomic conditions are built from a small number of predefined predicates that take constants standing for words of the ACE text as their arguments. Furthermore, each logical atom gets an index relating it to the sentence of the ACE text from which it was derived.

Contents

1 Introductory Notes

This technical report describes the representation of discourse representation structures (DRSs) derived from version 5 of Attempto Controlled English (ACE 5). It uses illustrative ACE examples, but does not describe ACE itself. For a complete description of the ACE language please refer to the Attempto web site [1]. An abstract grammar for ACE 5 – aimed to the linguist – will be made public soon. For the old version 4 of ACE, the abstract grammar is provided in [4].

We expect the reader to be familiar with the basic notions of Discourse Representation Theory (DRT) [5] as, for instance, introduced in [2]. Consult [3] for the semantics of modality and sentence subordination.

Section 2 introduces the notation used in this report. Sections 3 to 12 describe discourse representation structures derived from declarative ACE sentences, and section 13 those derived from ACE questions.

2 Notation

2.1 Basics

The ACE parser translates an ACE text unambiguously into a DRS representation.

The discourse representation structure derived from the ACE text is returned 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 '`=>`'. Furthermore, we use non-standard logical connectors for possibility '`<>`', necessity '`[]`', negation as failure '`~`', and a connector for the assignment of variables to sub-DRSs '`:`'.

A DRS like

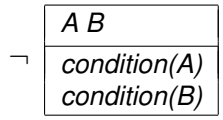
```
drs([A,B],[condition(A),condition(B)])
```

is usually pretty-printed as

A B
condition(A) *condition(B)*

2.2 Classical Negation

A negated DRS like



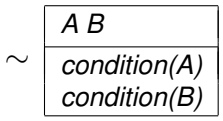
is internally represented as

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

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

2.3 Negation As Failure

A DRS that is negated using negation as failure (NAF) is marked with a tilde sign:



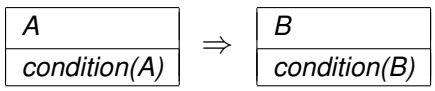
It is represented as

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

We have defined `~/1` as a prefix operator which stands for negation as failure.

2.4 Implication and Disjunction

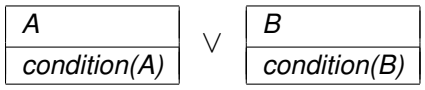
In a DRS, all variables are thus existentially quantified unless they stand in the restrictor of an implication. The implication



is internally represented as

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

The disjunction



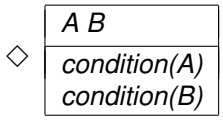
is likewise internally represented as

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

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

2.5 Possibility and Necessity

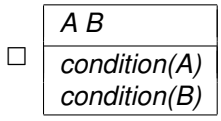
Possibility and necessity (see next subsection) are modal extensions for DRSs. Consult [3] for details about such modal constructs and their representations in first-order logic. Possibility is represented with a diamond sign



and is internally represented as

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

Necessity is represented with a box sign



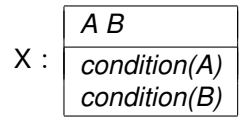
and is internally represented as

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

The prefix operators `<>/1` and `[]/1` are used to represent possibility and necessity, respectively.

2.6 Sentence Subordination (*that*-Subordination)

For sentences like 'John believes that Mary sleeps' we need an extended DRS syntax. For that reason we introduce a new notation that allows to attach labels to sub-DRSs. Consult [3] for details.



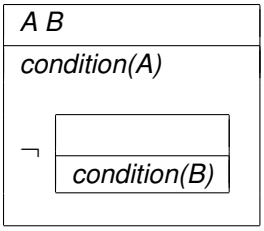
This is internally represented as

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

The infix operator `:/2` is used to attach labels to sub-DRSs.

2.7 Nesting

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



is represented as

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

2.8 Flat Notation

The discourse representation structure uses a reified, or 'flat' notation for logical atoms. For example, the noun *a card* that customarily would be represented as

```
card(A)
```

is represented here as

```
object(A, atomic, 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.

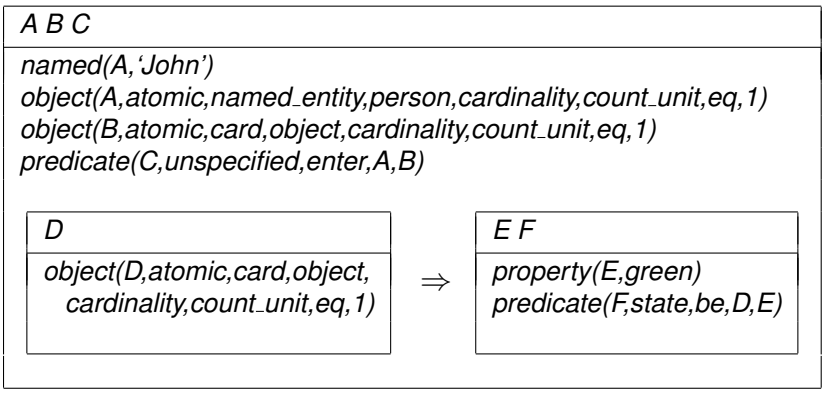
2.9 Sentence Numbers

Logical atoms occurring in `drs/2` are actually written as `Atom-I` (using an infix operator `-/2`) where the number `I` refers to to the sentence from which `Atom` was derived.

The example text

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

the DRS of which is



will thus internally be represented as

```
drs([A,B,C],[named(A,'John')-1,
object(A,atomic,named_entity,person,cardinality,count_unit,eq,1)-1,
object(B,atomic,card,object,cardinality,count_unit,eq,1)-1,
predicate(C,unspecified,enter,A,B)-1,
drs([D],[object(D,atomic,card,object,cardinality,count_unit,eq,1)-2])=>
drs([E,F],[property(E,green)-2,predicate(F,state,be,D,E)-2])]).
```

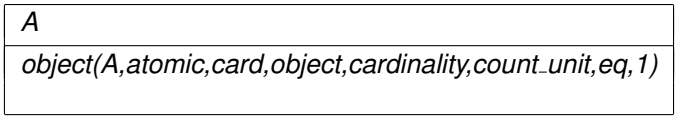
The following sections provide the discourse representation structures for a selected number of ACE sentences in the form they will be output by the ACE parser. Logical atoms, however, are represented without the number pointing to the sentence from which they were derived. Refer to the index at the end of this document if you want to find an explanatory DRS for a particular predicate that you saw in an ACE 4 DRS.

Using illustrative ACE examples this report completely describes the language of DRSs derived from ACE texts. For a complete description of the ACE language itself please refer to the Attempto web site [1].

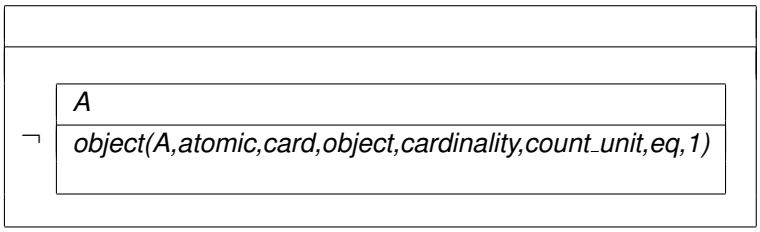
3 Noun Phrases

3.1 Singular Countable Noun Phrases

a card

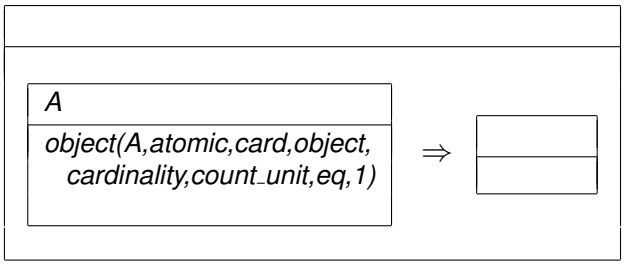


no card

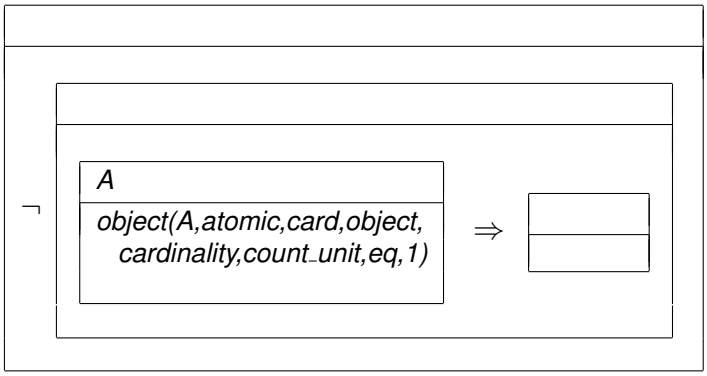


Note that the representation of "*no card*" depends on the context (see section 9.1.1).

every card

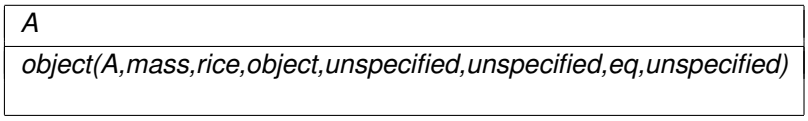


not every card

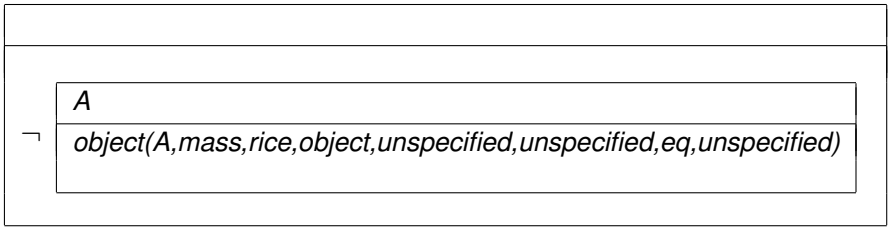


3.2 Mass Nouns

some rice

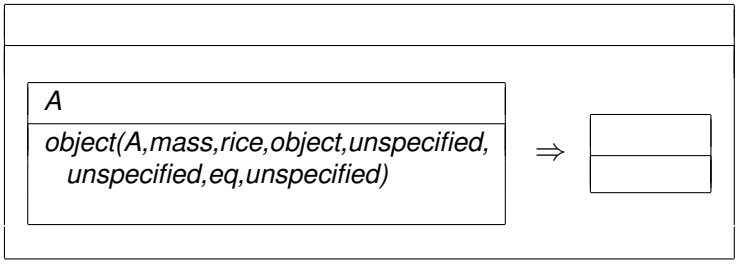


no rice

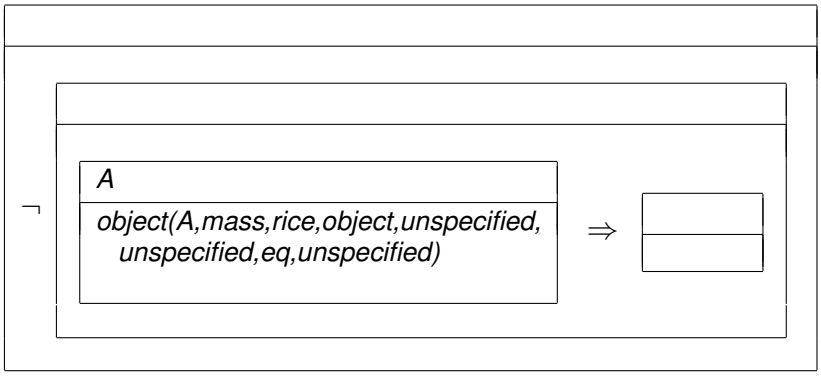


Note that the representation of "*no rice*" depends on the context (see section 9.1.1). Furthermore, the determiner *no* is ambiguous between countable and mass. For nouns that can be countable or mass, e.g. *money*, preference to countable is given. Mass reading can be forced by using sentential negation, e.g. *It is false that some money is omnipotent*.

all rice



not all rice



3.3 Proper Names

John

A
named(A,'John') *object(A,atomic,named_entity,person,cardinality,count_unit,eq,1)*

3.4 Plural Noun Phrases

some cards

A
object(A,group,card,object,cardinality,count_unit,geq,2)

***2** cards*

A
*object(A,group,card,object,cardinality,count_unit,**eq,2**)*

***five** cards*

A
*object(A,group,card,object,cardinality,count_unit,**eq,5**)*

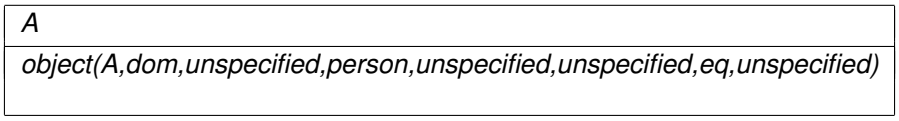
no cards

¬ | *A* *object(A,group,card,object,cardinality,count_unit,eq,1)*

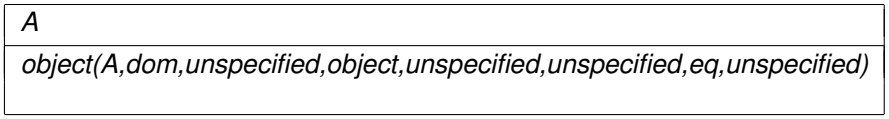
Note that the representation of "*no cards*" depends on the context (see section 9.1.1).

3.5 Indefinite Pronouns

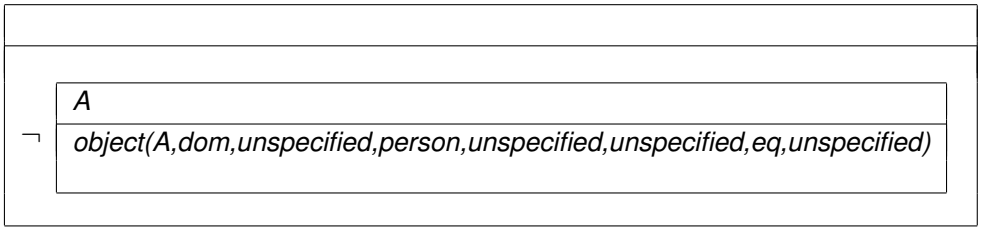
someone / somebody



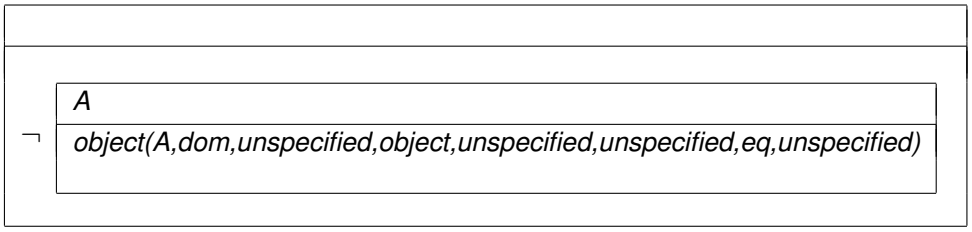
something



no one / nobody

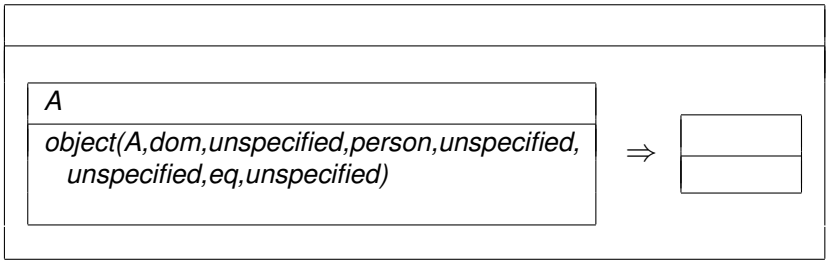


nothing

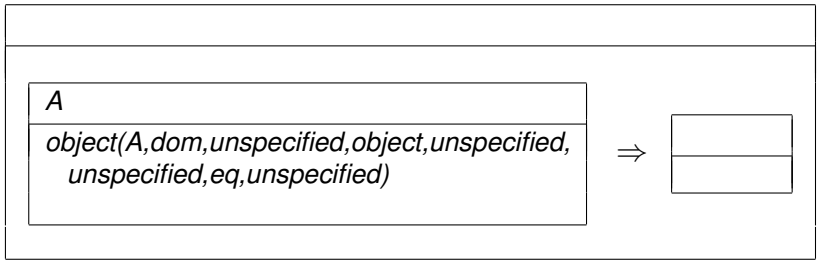


Note that the representations of "*no one*", "*nobody*", and "*nothing*" depend on the context (see section 9.1.1).

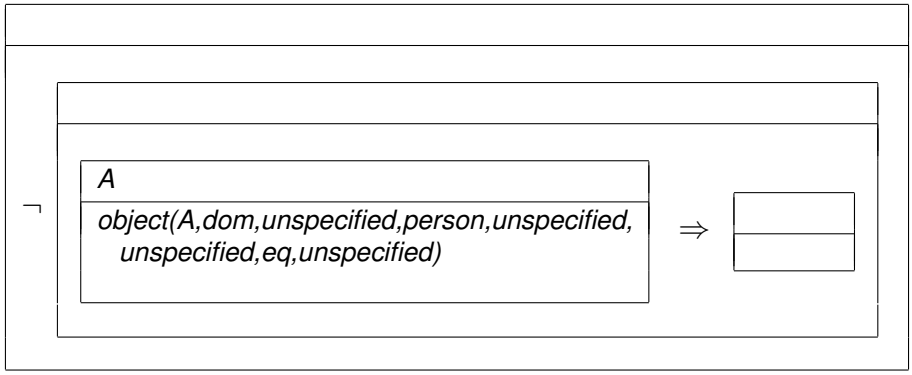
everyone / everybody



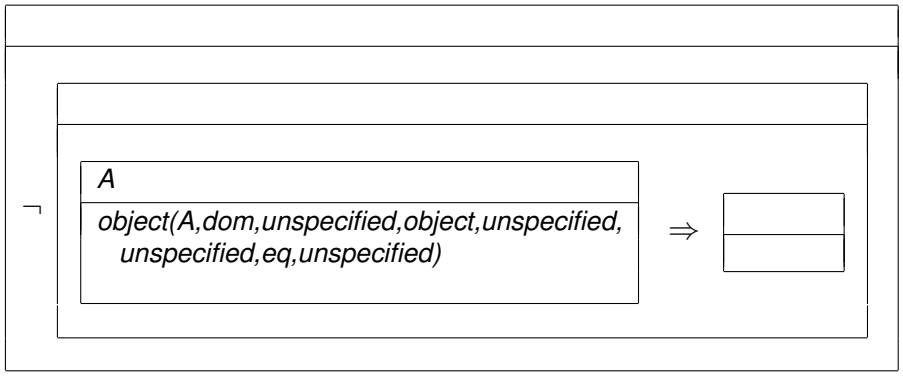
everything



not everyone / not everybody



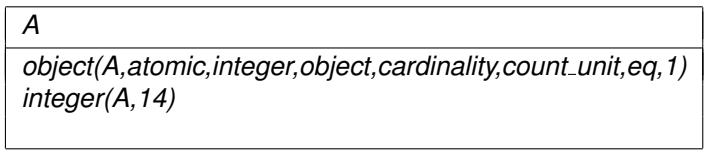
not everything



3.6 Numbers and Strings

Note: Numbers and strings are only allowed in copula (see section 4.4.4).

14



”abc”

A
object(A,atomic,string,object,cardinality,count_unit,eq,1) *string(A,abc)*

3.7 Generalised Quantors

at least 2 *cards*

A
*object(A,group,card,object,cardinality,count_unit,**geq,2**)*

at most 2 *cards*

A
*object(A,group,card,object,cardinality,count_unit,**leq,2**)*

more than 2 *cards*

A
*object(A,group,card,object,cardinality,count_unit,**greater,2**)*

less than 2 *cards*

A
*object(A,group,card,object,cardinality,count_unit,**less,2**)*

3.8 Noun Phrase Conjunction

a customer and a clerk

A B C
object(A,atomic,customer,person,cardinality,count_unit,eq,1) *object(B,atomic,clerk,person,cardinality,count_unit,eq,1)* *proper_part_of(C,A)* *proper_part_of(C,B)* *object(C,group,unspecified,unspecified,cardinality,count_unit,eq,2)*

3.9 Measurement Noun Phrases

2 kg of *apples*

A
*object(A,group,apple,object,**weight,kg,eq,2**)*

2 kg of *rice*

A B
*object(A,mass,rice,object,**weight,kg,eq,2**)*

4 Verb Phrases

Note: the version of the currently used large lexicon, *clex*, does not distinguish event from state verbs. The distinction is thus unspecified.

4.1 Intransitive Verbs

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

A ***B***
object(A,atomic,customer,person,cardinality,count_unit,eq,1) ***predicate(B,unspecified,wait,A)***

4.2 Transitive Verbs

The following two sentences are parsed identically.

John ***enters*** *a card.*
A card ***is entered by*** *John.*

A B ***C***
named(A,'John') *object(A,atomic,named_entity,person,cardinality,count_unit,eq,1)* *object(B,atomic,card,object,cardinality,count_unit,eq,1)* ***predicate(C,unspecified,enter,A,B)***

4.3 Ditransitive Verbs

The following four sentences are parsed identically.

*A clerk **gives** a password **to** a customer.*
*A clerk **gives** a customer a password.*
*A password **is given to** a customer **by** a clerk.*
*A customer **is given** a password **by** a clerk.*

*A B C **D***
object(A,atomic,clerk,person,cardinality,count_unit,eq,1) *object(B,atomic,password,object,cardinality,count_unit,eq,1)* *object(C,atomic,customer,person,cardinality,count_unit,eq,1)* ***predicate(D,unspecified,give,A,B,C)***

4.4 Copula

4.4.1 Copula and Predicative Adjectives

*A card **is valid**.*

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

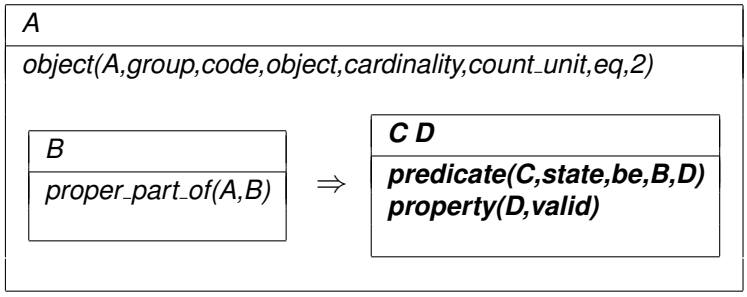
*A card **is valid and correct**.*

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

*2 codes **are valid**.*

*A **B C***
object(A,group,code,object,cardinality,count_unit,eq,2) ***predicate(B,state,be,A,C)*** ***property(C,valid)***

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



4.4.2 Copula and Noun Phrase

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

*A **B** C*
named(A,'John') *object(A,atomic,named_entity,person,cardinality,count_unit,eq,1)* ***predicate(B,state,be,A,C)*** *object(C,atomic,customer,person,cardinality,count_unit,eq,1)* *property(C,rich)*

4.4.3 Copula and Prepositional Phrase

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

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

4.4.4 Copula and Number or String

*A value **is** 14.*

*A **B** C*
object(A,atomic,value,object,cardinality,count_unit,eq,1) *object(B,atomic,integer,object,cardinality,count_unit,eq,1)* *integer(B,14)* ***predicate(C,state,be,A,B)***

*A text **is** "abc".*

*A **B** C*
object(A,atomic,text,object,cardinality,count_unit,eq,1) *object(B,atomic,string,object,cardinality,count_unit,eq,1)* *string(B,abc)* ***predicate(C,state,be,A,B)***

4.5 Coordinated Verb Phrases

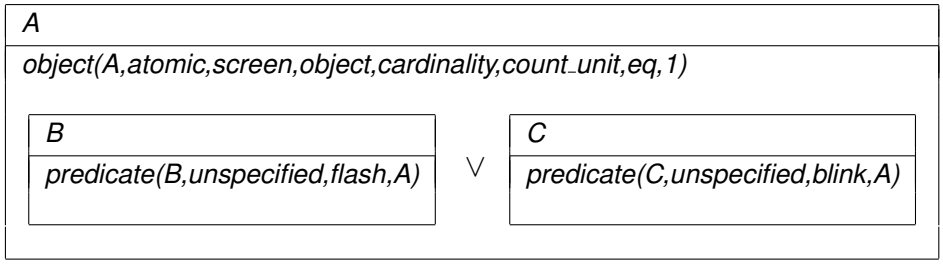
4.5.1 Verb Phrase Conjunction

A screen flashes and blinks.

A B C
object(A,atomic,screen,object,cardinality,count_unit,eq,1) *predicate(B,unspecified,flash,A)* *predicate(C,unspecified,blink,A)*

4.5.2 Verb Phrase Disjunction

A screen flashes or blinks.



5 Modifying Nouns and Noun Phrases

5.1 Adjectives

5.1.1 Simple Adjectives

*A **rich** customer waits.*

A B
object(A,atomic,customer,person,cardinality,count_unit,eq,1) ***property(A,rich)*** *predicate(B,unspecified,wait,A)*

5.1.2 Comparatives and Transitive Adjectives

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

A B C D
named(A,'John') *object(A,atomic,named_entity,person,cardinality,count_unit,eq,1)* *object(B,atomic,customer,person,cardinality,count_unit,eq,1)* *predicate(C,state,be,B,D)* ***property(D,richer_than,A)***

5.1.3 Adjective Conjunction

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

A B
object(A,atomic,customer,person,cardinality,count_unit,eq,1) ***property(A,old)*** ***property(A,rich)*** *predicate(B,unspecified,wait,A)*

5.2 Appositions

5.2.1 Quoted Strings

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

A B C
object(A,atomic,customer,person,cardinality,count_unit,eq,1) *predicate(B,unspecified,enter,A,C)* *object(C,atomic,password,object,cardinality,count_unit,eq,1)* ***quoted_string(C,'Jabberwocky')***

5.2.2 Variables

*A **customer X** greets a clerk. The clerk is happy. **X** is glad.*

***A** B C D E F G*
object(A,atomic,customer,person,cardinality,count_unit,eq,1) *object(B,atomic,clerk,person,cardinality,count_unit,eq,1)* *predicate(C,unspecified,greet,A,B)* *predicate(D,state,be,B,E)* *property(E,happy)* ***predicate(F,state,be,A,G)*** *property(G,glad)*

Note: Variables do not appear in the DRS. They only establish anaphoric references.

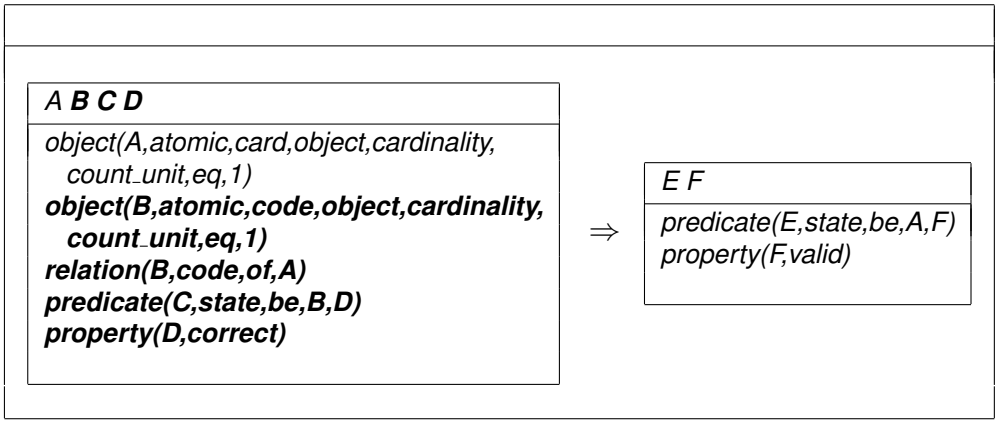
5.3 Relative Sentences

5.3.1 Simple Relative Sentences

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

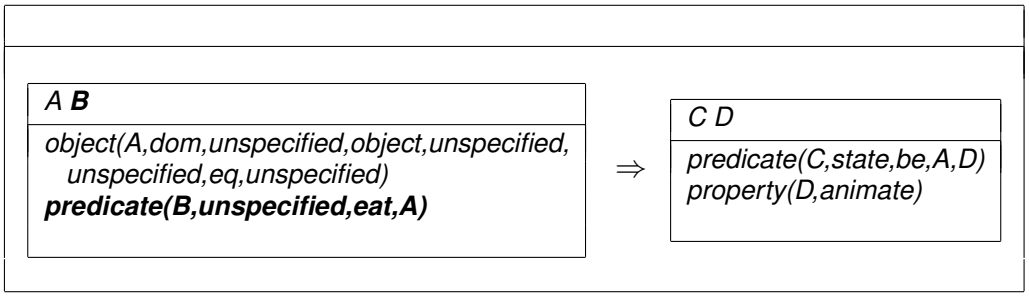
*A B C **D E***
object(A,atomic,customer,person,cardinality,count_unit,eq,1) *object(B,atomic,card,object,cardinality,count_unit,eq,1)* *predicate(C,unspecified,enter,A,B)* ***predicate(D,state,be,B,E)*** ***property(E,valid)***

Every card ***the code of which is correct*** *is valid.*



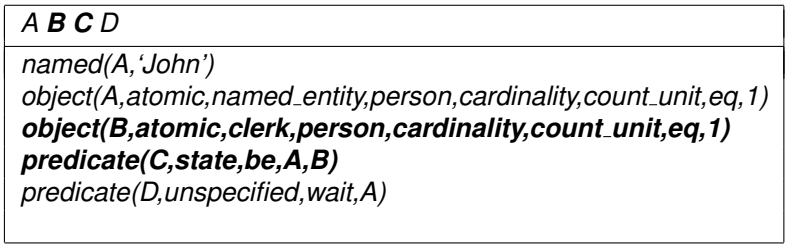
5.3.2 Relativized Indefinite Pronouns

Everything ***which eats*** *is animate.*



5.3.3 Relativized Personal Pronouns

John ***who is a clerk*** *waits.*



5.3.4 Relativized Variables

There is a card X. X ***which a customer possesses*** *is valid.*

A ***B C*** *D E*
object(A,atomic,card,object,cardinality,count_unit,eq,1) ***object(B,atomic,customer,person,cardinality,count_unit,eq,1)*** ***predicate(C,unspecified,possess,B,A)*** *predicate(D,state,be,A,E)* *property(E,valid)*

5.3.5 Relative Sentence Conjunction

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

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

5.3.6 Relative Sentence Disjunction

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

A B C
object(A,atomic,customer,person,cardinality,count_unit,eq,1) *object(B,atomic,card,object,cardinality,count_unit,eq,1)* *predicate(C,unspecified,enter,A,B)*

D E		***F G***
predicate(D,state,be,B,E) ***property(E,green)***	$\vee$	***predicate(F,state,be,B,G)*** ***property(G,valid)***

5.4 *of*-Prepositional Phrases

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

*A **B** C D*
object(A,atomic,surface,object,cardinality,count_unit,eq,1) ***object(B,atomic,card,object,cardinality,count_unit,eq,1)*** ***relation(A,surface,of,B)*** *object(C,atomic,color,object,cardinality,count_unit,eq,1)* *property(C,green)* *predicate(D,unspecified,have,A,C)*

5.5 Possessive Nouns

Possessive nouns are introduced by a possessive pronoun or a Saxon genitive. While possessive nouns are equivalent to *of* PPs, Saxon genitives in general are not because of the scoping rules of quantifiers:

- a man's dog (1 man with 1 dog) $=$ a dog of a man (1 man with 1 dog)
- every man's dog (several men each with 1 dog) $\neq$ a dog of every man (1 dog of several men)

***The customer's** card is valid.*

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

Note: There are no recursive Saxon genitives. "A customer's card" is in ACE, but "A customer's card's code" is not.

*There is a customer. **His** code is correct.*

A B C D
object(A,atomic,customer,person,cardinality,count_unit,eq,1) *object(B,atomic,code,object,cardinality,count_unit,eq,1)* ***relation(B,code,of,A)*** *predicate(C,state,be,B,D)* *property(D,correct)*

6 Modifying Verb Phrases

6.1 Adverbs

*A customer **quickly** enters a card.* $\Leftrightarrow$ *A customer enters a card **quickly**.*

A B C
object(A,atomic,customer,person,cardinality,count_unit,eq,1) *object(B,atomic,card,object,cardinality,count_unit,eq,1)* *predicate(C,unspecified,enter,A,B)* ***modifier(C,manner,none,quickly)***

6.2 Prepositional Phrases

Prepositional phrases create `modifier/4`-predicates which have always the type `unspecified`.

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

*A B C **D***
named(A,'John') *object(A,atomic,named_entity,person,cardinality,count_unit,eq,1)* *object(B,atomic,card,object,cardinality,count_unit,eq,1)* *predicate(C,unspecified,enter,A,B)* ***object(D,atomic,bank,object,cardinality,count_unit,eq,1)*** ***modifier(C,unspecified,in,D)***

7 Composite Sentences

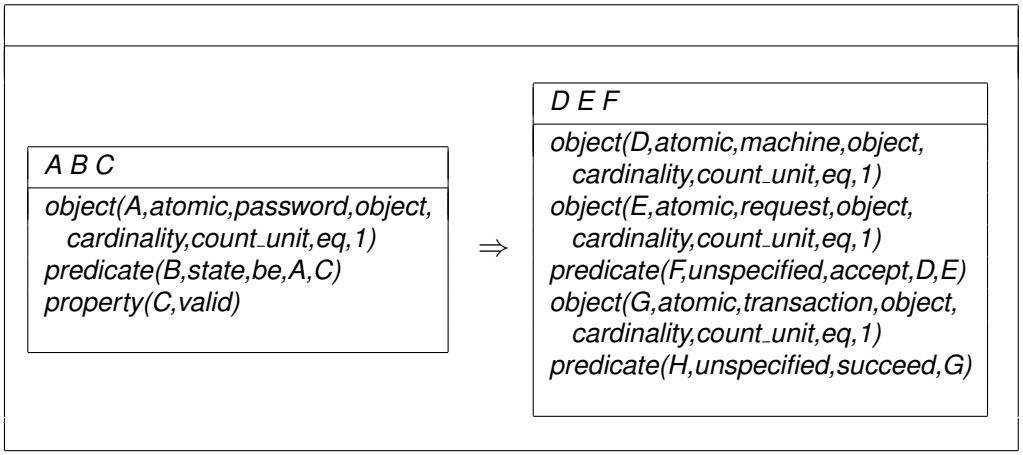
7.1 Conditional Sentences

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

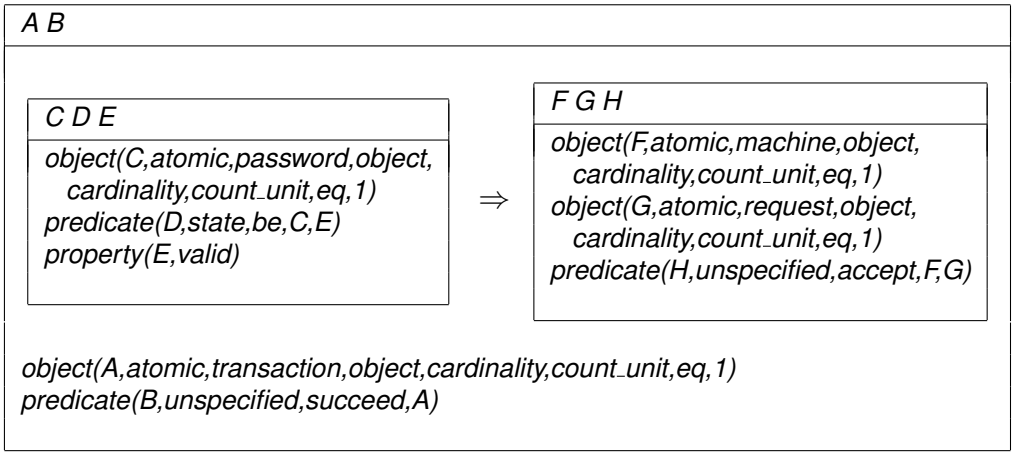
A B C *object(A,atomic,password,object,* *cardinality,count_unit,eq,1)* *predicate(B,state,be,A,C)* *property(C,valid)*	$\Rightarrow$	*D E F* *object(D,atomic,machine,object,* *cardinality,count_unit,eq,1)* *object(E,atomic,request,object,* *cardinality,count_unit,eq,1)* *predicate(F,unspecified,accept,D,E)*

Conditional sentences always take wide scope. Narrow scope requires starting a new sentence.

If the password is valid then the machine accepts the request and the transaction succeeds.



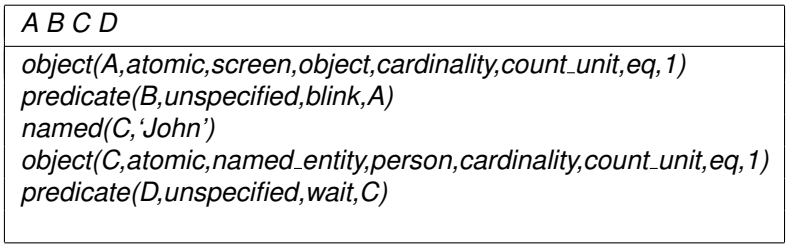
If the password is valid then the machine accepts the request. The transaction succeeds.



7.2 Coordinated Sentences

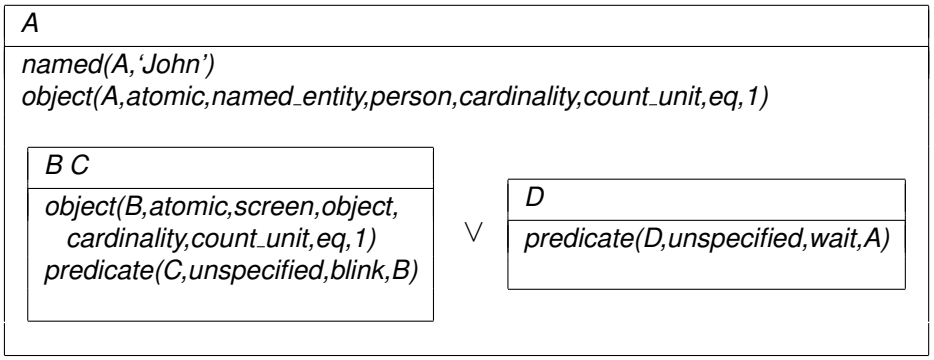
7.2.1 Sentence Conjunction

The screen blinks and John waits.



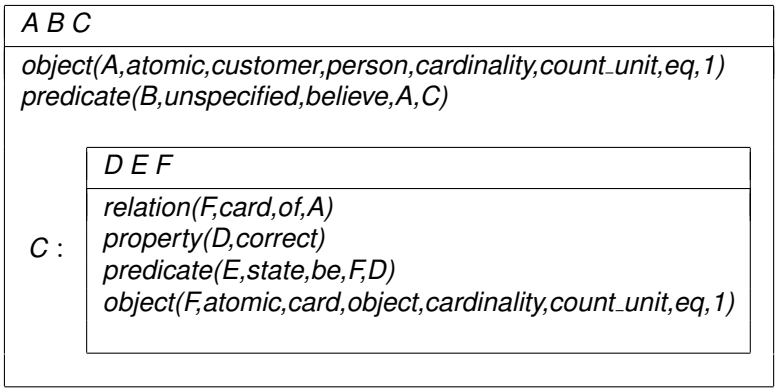
7.2.2 Sentence Disjunction

A screen blinks or John waits.



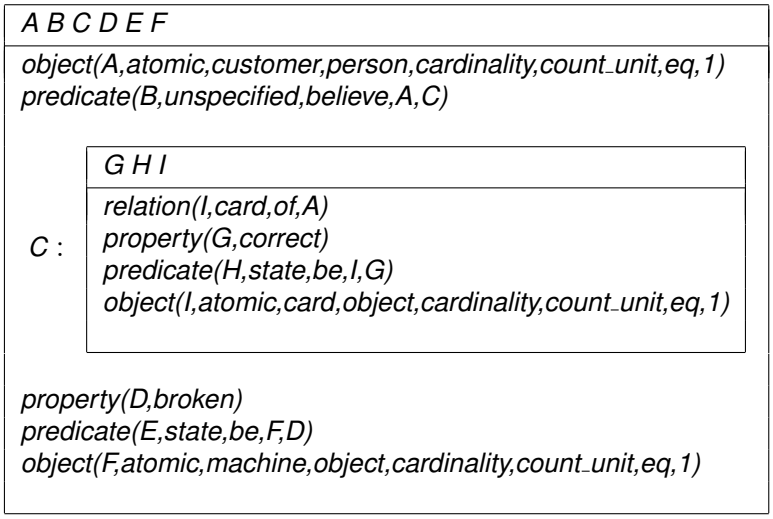
7.3 Sentence Subordination

*A customer believes **that** his own card is correct.*

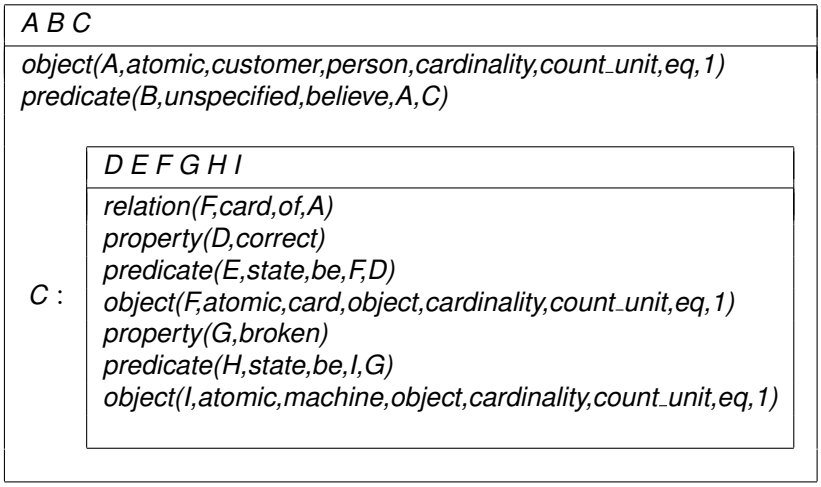


Sentence subordination takes narrow scope unless the word "*that*" is repeated.

*A customer believes **that** his own card is correct and the machine is broken.*



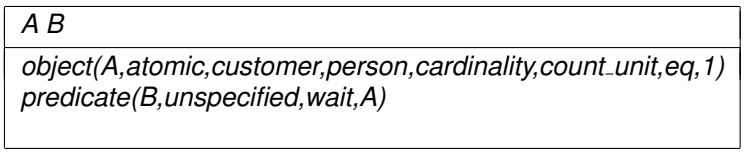
*A customer believes **that** his own card is correct and **that** the machine is broken.*



7.4 Positive Sentence Marker

For consistency reasons, we support the sentence-initial phrase "*It is true that ...*". It does not make much sense for normal sentences but it is useful for macros (see section 11).

***It is true that** a customer waits.*



8 Quantified Sentences

8.1 Existential Quantification

A card ... $\Leftrightarrow$ *There is a card.*

A
object(A,atomic,card,object,cardinality,count_unit,eq,1)

John enters a card.

A B C
named(A,'John') *object(A,atomic,named_entity,person,cardinality,count_unit,eq,1)* *object(B,atomic,card,object,cardinality,count_unit,eq,1)* *predicate(C,unspecified,enter,A,B)*

8.2 Universal Quantification

John enters every code.

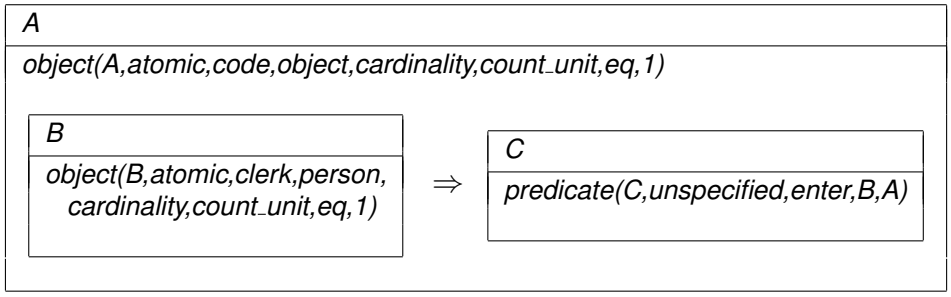
A		
named(A,'John') *object(A,atomic,named_entity,person,cardinality,count_unit,eq,1)*		
B *object(B,atomic,code,object,* *cardinality,count_unit,eq,1)*	$\Rightarrow$	*C* *predicate(C,unspecified,enter,A,B)*

8.3 Global Quantification

8.3.1 Global Existential Quantification

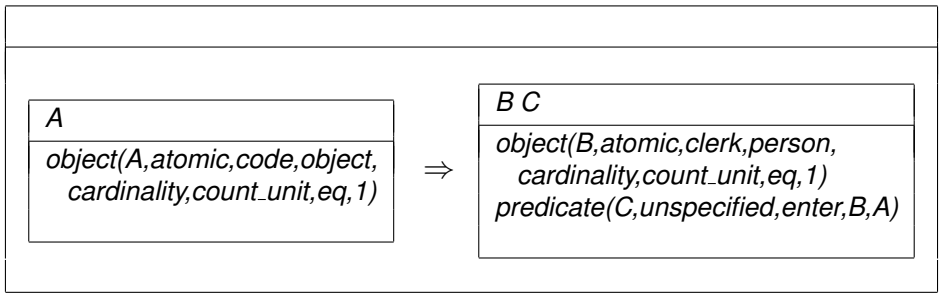
There is a code such that every clerk enters it and *There is a code that every clerk enters* lead to an identical DRS.

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



8.3.2 Global Universal Quantification

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



9 Negation

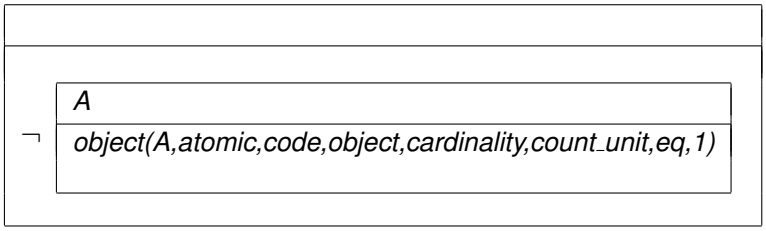
Unless stated otherwise, we talk about classical negation. For negation as failure see the last subsection.

9.1 Quantor Negation

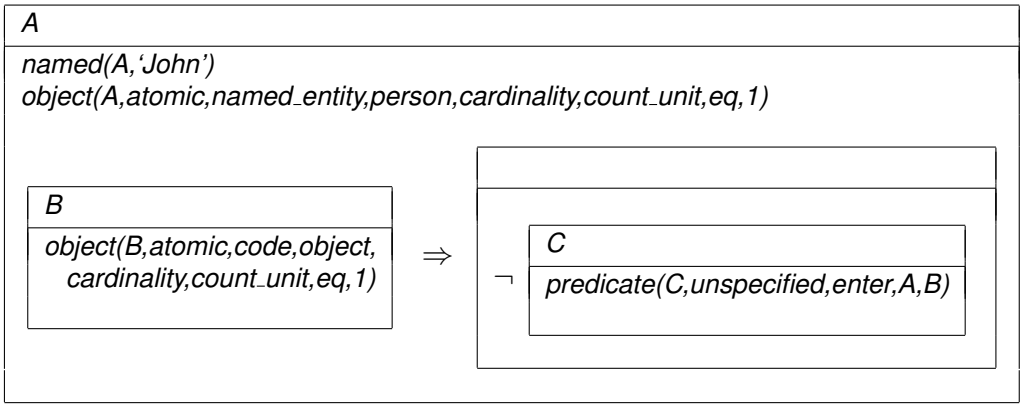
9.1.1 Negated Existential Quantor

Note that negated existential quantors can produce different DRS representations, depending on the context. Within "*there is ...*", a negated sub-DRS is created. Otherwise, we get an implication with a negated sub-DRS on the right hand side.

There is no code.

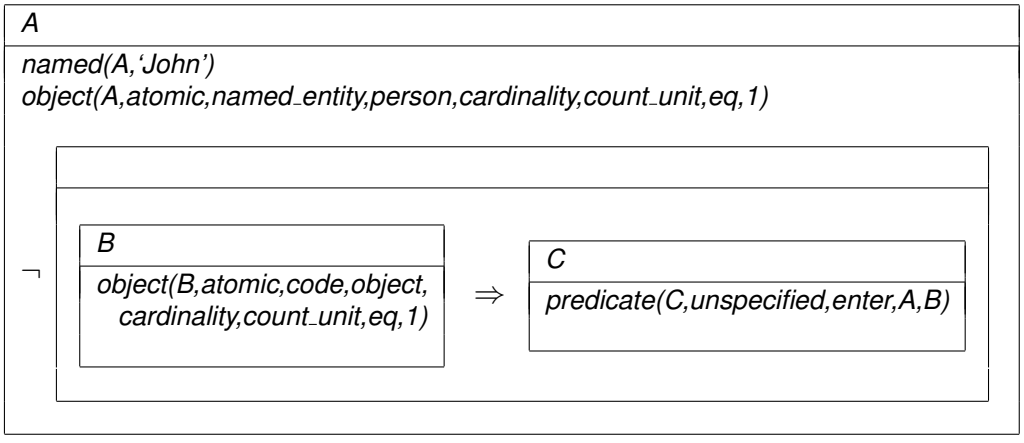


John enters no code.



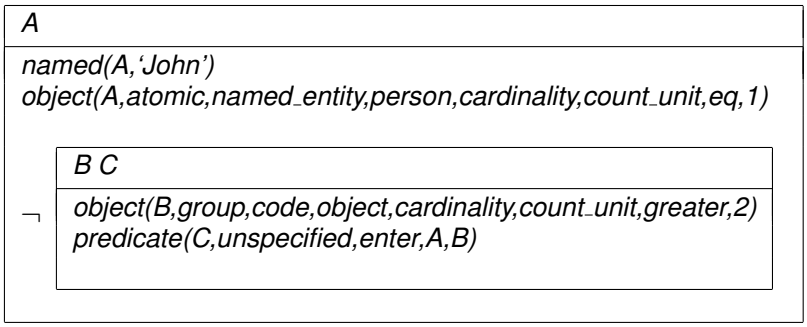
9.1.2 Negated Universal Quantor

John enters not every code.

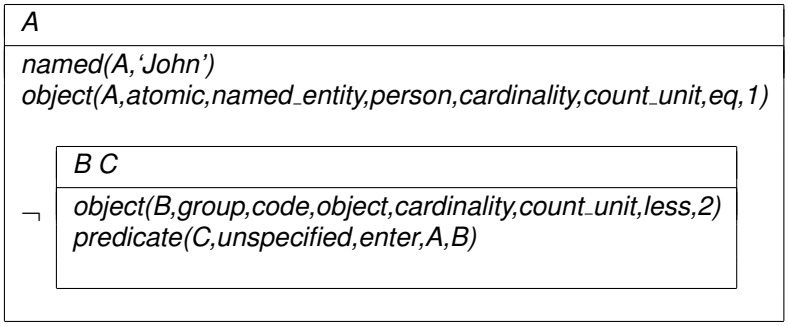


9.1.3 Negated Generalised Quantors

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



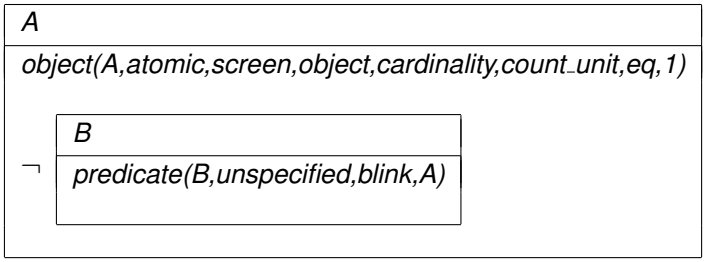
*John enters **not less than 2** codes.*



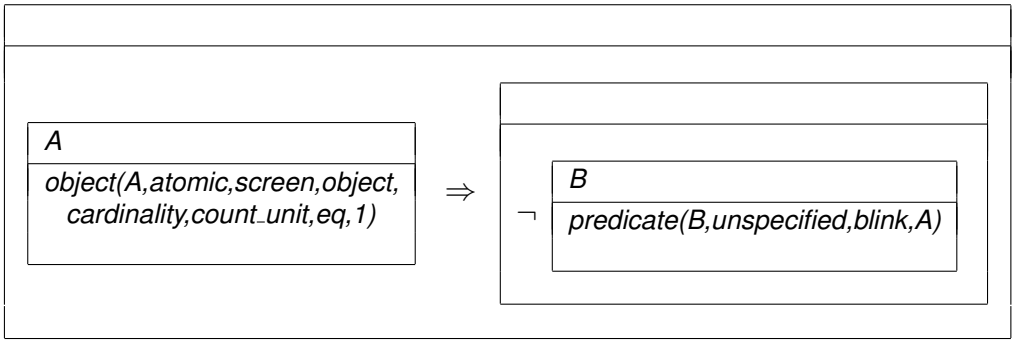
9.2 Verb Phrase Negation

9.2.1 Intransitive Verbs

A screen does not blink.

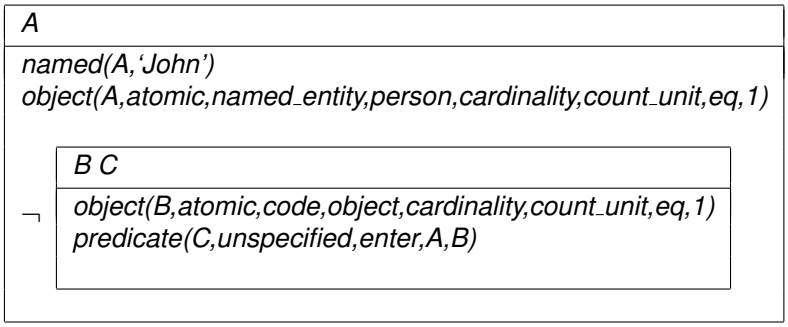


Every screen does not blink.

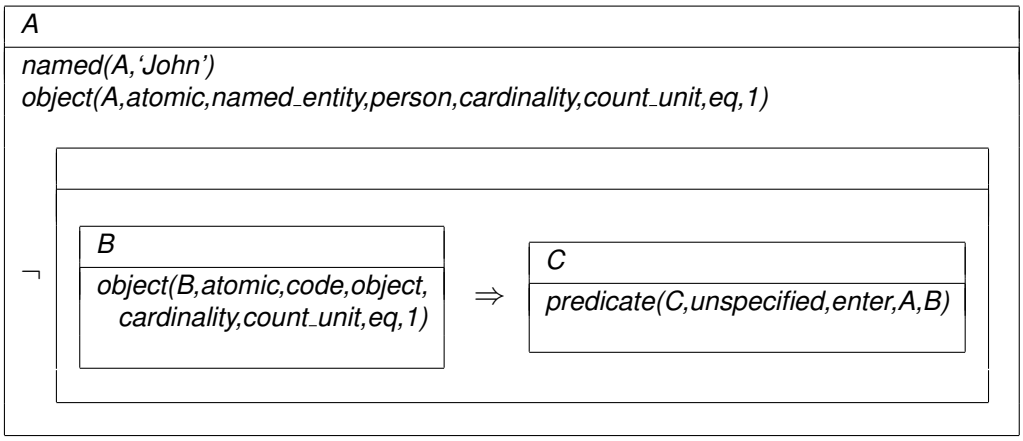


9.2.2 Transitive Verbs

John does not enter a code.

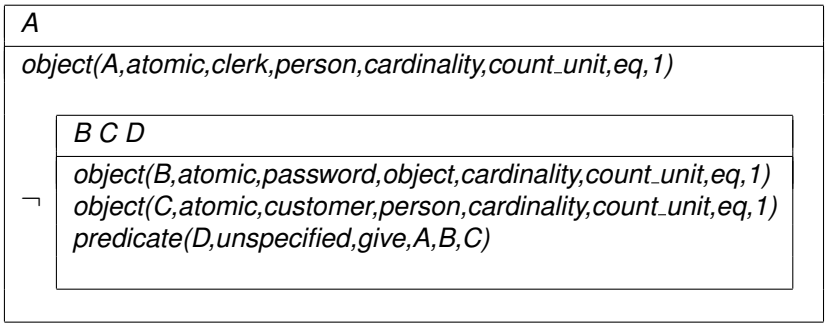


John does not enter every code.



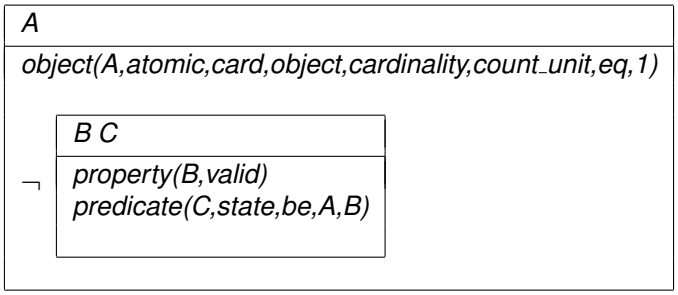
9.2.3 Ditransitive Verbs

A clerk does not give a password to a customer.



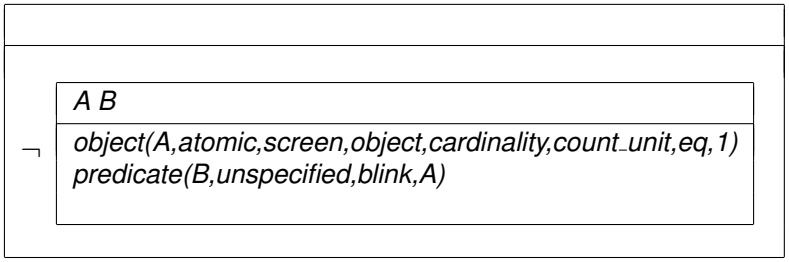
9.2.4 Copula

A card is not valid.

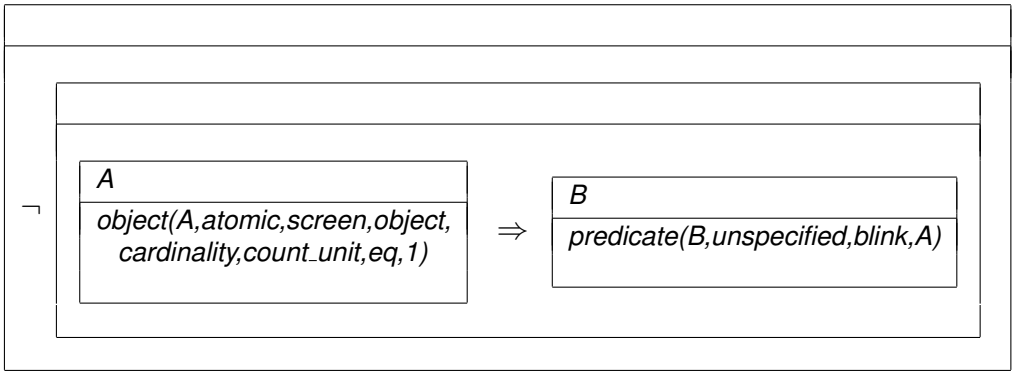


9.3 Sentence Negation

It is false that *a screen blinks.*

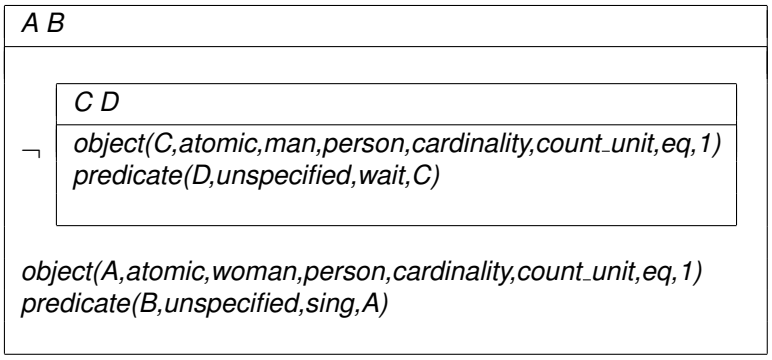


It is false that *every screen blinks.*

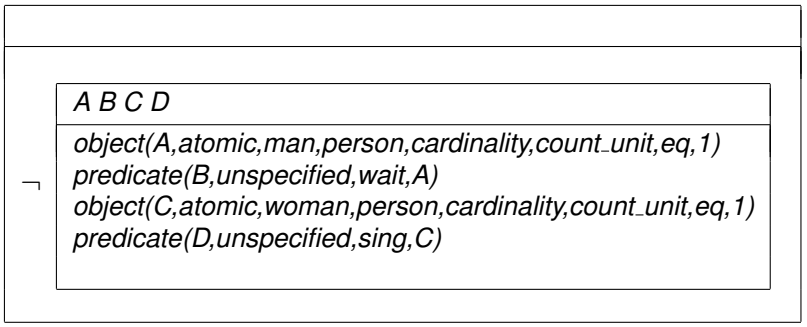


Sentence negation takes narrow scope, but wide scope can be triggered by repeating the *that* complementizer. Compare the following two examples.

It is false that *a man waits and a woman sings.*



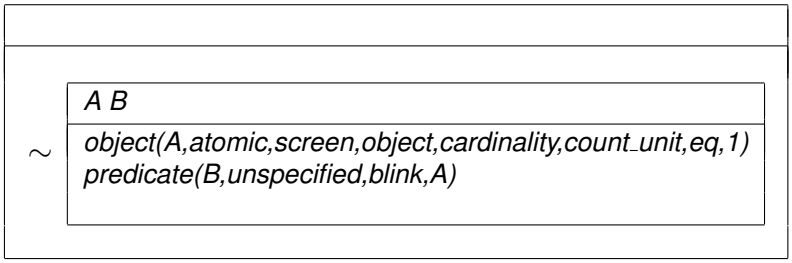
It is false that *a man waits and* ***that*** *a woman sings.*



9.4 Negation as Failure

The only way to express negation as failure (NAF) is to use the predefined phrase "*It is not provable that ...*".

It is not provable that *a screen blinks.*



Concerning scoping, it behaves like the classical sentence negation ("*It is false that ...*") explained in the previous subsection.

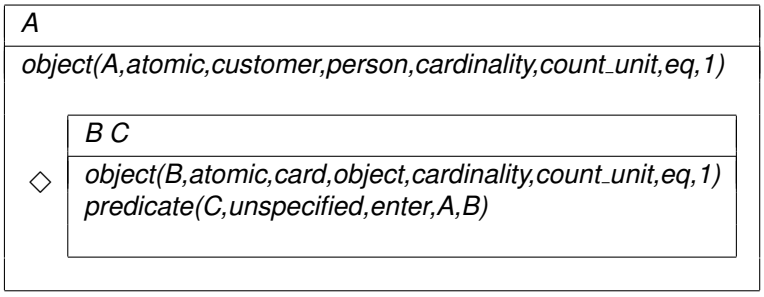
10 Modality

Each of the two forms of modality (possibility and necessity) can be represented in two different ways. First, we can use the modal auxiliary "*can*" or "*must*", respectively. Second, we can use the sentence-initial phrase "*It is possible that...*" or "*It is necessary that ...*", respectively. Negation of these constructs is also allowed (see below for details).

Note that "*a customer can enter a card*" is not equivalent to "*it is possible that a customer enters a card*" (see below).

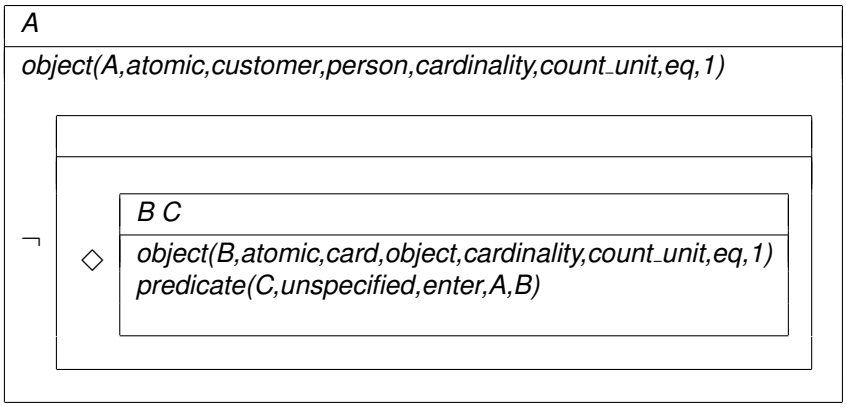
10.1 Possibility

A customer ***can*** *enter a card.*

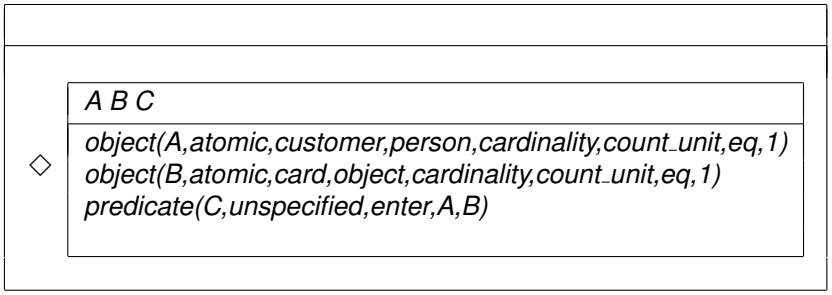


The following three sentences are equivalent.

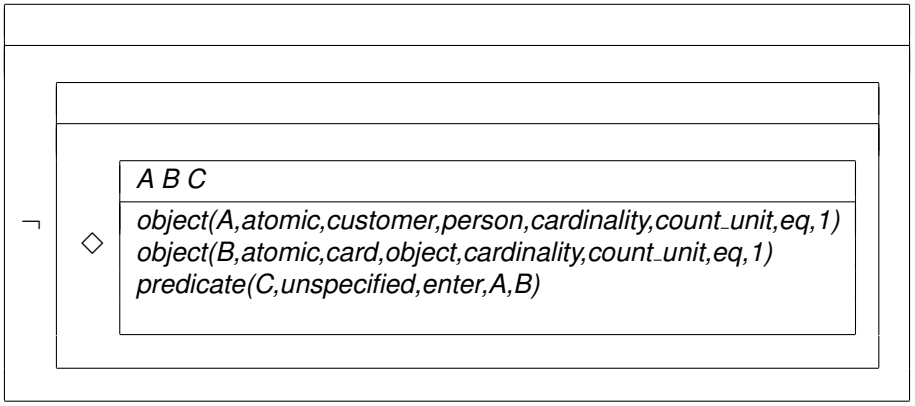
*A customer **can't** enter a card.*
*A customer **cannot** enter a card.*
*A customer **can not** enter a card.*



***It is possible that** a customer enters a card.*



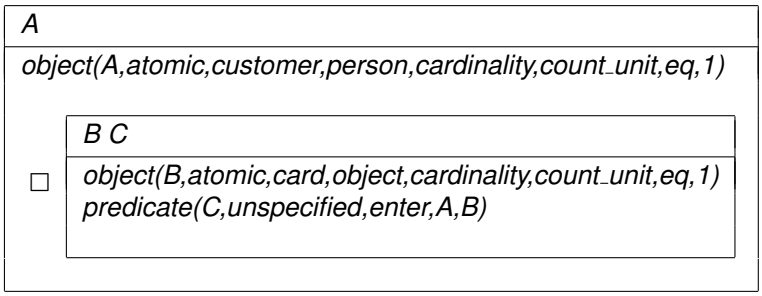
***It is not possible that** a customer enters a card.*



10.2 Necessity

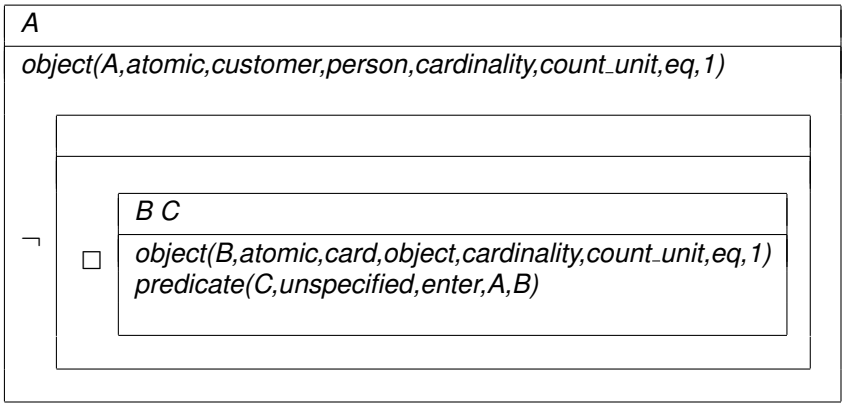
The two synonyms "*must*" and "*has to*" can be used.

*A customer **must** enter a card.*
*A customer **has to** enter a card.*

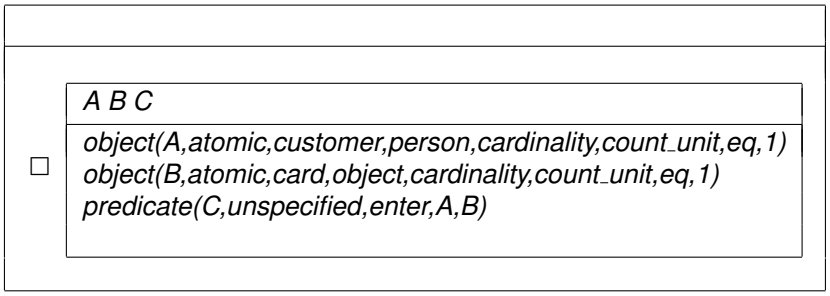


For the negation, only "*does not have to*" is allowed.

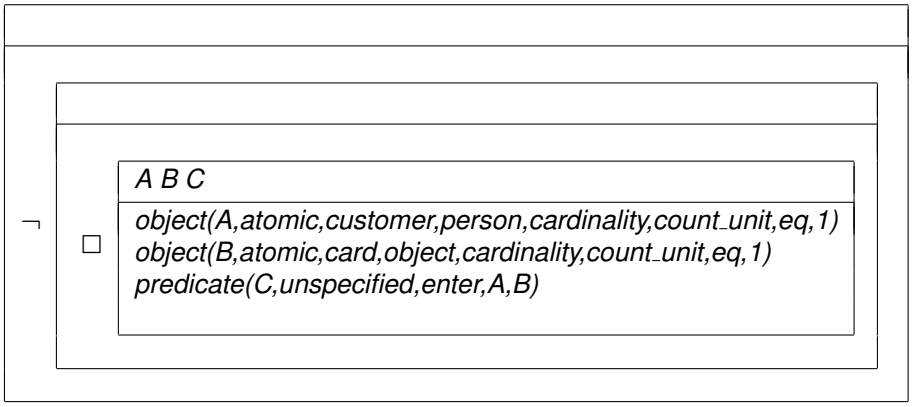
*A customer **does not have to** enter a card.*



***It is necessary that** a customer enters a card.*



It is not necessary that *a customer enters a card.*

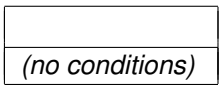


11 Macros

11.1 Macro Definitions

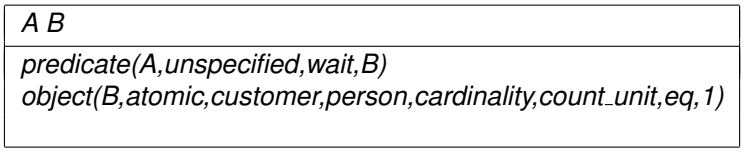
A macro definition has the form "*Proposition [V]: [S]*" where *[V]* is a new variable and *[S]* is a valid ACE Sentence. As long as there is no reference to this macro, no conditions are added to the DRS.

Proposition P: *A customer waits.*



Alternatively, we can use macros of the form "*Fact [V]: [S]*". In this case the contained conditions are added to the top-most level of the DRS, even if there are no references.

Fact P: *A customer waits.*



This is equivalent to "*Proposition P: A customer waits. It is true that P.*".

11.2 References to Macros

There are multiple possibilities to refer to a macro (we assume that there is a preceding macro definition with the variable *P*):

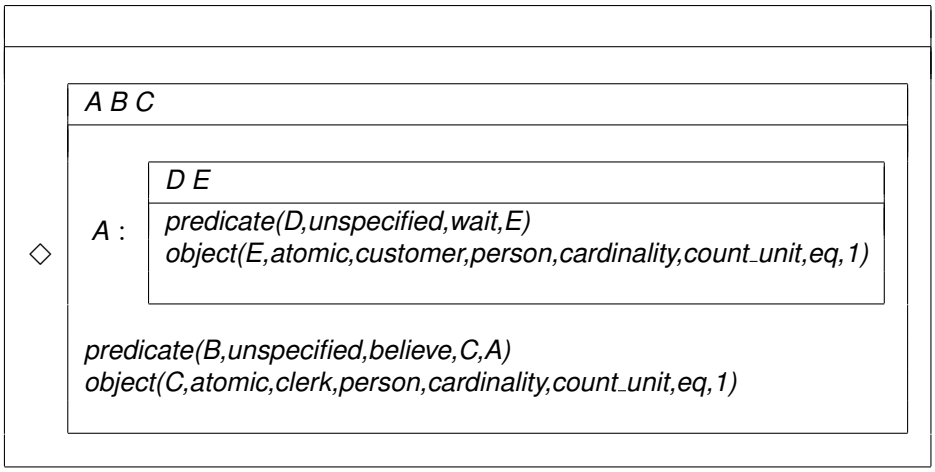
- It is true/false that P.

- It is (not) possible/necessary that P.
- It is not provable that P.
- John believes that P. *(or any other sentence subordination)*

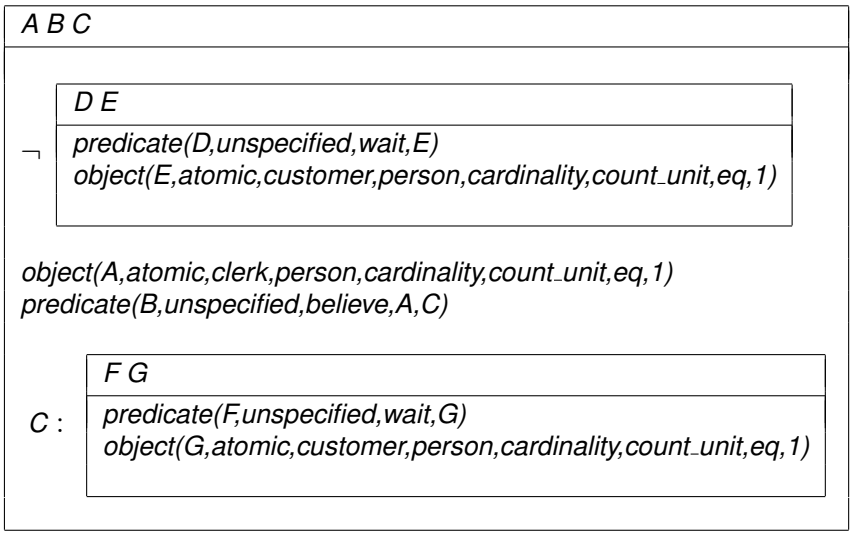
During parsing, each occurrence of a macro is replaced by its definition. Thus, the DRS does not contain any information about macros.

Two examples:

Proposition P: *A customer waits.*
Proposition Q: *A clerk believes that* ***P****.*
It is possible that ***Q****.*



Proposition P: *A customer waits.*
It is false that ***P****.*
A clerk believes that ***P****.*



12 Plural Interpretations

In this section, we present the eight readings of the natural English sentence

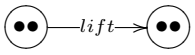
2 girls lift 2 tables.

which can be expressed in ACE 4. Note that reading 4 has two interpretations, the second of which is given as reading 4' at the end of the section. For background information on the disambiguation of plurals consult [6] and [7].

In ACE, a plural noun phrase has a default collective reading. To express a distributive reading, a noun phrase has to be preceded by the marker *each of*. 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.

12.1 Reading 1

$girls$ $tables$

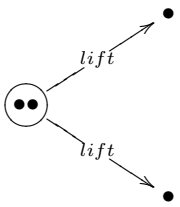


2 girls lift 2 tables.

A B ***C***
object(A,group,girl,person,cardinality,count_unit,eq,2) *object(B,group,table,object,cardinality,count_unit,eq,2)* ***predicate(C,unspecified,lift,A,B)***

12.2 Reading 2

$girls$ $tables$



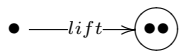
2 girls lift each of 2 tables.

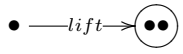
A B
object(A,group,girl,person,cardinality,count_unit,eq,2) *object(B,group,table,object,cardinality,count_unit,eq,2)*

C		***D***
proper_part_of(B,C)	⇒	***predicate(D,unspecified,lift,A,C)***

12.3 Reading 3

$girls$ $tables$





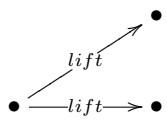
Each of 2 girls lifts 2 tables.

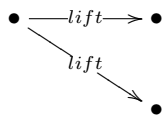
A
object(A,group,girl,person,cardinality,count_unit,eq,2)

B		*C* ***D***
proper_part_of(A,B)	⇒	*object(C,group,table,object,* *cardinality,count_unit,eq,2)* ***predicate(D,unspecified,lift,B,C)***

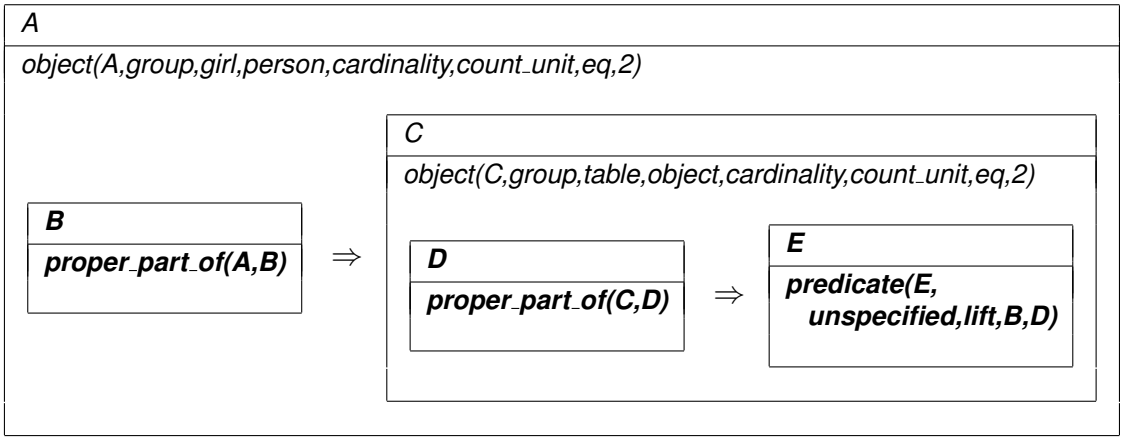
12.4 Reading 4

$girls$ $tables$





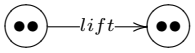
Each of 2 girls lifts each of 2 tables.



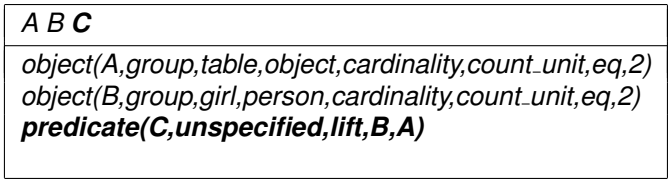
12.5 Reading 5

Reading 5 is identical to reading 1.

$girls$ $tables$

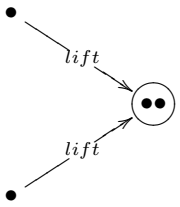


There are 2 tables such that 2 girls lift the tables.



12.6 Reading 6

$girls$ $tables$



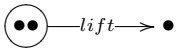
There are 2 tables such that each of 2 girls lifts the tables.

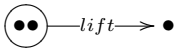
A B
object(A,group,table,object,cardinality,count_unit,eq,2) *object(B,group,girl,person,cardinality,count_unit,eq,2)*

C		***D***
proper_part_of(B,C)	$\Rightarrow$	***predicate(D,unspecified,lift,C,A)***

12.7 Reading 7

$girls$ $tables$





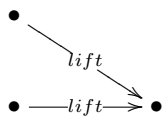
For each of 2 tables 2 girls lift it.

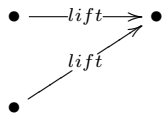
A
object(A,group,table,object,cardinality,count_unit,eq,2)

B		*C* ***D***
proper_part_of(A,B)	$\Rightarrow$	*object(C,group,girl,person,* *cardinality,count_unit,eq,2)* ***predicate(D,unspecified,lift,C,B)***

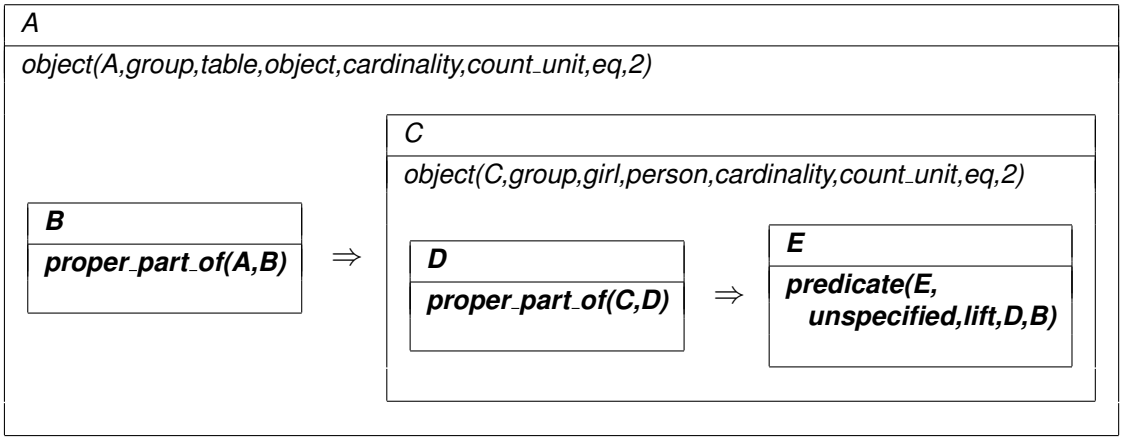
12.8 Reading 8

$girls$ $tables$



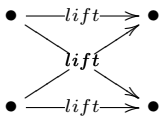


For each of 2 tables each of 2 girls lifts it.

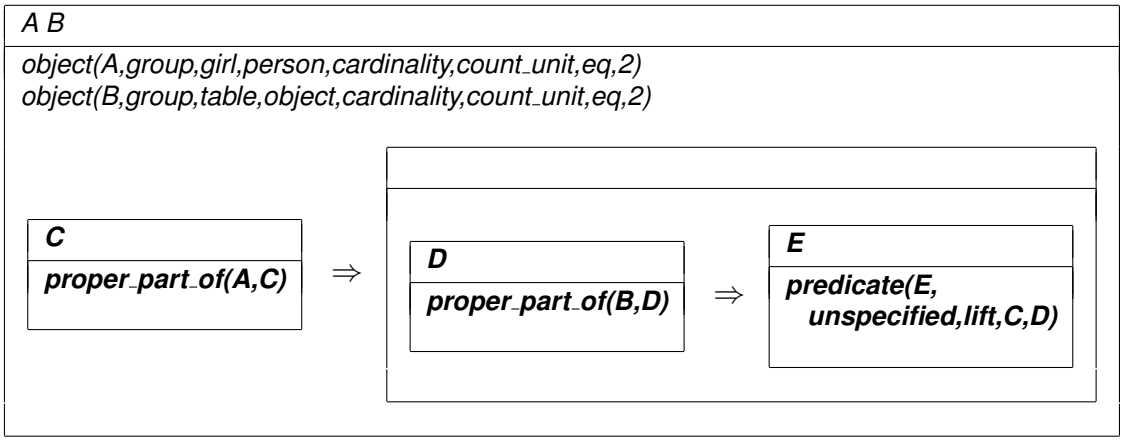


12.9 Reading 4'

$girls$ $tables$



There are 2 girls and there are 2 tables such that each of the girls lifts each of the tables.



13 ACE Questions

13.1 Yes/No-Questions

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
named(A,'John') *object(A,atomic,named_entity,person,cardinality,count_unit,eq,1)* *object(B,atomic,card,object,cardinality,count_unit,eq,1)* *predicate(C,unspecified,enter,A,B)*

Is the card valid?

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

13.2 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) ***query(B,what)*** *predicate(C,unspecified,enter,A,B)*

***Which customer** enters a card?*

A B C
query(A,which) *object(A,atomic,customer,person,cardinality,count_unit,eq,1)* *object(B,atomic,card,object,cardinality,count_unit,eq,1)* *predicate(C,unspecified,enter,A,B)*

13.3 How-Questions

How-questions ask for details of an action. Concretely they ask for the verb modifications introduced by adverbs and prepositional phrases, independently whether these modifications relate to times, locations, durations, manners etc. How-questions are translated as their declarative counterparts but contain an additional condition for the query word 'how'.

How *does John enter a card?*

A B C D E F G
named(A,'John') *object(A,atomic,named_entity,person,cardinality,count_unit,eq,1)* *object(B,atomic,card,object,cardinality,count_unit,eq,1)* *predicate(C,unspecified,enter,A,B)* ***query(E,D,how)*** *modifier(C,unspecified,E,D)*

A Appendix: Predicate Declarations

integer(X,Integer)

X		discourse referent of the object that is denoted by the integer

modifier(X,K,Preposition,Y/Adverb)

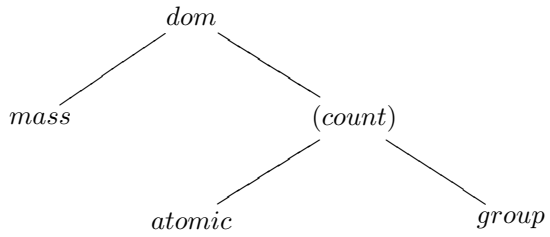
X		discourse referent of the event or state that is modified
K	$\in$	{unspecified, location, origin, direction, time, start, end, duration, instrument, comitative, manner, ...}
Y		discourse referent of an object, i.e. the NP of the modifying PP

named(X,ProperName)

X		discourse referent of the object that is named

object(X,S,Noun,T, K, I, J, N)

X		discourse referent of the object that is denoted by the noun
Noun		the noun that represents the object or 'named_entity' for proper names. For strings and integers that have no noun attached, 'string' or 'integer' is used.
T	$\in$	{person, object, ...}
K	$\in$	{cardinality, weight, size, length, volume, dimension, ...}
I	$\in$	{count_unit, unit, kg, cm, liter, ...}
J	$\in$	{eq, leq, geq, greater, less}
N		a number, or unspecified
S	$\in$	{atomic, group, mass, dom}



predicate(E,D,Verb,X)

E		discourse referent of the event or state that is denoted by the verb
D	$\in$	{unspecified, event, state, ...}
X		discourse referent of the subject

predicate(E,D,Verb,X,Y)

E		discourse referent of the event or state that is denoted by the verb
D	$\in$	{unspecified, event, state, ...}
X		discourse referent of the subject
Y		discourse referent of the direct object

predicate(E,D,Verb,X,Y,Z)

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

proper_part_of(X,Y)

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

property(X,IntransitiveAdjective)

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

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

query(X,Q)

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

query(P,Y,Q)

P		preposition
Y		discourse referent of an object, i.e. the NP of the modifying PP or an adverb
Q	$\in$	{how, ...}

quoted_string(X,String)

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

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

$X \xrightarrow{Relationof} Y$

string(X,String)

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

References

[1] Attempto project. Attempto website, 2006. 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. http://csli-publications.stanford.edu/site/1575861607.html.

[3] Johan Bos. Computational Semantics in Discourse: Underspecification, Resolution, and Inference. *Journal of Logic, Language and Information*, 13(2):139–157, 2004.

[4] Stefan Höfler. The Syntax of Attempto Controlled English: An Abstract Grammar for ACE 4.0. Technical Report ifi-2004.03, Department of Informatics, University of Zurich, Zurich, Switzerland, 2004.

[5] Hans Kamp and Uwe Reyle. *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer Academic Publishers, Dordrecht/Boston/London, 1993.

[6] Uta Schwertel. Controlling Plural Ambiguities in Attempto Controlled English. In *Proceedings of the 3rd International Workshop on Controlled Language Applications*, Seattle, Washingtion, 2000.

[7] Uta Schwertel. *Plural Semantics for Natural Language Understanding — A Computational Proof-Theoretic Approach*. PhD thesis, University of Zurich, 2004.