Question Answering in a Natural Language Understanding System Based on Object–Oriented Semantics

Yuriy Ostapov

Institute of Cybernetics of NAS of Ukraine, 40 Acad. Glushkova avenue, Kiev, Ukraine.
E-mail: yugo.ost@gmail.com

Abstract

Algorithms of question answering in a computer system oriented on input and logical processing of text information are presented. A knowledge domain under consideration is social behavior of a person. A database of the system includes an internal representation of natural language sentences and supplemental information. The answer *Yes* or *No* is formed for a general question. A special question containing an interrogative word or group of interrogative words permits to find a subject, object, place, time, cause, purpose and way of action or event. Answer generation is based on identification algorithms of persons, organizations, machines, things, places, and times. Proposed algorithms of question answering can be realized in information systems closely connected with text processing (criminology, operation of business, medicine, document systems).

1. Introduction

A computer understanding of natural language consists in capability of a program system to translate sentences into an *internal representation* so that this system generates adequate (i.e., valid) answers to questions to be asked by an user. Adequateness is that an answer (from the viewpoint of a researcher) solves enough correctly the problem that is contained in a question.

As the internal representation of natural language sentence must adequately map semantics of this statement, the most natural approach is in the simulation of facts contained in the sentence using a description of *real*

objects as well as actions and events connected with these objects. This approach was realized in the experimental system LEIBNIZ [5].

To form an answer it is necessary, in the first place, to execute the syntax and semantic analysis of a question, in the second place, to find an adequate solution for the problem under review.

It should be distinguished:

- *simple questions* — when all demandable information is contained in a database, and only it is necessary to find an answer.

- *logical questions* — when an inference plays a leading role for forming answers[1].

In this paper we are limited only simple questions. The answer is formed using the database of LEIBNIZ built from facts. This database consists of predicates describing behavior of persons, organizations, machines, things [5].

Analogous problems in reference to a knowledge domain under consideration (social behavior of a person) are studied in [6, 7, 8, 9, 10] . The essential distinction of our investigation is in the use of **object–oriented semantics** to represent facts from sentences. It should be pointed out that such approach is not attached to a specific design tool. Furthermore, a real information system can be built on the base of this technology by means of modern design tools: *Microsoft Visual Studio, JBuilder, Delphi,* and others. Information in this system will be saved and processed with the help of the living database management systems: *Oracle, Informix, MS SQL Server, DB2.*

The use of *Visual Prolog* [11] in the system LEIBNIZ is explained by research character of this system and permits with comparative ease to execute the syntax and semantic analysis as well as to organize an inference.

2. Syntax and semantic analysis of questions

The syntax analysis of a question is based on a description of grammatical structure (see [2]) using *Backus–Naur form*[2]. The question can be general or special:

---

[1]For example, it is necessary to detect on the base of indirect evidence who is a killer.

[2]We apply the names of grammatical constructions that were determined in [5]. The square brackets are used to point to possibility of absence for an element.

2

⟨question⟩ ::= ⟨general question⟩ | ⟨special question⟩

The general question demands the answer *Yes* or *No*:

⟨general question⟩ ::= ⟨verb ⟩ ⟨group of subject⟩[not] ⟨the rest of predicate⟩ ⟨construction controlled with predicate⟩

The verb at the beginning of the question is *to be, to have, to do* in the present or past indefinite, the modal verb or *will (would, should).*
The special question begins with an interrogative word or group of interrogative words and can be addressed to an object or adverbial modifier (or their properties):

⟨special question⟩ ::= [⟨preposition⟩] ⟨interrogative word or group of interrogative words⟩ [⟨group of noun ⟩] ⟨verb⟩ ⟨group of subject⟩[not] ⟨the rest of predicate⟩ ⟨construction controlled with predicate⟩

The interrogative word is *when, where, what, whom,* and other. The group of interrogative words is, for example, *how much, how long.* The verb after an interrogative word(s) is *to be, to have, to do* in the present or past indefinite, the modal verb or *will (would, should).*

The special question can be addressed to a subject:

⟨special question⟩ ::= ⟨*who* or *what*⟩ ⟨predicate ⟩ ⟨group of objects ⟩ ⟨group of adverbial modifiers⟩

or to a property of subject:

⟨special question⟩ ::= ⟨*which, what, whose, how many, how much*⟩ ⟨group of noun⟩ ⟨predicate⟩ ⟨group of objects⟩ ⟨group of adverbial modifiers⟩

The special question with a compound name predicate as well can be addressed to a subject :

⟨special question⟩ ::= ⟨*who* or *what*⟩ ⟨predicate⟩ ⟨basic noun phrase⟩ ⟨group of adverbial modifiers⟩

The predicate after the interrogative word *who* or *what* is *to be* in the indefinite tense or the modal verb + *be*.

The semantic analysis is executed after the parser and consists in building predicates: *person, organization, machine, thing, action, event, place, time,* and others [5] .

3. Identification of objects

Algorithms of identification are used when it is necessary to form an inquiry answer. Identification is the comparison of two objects (predicates) pointed to the predicate of action (or event) from a question and the similar predicate of a database to detect their sameness[3].

Arguments of predicates will be referred to as *fields*. The first predicate in algorithms of identification presented below corresponds to the question, and the second predicate to the database.

3.1. Identification of persons

A person is described with the predicate *person*:

1. Code of *person*.
2. Designation of person.
3. Sex.
4. First name.
5. Last name.
6. Additional data (other names, honorary title and degree).
7. Place of birth (code of *place*).
8. Nationality.
9. Mother tongue.
10. Other tongues (parallel with mother).
11. Place of residence (code of *place*).
12. Description of face.
13. Description of nose.
14. Description of constitution.
15. Description of eyes.

---

[3]It should be pointed out that proposed algorithms have a certain degree of credibility, which depends on concrete conditions of application.

16. Description of hair.
17. Date of birth (code of *time*).
18. Stature.
19. Temperament.
20. Psychological type.
21. Profession.

The identification algorithm of two predicates *person*:

1. If the first predicate contains only the field *first* or *last name*, then we go to step 2, or else to step 3.

2. If the field *first* (or *last) name* from the first predicate coincides accordingly with the field *first* (or *last) name* from the second predicate, then the objects are identical, otherwise the objects are not identical. The algorithm is completed.

3. If the first predicate contains only the fields *first* and *last name*, then we go to step 4, or else to step 5.

4. If the fields *first* and *last name* from the first predicate coincide accordingly with the fields *first* and *last name* from the second predicate, then the objects are identical, otherwise the objects are not identical. The algorithm is completed.

5. If the first predicate contains the field *first* (or *last) name* and a property, then we go to step 6, otherwise the algorithm is completed.

6. If the field *first* (or *last) name* and this property from the first predicate coincide accordingly with the field *first* (or *last) name* and the same property from the second predicate, then the objects are identical, otherwise the objects are not identical. The algorithm is completed.

Furthermore, in specific cases one can apply also the comparison using only the field *designation of person*.

Analogous algorithms can be built for the predicates *organization, thing, and machine*.

3.2. Identification of places:

A place of action or event is described with the predicate *place*.

1. Code of *place*.
2. Country.

3. Type of region (*state, province*).

4. Geographical name of region.

5. Territorial entity(*town, village*).

6. Geographical name of territorial entity.

7. Location (*street, square, park, line*).

8. Name of location.

9. Construction (*house, theatre, station, industrial object*).

10. Name of construction.

11. Additional information for construction (*stairs, roof, garret, floor*).

12. Designation of final location (*apartment, hall, office, restaurant, cafe*).

13. Designation of room (*bathroom, bedroom, living room, kitchen*)

The identification algorithm of two predicates *place*:

1. If the first predicate contains only the field *town*, then we go to step 2, or else to step 3.

2. If the field *town* from the first predicate coincides with the field *town* from the second predicate, then the objects are identical, otherwise the objects are not identical. The algorithm is completed.

3. If the first predicate contains only the fields *town* and *street*, then we go to step 4, or else to step 5.

4. If the fields *town* and *street* from the first predicate coincide accordingly with the fields *town* and *street* from the second predicate, then the objects are identical, otherwise the objects are not identical. The algorithm is completed.

5. If the first predicate contains the fields *town, street,* and a number of house (or *name of construction*), then we go to step 6, or else to step 7.

6. If the fields *town, street,* and the number of house (or *name of construction*) from the first predicate coincide accordingly with the fields *town, street,* and the number of house (or *name of construction*) from the second predicate, then the objects are identical, otherwise the objects are not identical. The algorithm is completed.

7. If the first predicate contains the fields *town, street,* a number of house, and number of apartment, then we go to step 8, or else to step 9.

8. If the fields *town, street,* the number of house, and number of apartment from the first predicate coincide accordingly with the fields *town, street,* the number of house, and number of apartment from the second predicate,

then the objects are identical, otherwise the objects are not identical. The algorithm is completed.

9. If the first predicate contains the fields *town, street,* a number of house, and location in house (*stairs, roof, garret, floor*), then we go to step 10, otherwise the algorithm is completed.

10. If the fields *town, street,* the number of house, and this location in house from the first predicate coincide accordingly with the fields *town, street,* the number of house, and the same location in house from the second predicate, then the objects are identical, otherwise the objects are not identical. The algorithm is completed.

3.3. Identification of times

A time of action or event is described with the predicate *time*:

1. Code of *time.*
2. Year.
3. Season (*spring, summer, autumn, winter*).
4. Month.
5. Day in month.
6. Day of the week.
7. Holyday (*New Year, Christmas, Easter, and others*).
8. Part of day (*morning, afternoon, evening, night*).
9. Hours.

The identification algorithm of two predicates *time*:

1. If the first predicate contains only the field *year*, then we go to step 2, or else to step 3.

2. If the field *year* from the first predicate coincides with the field *year* from the second predicate, then the objects are identical, otherwise the objects are not identical. The algorithm is completed.

3. If the first predicate contains only the fields *year* and *season* (or *month*), then we go to step 4, or else to step 5.

4. If the fields *year* and *season* (or *month*) from the first predicate coincide accordingly with the fields *year* and *season* (or *month*) from the second predicate, then the objects are identical, otherwise the objects are not identical. The algorithm is completed.

5. If the first predicate contains the fields *year, month,* and *day in month,* then we go to step 6, or else to step 7.

6. If the fields *year, month,* and *day in month* from the first predicate coincide accordingly with the fields *year, month,* and *day in month* from the second predicate, then the objects are identical, otherwise the objects are not identical. The algorithm is completed.

7. If the first predicate contains the fields *year, month, day in month,* and *hours* (or *part of day*), then we go to step 8, otherwise the algorithm is completed.

8. If the fields *year, month, day in month,* and *hours* (or *part of day*) from the first predicate coincide accordingly with the fields *year, month, day in month,* and *hours* (or *part of day*) from the second predicate, then the objects are identical, otherwise the objects are not identical. The algorithm is completed.

4. Answer generation for general questions

Answer generation depends essentially on the character of action contained in a question. Therefore, at first we consider in more detail the semantic classification of verbs:

JOB — long purposeful occupation (*job, sport, studies*);
PROPEL — applying a force to an object;
MOVE — moving a body part;
INGEST — ingesting something inside;
EXPEL — expelling something from a subject;
GRASP — grasping an object;
GO — displacement of a subject;
TRANSFER — change of general relation for a subject (*to buy, to sell, to come into fortune*);
FEEL — perception of a subject (*to see, to hear, to touch*);
MESSAGE — transmission of information between a subject and object;
BE — identity of a subject and object, existence of a subject or connection between a subject and certain class of objects;
CHANGE — transition of a subject to another state;
CREATE — thinking (*decision-making, problem-solving, prediction*);
DO — an action.

This classification is founded on the fundamental investigations in [7] (with some additions and modifications).

The predicate *action* is formed for the codes PROPEL, MOVE, INGEST, EXPEL, GRASP, GO, TRANSFER, BE, DO, the predicate *job* is built for the code JOB, the predicate *message* is generated for the code MESSAGE, the predicate *intelligence* corresponds to the codes FEEL, CREATE. To form the predicate *event* the code CHANGE is necessary.

Consider an algorithm of answer generation for a general question when an action in this question is described with the predicate *action*[4] :

1. If the predicate *action* from the question describes a physical effect or change of general relation (the codes PROPEL, MOVE, INGEST, EXPEL, GRASP, GO, TRANSFER), then we go to step 2, or else to step 3.

2. All the predicates *action* that have the same values for fields *semantic type of action, negation of action, tense, type of tense* as the predicate *action* of the question are selected from a database. Each such predicate *action* is compared with the predicate *action* from the question to establish identity of verbs (in terms of synonyms), subjects, direct, indirect (or prepositional) objects, locations, times, purposes, and ways of actions. If a given predicate *action* from the database satisfies all these conditions, then the message *Yes* is displayed. Otherwise an inference is executed[5] .

3. If the predicate *action* from the question has the code BE, then all the predicates *action* that have the code BE are selected from the database. Each such predicate *action* is compared with the predicate *action* from the question to establish identity of subjects, locations, and times of actions. If a given predicate *action* from the database satisfies all these conditions, then the message *Yes* is displayed. The algorithm is completed. Otherwise we go to step 4.

4. All the predicates *job, message, intelligence, event* are selected from the database. Each such predicate is compared with the predicate *action* from the question to establish identity of subjects, locations, and times of actions or events. If a given predicate from the database satisfies all the conditions, then the message *Yes* is displayed. The algorithm is completed.

---

[4]The structure of *action* is given in [5].

[5]Algorithms of inference are not expounded in this paper as reasonably complex character of this problem demands a special consideration.

Otherwise the message *No* is displayed. The algorithm is completed.

Consider else an algorithm of answer generation for a general question when an event in this question is described with the predicate *event*[6]. In such a situation all the predicates *event* that have the same *scale* as the predicate *event* of the question are selected from the database. Each such predicate *event* is compared with the predicate *event* from the question to establish identity of verbs (in terms of synonyms), subjects, locations, and times of events. If a given predicate *event* from the database satisfies all these conditions, then the message *Yes* is displayed. Otherwise the message *No* is displayed.

When an action in a general question is described with the predicate *job, message* or *intelligence*, an answer for such question is formed analogously.

As regards reliability of answers for general questions, it should be mentioned that one depends on credibility of identification algorithms.

5. Answer generation for special questions

Answer generation for special questions is determined with two factors:

- the *type of predicate* for an action or event in a question;

- an *interrogative word* (or a *group of interrogative words*), which points to a solvable problem.

If a problem can not be solved with a selection from a database, then an inference may help in the solution of this task. Furthermore, a knowledge domain under consideration (social behavior of a person) must be extra restricted to provide the solution of real tasks in business, criminology, medicine, and the like.

Consider an algorithm of answer generation for a special question when an action in this question is described with the predicate *action* [7]:

1. If the question is addressed to a *subject* of action (the interrogative

---

[6]The structure of the predicate *event* is given in [5].

[7]Considering complexity of the algorithm for forming such answer, we give the reductive variant having excluded some checks and alternatives.

word *who* or *what*), and the action has the code PROPEL, MOVE, INGEST, EXPEL, GRASP, GO, TRANSFER, then we go to step 2, or else to step 3.

2. All the predicates *action* that have the same values for fields *semantic type of action, negation of action, tense, type of tense* as the predicate *action* of the question are selected from the database. Each such predicate *action* is compared with the predicate *action* from the question to establish identity of verbs (in terms of synonyms), direct, indirect (or prepositional) objects, locations, times, purposes, and ways of actions. If a given predicate *action* from the database satisfies all these conditions, then the *subject* from this predicate is displayed. When all the selected predicates have been checked, and there is a true answer, then the algorithm is completed. Otherwise the inference is executed.

3. If the question is addressed to a *property of the subject* [8], and the action has the code PROPEL, MOVE, INGEST, EXPEL, GRASP, GO, TRANS-FER, then we go to step 4, or else to step 5.

4. All the predicates *action* that have the same values for fields *semantic type of action, negation of action, tense, type of tense* as the predicate *action* of the question are selected from the database. Each such predicate *action* is compared with the predicate *action* from the question to establish identity of verbs (in terms of synonyms), subjects, direct, indirect (or prepositional) objects, locations, times, purposes, and ways of actions. If a given predicate *action* from the database satisfies all these conditions, then the *property of the subject* from this predicate is displayed. When all the selected predicates have been checked, and there is a true answer, then the algorithm is completed. Otherwise the answer *The question can not be executed* is displayed, and algorithm is completed.

5. If the question is addressed to a *direct object* of action (the interrogative word *what* or *whom*), and the action has the code PROPEL, MOVE, INGEST, EXPEL, GRASP, GO, TRANSFER, then we go to step 6, or else to step 7.

6. All the predicates *action* that have the same values for fields *semantic type of action, negation of action, tense, type of tense* as the predicate *action* of the question are selected from the database. Each such predicate *action* is compared with the predicate *action* from the question to establish identity of verbs (in terms of synonyms), subjects, indirect (or prepositional)

---

[8]It is discovered with the parser of the question.

objects, locations, times, purposes, and ways of actions. If a given predicate *action* from the database satisfies all these conditions, then the *direct object* from this predicate is displayed. When all the selected predicates have been checked, and there is a true answer, then the algorithm is completed. Otherwise the inference is executed.

7. If the question is addressed to an *indirect object* of action (the interrogative word *whom*), and the action has the code PROPEL, MOVE, INGEST, EXPEL, GRASP, GO, TRANSFER, then we go to step 8, or else to step 9.

8. All the predicates *action* that have the same values for fields *semantic type of action, negation of action, tense, type of tense* as the predicate *action* of the question are selected from the database. Each such predicate *action* is compared with the predicate *action* from the question to establish identity of verbs (in terms of synonyms), subjects, direct objects, locations, times, purposes, and ways of actions. If a given predicate *action* from the database satisfies all these conditions, then the *indirect object* is displayed. When all the selected predicates have been checked, and there is a true answer, then the algorithm is completed. Otherwise the inference is executed

9. If the question is addressed to a *time* of action (the interrogative word *when*), and the action has the code PROPEL, MOVE, INGEST, EXPEL, GRASP, GO,TRANSFER, then we go to step 10, or else to step 11.

10. All the predicates *action* that have the same values for fields *semantic type of action, negation of action, tense, type of tense* as the predicate *action* of the question are selected from the database. Each such predicate *action* is compared with the predicate *action* from the question to establish identity of verbs (in terms of synonyms), subjects, direct, indirect (or prepositional) objects, locations, purposes, and ways of actions. If a given predicate *action* from the database satisfies all these conditions, then the *time* of action from this predicate is displayed. When all the selected predicates have been checked, and there is a true answer, then the algorithm is completed. Otherwise the inference is executed.

11. If the question is addressed to a *place* of action (the interrogative word *where*), and the action has the code PROPEL, MOVE, INGEST, EXPEL, GRASP, GO, TRANSFER, then we go to step 12, or else to step 13.

12. All the predicates *action* that have the same values for fields *semantic type of action, negation of action, tense, type of tense* as the predicate *action* of the question are selected from the database. Each such predicate *action* is compared with the predicate *action* from the question to establish identity of verbs (in terms of synonyms), subjects, direct, indirect (or prepo-

sitional) objects, times, purposes, and ways of actions. If a given predicate *action* from the database satisfies all these conditions, then the *location* of action from this predicate is displayed. When all the selected predicates have been checked, and there is a true answer, then the algorithm is completed. Otherwise the inference is executed.

13. If the question is addressed to a *way* of action (the interrogative word(s) *how, in that way*), and the action has the code PROPEL, MOVE, INGEST, EXPEL, GRASP, GO, TRANSFER, then we go to step 14, or else to step 15.

14. All the predicates *action* that have the same values for fields *semantic type of action, negation of action, tense, type of tense* as the predicate *action* of the question are selected from the database. Each such predicate *action* is compared with the predicate *action* from the question to establish identity of verbs (in terms of synonyms), subjects, direct, indirect (or prepositional) objects, locations, times, and purposes of actions. If a given predicate *action* from the database satisfies all these conditions, then the *way* of action from this predicate is displayed. When all the selected predicates have been checked, and there is a true answer, then the algorithm is completed. Otherwise the inference is executed.

15. If the question is addressed to a *purpose* of action (the group of interrogative words *what for, for what purpose*), and the action has the code PROPEL, MOVE, INGEST, EXPEL, GRASP, GO, TRANSFER, then we go to step 16, or else the algorithm is completed .

16. All the predicates *action* that have the same values for fields *semantic type of action, negation of action, tense, type of tense* as the predicate *action* of the question are selected from the database. Each such predicate *actions* is compared with the predicate *action* from the question to establish identity of verbs (in terms of synonyms), subjects, direct, indirect (or prepositional) objects, locations, times, and ways of actions. If a given predicate *action* from the database satisfies all these conditions, then the *purpose* of action from this predicate is displayed. When all the selected predicates have been checked, and there is a true answer, then the algorithm is completed. Otherwise the inference is executed.

When an action or event in a special question is described with the predicate *job, message, intelligence* or *event*, an answer for such question is formed analogously. Similar to general questions, reliability of answers for special questions depends on credibility of identification algorithms.

6. Example (sea story)

Consider a story:

Mister Brown was a mate on a ship fifteen years ago. The captain came up to the mate one morning. The captain said that he heard strange voice at night. This voice said in his ear to sail north-west. The captain told the mate to sail north-west. One of the men saw something black in the sea the next day. The captain looked through glasses. He said that there is small boat there with man. The captain ordered the men to save the man. Soon the men reached the small boat. The man was fast asleep. He went on sleeping while the men took him in boat. When the man was aboard the ship, the man suddenly opened his eyes. The man cried out loudly where he is. Captain said that ship's company saved him. The man asked if the captain ordered to take him from small boat. Captain answered that he ordered to take him. Then the man messaged that he executed a record voyage from New York to Liverpool on small boat.

Questions and answers formed by the system LEIBNIZ are given in the table:

Was Brown a mate?	Yes
Who was a mate?	Brown
What did the voice say?	to sail north-west
Who ordered the men?	captain
What did the captain order?	to save man
What did the men reach?	boat
What did the captain say?	ship's company saved man
Who cried out loudly?	the man
What did the man message?	the man did voyage

7. Conclusion

As mentioned in our paper [5], we use an ontological approach to the problem of natural language understanding. The ontological approach proceeds from the assumption that a natural language maps the structure of the outside world. Each sentence describes facts (or fact) interpreting actions and events for *real objects*. Therefore, it is necessary to model directly these objects, actions, and events.

14

As a knowledge domain, we consider social behavior of a person. These problems are studied in social psychology [4]. However, when we are dealing with computer technologies, to solve real problems the knowledge domain must be pointed in more detail. For example, it can view the domain of criminology (the revelation of criminal offences) [3] or business management (the use of informal data about organizations, persons, goods) [1].

The advantage of the approach considered in our paper (as compared with living information systems in business, criminology, medicine) is that extensive text information can be used *for logical processing*[9].

By this means the technology proposed the author discovers the perspective for successful solution of problems that so far were outside computer technologies. First of all this is concern of domains of human activity where at the moment formalization of data is connected with essential system losses (criminology, business management, medicine, document systems).

References

[1] R.B.Chase, R.F.Jacobs, W.J.Aquilano, Operations Management for Competitive Advantage, McGraw-Hill, 2004.

[2] R.Quirk, S.Greenbaum, G.Leech, J.Svartvik, A University Grammar of English, Longman, London, 1973.

[3] P.Manning, The technology of Policing: Crime Mapping, Information Technology, and the Rationality of Crime Control (New Perspective in Crime, Deviance, and Law), NYU Press, 2008.

[4] D.G.Myers, Social psychology, McGraw-Hill, 2002.

[5] Yu.Ostapov, Object-oriented semantics of English in natural language understanding system, arXiv:1109.5798v1[cs.CL] 27 Sep 2011.

[6] M.J.Pazzani,C.Engelman, Knowledge based questions answering, The MITRE Corp., ACL Anthology.

---

[9]Available text information in living systems, as a rule, can not be processed in full measure to solve topical problems. For example, it is difficult to calculate a criminal on the base of indirect evidence as only a little part of information about criminal offences is formalized in a database.

[7] R.C.Schank, N.M.Goldman, C.J.Rieger, C.K.Riesbeck, Conceptual Information Processing, North–Holland, Amsterdam, 1975.

[8] R.W.Smith, D.R.Hipp, A.W.Biermann, An Architecture for Voice Dialog System Based on Prolog-Style Theorem Proving, Computational Linguistics, vol. 21, Number 3, 1995.

[9] B.H.Thompson, F.B.Thompson, Introducing ASK, A Simple Knowledge System, California Institute of Technology, ACL Anthology.

[10] P.Velardi, M.T.Pazienza, M.Fasolo, How to Encode Semantic Knowledge: A Method for Meaning Representation and Computer-Aided Acquisition, Computational Linguistics, vol. 17, Number 2, 1991.

[11] Visual Prolog Version 5.0. Language Tutorial, Prolog Development Center, Copenhagen, 1997.