Educational Robotics

BA thesis in Computer Systems Engineering

The Maersk Mc-Kinney Moller Institute
for Production Technology

Authors:
Steffen Jensen
Mikkel Meister Pedersen

Supervisor:
Henrik Hautop Lund

Abstract

The objective of this thesis is to explore the possibility of using a modular robotic entity, the I-BLOCK, for educational purposes. The I-BLOCKs allow for programming by building and this project is centered around a linguistics scenario supporting this concept. The linguistics scenario developed and implemented during this project makes a user capable of constructing basic sentence structures by using the I-BLOCKs. Whether this approach is suitable for educational purposes is tested with elementary school pupils of 9-10 years of age. Furthermore, results from testing performed in Siena, Italy with a child having language problems is presented and discussed. It is concluded that the use of I-BLOCKs for solving language tasks makes it easy for elementary school pupils to experiment with sentence structures. However, the structure tested in Siena, Italy is found to be inappropriate for children with language problems.

Contents

List of Figures

Chapter 1

Preface

THIS thesis has been written as a part of the Bachelor degree exam in Computer Systems Engineering[1] undertaken by the authors. The project was initiated on March 15th 2003 and concluded on October 1st 2003 and was supervised by professor Henrik Hautop Lund[2]. Furthermore, co-supervision was done by Luigi Pagliarini.

1.1 Home page

This report, as well as movie clips and source code, can be found on the address www.adaptronics.dk. Navigate to the page with bachelor theses and locate the category Educational Robotics.

1.2 Reading directions

When reading this thesis, the reader will encounter words and symbols typeset different than the rest of the text. This is done to give the reader an easy way of distinguishing e.g. source code variables from ordinary text. The layout is as follows:

- Source code is typeset: `int example = 0;`
- Source code files are typeset: example.c
- Litterature references are indicated by a number within brackets: "As stated by Jakob Nielsen [3] ..."

In the following, a short description of each chapter is given.

Chapter 1 - Preface

The chapter you are reading now.

Chapter 2 - Introduction

An introduction to the concept of Educational Robotics and I-BLOCKs.

Chapter 3 - Analysis

A short overview of basic differences in the Danish language compared to the English.

Chapter 4 - Software Implementation

The authors give a fulfilling description of two implementations of the linguistics scenario.

[1] Visit www.mip.sdu.dk

[2] Visit www.mip.sdu.dk/∼hhl

Chapter 5 - School Sessions

The authors reflect on the practical experience obtained by letting school children work/play with the I-BLOCKs.

Chapter 6 - Results from Siena, Italy

This chapter discusses the results gained from the performed I-BLOCK testing in Siena, Italy. The testing has been conducted by Valentina Palma of the University of Siena.

Chapter 7 - Discussion and Conclusion

The authors discuss the obtained results and draw a number of conclusions from these.

Appendix A - Questionnaires

The questionnaires used for evaluating the school sessions are located here.

Appendix B - The Living Tree

The article about The Living Tree [7].

Appendix C - CD-ROM Contents

A table of contents for the CD-ROM accompanying this thesis.

Appendix D - Author Indication

This appendix indicates, what has been written by Steffen Jensen and Mikkel Meister Pedersen in this thesis. It should be emphasised that the project and thesis has been concluded and written in cooperation. Only the two prototypes discussed in chapter 4 have been made and documented individually. A table of author indications can be found in appendix D.

1.3 Thanks

We want to thank our supervisor Henrik Hautop Lund and our co-supervisor Luigi Pagliarini. A very special thanks goes to Jakob Nielsen for helping the authors on various I-BLOCK related issues. Furthermore we would like to thank Valentina Palma and Alessia Rullo of University of Siena, Italy, for collaborating with us on the development of the linguistics scenario. Last, but by no means least, we would like to thank the children in 4.a at Aaloekkeskolen in Odense, Denmark, as well as their teacher Ole Soerensen, for having patience, when the I-BLOCKs were behaving strange.

Chapter 2

Introduction

"The goal of education is the advancement of knowledge and the dissemination of truth."

- John Fitzgerald Kennedy

LATELY, a number of rapid robot prototype systems have allowed robotics to be given to the hands of children. This has often been with an educational goal in mind. However, there exist very little empirical data that shows the advantage and disadvantages of using robotics with children. In this project, we will set up quantitative and qualitative measurements and perform empirical tests with children to collect data regarding the suitability of the approach of an open robotic construction system. In general, the project aims to design, develop and experimentally assess a concept of flexible and physical components to manipulate conceptual structures. This will be achieved through a system that exploits the affordances of physical objects in creating physical interfaces for collaborative and distributed learning environments. The project will explore the possibility of allowing young children to develop and interact with processing systems by manipulating physical objects in different environments. As a starting point, the project will use a set of basic intelligent blocks (I-BLOCKs) with individual processing and communication power, and we may look at how to increase transparency of individual building block functionality. Each construction/configuration of I-BLOCKs may represent a conceptual structure (a narrative, a piece of music, etc.) that will be interpreted and represented on a shared communication medium. Connectivity and behaviour of such structures will be defined by the physical connectivity between the blocks. Scenarios may be developed in collaboration with Prof.ssa Marti's group at University of Siena, and the developed, implemented system will be tested with school children in an elementary school, the results will be thoroughly reported, and may be used for further development.

2.1 I-BLOCKs - an introduction

The original concept of "programming by building" using intelligent artefacts is developed by the supervisor on this project, Henrik Hautop Lund. In his article "Intelligent artefacts" [1], Lund states the importance of hands-on experimentation as the basis for cognitive development. This approach is based on the work in educational psychology

by J. Piaget and L.S. Vygotsky. By using a physical entity, the Intelligent Artefact, the user becomes capable of creating physical structures with corresponding functionality.

The hardware implementation of the intelligent artefacts has been made in LEGO DUPLO bricks. It should be strongly emphasised that the hardware used in this project has not been made by the authors. As stated by Jakob Nielsen [3] in his thesis, the intelligent bricks, or I-BLOCKs, have been designed at the Maersk Mc-Kinney Moller institute and physically made by the company μLab and a technician at the Maersk Mc-Kinney Moller Institute. The further development of the bricks has been an ongoing process between Jakob Nielsen, μLab and Henrik Hautop Lund. It should also be noted that in our thesis, the terms *brick*, *intelligent brick/artefact* and *I-BLOCK* can be used interchangeably.

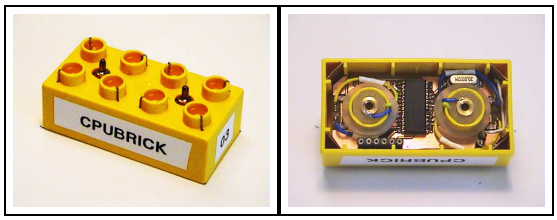


Figure 2.1: A standard CPU brick

structure work, each brick is also equipped with power connections. These can be seen in figure 2.1 as the thin wires on the studs.

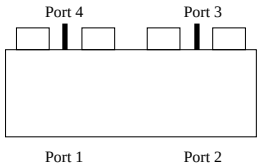


Figure 2.2: Schematic view showing the communication ports.

2.1.1 Specifications

This section will describe the bricks used in this project. Each brick is a modified DUPLO brick, as shown in figure 2.1. As described by Jakob Nielsen [3], the processing power in each brick is "...the Microchip PIC 16F876, which include a RISC CPU and 8Kbytes of flash program memory, 368 bytes of data memory and 256 bytes of EEprom data memory.". Furthermore, the PIC micro controller offers speeds at up to 20 MHz and includes an A/D converter as well as timers and PWM generators. Each brick is equipped with four communication ports, two on top of the brick and two on the bottom. This makes the brick capable of communicating with as many as four neighbour bricks at the same time. The communication ports are placed as shown in figure 2.2 and can also be seen on figure 2.1. To distribute the power needed to make an I-BLOCK

2.1.2 Brick types used

A number of different bricks has been used in this project. In general all the bricks contain the same processing power, however the functionality vary according to the brick type. This section will give a brief presentation of each brick.

LCD brick

This brick has a built-in LCD (two lines, eight characters per line) and three buttons for control. In this project, the LCD brick is used for presenting the constructed sentence. By pressing the buttons the user is able to scroll the sentence, as explained in the chapter concerning the software implementation. Figure 2.3 shows the LCD brick.

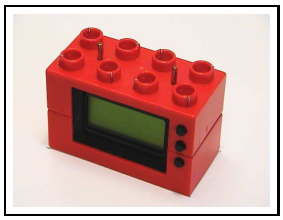


Figure 2.3: LCD brick

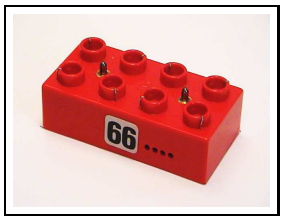


Figure 2.5: Speaker brick

LED brick

This brick has eight built-in light emitting diodes (LEDs), which can be turned on an off in any pattern decided by the programmer. These bricks are very useful for debugging, as the programmer for instance can switch on one set of diodes, when a brick is receiving data, and another set when transmitting. The LED bricks have been used extensively as elements in a sentence (words). However other bricks (e.g. microphone bricks) have been used for this purpose, when no more LED bricks were in stock. An LED brick is shown in figure 2.4.

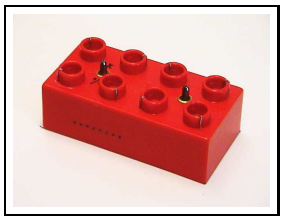


Figure 2.4: LED brick

Speaker brick

This brick has a built-in speaker, which is capable of producing sounds with certain frequencies determined by the program. In this project, the speaker brick is used for generating an audio feedback to the user, i.e. a nice tune on success and a beep tone upon failure. The speaker brick is shown in figure 2.5

2.1.3 Bricks as objects

When discussing how to represent the bricks in an abstract way, the authors have come to the conclusion that the definition of an object is the most fitting. In Computer Science, Object oriented methods are, of course, focused on the concept of objects. As described in Mathiassen et al. [5], an object is "*an entity with identity, state and behaviour*"[1]. This definition fits the I-BLOCKs very well and we encourage the reader from here on to think of the I-BLOCKs as physical representations of this definition. See chapter 7 for a discussion of this subject.

[1] Translation from Danish to English by the authors.

Chapter 3

Analysis

> "Get the habit of analysis - analysis will in time enable synthesis to become your habit of mind."
>
> - *Frank Lloyd Wright*

THIS project is concerned with the design and implementation of a linguistics scenario based on I-BLOCKs for use with children at 6-10 years of age. In this chapter we will describe the language theory used in the analysis of the scenario. Examples are presented in Danish with an accompanying English translation, thereby showing some basic grammatic differences in the two languages. Firstly, a brief introduction to the development of the linguistics scenario is given.

3.1 Brainstorming sessions

The idea of using the I-BLOCKs to solve language tasks were one of a series of initial ideas proposed by the authors at the beginning of this project. During two brainstorming sessions with students Valentina Palma and Alessia Rullo, University of Siena, Italy, as well as Jakob Nielsen of the Maersk Institute, Southern Denmark University, the idea for the linguistics scenario was further developed. The sessions were focused on, through brainstorming, the designing of one or more scenarios for indoor/outdoor learning/entertainment. The linguistics scenario is most appropriately placed in the indoor learning category, albeit the user of course is free to take the bricks outside. Besides the linguistics scenario, an outdoor entertainment scenario was developed. The Living Tree, a highly reconfigurable I-BLOCK structure, is the working title of this scenario, for which the authors collaborated with Jakob Nielsen on creating the first version of a simple prototype. This prototype is described in [7], which is included in appendix B, and has later on been improved extensively by Jakob Nielsen. However, the linguistics scenario is the only focus of this thesis.

3.2 Basic grammar

The initial idea was to make the user able to construct simple sentences like the one shown in figure 3.1. The capital letters above the sentences are A for Article, N for Noun and V for Verb. Webster's Dictionary defines these three as:

	A N V A N
Danish:	et barn spiser en is
English:	a child eats an icecream
	A N V A N

Figure 3.1: An example sentence.

Article

One of the three words, a, an, the, used before nouns to limit or define their application. A (or an) is called the indefinite article, the the definite article.

Noun

A word used as the designation or appellation of a creature or thing, existing in fact or in thought; a substantive.

Verb

A word which affirms or predicates something of some person or thing; a part of speech expressing being, action, or the suffering of action.

The sentence in figure 3.1 encourages a partitioning on the form

$$< \text{subject} > < \text{verb} > < \text{object} >$$

where the subject and the object are both consisting of an article and a noun. Using this partitioning we obtain a general structure, in which two structurally identical parts (subject and object) are linked together by a verb. In Danish, however, the subject and object are not always on the form presented in figure 3.1. As shown in figure 3.2, the article may succeed the noun. This is always the case with sentences in determinate form. Still, the partitioning holds. Webster's' Dictionary defines subject and object as:

Subject

One of the two main constituents of a sentence; the grammatical constituent about which something is predicated.

Object

A grammatical constituent that is acted upon; "the object of the verb".

	N A V NA
Danish:	barnet spiser isen
English:	the child eats the icecream
	A N V A N

Figure 3.2: An example in determinate form.

Observe the way in which the article is placed immediately after the noun with no separating space, reducing the Danish sentence to three words rather than five.

3.2.1 Questions

An enhancement of the scenario described so far is to make the user able to ask questions. In Danish this is done by placing the verb as the very first element in a sentence which of course is different from English where an extra verb is inserted in the beginning of the sentence. This difference is illustrated in figure 3.3.

	V A N A N
Danish:	spiser et barn en is ?
English:	does a child eat an icecream ?
	V A N V A N

Figure 3.3: An example of a question.

3.2.2 Adjectives

Another enhancement is the possibility of using adjectives. Figure 3.4 shows some possible ways of inserting an adjective (marked by A') The use of

```
                A  N    V  A  N  A'
Danish:         et barn spiser en is hurtigt

English:        a child eats an icecream fast
                A  N    V  A     N     A'

                A  A'   N    V   A  N
Danish:         et hurtigt barn spiser en is

English:        a fast child eats an icecream
                A  A'   N    V  A     N
```

Figure 3.4: Examples of using adjectives

adjectives introduces the opportunity of applying a certain property to a noun or a whole sentence, thereby giving the user more ways to construct sentences with different semantics.

3.3 Special considerations

As the reader may have noticed, the Danish language contains the articles "en" and "et". These are meant to be used with certain nouns. For instance "et barn" (a child) is correct whereas "en barn" is not. To ensure that this rule is not broken, we introduce the concept of class one articles/-nouns and likewise class two articles/nouns. "En" is the class one article and "et" is the class two article. Examples of class one nouns are "is" (ice/ice cream) and "bil" (car); class two examples are "fly" (aeroplane) and "hus" (house). Hence, it is only allowed to combine articles and nouns of the same class, as shown in figure 3.5. This makes it easier at the implementation level to decide, which articles and nouns are allowed to be put together. The third entry in figure 3.5 is incorrect because it combines

1. "en is" ✓
2. "et barn" ✓
3. "en fly" ✗

Figure 3.5: Class examples.

a class one article and a class two noun, thereby breaking the rule. Certainly, one has to know the Danish language to tell, which articles and nouns belong to the same class.
To formalise the basic grammatic rules, we have constructed a grammar[1] as shown in figure 3.6. Let G be the grammar (W, Σ, R, S). G is designed

$$
\begin{aligned}
W &= \{S, A_1, A_2, N_1, N_2, V, S', O\} \cup \Sigma \\
\Sigma &= \{en, et, barn, is, spiser\} \\
R &= \{S \to S'VO, \\
&\quad S' \to A_1N_1, \\
&\quad S' \to A_2N_2, \\
&\quad S' \to N_1A_1, \\
&\quad S' \to N_2A_2, \\
&\quad O \to S', \\
&\quad N_1 \to is, \\
&\quad N_2 \to barn, \\
&\quad A_1 \to en, \\
&\quad A_2 \to et, \\
&\quad V \to spiser\}
\end{aligned}
$$

Figure 3.6: An example of a grammar for the language over the alphabet Σ

to be a grammar for a small subset of the Danish language. The non-terminal symbols are S for sentence, A_1 for class one article, A_2 for class two article, N_1 for class one noun, N_2 for class two noun, V for verb, S' for subject and O for object.

[1] Consult Lewis and Papadimitriou [6] for further information concerning grammars.

Some strings in $L(G)$ are:

1. Et barn spiser en is
   A child eats an ice cream
2. En is spiser et barn
   An ice cream eats a child
3. Barnet spiser en is
   The child eats an ice cream

Our only interest is to decide whether a sentence is grammatically correct. Hence, sentence 2 is a correct sentence, even though it doesn't make any sense.

3.4 Summary

This chapter concerned the various grammatic rules, which are to be obeyed by the two implementations discussed in chapter 4. The class concept was introduced and explained as an important thing regarding the Danish language. Furthermore some suggestions for extensions of the linguistics scenario were given. Finally, the basic grammatic rules were formalised by the grammar shown in figure 3.6, with an example alphabet.

Chapter 4

Software Implementation

> **software**: something used or associated with and usually contrasted with hardware: as **a** : the entire set of programs, procedures, and related documentation associated with a system and especially a computer system; specifically : computer programs **b** : materials for use with audiovisual equipment
>
> *Webster's Dictionary*

DURING this chapter, the authors give a comprehensive overview of two different implementations of the linguistics scenario. Drawings of example sentence structures will be shown, as well as code excerpts from the implementations. The source code for the software implementation is written in ANSI C. For compilation, the CCS C Compiler [2] has been used in cooperation with the MPLAB programming environment. Furthermore, the communication protocol written by Jakob Nielsen [3] has been used for sending integer values and data types. The code written by the authors as well as that written by Jakob Nielsen is included on the CD-ROM accompanying this Bachelor thesis. A table of contents for the CD-ROM is found in appendix C.

4.1 Why two prototypes?

This chapter concerns the design and implementation of two prototypes. Although Prototype 2 was the first working prototype, it was the last one tested during the school sessions, hence the name Prototype 2. When developing this prototype, an idea for another approach came up. This resulted in the development of Prototype 1, named so due to the fact that it was the first prototype tested during the school sessions. The reason for creating two different prototypes may not be obvious to the reader. However, implementing the linguistics scenario has been troublesome. The authors have been confronted with a variety of hardware related problems, the most prominent one being the power connections. This made it very hard to determine the probable cause of failure when testing the written code. Not always knowing whether a failure was caused by a bug in the code[1] or simply by unstable power connections, the authors decided to implement two different approaches to solving

[1] Though not being able to spot any when reading it.

the linguistic task, thereby having two options to choose from when doing the testing with children. I.e. if one prototype seemed to work better than the other, this would be chosen. As clarified by this and the next chapter, the two prototypes, though containing differences, were both successfully implemented and tested.

4.2 Data types for identification

This section explains the data types used to identify each brick in the context of the other bricks. The naming convention for the data types is `SEND_` followed by a brief descriptive suffix. To fully understand the following, the reader needs to be acquainted with the material concerning language properties, presented in chapter 3.

4.2.1 Types in detail

In the next 23 entries the data types and their relation to one another will be explained in detail. The data types have been implemented to resemble, the theory about grammar discussed in chapter 3.

1. `SEND_ARTICLE_CLASS1`
   If a class one article is placed as the very first element of a sentence, this data type is passed on.

2. `SEND_ARTICLE_CLASS2`
   Explanation follows from the previous entry.

3. `SEND_NOUN_CLASS1`
   If a class one noun is placed as the very first element of a sentence, this data type is passed on

4. `SEND_NOUN_CLASS2`
   Explanation follows from the previous entry.

5. `SEND_SUBJECT`
   If an article of class one receives `SEND_NOUN_CLASS1`, the data type `SEND_SUBJECT` is passed on. This is also the case if a noun of class one receives `SEND_ARTICLE_CLASS1`. The case for class two articles and nouns follows the exact same rules.

6. `SEND_FIRST_PART`
   When a verb receives the data type `SEND_SUBJECT`, it means that the verb is working as a link between a subject and an object. In this case the data type `SEND_FIRST_PART` is passed on to the object side.

7. `SEND_OBJECT_NOUN_CLASS1`
   This data type is passed on from a class one noun brick that receives the `SEND_FIRST_PART` data type. `SEND_OBJECT_NOUN_CLASS1` indicates that we are now on the object side of the sentence and the next brick is the terminator of the sentence (except if an adjective is finishing the sentence) and should be a class one article.

8. `SEND_OBJECT_NOUN_CLASS2`
   Explanation follows from the previous entry.

9. `SEND_OBJECT_ARTICLE_CLASS1`
   This data type is passed on from a class one article brick that receives the `SEND_FIRST_PART` data type. `SEND_OBJECT_ARTICLE_CLASS1` indicates that we are now on the object side of the sentence and the next brick is the terminator of the sentence (except if an adjective is finishing the sentence) and should be a class one noun.

10. SEND_OBJECT_ARTICLE_CLASS2
    Explanation follows from the previous entry.
11. SEND_OBJECT
    Same functionality as the previous four entries, except no indication of class is made.
12. SEND_SENTENCE
    The very last element of a sentence sends this data type to indicate that the sentence is complete.
13. SEND_VERB
    In case a verb is beginning the sentence, the user is constructing a question. SEND_VERB is passed on from the verb to the subject.
14. SEND_QUESTION_NOUN_CLASS1
    This data type is passed on from a class one noun, if the noun succeeds a verb. Please note that the verb should be starting the sentence, thereby creating a question.
15. SEND_QUESTION_NOUN_CLASS2
    Explanation follows from the previous entry.
16. SEND_QUESTION_ARTICLE_CLASS1
    Again, the explanation follows. In this case a class one article is succeeding the verb.
17. SEND_QUESTION_ARTICLE_CLASS2
    Explanation follows from the previous entry.
18. SEND_QUESTION_FIRST_PART
    This data type is sent from the last element of the subject (article or noun of either class) to notify the rest of the structure that it is on the object side of the sentence.
19. SEND_QUESTION_OBJECT_NOUN_CLASS1
    This data type has the same functionality as SEND_OBJECT_NOUN_CLASS1 only differing from this in indicating that the sentence is a question.
20. SEND_QUESTION_OBJECT_NOUN_CLASS2
    Explanation follows from the previous entry.
21. SEND_QUESTION_OBJECT_ART_CLASS1
    This data type has the same functionality as SEND_OBJECT_ARTICLE_CLASS1 only differing from this in indicating that the sentence is a question.
22. SEND_QUESTION_OBJECT_ART_CLASS2
    Explanation follows from the previous entry.
23. SEND_QUESTION_SENTENCE
    This data type is sent from the very last element in a question, thereby indicating that the sentence is complete.

The data types are defined as integer values in the file lib.h.
To assist the written explanation of the data types, the following sections will present visual examples of how the data types are used.

4.3 String transmission

The first problem discovered was, that it was not possible to send a string from one brick to another in the original code base [3]. This problem has been solved, by implementing functions for sending and receiving strings. The two functions, `send_string` and `receive_string`, are located in the header file transmit.h. The idea is to achieve synchronisation between the involved bricks, so that the receiving brick is ready when the sending brick is sending. This can be considered as a simple form of negotiation; actually the only thing negotiated is when to send the string.

4.3.1 Sending strings

At figure 4.1 the flowchart shows how a string is sent when `send_string` is called. The sender will

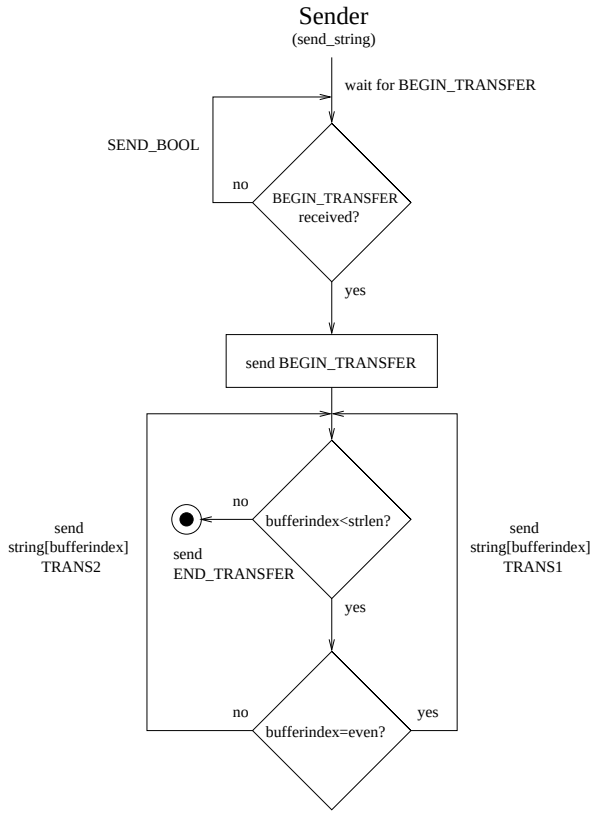


Figure 4.1: Flowchart describing function send_string.

wait for data type BEGIN_TRANSFER while sending the boolean value true to the receiving brick indicating it is ready to transmit the string. The reason why the sender has to wait for BEGIN_TRANSFER is to make sure that the receiving brick has called `receive_string` and therefore is ready to receive a string. When a synchronised state is achieved, the sending brick will send BEGIN_TRANSFER to the recipient and then start transmitting the string. To ensure that each index in a string is received just once it has been necessary to use two data types TRANS1 and TRANS2 to distinguish even and odd indices. I.e. even indices are send as TRANS1 and odd as TRANS2, as shown at the lower part of figure 4.1. When the string has been sent, the data type END_TRANSFER is sent to indicate that the transfer has completed.

4.3.2 Receiving strings

Figure 4.2 shows a flowchart of how strings are received. When the recipient has received the boolean value true it will send BEGIN_TRANSFER to the sending brick to show that it is ready for receiving a string. Function `receive_string` knows that while it has not received END_TRANSFER, it will first receive TRANS1 and then TRANS2 and then this pattern is repeated until END_TRANSFER is received. If an error should occur, say data of type TRANS2 is not received as expected according to the transmission pattern, then `receive_string` will time out waiting for data and the receiving state will be terminated. This is done because there is no reason to continue receiving a string containing errors.

4.4 Common functionality

The two prototypes have some common features, namely the speaker brick and the wizard brick.

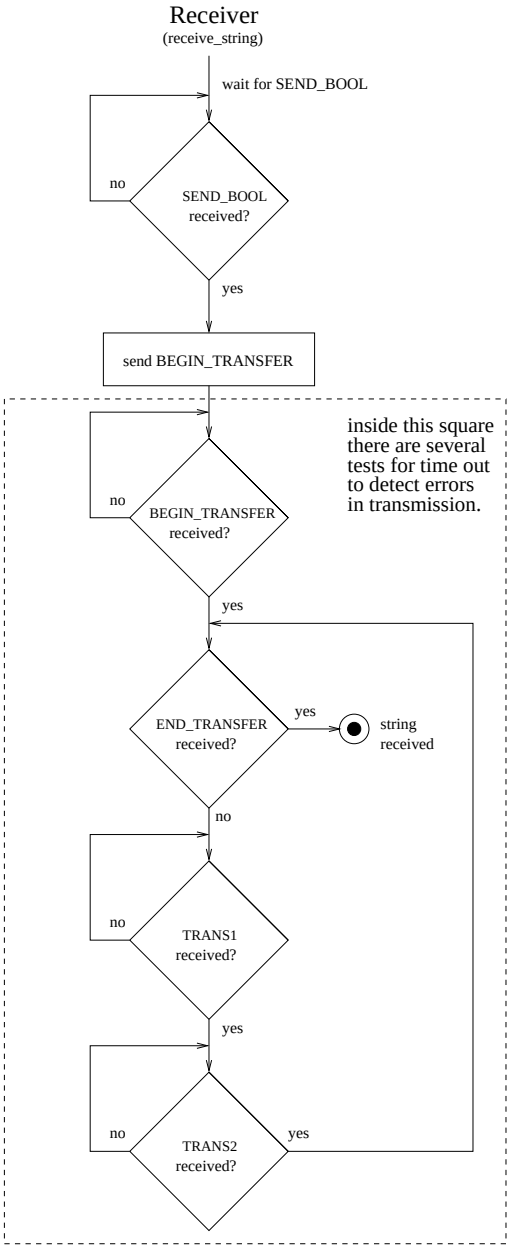


Figure 4.2: Flowchart describing function receive_string.

4.4.1 The wizard brick

The program for the wizard brick is located in wizard.c. The wizard brick is a LCD brick used for displaying strings. The name for the brick is inspired by Valentina Palma [4]. The wizard will wait for the completed sentence to be received. The wizard will receive the string using the function `lcd_receive_string` located in lcd.h. The reason why `lcd_receive_string` is used instead of `receive_string` is that it has some advantageous features for use in the LCD brick. The advantages is some output messages, and the buttons on the LCD brick is used, so when the LCD brick is ready for receiving the string, it is up to the user to decide when the string should be received. Finally the wizard brick will display the string using the function `lcd_scroll_down`, located in lcd.h. `lcd_scroll_down` uses the buttons on the LCD brick. The buttons are used to scroll the display, the top button are used to scroll upwards, and the bottom button scrolls downwards. This behaviour is much the same as the one found in mobile phones when reading SMS messages, and it is on purpose this behaviour is chosen, taking the users in to consideration, because of their knowledge of how mobile phones are operated.

4.4.2 The speaker brick

The program for the speaker brick is located in speaker.c. The speaker brick will wait for the data type `SEND_SOUND`, indicating that parsing and transmitting the sentence has finished, and the speaker should start playing a happy or a sad tune depending on the result. When `SEND_SOUND` is received, the value send with `SEND_SOUND`, either `HAPPY` or `SAD`, corresponds to the tune played. The tune will be played using functionality from func.h, the tunes to play are defined in func.h. A

tune is defined by two arrays `music_tones` and `music_dura`. `music_tones`, where each place in the array corresponds to a note, defines in which order the notes should be played, and `music_dura` of same length as `music_tones`, is the duration of each note.

4.5 Prototype 1

At figure 4.3 there is a I-BLOCK structure representing a sentence. The basic part of the structure is the three types of bricks, used to build a simple sentence, noun, article and verb. The LCD is only used for output to the user, and the speaker will play a small tune when a sentence is build correctly.

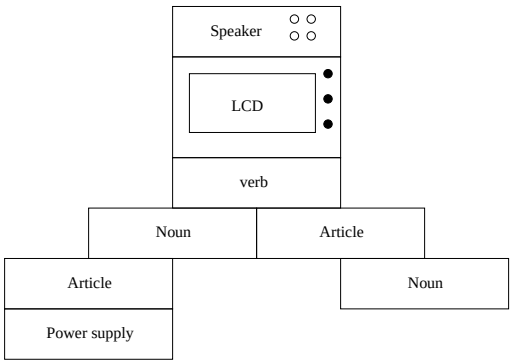


Figure 4.3: A Structural representation of a sentence.

4.5.1 Basic functionality

The implementation of the basic part bricks can be divided into smaller parts. First the structure will initialise itself. The next task is to parse the sentence to check whether it's a syntactically correct built sentence or not. If the parsing has ended

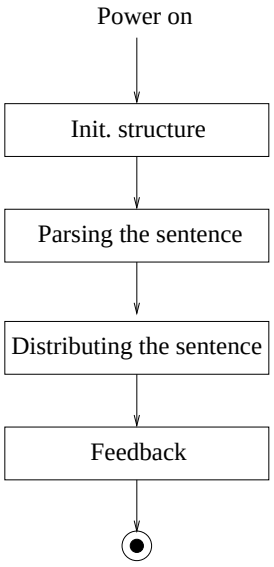


Figure 4.4: General flow in both bricks and the entire structure.

successfully, the strings will be collected and concatenated. Then the sentence will be distributed to all the bricks, enabling the possibility of placing the LCD brick at any given position in the structure. The final task is to generate aural and visual feedback to the user. At figure 4.4 an overview of the general flow in the program is shown. Before a more specific description of the different tasks are given, the rules for parsing will be defined.

Defining rules for parsing

The verb brick has an important role in the structure, because it is used to start parsing the sentence. The verb brick is chosen, because it can be seen as the middle part in a sentence of the kind discussed in chapter 3. At figure 4.5 there is a verb brick divided into four parts. The division is only imaginary, but useful to create some rules on how to interpret the structure, only by knowing where it

is connected to other bricks. The program used for the verb brick is located in verb.c There are three

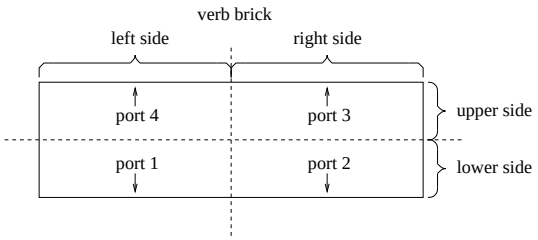


Figure 4.5: The verb brick divided into four parts.

rules, and they are defined as follows:

1. If only there are connections on the lower side of the brick the subject must be on the left side and the object on the right side.
2. If only there are connections on the upper side of the brick the subject must be on the left side and the object on the right side.
3. If there are connections on both the upper side and the lower side of the brick the subject is on the lower side and the object on the upper side.

If none of the three rules are fulfilled, it is not a valid configuration, and an error message will be generated, as output to the user instead of the sentence.

Initialising the structure

When the structure is powered on the bricks will all send out an initialising signal, `SEND_INIT`, on all their ports. This will continue until all the bricks have received the initialisation signal. This is done to initialise the structure, so the neighbour brick are able to detect, on which ports they are connected. When the structure has been initialised it is up to the verb brick to determine which of the three rules should be used for parsing.

Parsing the sentence

To parse the sentence obviously some method must be chosen. The problem is that we want the structure to act as a single unit achieved by the different bricks. When the structure is working in a distributed manner like this, it is not possible to parse it in a usual way, where you after reading one token know what to expect as the next token. This problem is solved by letting a given brick know which inputs are accepted, and then on behalf of the input determine whether to continue parsing or generate an error.

The parsing can be divided into two parts, subject part and object part, to take advantage of the parsing rules earlier defined and with respect to the basic grammar in section 3.2. To exemplify how the parsing is done, let the structure at figure 4.6 be the example. First the subject part of the sentence will be parsed. This is done by letting the verb brick send it's own type on the port where the subject part is connected. When the noun brick receives a verb as input it will pass it's own type to the next brick in the subject part. The article brick will received the noun type it will check whether the noun is the same class as itself. Now parsing of the subject part has ended successfully, and a success signal will be send back to verb brick. If parsing of the subject part is successful, the object part is parsed in the same manner.

Collecting the words

If the parsing has ended successfully, the single words in the sentence can be collected and concatenated. To optimise the performance of the structure instead of just sending a success signal back to the verb brick when the parsing has been successful, the success signal consist of the strings so

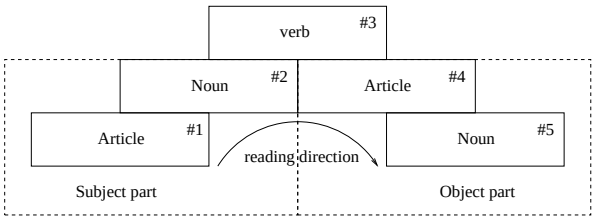


Figure 4.6: Order of concatenating strings.

far concatenated and a data type indicating success. In the subject part at figure 4.6 the article and the noun are concatenated in the reading direction which is expected, article 1 + noun 2, in a depth first kind of way. If the same strategy is applied to the object part of the sentence, the strings will be concatenated in opposite order of the reading direction. This can be prevented if the structure knows whether it's from the subject part or the object part the strings are collected. To notify each brick of which part of the structure it belongs to, the verb brick will, when it starts parsing, send out the value `subject` combined with it's own data type for the subject part and the value `object` for the object part. Now it's easy to find out whether it's the subject part or the object part the strings to be concatenated belong to. This is done just by looking at the value received with the type of the preceding brick. After the strings have been collected and concatenated in the verb brick, the complete sentence will be distributed to the rest of the structure.

Distributing the sentence

When the sentence is completed it is located in the verb brick. The sentence now has to be distributed to the rest of the structure to enable the possibility of placing the LCD brick anywhere in structure. First the verb brick will send out the data type `SEND_SENTENCE` to the subject part and then afterwards to the object part. When any given brick receives data type `SEND_SENTENCE` it knows that it will receive the complete sentence, and then pass it on to the next brick in the structure. After the sentence has been distributed an appropriate feedback to the user will be generated, depending on whether an error has occurred or not.

Feedback

When the completed sentence has been distributed successfully, first the structure will decide if any error has occurred. Then it will send one of the values `HAPPY`or `SAD` as the data type `SEND_SOUND`. This is intended to the speaker brick so if the speaker brick is attached to the brick it will play a happy or a sad tune. If the speaker brick is not attached to the brick, `SEND_SOUND` doesn't have any effect but, it is easier just to send it, instead of detecting whether the speaker brick is attached or not. Now a light pattern is lit on the LEDs on the brick indicating success. The task for the brick is now completed, but if it receives `SEND_DISPLAY` on one of it's ports it knows it is a request from the LCD brick and it will send the completed sentence to the LCD brick so it can be presented to the user.

4.5.2 Code excerpts

The noun brick and the article brick have very similar behaviours, and because of that there is just one program used to program both types of bricks. The program is located in common.c. The way to choose which type of brick, the program is used in, is to initialise a variable called `brick_type` to a value between 1 and 4 according to the brick type. The part of the program taking care of choosing the brick type is shown in figure 4.7. If `brick_type` is set to 1, the program will enter case 1, it means

```
switch(brick_type){
 case 1:
   own_type = SEND_ARTICLE_CLASS1;
   receive_type = SEND_NOUN_CLASS1;
   strcpy(stringbuffer, "en");
   break;
 case  2:
   own_type = SEND_ARTICLE_CLASS2;
   receive_type = SEND_NOUN_CLASS2;
   strcpy(stringbuffer, "et");
   break;
 case 3:
   own_type = SEND_NOUN_CLASS1;
   receive_type = SEND_ARTICLE_CLASS1;
   strcpy(stringbuffer,inputstring);
   break;
 case 4:
   own_type = SEND_NOUN_CLASS2;
   receive_type = SEND_ARTICLE_CLASS2;
   strcpy(stringbuffer,inputstring);
   break;
 }
```

Figure 4.7: A code excerpt from common.c to determine brick type.

that the brick itself will be an article class one and besides the verb brick can be correctly placed after a brick of type noun class one. The string is predefined to "en"[2] because of the definition of article class one. More general `own_type` is the bricks own type. `receive_type` is the possible input type besides verb. If the brick is chosen to be a noun `inputstring` has to contain an appropriate noun, but if the brick is an article the string is predefined.

4.6 Prototype 2

This prototype uses the data types presented in section 4.2.1 as the only indication of the identity of a brick. The function `send_port(int ports, int value, int kind)` is used to send the different data types. The `value` field is always '1' (a "true" value). Some code might be saved by utilising the `value` field as well, but the strict use of data types with long, descriptive names seems more friendly to the programmer.

4.6.1 Basic functionality

The tower-like structure in figure 4.8 shows, how the data types are sent through the structure to create the sentence "et barn spiser en is" (a child eats an ice cream). Each brick listens for input and acts according to this. When a brick has received a valid data type, it calls the `receive_string` function and thereby receives the string from the same brick that sent the received data type. Furthermore the brick adds its own string to the received string, sends the appropriate data type to all output ports and finally transmits the string. This algorithm is formalised, in a very general fashion, in algorithm 1. Here a *language data type* is one of the data types described in section 4.2.1.

This particular implementation uses a "start"

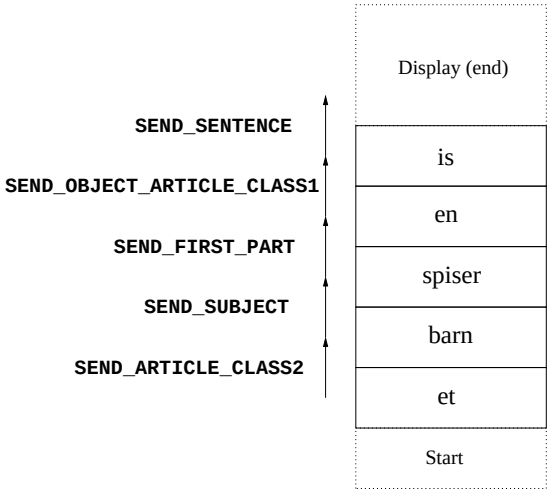


Figure 4.8: Sending data types

[2] Danish article corresponding to the English "a".

Algorithm 1: General algorithm

```
initialisation;
while true do
    if language data type is received then
        receive string s from sender;
        send appropriate data type to recipient;
        add own string, s', to s: s = s + s';
        send s to recipient;
    end
end
```

brick (placed at the bottom of the structure in figure 4.8) to indicate, which brick should be the first in the sentence. This concept defines the rule that the user should always begin the sentence with the "start" brick and end it with the display brick[3]. In this way, the user can build sentences like the one shown in figure 4.8, i.e. read from the bottom and up. But this sentence could also be reversed by switching the "start" brick and the display. In this case, the sentence should be read from the top down resulting in the sentence "isen spiser barnet" (the ice cream eats the child). Clearly, this is nonsense. However, the sentence is grammatically correct, and deciding, whether the semantics of the sentence is sensible, is, as earlier noted[4], not of our interest. The "start" brick sends the `SEND_START` data type on all ports to notify its presence. Note, that although the user has to begin the sentence with a start brick, the control of the structure is in no way centralised. As mentioned earlier, the start brick only indicates the beginning of the sentence. The rest is decided by each brick.

[3] I.e. this implementation does *not* allow the user to place the display at any given position of the structure.

[4] In section 3.3

One of the main ideas of this project is that the user should be able to construct a single sentence in many ways, thereby obtaining a variety of different structures. For instance, the structure in figure 4.9 is another way of constructing the sentence in figure 4.8. In short, the resulting sentence is the same in the two figures, yet the morphology is quite different. The possibility of

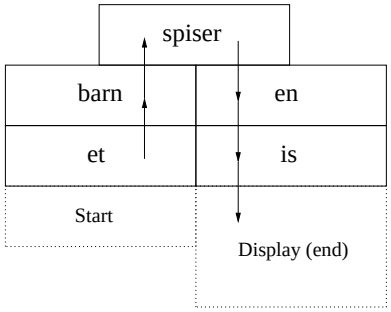


Figure 4.9: Same sentence - different shape

making many different shapes is ensured by letting each brick "remember", where the last input was sent from. Once the input port(s) is/are known, the output ports are equal to all ports except the input port(s). Hence, there might be situations, where some output ports are not attached to any other brick, as shown in figure 4.10, but this is alright, since no harm is done sending data to an unconnected port. In figure 4.10, provided that the user starts the sentence from the bottom, port 1 is the input port for the brick "barn" (child) and ports 2,3,4 are all output ports.

4.6.2 Extension one - Questions

A feature of this implementation is the concept of making the user able to ask questions using the I-BLOCKs. As mentioned in chapter 3, this is done in Danish by placing the verb as the very first ele-

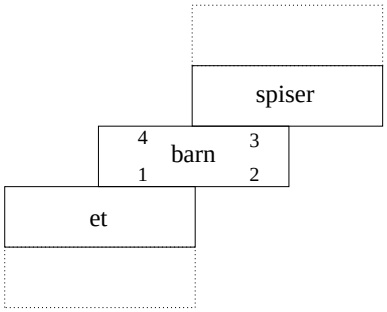


Figure 4.10: Ports for in- and output

ment of the sentence. An example of this is shown in figure 4.11, which shows one possible way of constructing the sentence "spiser barnet isen?" (does the child eat the ice cream?). Notice the way in which the part of the sentence succeeding the verb still naturally divides itself into a subject and an object. Following the associated numbers of the

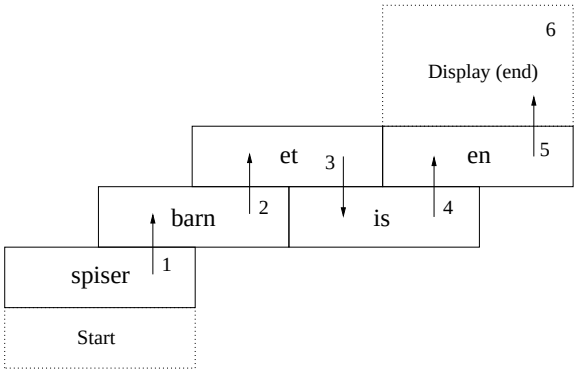


Figure 4.11: Question

bricks in figure 4.11, we obtain the flow described in the next 6 entries:

1. The verb "spiser" begins the sentence. The data type `SEND_VERB` is sent from this brick.

2. The second brick is a class two noun, "barn". By receiving the data type `SEND_VERB`, this bricks knows that it is part of a question and proceeds by sending the data type `SEND_QUESTION_NOUN_CLASS2`.

3. This brick is a class two article, thereby matching the preceding noun. Hence, the article has received valid input and subsequently sends the data type `SEND_QUESTION_FIRST_PART`.

4. Having reached the object side, we encounter a class one noun. This noun receives the data type `SEND_QUESTION_FIRST_PART` and now sends the data type `SEND_QUESTION_OBJECT_NOUN_CLASS1`.

5. This is the final brick in the sentence. Having just received the data type `SEND_QUESTION_OBJECT_NOUN_CLASS1` it knows, that: 1) it is part of a question, 2) it is on the object side and 3) the previous brick is a class one noun. Using this information, and being a class one article itself, the brick now sends the data type `SEND_QUESTION_SENTENCE`. Finally a question mark '?' is added to the string before being passed on.

6. The display brick now receives the data type `SEND_QUESTION_SENTENCE` followed by the string meant for presentation.

Recall, that during every step, the strings are sent alongside the data types, resulting in the complete sentence being shown on the display.

4.6.3 Extension two - Adjectives

As mentioned in chapter 2, adjectives introduces the opportunity of applying a certain property to a noun or a sentence. The sentence "barnet spiser isen" (the child eats the ice cream) for instance could be expanded to "barnet spiser

isen hurtigt" (the child eats the ice cream fast), thereby explaining in what way the child is eating the ice cream.

The use of adjectives is an option, i.e. sentences can be constructed *without* adjectives. Thus the main idea when using adjectives is that these are inserted at some valid position without telling any of the other bricks. The only thing done by an adjective is an addition of its own string to the string, it receives. Naturally, if the other bricks should not notice the presence of an adjective, the adjective simply sends on the same data type as the one received. Algorithm 2 formalises the concept. Notice, that the algorithm indicates possible string changes. Whether this is done, depends on the position and the type of brick, the adjective is inserted after. If the adjective is inserted after the final brick of the sentence then the adjective will receive a complete sentence finalised by a dot or a question mark ('.' or '?'). The adjective should now remove this final symbol, add its own string to the string just received and finally insert the previously deleted symbol at the sentence end. If, on the other hand, the adjective is inserted somewhere *inside* a sentence, normally no change should be made to the string. There is, however, a special case to be considered. If the adjective is inserted after a class one article, the adjective must remove the ending 't'. An example of this is shown here.

Algorithm 2: Adjective algorithm

```
initialisation;
while true do
    if position is valid then
        receive string s from sender;
        send received data type to recipient;
        alter s, if necessary;
        add own string, s', to s: s = s + s';
        alter s, if necessary;
        send s to recipient;
    end
end
```

1. et barn spiser en is hurtigt
   a child eats an ice cream fast

2. et barn spiser en hurtig is
   a child eats a fast ice cream

Observe, that in the lower sentence, the 't' has been removed from the adjective "hurtigt". It should be stressed that in the implementation, all adjectives initially belong to class two articles. That is, the identity of the article, the string `article` in the program, should for instance be "hurtigt" not "hurtig", "langt" not "lang" and so on. Then, if the adjective is inserted after a class one article, the ending is changed, as described earlier. Having read this, the reader might ask: "Why not simply divide adjectives into classes one and two like nouns and articles?". Clearly, it seems more appropriate as well as more user friendly, that the user has both classes of adjectives available in separate bricks. The authors couldn't agree more, but at the time of testing we only had a limited amount of bricks available - hence the solution of keeping both classes in the same brick. However, the required implementation changes to obtain two separate classes are minimal and may be completed in a future release.
Figure 4.12 shows the possible ways of inserting an adjective in an example structure (a child eats an ice cream). If, for instance, the adjective is inserted after the verb "spiser", it will receive the data type `SEND_FIRST_PART`. Following algorithm 2, the adjective simply executes the required string operations, sends the data type `SEND_FIRST_PART` and ultimately sends the string. Notice that this

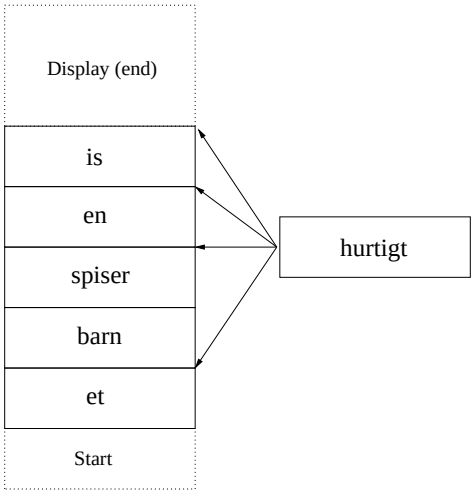


Figure 4.12: Ways of inserting an adjective in a sentence in indeterminate form

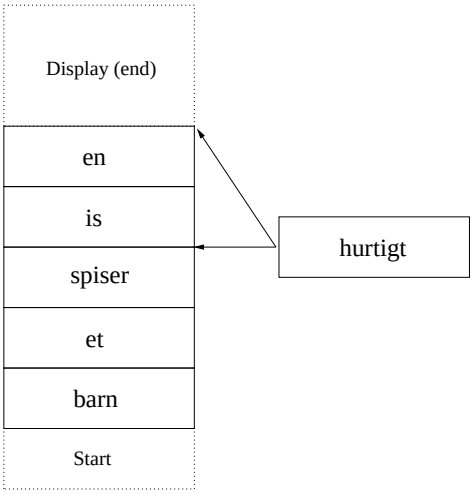


Figure 4.13: Ways of inserting an adjective in a sentence in determinate form

sentence is in indeterminate form, thus allowing the insertion of an adjective in between an article and a noun. The structure in figure 4.13 shows a sentence in determinate form (the child eats the ice cream). Here it is not allowed to insert the adjective between the article and the noun, as e.g. "et barn" becomes "barnet" in determinate form. Thus, sentences in determinate form reduces the number of ways of inserting an adjective.

As a final note, we should add that only class two adjectives (e.g "hurtigt") may end a sentence.

4.6.4 Code excerpts

The previous sections described this implementation in an algorithmic manner, i.e. *what* is done by each type of brick. This section will go through excerpts of essential code[5], thereby showing *how* things are done.

As indicated in algorithm 1 and 2, every program runs in an infinite loop. For each iteration, a check for connections is made. Furthermore in- and output ports are found. This is shown in figure 4.14, an excerpt of code taken from the source code file noun.c. When these function

```
while(TRUE) {
  check_configuration();
  get_input();
  input_ports = get_input_ports();
  output_ports = get_output_ports();
```

Figure 4.14: Function calls made for each iteration

calls are completed, the testing for data types is commenced. The very first test done by every brick (except the adjective) is for the data type SEND_START. The code in figure 4.15, also from noun.c, shows, how the brick reacts to this data type. Initially, the noun data type (decided by

[5] Please note, that original comments in the code are excluded due to typographical reasons.

```
if(is_data_type(SEND_START)) {
  output_c(1);
  send_port(output_ports, 1, noun_class);
  noun[0]-=32;
  delay_ms(10+(find_random_number()/10));
  send_port(output_ports, 42, SEND_BOOL);
  send_string(noun, output_ports);
}
```

Figure 4.15: Testing for the start data type

the noun_class flag) is send to all output ports. The subtraction noun[0]-=32 capitalises the first letter of the noun, since this is the word beginning the sentence. After a random delay[6], the string is sent to all output ports.
The noun_class flag just mentioned is set at the very beginning of each program, where the class concept is used. At the same time, other relevant class flags are set. An excerpt is shown in figure 4.16. As described in the comment above

```
//int article_class = SEND_ARTICLE_CLASS1;
int article_class = SEND_ARTICLE_CLASS2;
//int noun_class = SEND_NOUN_CLASS1;
int noun_class = SEND_NOUN_CLASS2;
```

Figure 4.16: Flags indicating noun and article classes

the variables (excluded), the programmer should only uncomment similar classes to ensure, that the grammatic rules are obeyed.

In the above example, where a brick starts a sentence, no string is received, since there are no preceding language bricks. However, if a brick is placed somewhere else than at the beginning, it needs to receive the string sent by its predecessor. The code for this is shown in figure 4.17. If

```
if(!received) {
  while(!receive_string(string_buffer,
            input_ports)) delay_ms(2);
  received = 1;
  strcatchar(string_buffer,' ');
  strcat2(string_buffer, noun);
}
```

Figure 4.17: Receiving strings

the value of the received variable is false, the program loops, until the string is received. The received string is kept in a character array called string_buffer. Afterwards a white space is inserted and the noun is appended to the received string.

In case an adjective is inserted after a class one article, the ending 't' character of the adjective should be removed (described in section 4.6.3). The code for this is shown in figure 4.18 After this is done, the same data type as the one

```
  string_length =
        get_stringlength(string_buffer);
  string_buffer[string_length-2] = ' ';
```

Figure 4.18: Deleting the ending character

received is passed on to the next brick followed by the string.

4.6.5 Code optimisation

Duplicate code is never desirable. Therefore some parts of the code were put into separate functions to be called at the required stages. However, the attempt to avoid duplicate code has so far been unsuccessful. Communication between the bricks

[6] A random delay is chosen to eliminate the risk of continuously communicating in an asynchronous manner. I.e a random delay may help the communication to once again get synchronised.

simply stops working. The reason for this has not been clarified.
The source code includes a lot of testing for input data types. This is currently done by using `...else if...` quite a number of times. This could have been optimised by using the C construct `switch(variable)`, which is faster than the previously mentioned method. However, the faster method only works when testing on a single variable. When making the Italian version of the scenario used in Siena, Italy, testing on more than one variable at the same time was done. Therefore, the switch construct was abandoned.

4.7 Summary

This chapter has given a clear and concise presentation of the two implemented prototypes. Each prototype has been implemented successfully and each offer some different functionality though centered around the implementation of basic grammar, as discussed in chapter 3. Both implementations use the display brick for presenting the constructed sentence as well as a speaker brick used for giving the user relevant feedback. Even though the outcome of the two implementations has been positive, the process of implementing them has been troublesome. Especially the brick connections have caused quite serious problems, as explained earlier. However, *when* the connections work the prototypes work as intended.
In future releases more functionality could be implemented, for instance the use of prepositions, thereby making it possible for the user to construct even more advanced sentence structures. This, however, requires more stability wrt. the connections, than is currently offered, due to the increased size of the structure.

Chapter 5

School Sessions

> "Learning is an active process in which people build knowledge from their world experiences. Ideas aren't being acquired from someone who can transfer them, but they are constructed."
>
> - *Jean Piaget*

As indicated by the title of this project, education is a key factor. Hence, a very important aspect of the project is the actual testing of the linguistics scenario with school children. Preliminary testing has, of course, been conducted by the authors, thereby showing weaknesses to be corrected at the implementation level. This eventually lead to two different prototypes, as described in the previous chapter, which have been tested with school children in 4^{th} grade[1].

5.1 Introduction

The testing took place at Aaloekke-skolen, Odense and was conducted during the last week of August and the first week of September 2003 with a distribution of two sessions a week. A lot of practical experience was gained from the sessions and this is what the authors reflect on in this chapter. Each session was of a duration of approximately 45 minutes. The class taught during the sessions was Natur og Teknik[2]. The subject of these lessons normally isn't Danish grammar. However using the I-BLOCKs introduced some technical issues for the pupils.

The authors wanted the entire testing period to be iterative, which certainly is the main reason for visiting the school four times. Using this approach, the authors were able to improve the implementation of the linguistics scenario based on the feedback from the pupils as well as the teacher. Figure 5.1 shows the phases in the iteration process. For evaluation of the outcome of the sessions, two kinds of questionnaires have been used: one for the pupils and one for the teacher. The questionnaires in their original form[3] are located in appendix A. Video recordings documenting the sessions can be found on the CD-ROM accompanying this thesis

[1] 9-10 years of age.

[2] Science class.

[3] Though translated from Danish to English by the authors.

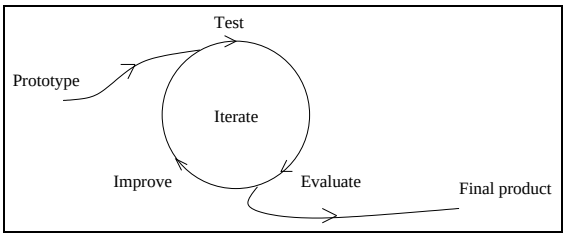


Figure 5.1: The flow of the iterative process

(See appendix C).

5.2 Session 1

The first session at Aaloekke-skolen, Tuesday, August 26^{th}, 2003.

5.2.1 Goal

This session was meant as an introduction to the I-BLOCKs. No assignments were appointed to the children, who were allowed to play and experiment with the bricks.

5.2.2 The session

The authors began by giving a short introduction to the bricks, as shown in figure 5.2, explaining how the different kinds of bricks are used, and how sentences are constructed, shown in figure 5.3. Furthermore, a brief technical explanation of the bricks were given explaining how they communicate with each other. First the class was divided into three groups of about four pupils. To prepare the pupils for the assignments proposed in session 2, they were now allowed to play and get hands-on experience with the bricks, as shown in figure 5.4. Due to severe power connection problems, this session unfortunately wasn't that successful. Another unforeseen problem was the sudden malfunction of one

Figure 5.2: Steffen explaining how the bricks work.

of the power adapters used to power the I-BLOCK structures. All these problems made the children lose interest in the bricks quite early in the session, which of course is very unfortunate. A positive aspect from this session was, that the pupils were quick to understand the way a sentence is represented by the bricks, and how to build determinate or indeterminate nouns.

5.2.3 Experiences

One of the experiences gained from this session was the importance of giving the user a feedback, when some error in the structure occurred. The reason for this is that the pupils quickly get impatient, when no visible feedback is given upon failure. By giving the user an easy-to-understand feedback, he or she can react accordingly in the appropriate way.

5.3 Session 2

The second session at Aaloekke-skolen, Friday, August 29^{th}, 2003.

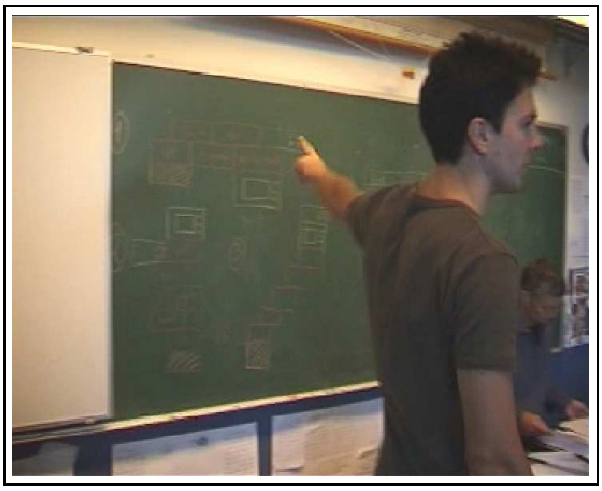


Figure 5.3: Mikkel giving examples of structures.

Figure 5.4: Pupils experimenting with the bricks.

5.3.1 Goal

The goal of this session was to make the pupils solve simple grammatic tasks using the bricks. Papers presenting the following tasks were handed out at the beginning of the session.

Determinate and indeterminate form

1. Use the bricks to build sentences with nouns in determinate form. Draw the sentence structures on paper.

2. Use the bricks to build sentences with nouns in indeterminate form. Draw the sentence structures on paper.

Draw shapes

1. Use the bricks to build sentences with different shapes/structures.

2. Draw the different structures.

5.3.2 Improvements of the prototype

The bricks had been improved for this session, because in session 1 it was only possible to gain a visual positive feedback. The improvement consisted of adding some time limits to the bricks. Hence if a structure wasn't able to verify the sentence within the time limit an error was generated and displayed to the user. This improvement prevents the user from waiting for a longer time period, uncertain whether the structure is valid or not. This hopefully also will prevent the pupils getting impatient.

5.3.3 The session

The pupils seemed very interested to solve the tasks presented. This was a bit surprising because of lack of interest in the second half due to the problems mentioned in session 1. It was nice to see, that the pupils were willing to give the bricks another chance. Figure 5.5 shows a pupil scrolling through a constructed sentence. The pupils also seemed a little more experienced this time, so the goal for

session 1 had been achieved and the pupils had gained a little experience and confidence with the bricks. The three groups started solving the tasks. To show how the pupils solved the tasks, a series of four pictures presenting the different steps are shown. When a structure had been built, figure 5.6, one or more pupils drew the structure, as shown in figure 5.7. Some pupils started painting their drawings which was unexpected, because the authors had imagined that the pupils just would draw sketches and not afterwards paint them. At figure 5.8 there is a colorful painting of the structure from figure 5.6. Some pupils also wrote down the builded sentences as shown in figure 5.9 An unsatisfying thing about the pupils started painting their drawings is, that instead of building lots of different structures the pupils where painting which wasn't the intension. A reason why the pupils would rather paint than build structures could be, that also during this session there were connection problems. Allthough that there were problems with the connections, the pupils didn't get impatient as quickly

Figure 5.5: An example on how a valid structure might look like.

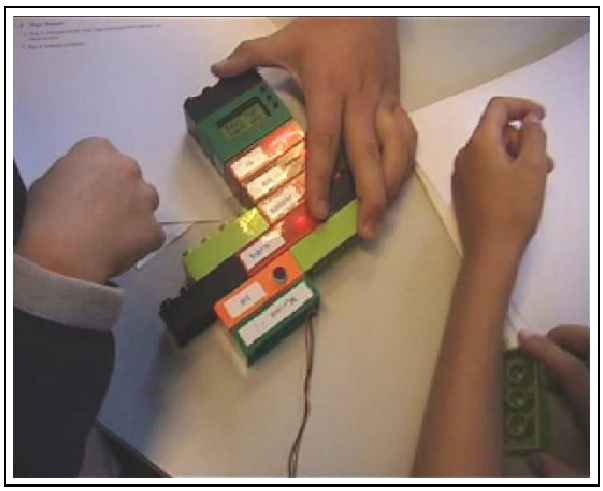


Figure 5.6: Another example of a valid structure.

Figure 5.7: A pupil drawing the structure from figure 5.6.

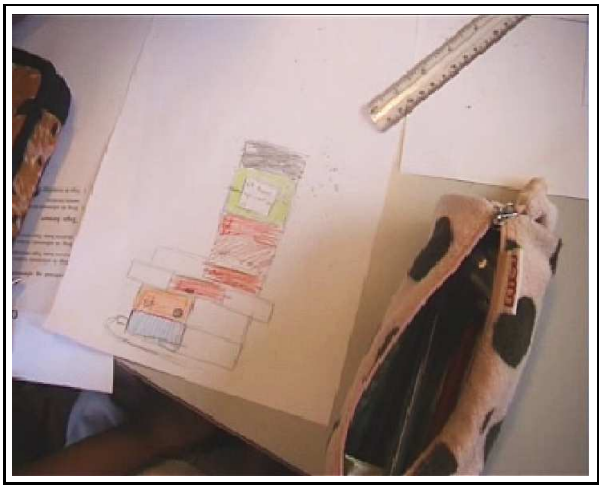


Figure 5.8: A painted drawing of figure 5.6.

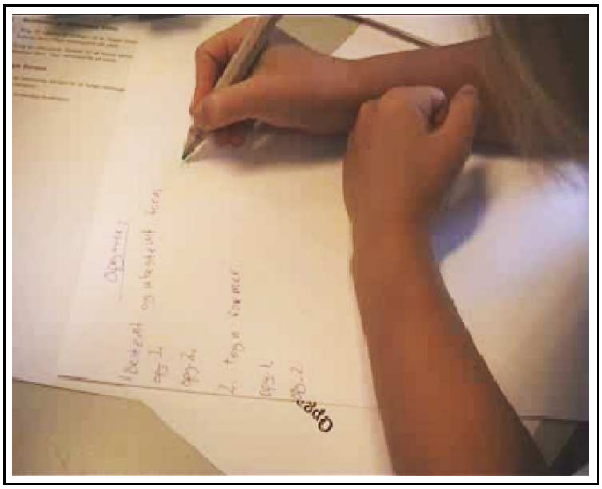


Figure 5.9: A pupil writing sentences after they have been built.

as in session 1. The reason for that could be the improvements in the prototype prior to this session. The pupils noticed the improvements right away, and it seemed easy for them to understand that when an error message was displayed, they had to try once again or build another structure. Generally the bricks worked slightly better in this session than in the first session.

5.3.4 Experiences

An experience gained from this session was that the improvement in the bricks actually were improvements. Another experience gained was that we had to be more explicit about whether a task is important or not. For instance it should be emphasised that the pupils should concentrate on building more structures. A last experience gained is that it's impossible to keep the pupils attention if we keep having connection problems.

5.4 Session 3

The third session at Aaloekke-skolen, Tuesday, September 2^{nd}, 2003.

5.4.1 Goal

The goal for this session was to complete the assignment proposed in session 2. In explicit the pupils should build as many different structures representing the same sentence as possible. They were explicitly told not to paint, but only draw sketches. How to draw the sketches also was exemplified on the black board.

5.4.2 Improvements of the prototype

For this session we brought along the final prototype. The final prototype is slightly different compared to the preceding prototypes, the changes

is mentioned in section 4.6.2 and 4.6.3. Improvements compared to the preceding prototypes is the support for new brick types and the extended grammar. I.e. the possibility to build questions and using adjectives combined with the speaker brick for aural feedback.

5.4.3 The session

The pupils started building right away after a small presentation of the new types of bricks, and the guidance on how to draw the sketches. Once again the pupils were very engaged in solving the task, even though they were impatient and some even seemed bored at the end of the last session. Especially the speaker brick gained a lot of attention, and the pupils were really pleased with the aural feedback. The pupils also liked the ability of using adjectives, maybe because they found it more challenging when they were able to build more complex sentences. The pupils didn't find it difficult to build questions or sentences containing adjectives. Figure 5.10 shows a structure representing a more complex sentence, according to the extended grammar. The most positive observation during this session was the power connections on the bricks, which this time generally speaking didn't cause any troubles. There is no exact explanation, why the bricks worked for this session and not the two previous sessions. A reason could be that pupils have become more experienced with the bricks, due to that fact, they are more capable of operating them. Figure 5.11 is a closeup of a structure containing the new brick types, on top of the LCD brick the speaker and beneath the adjective brick. The pupil is scrolling through the sentence represented by the structure.

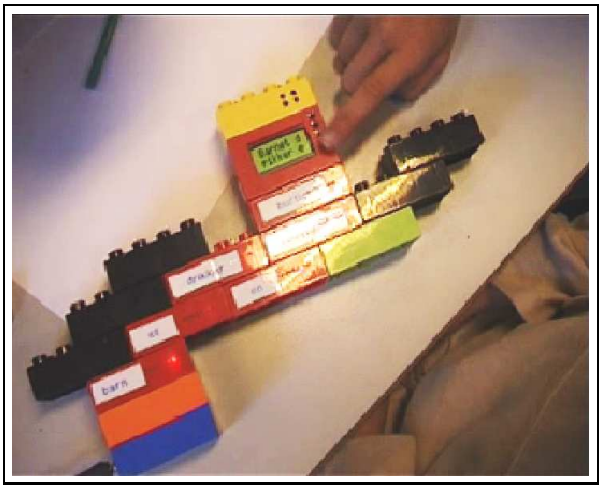


Figure 5.10: A structure supporting extended grammar.

Figure 5.11: A closeup of an extended structure

5.4.4 Experiences

As assumed the pupils seemed more interested when the bricks worked as intended without connection problems. The idea about presenting an aural feedback, worked very well, and it encouraged the pupils to continue building even though they from time to time got a negative feedback. The pupils also quickly accepted the adjective bricks, and were able to use them almost right away. This indicates that the pupils find the mapping from words to bricks easy.

5.5 Session 4

The fourth session at Aaloekke-skolen, Friday, September 5^{th}, 2003.

5.5.1 Goal

The objective with this session was to get an impression, of what the pupils and the teacher thought about the three earlier sessions. Questionnaires were handed out in two different versions: a version for the pupils and a version for the teacher. The contents of the two versions is essentially the same. The language used in the pupil version is, however, somewhat simpler than that of the teacher version, considering the age and intellectual capabilities of the pupils.

5.6 Results from questionnaires

This section evaluates the results obtained from the questionnaires.

5.6.1 Pupils

In this section the results from the pupils questionnaires will be presented. Figure 5.12 shows statistics over the answers from the pupil questionnaires

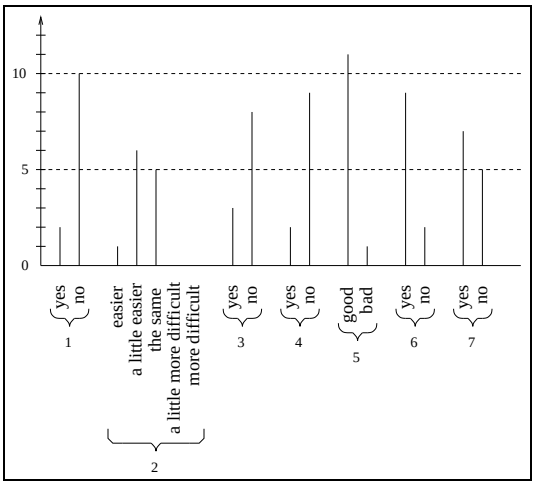


Figure 5.12: Statistics over the pupil answers

(questions 1-7). Note, that not all the columns sum up to the number 12 (the number of answered questionnaires). This is due to the fact that some pupils mistakingly had answered "Yes" as well as "No" or not at all answered the question, thereby making the answering inconclusive. In the following, the authors comment on the answers given by the pupils.

1. Has it been difficult to use the bricks?

One of the important conclusions to be made from this is the fact that the pupils in general didn't think that the bricks were too hard to work with (considering the earlier mentioned connection problems).

2. Have the bricks made it easier or more difficult to understand Danish?

This question is meant to clarify, whether the pupils approved of the I-BLOCK concept as a tool for learning Danish grammar. An important conclusion is that about 50 per cent of pupils think that the I-

BLOCKs have made it a little easier to understand grammar; the other half didn't think it was any different than ordinary elementary school methods. A positive fact is that no pupils thought that the I-BLOCKs had made it harder or a little harder to understand grammar.

3. What do you think about the bricks shape/connections? Is it too difficult to put them together?

Considering the sometimes unstable performance by the bricks, this is a very relevant question to ask the pupils. As can be seen from column 3 in figure 5.12, two thirds of the pupils didn't think the bricks were too difficult to connect. This outcome was expected, considering the answers to the first question. Question 3 was simply meant as a double check.

4. Has it been difficult to work with the other pupils to solve the problems with the bricks?

Learning children to cooperate while solving a task is very important, hence this question. 75 per cent of the pupils thought that cooperating with the other pupils on solving the problems went well. This question was included to get an overall idea of the pupils attitude towards working with the bricks.

5. What do you think about the way sentences are build with the bricks?

This question is a reconfirmation of question 2. More than 90 per cent of the pupils liked the idea of constructing sentences with I-BLOCKs. One pupil didn't like the idea. However, this pupil also answered "Yes" in question 3, indicating that the bricks were hard to put together. This may have influenced the answer for this question. However, about 90 per cent satisfaction is more, than the authors could have hoped for.

6. Has it been fun working with the bricks?

Nine pupils (75 per cent) answered "Yes" to this question. Two pupils answered "No" and one pupil answered inconclusively. The two pupils, who answered "No", also indicated that they thought it had been difficult cooperating with the other pupils. But in general, the majority of the class liked working with the bricks.

7. Would you like I-Bricks to be used in the education?

A small majority (seven pupils) answered "Yes" to this question. About half of these pupils wrote that they thought the bricks were fun to work with. The other half didn't give any reason. One of the five remaining pupils (the pupils who answered "No") explained that the bricks didn't work very well and therefore not could be used in the education. The other four didn't give any further explanation.

8. Do you have any suggestions how to make the bricks better?

This is of course a quite hard question to answer, nonetheless relevant. Two pupils stated that the connections should be improved. This is probably due to the fact that there were a lot of problems with the brick during the first two sessions. The remaining pupils didn't have any suggestions.

9. Write three good things about I-Bricks

Generally most pupils say that it's fun working with the bricks. This is just a confirmation of question six where 75 per cent agreed that it was fun.

10. Write three bad things about I-Bricks

Only half of the pupils have filled in an answer for this question. Four of the pupils is of the opinion,

that the bricks don't work very well. Two pupils stated that the bricks are boring to work with.

11. Do you have any suggestions for other problems that can be solved with I-Bricks

One of the pupils suggested the possibility of constructing songs e.g. "Lille Peter Edderkop"[4]. Another pupil suggested math as a possibility.

5.6.2 Teacher

In this section the result from the teacher questionnaire will be presented.

1. How do you rate the pupils' understanding concerning the use of I-Bricks? Is it easy/hard for them to understand the concept?

The teacher rated this as being easy.

2. Have the pupils in general obtained confidence with the bricks?

Considering the rather novel concept of using I-BLOCKs in education, this is indeed a very relevant question. The teacher answered "Yes".

3. Have the bricks made it easier/harder for the pupils to understand rules about grammar?

The teacher stated that the bricks made it easier for the pupils.

4. What do you think about the shape and connections of the bricks? Is it hard for the pupils to put them together?

The teacher answered "No".

[4] "Itsi-Bitsi Spider"

5. Have the pupils been good at cooperating in trying to solve the appointed tasks with the bricks?

The teacher answered "Yes".

6. What do you think about the way in which sentences are being constructed using the bricks?

The teacher found it to be a good idea.

7. Did the pupils have fun while working with the bricks?

Although probably hard to measure, the teacher might be in a position to decide, whether this was the case. The teacher answered "Yes".

8. Do you believe that I-bricks is a tool which might be used for educational purposes in the future?

The teachers' opinion on this is very important. The teacher answered "No" and explained that the cost-benefit ratio would probably be to high.

9. Do you have any suggestions as to improve the I-Bricks concept? Please write below.

-

10. Please mention as many as three good things about the I-Bricks concept

The teacher stated that it is easy to experiment with sentence structures. It is more fun to work with, than the ordinary textbook.

11. Please mention as many as three bad things about the I-Bricks concept

In general it is probably too expensive for common use in elementary school.

12. Are the pupils at this level too young/old to use the bricks?

It is relevant to get a statement from the teacher on, whether the pupils are too young or old. Hence this question. The teacher stated that the pupils were of appropriate age.

13. If you answered "Yes" in the previous question, what educational level would be better?

-

14. Can you suggest any other exercises for the pupils to solve with the I-Bricks? Please write below.

This question is meant for the teacher to give examples of other tasks for the children to solve using the I-BLOCKs. The teacher gave the following suggestions: math such as greater than, less than, multiplication and addition. Furthermore he mentioned foreign language teaching.

The conclusion: Judging from the answers, the teacher in general has a very positive attitude towards the I-BLOCKs. Especially the way in which the pupils are able to experiment with sentence structures is emphasised as a nice feature. However, the teacher assumes that the cost-benefit ratio is too high therefore making the I-BLOCKs inattractive to schools in general. This assumption is, however, not based on any cost information given by the authors.

5.7 Summary

The process of testing the linguistics scenario has been highly iterative. Each session has resulted in feedback from the pupils as well as the teacher, giving the authors the means to improve the prototype for the following session. One of the issues not prone to immediate improvement was the brick connections. This is due to the fact that improvements in this direction will require a redesign of the bricks. However, when the connections were working as intended, the pupils had a good time solving the appointed task. The most important improvement of the prototype during the iterative process, probably was the aural and visual feedback to the user, which increased the attractiveness of the bricks significantly. Another important observation during the sessions, was the ease, for the pupils, to understand the mapping from words to bricks. This is probably due to the fact that even though the morphology can change, the sentences are still read in a sequential manner.

The pupils generally found it interesting working with the bricks. As expected some of the pupils pointed out the problems concerning the power connections. This is most likely the reason only well over half of the pupils stating that they would like the I-BLOCKs to be used in everyday teaching.

The opinion of the teacher about the I-BLOCKs and the concept in general is very important. The most interesting statement made by the teacher was that of the I-BLOCKs making it easy for the pupils to experiment with sentence structures. However, the teacher also stated that the cost-benefit ratio regarding the use of I-BLOCKs for educational purposes probably will be too high. As earlier mentioned, the authors haven't provided the teacher with any information about the price of the I-BLOCKs.

Chapter 6

Results from Siena, Italy

"Fundamental progress has to do with the reinterpretation of basic ideas."

- Alfred North Whitehead

ANOTHER area, in which the use of I-BLOCKs may be relevant, is that of children with linguistic problems. During the first half of September 2003, a set of I-BLOCKs has been tested in Italy with a child of age 10 having this kind of learning difficulties. The Danish version of the implementation was reprogrammed to support a small subset of the Italian language. More accurately, the ability of placing an article immediately after a noun[1], as can be done in Danish, was removed, since this is not possible in Italian. The testing was conducted by Valentina Palma of the University of Siena, Italy.

6.1 Results from testing

Traditionally, special cards with different symbols and shapes have been used for helping children with linguistics problems learning sentence structures. The cards vary in shape according to the content they transmit.

- A long and narrow green rectangle for articles.
- Rectangular icon with the name print on.
- Red arrow for the verb.
- Small yellow square for preposition.

An example sentence constructed with these cards is shown in figure 6.1. Note the sequential manner, in which the sentence is constructed. The sentence reads: "the doctor examines the child". Figure 6.2

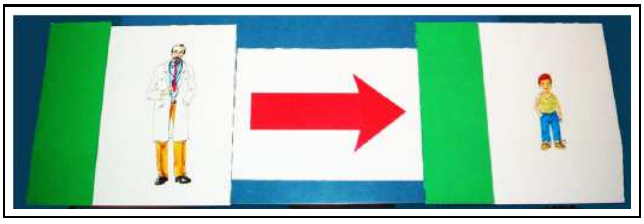


Figure 6.1: An example sentence constructed with the cards

shows a sentence built with I-BLOCKs, which has been used for testing in Siena, Italy. Note, that in

[1] E.g the noun "barnet" is composed by the noun "barn" and the article "et"

this configuration, the article bricks (4-stud DUPLO) are half the size of the noun and verb bricks (8-stud DUPLO). This has been made by request from Valentina Palma due to the varying size of the cards described earlier. Also note that the black brick next to the article "il" in the subject of the sentence is present only to support the structure. I.e., the black brick is an ordinary DUPLO brick with no processing power. In her Masters thesis [4],

Figure 6.2: The sentence used for testing in Siena, Italy.

Palma argues, that "transferring such a task to I-Bricks could be a good idea: the added value that the bricks can offer (scaffolding, external representations, meaningful feedback according to linguistic problem that child has) can really facilitate and help the child in the structuring sentence task". The results from the testing were, as described in [4]:

+ The child found it amusing to build sentences with the I-BLOCKs and was curious about the feedback given by the speaker.

+ The pieces were big enough to be manipulated in an easy way and to hold a linguistic label or an icon.

However,

÷ The connection between the bricks didn't always work, which means that it can be difficult to obtain the expected result.

÷ The mapping isn't very natural to a child, because the child is used to have a sequential order from using the above mentioned cards and not an overlapping of the bricks.

÷ The overlapping can be difficult, because you need a normal brick to support the structure. To insert the display-brick at the end can be an operation hard to do for very young children.

÷ To foresee a point to start and another to stop is a good expedient in the written language, but not for the spoken language. The younger children can have difficulties understanding this.

In short, the number of cons exceed the number of pros. This was confirmed by a speech therapist, who, instead of the proposed structure in figure 6.2, suggested the configuration shown in figure 6.3, which should be read from the top to the bottom. As implemented now, the linguistics scenario doesn't support the building of this kind of structure. However, the user is free to put the bricks on top of each other[2] thereby obtaining a tower structure as shown in chapter 4, figure 4.8, which is not far from the structure shown in figure 6.3. The child did seem to be interested in building sentences with the bricks, in spite of the earlier mentioned cons. The reader should recall that the testing done in Italy by Palma is very different from the testing done by the authors in Denmark. Firstly the tools

[2] Still in a grammatically correct fashion.

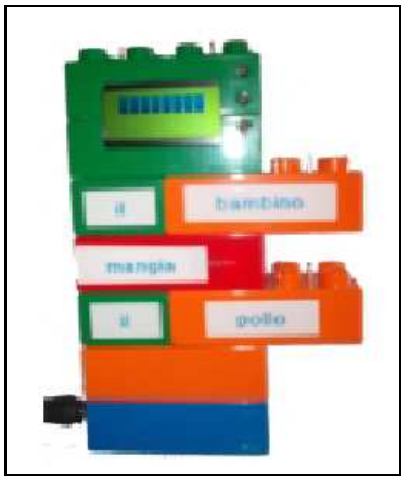


Figure 6.3: The configuration proposed by the speech therapist.

that the I-BLOCKs should be a substitute for are a textbook in Danish as opposed to linguistics cards. Secondly, the Danish pupils involved in the testing did not have any linguistic problems, at least not compared to the average elementary school pupil.

6.2 Summary

The testing conducted with I-BLOCKs in Siena, Italy has lead to the conclusion that the structure shown in figure 6.2 is inappropriate for children with linguistic problems. However, when consulting an expert (a speech therapist), another structure has been proposed, as shown in figure 6.3. This structure could, as explained by Palma in [4], be very useful for the reading task. Also, the size of the bricks made it easy for the child to build a structure. The child furthermore found it very amusing to build sentences with the I-BLOCKs, curiously awaiting the feedback given by the speaker brick.
The enhancement of the linguistics scenario to support the Italian language didn't cause any trouble whatsoever. As earlier noted, only a small redesign of the code was necessary.

Chapter 7

Discussion and Conclusion

> "It is better to debate a question without settling it than to settle a question without debating it."
>
> *- Jeseph Joubert*

THIS chapter discusses the relation between the theory used and the obtained results. Furthermore the discussion contains an evaluation about whether the results are as expected. Based on the discussion, the conclusion will present a summary of the most important results.

7.1 Discussion

This discussion will treat the essential subjects of the thesis in a chronological manner.

Analysis

The role of this chapter is to discuss the theoretical issues concerning the grammar used for this project. Basic elements of a sentence are introduced and formalised by a context-free grammar. Furthermore the class concept is explained, stating rules about the combination of articles and nouns. Also a couple of extensions are presented, namely the ability of asking questions and using adjectives. Of course, more advanced grammar, e.g. the use of prepositions, would be desirable. However, this would add to the complexity at the implementation level, and as such the authors decided to have a basic framework to develop from still having in mind the possibility of further extensions.

Software implementation

The authors have successfully implemented two prototypes, each offering some different functionality though centered around the implementation of basic grammar. Even though the outcome of the two implementations has been positive, the process of implementing them has been troublesome. Especially the brick connections have caused quite serious problems, as explained earlier. However, *when* the connections work the prototypes work as intended.
Though both centered around basic grammar, the two prototypes offer some different functionality, i.e. ability to place the LCD display at any given position on the structure as opposed to the ability of

asking questions and using adjectives. A future version of the linguistics scenario might contain both of these properties.
In future releases more functionality could be implemented, for instance the use of prepositions, thereby making it possible for the user to construct even more advanced sentence structures. This, however, requires more stability wrt. the connections, than is currently offered, due to the increased size of the structure.

School sessions

A lot of experience was gained from the iterative testing process. Each session has resulted in feedback from the pupils as well as the teacher. One of the issues not possible to improve at the moment, was the brick connections. This is due to the fact that improvements in this direction will require a redesign of the bricks. The most important improvement of the prototype during the iterative process, probably was the aural and visual feedback to the user, which increased the attractiveness of the bricks significantly. Another important observation during the sessions, was the ease, for the pupils, to understand the mapping from words to bricks. The pupils generally found it interesting working with the bricks. The most interesting statement made by the teacher was that of the I-BLOCKs making it easy for the pupils to experiment with sentence structures. However, the teacher also stated that the cost-benefit ratio regarding the use of I-BLOCKs for educational purposes probably will be too high.
It would be relevant to compare the results from the school sessions using the I-BLOCKs with a focus group *not* using the IBLOCKs, but another alternative tool. However the teachers opinion gives some idea as to the differences between ordinary methods (a textbook) and the I-BLOCKs.

Results from Siena, Italy

The results from Siena, Italy showed that the structure used for testing is inappropriate for children with linguistic problems. However, a speech therapist has proposed a different kind of structure, for this particular task, which also could useful for the reading task.

Recommendations

Based on the experience gained during this project, the authors have a number of recommendations for the future development of the I-BLOCKs. This involves hardware issues, as well as software issues.

Power connections: One of the earliest and greatest problems discovered, was the power connection on the bricks. This is confirmed by the sessions with the pupils and by Palma [4]. The connections on the bottom side of the brick is working acceptable or easy fixable if they have been bend during use. On the top side of the brick it's more difficult. The reason why is because of the realisation of the connections. The connection is realised by a thin metal wire on each of the studs. Hence the points of contact are very small. It is possible to adjust the connections by hand to make them work probably for a while. However after using the bricks several times the connections get inaccurate due to the fact that the metal wire gets strained and thereby is able to move out of position. A temporary solution could be to cover the studs on the top side with some kind of metal film or metal coating, to increase the point of contact. For future use, a redesign of the bricks would probably be necessary. This could be suggested for further studies, to achieve a better and more reliable solution.

The protocol: The protocol developed for the I-BLOCK communication can, as described by Jakob Nielsen in [3], "be considered reliable within reason". However, the data packages sent between the bricks only consist of an 8-bit data field, a split field, a 5-bit type field and a stop field. This is, as explained in [3], due to the fact that the amount of code being transmitted between each interrupt must be limited to ensure the entire data package being correctly received by the recipient. In future versions of the I-BLOCKs, it would be relevant to consider the opportunity of making a reliable protocol in the words true meaning. I.e. a protocol that offers handshaking and possibly some kind of data consistency check. As stated by Jakob Nielsen, this has not been possible to implement with the current hardware configuration. This fact encourages a hardware redesign of the I-BLOCKs wrt. the microcontroller used.

Object oriented model: As mentioned in chapter 2 I-BLOCKs can be thought of as objects. For future versions of I-BLOCKs it could be interesting to investigate the possibilities of letting I-BLOCKs behave exactly as such objects. A benefit gained would be, that new developers familiar with object oriented languages such as JAVA, C++ etc. could use their expertise. The authors are of the opinion that it would help I-BLOCK programmers in the future, if they were able to make one brick call methods on other bricks. Another fact supporting this approach is that object oriented languages already are in widespread use in the software development world.

Future uses of I-BLOCKs: The idea of solving language tasks using I-BLOCKs could be projected to other educational areas. In [3] Jakob Nielsen developed and tested an I-BLOCK structure supporting basic mathematics. Though this has never been tested with elementary school pupils, the authors presume that it could be tested in this environment obtaining the same degree of success as the linguistics scenario.
Another suggestion for the use of I-BLOCKs is that of studying physical properties of matter. An I-BLOCK could be programmed to contain the properties of a certain element[1], thereby making the user capable of constructing molecular structures.
A suggestion for college students could be to teach object oriented methods and languages using I-BLOCKs. However, this of course requires the above mentioned support for object orientation in the I-BLOCKs.

7.2 Conclusion

The objective of this thesis has been to explore the possibility of using a modular robotic entity, the I-BLOCK, for educational purposes. The I-BLOCK allows for programming by building and this project has been focused on the development of the linguistics scenario supporting this concept.
Only a subset of the Danish language has been used. However, this has been sufficient to make the user capable of constructing basic sentences and experimenting with these.
The implementation has been successful for both prototypes. Though containing differences they are centered around the same basic functionalities.
A lot of experiences has been gained during the four school sessions. Judging the pupils and the teachers answers from the questionnaires they liked the concept. The pupils quickly understood the mapping from ordinary sentences to I-BLOCK structures.
According to the results from Siena, Italy, the prototype is not suitable for children with linguistic

[1] From the periodic table of elements.

problems, however another structure has been proposed by a speech therapist. This structure could be considered in future prototypes.
The implementation of the linguistic scenario can't be considered as a final product; a number of problems are still to be solved. However, the testing conducted using the I-BLOCKs has lead to some interesting results, hopefully encouraging further investigation in the field of Educational Robotics.

Appendix A

Questionnaires

A.1 Questionnaire - pupils

The questions below are addressed to *the pupils* and are presented in random order.

1. Has it been difficult to use the bricks?
   - Yes □
   - No □
2. Have the bricks made it easier or more difficult to understand Danish?
   - Easier □
   - A little easier □
   - The same □
   - A little more difficult □
   - More Difficult □
3. What do you think about the bricks shape/-connections? Is it to difficult to put them together?
   - Yes □
   - No □
4. Has it been difficult to work with the other pupils to solve the problems with the bricks?
   - Yes □
   - No □
5. What do you think about the way sentences are build with the bricks?
   - Good □
   - Bad □
6. Has it been fun working with the bricks?
   - Yes □
   - No □
7. Would you like I-Bricks to be used in the education?
   - Yes □
   - No □
     Why not?

     _____
     _____
     _____
8. Do you have any suggestions how to make the bricks better?

   _____

9. Write down three good things about I-Bricks.

10. Write down three bad things about I-Bricks.

11. Do you have any suggestions for other problem that can be solved with I-Bricks

A.2 Questionnaire - teacher

The questions below are addressed to *the teacher* and are presented in random order.

1. How do you rate the pupils' understanding concerning the use of I-Bricks? Is it easy/hard for them to understand the concept?
   - Easy □
   - Medium □
   - Hard □
2. Have the pupils in general obtained confidence with the bricks?
   - Yes □
   - No □
3. Have the bricks made it easier/harder for the pupils to understand rules about grammar?
   - Easier □
   - A little easier □
   - No change □
   - A little harder □
   - Harder □
4. What do you think about the shape and connections of the bricks? Is it hard for the pupils to put them together?
   - Yes □
     Why?
   - No □
5. Have the pupils been good at cooperating in trying to solve the appointed tasks with the bricks?
   - Yes □
   - No □
     Why not?
6. What do you think about the way in which sentences are being constructed using the bricks?
   - Good idea □
   - Bad idea □
     Why?

7. Did the pupils have fun while working with the bricks?

   - Yes □
   - No □
     Why not?

8. Do you believe that I-bricks is a tool which might be used for educational purposes in the future?

   - Yes □
   - No □
     Why not?

9. Do you have any suggestions as to improve the I-Bricks concept? Please write below.

10. Please mention as many as three good things about the I-Bricks concept

11. Please mention as many as three bad things about the I-Bricks concept

12. Are the pupils at this level too young/old to use the bricks?

    - Yes, too young □
    - Yes, too old □
    - Appropriate age □

13. If you answered "Yes" in the previous question, what educational level would be better?

    - 1. grade □
    - 2. grade □
    - 3. grade □
    - 4. grade □
    - 5. grade □
    - 6. grade □

14. Can you suggest any other exercises for the pupils to solve with the I-Bricks? Please write below.

Appendix B

The Living Tree

This appendix contains the report about The Living Tree [7].

The Living Tree

Maersk Mc-Kinney Moller Institute for Production Technology, Denmark
Jacob Nielsen, raider@mip.sdu.dk
Steffen Jensen, mrhunt@mip.sdu.dk
Mikkel Meister Pedersen, ronnie99@mip.sdu.dk

Abstract:

This document describes the programming and testing of The Living Tree, *a simple prototype constructed with I-bricks to simulate the concept of an artificial tree constructed with I-blocks.*

Specification

The Living Tree is a specification of the "art in park" scenario. It is an installation that can be configured by the user. The tree is able to react to and change the environment, in which it is situated. The behaviour of the tree is decided by wind, temperature and light conditions. The specific behaviour of each parameter is:

1. Wind – the tree moves randomly according to wind.
2. Temperature – the tree changes colour according to temperature.
3. Light – the tree moves its light sensors towards the light source.

Bricks used

Push-button brick:

The wind moving the tree is activated by the push-button brick. Pressing a button on this kind of brick causes the tree to move in a random manner.

Turn-button brick:

Activating this brick simulates the temperature changing the colour of the tree. When the user turns the button, the LED-brick changes its light sequence accordingly.

LED-brick (CPU):

The temperature changing the colour of the tree is simulated by a LED-brick. This brick is equipped with eight light emitting diodes, which can be activated by a turn-button brick.

LDR-brick:

The movement of the tree towards the light source is simulated by the use of LDR-bricks, which are light sensitive.

Top-turned motor brick (Servo):

All movement in the tree is done by motor-bricks, which are able to turn the top part.

Battery brick:

The power supply for the entire system.

Because of the limitations of the battery, this brick can power no more than 3 servo bricks.

Software design

The concept of the software design is illustrated in figure 1. This figure shows the relationship between the bricks and how they are able to communicate.

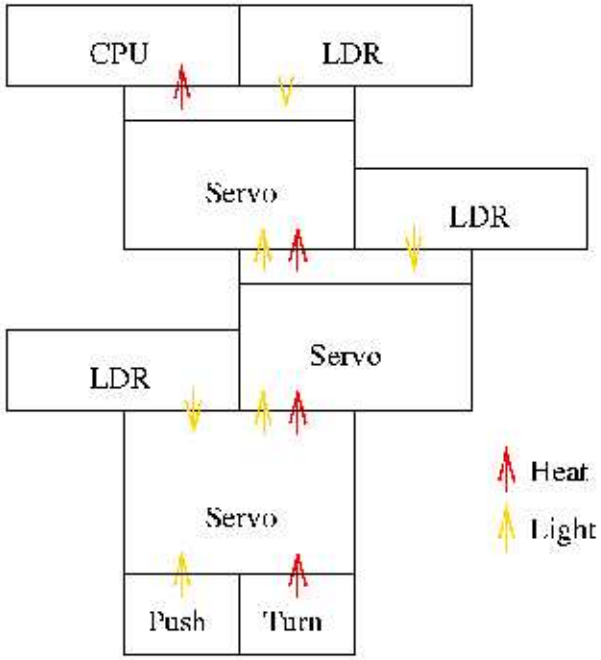


Figure 1: Schematic view of internal communication in The Living Tree

As can be seen from the figure, the push-button is transmitting "light" data types in spite of the fact that the push-button should simulate the wind behaviour. This is done to ease the programming, as this solution uses an already defined data type. Since this is a simple prototype, and the end product should move according to *real* wind, this is an acceptable solution.

Heat values (red arrows) are transmitted directly through the motor-bricks, whereas the light values (yellow arrows) are OR'ed together (bitwise OR) before being transmitted.

Figure 1 shows that the CPU-brick only receives heat values. Actually it also receives light values, but these are ignored.

When connecting the bricks the user has to make sure that all sensor bricks are placed directly on the motor bricks. This is due to the fact, that sensor bricks only transmit output values and are not receiving any values from other bricks.

Testing and re-design

In the process of developing The Living Tree, we have used an iterative approach. This means that every step of the programming has been tested before proceeding to the next step. This way we discovered the disability of the bricks to handle more than one data type transmitted from one brick. The solution to this problem was to introduce specific data types for light and heat values.
The data flow in the tree is from the bottom and up. The reason for this is that sensors in the lower part of the tree should be able to activate the upper part of the tree, but not the other way around. Because of this data flow the turn-button at the moment has to be placed at the same level as or lower than the LED-brick.

End prototype

With the developed prototype it is possible to construct different structures, still obtaining the behaviour of a living tree, under the constraints previously described. By constructing the tree in the right fashion it is possible to obtain the behaviour of a sunflower, i.e. movement towards a light source. It is however also possible to obtain several different behaviours or movements.
Figure 2 and 3 show one possible configuration from different views.

Figure 2: The Living Tree, back view

Figure 3: The Living Tree, front view

It should be noted that this is just one of the possible configurations. Numerous different configurations (structures and behaviour) were built and tested.

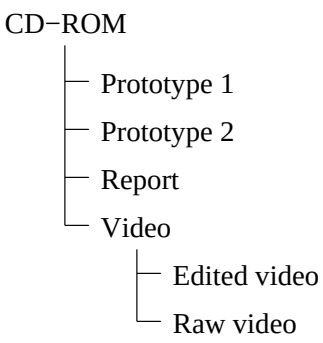
Conclusion

We conclude that it is possible to build simple tree-like structures of different kinds, which are able to simulate the behaviours of a real tree, or at least fulfil the specification of The Living Tree. Furthermore we conclude that there are limitations concerning the power supply and the communication between the bricks, which hopefully are to be solved in the future.
The Living Tree could be made more interactive by adding sound sensors and speech recognition/synthesis devices, making people able to communicate with the tree and thereby justifying the concept of a *living* tree.

Appendix C

CD-ROM Contents

The folders Prototype 1 and Prototype 2 contain the software implementations of the respective prototypes as MPLAB projects. The Video folder contains two subfolders, Edited video and Raw video. All video files are in .wmv format.



Appendix D

Author Indication

	Steffen Jensen	Mikkel Meister Pedersen
Chapter 1 - Preface	✓	✓
Chapter 2 - Introduction		✓
Chapter 3 - Analysis		✓
Chapter 4 - Software Implementation p11-13		✓
Chapter 4 - Software Implementation p13-19	✓	
Chapter 4 - Software Implementation p19-25		✓
Chapter 5 - School Sessions p27-28	✓	✓
Chapter 5 - School Sessions p28-33	✓	
Chapter 5 - School Sessions p33-36	✓	✓
Chapter 6 - Results from Siena, Italy		✓
Chapter 7 - Discussion and Conclusion	✓	✓
Appendix A - Questionnaires	✓	✓

Bibliography

[1] H. H. Lund
Intelligent Artefacts
In Sugisaka and Tanaka (eds.) Proceedings of 8th International Symposium on Artificial Life and Robotics.
ISAROB, Oita 2003.

[2] Custom Computer Services Inc.
C Compiler Reference Manual
2002

[3] Jakob Nielsen
Intelligent Bricks
Masters thesis, The Maersk McKinney-Moller Institute for Production Technology
2001

[4] Valentina Palma
I-Brick e ritardo del linguaggio
Masters thesis, University of Siena, Italy
2003

[5] Lars Mathiassen, Andreas Munk-Madsen, Peter Axel Nielsen, Jan Stage
Objekt Orienteret Analyse og Design
Marko
2001 3rd edition (in Danish)
ISBN: *87-7751-153-0*

[6] Harry R. Lewis, Christos H. Papadimitriou
Elements of The Theory of Computation
Prentice Hall
1998 2nd edition
ISBN: *0-13-262478-8*

[7] Jakob Nielsen, Steffen Jensen, Mikkel Meister Pedersen
The Living Tree
The Maersk McKinney-Moller Institute for Production Technology
2003