RATH

A Relational Adaptive Tutoring Hypertext WWW–Environment[1]

Cord Hockemeyer[2]

Department of Psychology
University of Graz, Austria
E–Mail: Cord.Hockemeyer@kfunigraz.ac.at

November 23, 1997

[1]The RATH prototype documented in this report is one result of the research project "A System for Developing Hypermedia–Based Teaching Components of Intelligent Tutoring Systems" supported by General Direction XII at the European Commission (Grant ERBCHBICT941599, HCM Research Fellowship) and by the University of Graz, Austria

[2]In cooperation with Dietrich Albert, E–Mail: Dietrich.Albert@kfunigraz.ac.at

The software system described in this manual can be accessed via the World Wide Web at the address `http://wundt.kfunigraz.ac.at/rath/` . For questions about `RATH` send an email to `rath@wundt.kfunigraz.ac.at` .

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This documentation describes RATH[1], a prototype for a **R**elational **A**daptive **T**utoring **H**ypertext WWW–environment. This program prototype was developed within the framework of a HCM research fellowship of the European Commission, DG XII (Grant ERBHCBICT941599) to Theo Held (University of Heidelberg; now University of Halle–Wittenberg) and Cord Hockemeyer (Technical University of Braunschweig) under the auspices of Dietrich Albert (University of Graz).

RATH's theoretical foundation lies in a relational model of hypertext structures (Albert, Hockemeyer and Held, 1997) and its combination with the theory of knowledge space theory (Albert and Hockemeyer, 1997) which have been supported by the same grant of the European Commission. Since this text is primarily a technical documentation the underlying theoretical basis is not explained in detail (this has been described sufficiently deeply by Albert, Hockemeyer, and Held, 1997). We will only comment on the extent of realizing the theoretical concepts into practise with this prototype in Appendix A.

Throughout this documentation, familiarity with the tool used is assumed, i. e. there are no introductions into topics like HTML, Oracle, or UNIX system administration.

1.1 Acknowledgements

The following persons are appreciated for their contributions: Dr. Theo Held who was the predecessor of the fellowship (Held and Albert, 1995) and Gerhard Hermann for preparing parts of the teaching material, and Rainer Kaluscha for introducing and supporting working with the Oracle database system.

[1] This acronym is an old German word, today written as *Rat*, with the connotation of taking care and giving advice (for details see Grimms Deutsches Wörterbuch, Grimm and Grimm, 1893).

Especially, the author and his cooperator, Dietrich Albert, are full of gratitude for the support of this research by the General Direction XII at the European Commission.

1.2 Structure of the documentation

The following list illustrates the structure of the complete documentation:

Chapter 1: Introduction. A short overview is given of the scientific environment of RATH, and of the structure of the documentation.

Chapter 2: Architecture of RATH. An overview of the components of RATH, of processes, and of the flow of information is given.

Chapter 3: Using RATH This chapter is the main documentation for pupils (or students)[2] who want to use RATH for learning.

Chapter 4: Administrating Courses in RATH This chapters covers the administration of RATH on a course level, i. e. special functions available to the teacher or other persons leading and supervising a course.

Chapter 5: Installing RATH The installation of RATH includes the creation of a pseudo user, and configuration changes for the `httpd` and for Oracle.

Chapter 6: Authoring Courses for RATH Lessons in RATH are written in an extended HTML version called $HTML_{\varepsilon}$. This chapter gives a short introduction into $HTML_{\varepsilon}$ focusing on the differences to standard HTML.

Appendix A: RATH and the Theoretical Model Some aspects of the underlying theoretical model could not be realized directly. Thus, it may seem that there are differences between model and implementation. These "differences" are illustrated and explained.

Appendix B: Structure of the Database Tables The structure of the tables are explained by an annotated listing of the `create table` and `create sequence` SQL commands

Appendix C: List of Programs The programs building RATH are listed with annotations.

Appendix D: Structure of Teaching Material Based on the work of Held (1993) we developed the prerequisite structure for a small stochastic course.

[2] Throughout the documentation, we will simply use the term "pupil". However, this includes all kind of learning users of the system.

Appendix E: Illustrative Teaching Material This appendix contains the printed version of the stochastic course.

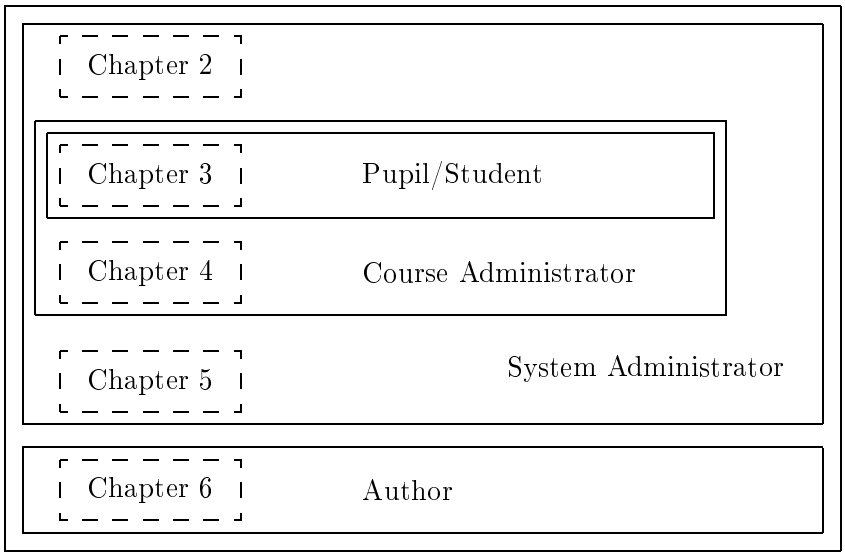

Figure 1.1: Reading hierarchy for different kinds of users

As a consequence, we obtain the reading hierarchy of the chapters 2–6 illustrated in Figure 1.1 as a recommendation for the different kinds of users of the system and readers of this documentation.

Chapter 2

Architecture of RATH

In this chapter, general design ideas in the development of RATH are presented. The knowledge of these ideas will be useful for understanding the other chapters although at least the usage documentation (Chapter 3) is sufficient as a stand alone documentation.

2.1 General architecture

2.1.1 Technical description of the architecture

The basic structure of the system's architecture is shown in Figure 2.1. The *complete hypertext* is a collection of hypertext components written in the *Hypertext Markup Language* (HTML). In a first step we extract structural information from these components and store them in a database as it is described by Albert et al. (1997). Additionally, we may select a sub hypertext, i. e. a subset of the components and of the links stored in the database. This is both done using offline[1] (relational) database functions. The sub hypertext and database obtained in this way are then used in the dialog with the user.

The user communicates via his/her WWW browser with the HTTPD (Hypertext transfer daemon, i. e. the WWW server). The HTTPD uses CGI programs to build user adapted components by modifying components from the sub hypertext according to internally stored user data. These CGI programs use relational database functions. The user adapted component is transferred via the HTTPD to the user's WWW browser and presented to the user.

This use of HTML has two major advantages: The first advantage is an easy and portable way to combine hypertext and relational databases. The second advantage is that we can use HTML documents even without

[1] The term *offline* here means that the sub system is not selected when the user wants to use it but it is selected once before and then constitutes an autonomous system. Correspondingly, we use the term *online* to denote that a certain component is selected and adapted to data stored about the user right in the moment it is requested by the user.

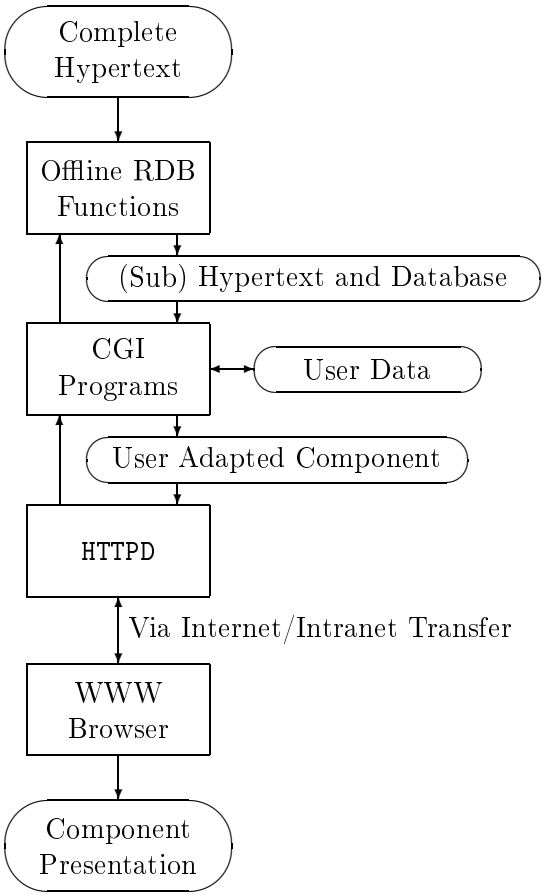

Figure 2.1: Architecture of RATH

an HTTPD and the additional programs offering the database functions. We just get a normal WWW hypertext. Therefore, a hypertext can be used even if our additional programs are not available (e. g. because of portability problems). The user misses some functionality but the hypertext is still usable.

The considerations described above lead to a system as it is shown in Figure 2.2 which contains only the WWW Browser selecting the components to be presented a) directly from the complete hypertext or b) from an earlier selected sub hypertext and presenting it to the user.

2.1.2 Application oriented description of the architecture

In the previous section, we focused on the technical aspects of the system's architecture. In this section, we look at the architecture with special respect

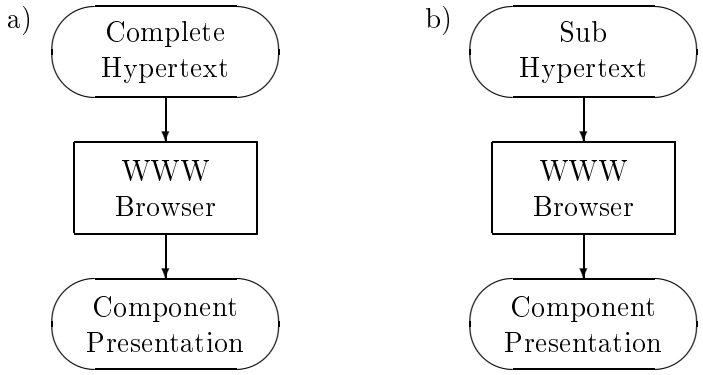

Figure 2.2: Using a) a complete hypertext or b) a preselected sub hypertext without a HTTPD

to intelligent tutoring systems (ITS). We illustrate this application oriented view to the system's architecture in Figure 2.3 which has the same structure as the technical illustration in Figure 2.1. We start with a hypermedia–based intelligent tutoring system which corresponds to the complete hypertext in the general technical approach. We have some offline functions for the selection of ITS sub systems. In this approach, it is realized by relational database functions. From this ITS sub system, certain components, e. g. lessons or exercises, are selected and adapted according to data stored about the pupil's knowledge and preferences. This selection and adaptation of single documents is done online by CGI programs. The adapted component is sent via the HTTPD to the pupil's WWW browser which presents the lesson, exercise, or test to the pupil. In case of exercises or tests, answers given by the pupil are transferred by the WWW browser via the HTTPD back to the CGI programs which evaluates the answers and updates the pupil database accordingly.

2.2 Security issues and user identification

There are two reasons for dealing with security issues: The most important point is the fact that, in an ITS, data about pupils' performances have to be stored. These must be treated as confidential. Another reason for dealing with security issues is the fact that RATH uses the World Wide Web (WWW) and, therefore, could be accessed by anyone. During the test phase, however, access shall be restricted to selected persons or user groups. Also afterwards, it may be reasonable to restrict the access, e. g. to paying users.

RATH administers two user groups: pupils and administrators. Currently, pupils may use the ITS itself while administrators may add or delete pupils. Additionally, they will also be able to get information about pupils' learning

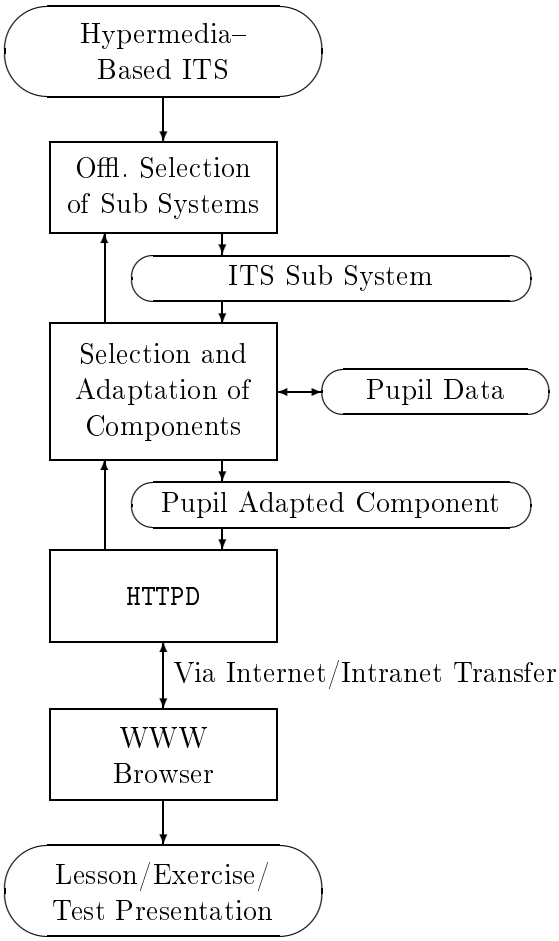

Figure 2.3: Architecture of a hypermedia–based intelligent tutoring system

progress. Thus, the administrator is identical to the pupil's teacher (or supervisor).

Administrators' and pupils' passwords are stored in different files. The administrators' password file can currently only be changed by the system (or WWW) administrator. Administrators can change their own password and they can add new pupils, delete existing pupils or change pupils' passwords. Pupils, on the other hand, can just access the ITS. Note that administrators can not directly access the ITS but need an additional entry in the pupil password file.

!

Warning:
Note that the mechanisms used in RATH are by no means a barrier against serious trials to break into the system. All authentication requests are done via the *Basic*

authentication Scheme of the HTTP–protocol (**RFC 2068, 1997**). The weakness of this procedure lies in the fact that user identity and password are transferred encoded, but not encrypted with each document request.

Future versions may use other procedures to reach a higher level of security, e. g. the *Digest Access Authentication* (**RFC 2069, 1997**). Currently, however, this is hardly supported by existing systems.

Chapter 3

Using RATH

Using RATH basically involves using a WWW browser, a task whose documentation is outside the scope of this manual. Currently, there is only one point to be mentioned which is due to the use of LaTeX2HTML in producing the online version of this documentation:,LaTeX2HTMLcan produce a navigation panel on top (and on long pages also on bottom) of each page. The navigation panels in this documentation contain the following buttons:

Previous Group: This button selects the previous section on the same (or higher) sectioning level.

Previous: This button selects the previous section independent of its sectioning level.

Up: Regarding the complete document as a tree structure, this button selects the parent node of the actual page.

Next: This button selects the next section independent of its sectioning level.

Next Group: This button selects the next section on the same (or higher) sectioning level.

Contents: This button selects the table of contents.

The document itself is segmented into several WWW pages by its logical structure: each section becomes an own WWW page containing the contents of the actual section (without subsections) and links to its subsections.

Chapter 4

Administrating courses in RATH

Administering courses in RATH contains two kinds of tasks: user administration in a technical sense, i. e. creation and deletion of new users (pupils), and administration in a pedagogical sense, i. e. supervising the pupils and their learning progress.

In order to become an administrator, an id number and password has to be requested from mailto:Cord.Hockemeyer@kfunigraz.ac.at or mailto:Dietrich.Albert@kfunigraz.ac.at .

An administrator has to identify him-/herself when he/she enters the administration section of RATH. Afterwards, no further identification is requested. The following menu offers four points to choose from: Showing a pupil's knowledge, adding a new pupil, changing a pupil's password, and deleting a pupil. All four possibilities present a form to the administrator.

4.1 Progress report

This function has only been rudimentarily implemented. Currently, after requesting the pupil's account name (see Figure 4.1), a report page (see Figure 4.2) is shown which simply consists of a direct Oracle output containing the titles of the lessons read and exercises solved correctly by the pupil.

Figure 4.1: WWW form for querying pupils' learning progress

Figure 4.2: Example progress report for the pupil 'test'

4.2 Adding a pupil

For adding a new pupil, the administrator has to fill out the form shown in Figure 4.3. The username is the name of the account, the real name is – according to UNIX traditions — denoted as comment in the second field. The last two fields contain the pupil's password which has to be entered twice to exclude typing errors.

Figure 4.3: WWW form for creating a new pupil

4.3 Changing a pupil's password

The form for changing a pupil's password is quite similar to that for adding a new pupil (see Figure 4.4), only the field `Comment/Real Name` has been omitted.

4.4 Deleting a pupil

The form for deleting a user only requests the pupil's account name.

Figure 4.4: WWW form for changing a pupil's password

4.5 Notification

Please note that in the three cases of user creation, password change, and user deletion a mail is sent to the central `RATH` administrator (the system's user with name 'RATH').

An example for such a notification mail is shown below.

```
From rath Tue Sep 23 17:00 MET 1997
Return-Path: <rath>
Received: by wundt.kfunigraz.ac.at (SMI-8.6/SMI-SVR4)
        id RAA20015; Tue, 23 Sep 1997 17:00:36 +0100
Date: Tue, 23 Sep 1997 17:00:36 +0100
From: rath (Relational Adaptive Tutoring Hypertext)
Message-Id: <199709231600.RAA20015@wundt.kfunigraz.ac.at>
To: RATH
Subject: Successful registration
Content-Type: text
Content-Length: 117

Successfully registered new RATH user

UserId: dietrich
Real Name: Dietrich Albert
Registered by: cord
Seems to be:
```

Chapter 5

Installing RATH

The installation of RATH can be divided into a number of subtasks:

- Creating a pseudo user `rath` and unpacking the RATH software in the home directory of the new user `rath`
- Configuring the httpd program (WWW server)
- Configuring Oracle
- Registering and analyzing the HTML files of a course in the database

In the following sections, the first three subtasks are described in more detail. It is assumed that you have system administrator privileges and some knowledge and experience with the compilation and installation of new software. The registry of documents is described in Section 6.4.

5.1 Unpacking and compiling the software

The software is packed in the `sources.tgz` file. After creating a user `rath` the software sources are unpacked with the commands

```
% gunzip sources.tgz
% tar -xf sources.tar
```

Compare the `makefile` and `rath-sql.h` with your system and perform changes where necessary (see also Section 5.3) and

```
% make all
```

5.2 Configuring the httpd

This documentation is written for the CERN httpd which was used in the development of the prototype. Although, for other web servers, some parts of the documentation may have to be adapted, the configuration options specified below should be applicable — probably with another syntax — to other server software easily.

5.2.1 Using the Common Gateway Interface (CGI) to connect the httpd and the database system

5.2.2 Configuring the httpd

Figure 5.1 shows the CGI configuration part of the `httpd.conf` file. These

```
Exec    /RATH/*                 /export/home/RATH/cgi/*
Exec    /RATHadmin/*            /export/home/RATH/cgi/*
```

Figure 5.1: Execution rules for CGI scripts

two lines specify that all WWW requests within the `/RATH/` or the `/RATHadmin/` directory trees should be redirected to the corresponding file (in a subdirectory, if appropriate) within the `/export/home/RATH/cgi/` directory. At this point it may seem unreasonable to define two different (virtual) directory trees and to map them to the same real directory but that is explained in Section 5.2.3 below.

5.2.3 Security, user identification, and administration

In our prototype, we use the CERN httpd as WWW server program. This has, of course, some influence on the realization of the access control and user identification methods. However, at least the NCSA httpd and the new Jigsaw WWW server from the W3C (WWW Consortium) have at least similar mechanisms for access control.

5.2.4 Access authentication in HTTP

In the following, we give a short introduction into the *challenge–response* authentication mechanism provided by HTTP (RFC 2068, 1997, section 11). Note that some familiarity with the concepts of TCP/IP in general and especially with those of HTTP is assumed.

If a client (i. e. WWW browser, also called user agent) sends a request for a address with access restrictions to a server, this server will reply with a 401 (Unauthorized) response message. This message is not a definite rejection but a *challenge* to the client which is necessarily combined with the

specification of a realm for which authentication is needed and with the authentication scheme. The realm value, in combination with the root URL[1], defines the *protection space*. The client may resend its request adding an authentification header field providing the credentials necessary for accessing addresses in the specified protection space. The authentication scheme given in the server's 401 response message specifies the type and encoding of the credentials. Using the *Basic* authentication scheme which is part of the HTTP protocol, e. g., the user-id and password, separated by a colon (":"), are sent within a base64 encoded string in the credentials.

> **Warning:**
> **Since base64 is only an encoding but no encryption, this means that user-id and password are sent through the net practically open readable (see also Section 2.2).**

!

5.2.5 Authentication in RATH

Despite its lacks, we use the Basic authentication scheme of HTTP for identification of pupils and administrators. Currently, we have one password file for administrators and another password file for pupils. The administrators' part of RATH provides WWW forms for changing the pupils' password file and for changing administrators' passwords.

The access restrictions themselves have to be specified in the httpd configuration file. Figure 5.2 shows the protection of the pupils' parts while

```
Protect /RATH/* {
  AuthType Basic
  ServerID RATH
  PasswordFile /export/home/RATH/etc/passwd
  GroupFile /export/home/RATH/etc/group
  UserID nobody
  GroupId nogroup
  GetMask All
  PostMask All
}
```

Figure 5.2: Protection rules for the pupils' parts of RATH

Figure 5.3 shows the protection of the administrators' parts. Note that according to Section 5.2.1 the directories /RATH/ and /RATHadmin/ actually do not contain documents but CGI scripts.

However, these different protections for the virtual directory trees /RATH/ and /RATHadmin/ explain why the CGI scripts are made accessible in two

[1] Basically, the root URL contains the access protocol and the hostname. Regarding the URL http://wundt.kfunigraz.ac.at/hockemeyer/Welcome.html, for example, the corresponding root URL is http://wundt.kfunigraz.ac.at.

```
Protect /RATHAdmin/* {
  AuthType Basic
  ServerID RATH-Admin
  PasswordFile /export/home/RATH/etc/passwd.admin
  GroupFile /export/home/RATH/etc/group.admin
  UserId RATH
  GroupId www
  GetMask All
  PostMask All
}

Protect /rath/Admin/* {
  AuthType Basic
  ServerID RATH-Admin
  PasswordFile /export/home/RATH/etc/passwd.admin
  GroupFile /export/home/RATH/etc/group.admin
  UserId RATH
  GroupId www
  GetMask All
  PostMask All
}
```

Figure 5.3: Protection rules for the administrators' parts of RATH

different directory trees: One script, e. g. for changing one's own password, may thus be used by pupils as well as by administrators — the password file to be changed can be selected by the CGI script in correspondence to the directory in which is was called.

5.3 Configuring Oracle

The configuration of Oracle basically consists of creating a new database user `rath`. This user must also have the privilege to create new tables (`grant resource`). We recommend to use the operating system's (`EXTERNALLY`) identification. If you prefer using explicit identification by username and (Oracle) password, the include file `rath-sql.h` in the `~rath/src` directory has to be changed accordingly before compiling the programs. Note, however, that, as a consequence, username *and* password will be hard–coded in the programs.

Additionally, the definition of the environment variables `ORACLE_SID` in the `~rath/oracle/.rath` file specifying the Oracle instance used has to be changed according to their values in the creation of the Oracle user rath.

Finally, you have to execute the `init-db` script in the `$RATH/bin` directory to create some tables.

Chapter 6

Authoring courses for RATH

Courses in RATH are built up by a network of linked hypertext documents. In the development of RATH, we decided to use HTML version 4.0. This version is not yet officially released but it has — in comparison to the older version 3.2 — the advantage that anchor and link types can be realized directly with the means of the language. With HTML 3.2, an extension of the language by own constructions would have been necessary.

We will not give an introduction into the authoring in HTML here — there exists enough literature about that. Instead we will concentrate on the way we realize the new concepts.

In Section 6.1, the means for bringing the concepts of the hypertext model into HTML are briefly described. This is briefly illustrated in an example text in Section 6.2. Finally, we comment in Section 6.3 on writing HTML texts with formal contents.

6.1 Realizing RATH concepts in HTML

6.1.1 Anchor properties

Introducing anchor (and, as a consequence, implicitly also link) properties we had primarily the specification of the relation between the linked documents in our mind. For this purpose, HTML provides the REL and REV attributes for the link and A (anchor) tags.

Currently, RATH only interprets source anchors having the REV attribute set to the value Prerequisite. The meaning of this attribute is that the destination of the link is a prerequisite for the current document. !

Note, however, that links with absolute destination addresses (e. g. href="http://..." or href="/...") or with destination addresses outside the directory tree of the current course (see below, Section 6.4) are not registered in the database nor evaluated by the user adaptive filter program.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<TITLE>Concluding Page</TITLE>
<!-- Please note the following prerequisite link! -->
<LINK REV="Prerequisite" HREF="node36.html">
</HEAD>
<BODY >
<h1>The End - Certification</h1>
This page could contain a possibility to print out a
certification form for the pupil who has completed the course.
<p>
<BR><HR>
<ADDRESS>
<I><a href="mailto:RATH@wundt.kfunigraz.ac.at">RATH</a>
(Relational Adaptive Tutoring Hypertext)<br></I>
</ADDRESS>
</BODY>
</HTML>
```

Figure 6.1: Example document for RATH

6.1.2 Presentation information

Currently, the specification of information how documents or links should be presented is not yet supported. In is planned, however, that document presentation information can be specified using the META tag of HTML. Information about link presentations, on the other side, may be specified using the CLASS attribute of the A (i. e. anchor) tag.

6.2 An example for a RATH document

Figure 6.1 contains an example HTML document. Currently, the main point is the LINK tag which says that the file node36.html is a prerequisite for the current file. The rest of the file is standard HTML but it may also serve as a suggestion for HTML writing style.

6.3 Writing HTML texts with formal contents

Bringing formal contents into WWW pages is an unsatisfactory job. With HTML version 3.1, extensions of the HTML language for mathematical symbols and formulas were introduced. However, these concepts were not accepted by the mainly used browsers (the only browser understanding the math extensions of HTML 3.1 was the arena browser of the WWW Consortium (W3C). Since the W3C has switched from Arena to Amaya, no browser supporting

the HTML 3.1 mathematical extensions is available at the time of finishing the system.

Currently, mathematical contents are generally produced using LaTeX2HTML, a tool that converts a LaTeX (or LaTeX 2ε) document into a number of HTML documents. Graphics and formulas are typeset using LaTeX and then converted into GIF pictures which are included as inline images into the HTML documents. This approach has also been used for producing the online version of the technical documentation for the RATH prototype and also for the teaching contents of the stochastic course.

Using LaTeX2HTML, however, has some disadvantages: First, the use of inline images induces higher use of processing time and also of bandwidth if the system is used via a network. Second, the quality of the images and their presentation within the browser is not always satisfactory. Besides that, it often does not fit to user specific font size selections in the browser. Finally, a third point comes up in combination with the additional links and link information used in RATH: These information have to be added manually after the conversion. These manual additions are, of course, lost after updates of the original sources and their re–conversion. Hopefully, these problems can be solved by new developments in the World Wide Web technology, e. g. described next.

One new approach is the definition of XML (XML, 1997). XML is a subset of SGML (ISO 8879, 1986); both languages are used to define markup languages. Two XML applications are MathML (MATHML, 1997) for mathematical markups and CML (see `http://www.venus.co.uk/omf/cml/`) for chemical markups. Depending on their success, we are considering to use XML and MathML in a later version of RATH.

6.4 Registering HTML documents in RATH

Currently, it is only possible to register a complete course at once. This means that all prior knowledge about pupils' performances is lost. This weakness will be solved in the future.

It is assumed that the course to be registered builds one directory tree. This means that (i) all documents belonging to the course are within this directory tree and (ii) all documents within this directory tree are supposed to belong to the course.

The registering itself is done with the program `generate-db` in the `$RATH/bin` directory. This programs accepts two parameters: source and destination directory. Default values are `$RATH/RATH` and `$RATH/registered` , respectively. For these directories, absolute path names must be specified. The program scans each file in the source directory (including subdirectories) and writes an enriched version (see Appendix C into the destination directory preserving the directory structure. The scan program also stores information about

!

the document and its anchors in the database.

Bibliography

Albert, D. & Hockemeyer, C. (1997). Adaptive and dynamic hypertext tutoring systems based on knowledge space theory. In B. du Boulay & R. Mizoguchi (Eds.), *Artificial Intelligence in Education: Knowledge and Media in Learning Systems* (pp. 553–555). Amsterdam: IOS Press.

Albert, D., Hockemeyer, C., & Held, T. (1997). *A relational model for hypertext structures*. Unpublished Manuscript.

Elmasri, R. & Navathe, S. B. (1989). *Fundamentals of database systems*. Addison–Wesley World Student Series. Redwood City, CA: The Benjamin/Cummings Publ.

Falmagne, J.-C. (1993). Stochastic learning paths in a knowledge structure. *Journal of Mathematical Psychology*, *37*, 489–512.

Grimm, J. & Grimm, W. (1893). *Deutsches Wörterbuch*, (Vol. 8). Leipzig, Germany: S. Hirzel.

Held, T. (1993). *Establishment and empirical validation of problem structures based on domain specific skills and textual properties.* Dissertation, Universität Heidelberg, Germany.

Held, T. & Albert, D. (1995). *A system for developing hypermedia–based teaching components of intelligent tutoring systems* . (Internal report), Karl–Franzens–Universität Graz, Austria.

ISO 8879 (1986). *International standard ISO 8879–1986. information processing, text and office systems, standard generalized markup language (SGML) = Traitement de l'information, systemes bureautiques, langage standard généralisé de balisage (SGML)* (1 ed.). International Organization for Standardization (ISO), Geneva, Switzerland.

MATHML (1997). *Mathematical markup language*. (W3C Working Draft WD-math-970794), World Wide Web Consortium (W3C). Online available at `http://www.w3.org/TR/WD-math-970704` .

RFC 2068 (1997). *Hypertext transfer protocol — HTTP/1.1*. (RFC 2068), Network Working Group.

RFC 2069 (1997). *An extension to HTTP: Digest access authentication* . (RFC 2069), Network Working Group.

XML (1997). *Extensible markup language (XML)* . (W3C Working Draft WD-xml-970807), World Wide Web Consortium (W3C). Online available at `http://www.w3.org/TR/WD-xml-970807` .

Appendix A

Remarks on the correspondence between the theoretical hypertext model and the practical implementation of RATH

There exist a few differences between the theoretical papers (Albert and Hockemeyer, 1997; Albert, Hockemeyer, and Held, 1997) and the practical implementation of RATH.

One feature that has not been realized in this prototype is the specification of presentation information for documents and for links. The evaluation of these information has been omitted in this version to keep the system simple. Note, however, that the specification of document presentation information is possible by the use of style sheets, a means that makes the storage of presentation information in the database superfluous.

In addition to that, the structure of the database tables has been modified slightly in a process of normalization (see, e. g., (Elmasri and Navathe, 1989) for details on that topic). The complete table structure (including user data) is shown in Appendix B.1.

Appendix B

Structure of the database tables

This chapter illustrates the structure of the database tables in the RATH system. Primarily this is done by showing the SQL creation commands for tables and sequences. Sequences in SQL are counters which can be used and incremented within one command. This feature ensures that each time a new value is selected, a crucial feature of identification numbers.

In Section B.1, the commands for the creation of all the tables are shown. In Section B.2 then the commands for creating the sequences are presented.

B.1 Tables

```
create table doc(
  did        number unique,
  filename   varchar(256) unique,
  title      varchar(1024)
);
```

The doc table consists of three columns: the document id, the file name, and the title of the document. The first two columns are defined as unique, i. e. each row must have a different value in this column.

```
create table source(
  sid        number unique,
  did        number
);

create table dest(
  aid        number unique,
```

```
  did         number,
  name        varchar(128)
);
```

These tables contain the source and destination anchors, respectively. We store the anchor id (sid or aid, respectively) and the document id. For destination anchors, we additionally store the name under which they can be accessed in `HTML`. Note that anchors which are both, source and destination anchors, at the same time are stored in both tables separately.

```
create table forms(
  did         number
);
```

This table contains the `did`s of all documents containing forms, i. e. of th doumetns containing exercises.

```
create table source_prop(
  sid         number,
  dir         varchar(5),
  sprop       varchar(32)
);

create table dest_prop(
  aid         number,
  dir         varchar(5),
  dprop       varchar(32)
);
```

These tables contain the properties of source and destination anchors. In the `HTML` documents we realize these properties by using `REV` and `REL` attributes. The `sid`/`aid` field contains the anchor identifier, the `dir` field the direction (`REL` or `REV`), and the `sprop`/`dprop` field contains the name of the property.

```
create table link(
  lid         number unique,
  sid         number,
  aid         number
);
```

The `link` table contains the links for which both, source and destination anchors are known. The anchors are identified only by their id (as opposed to the paper (Albert et al., 1997) where the document id's were also considered to be specified in the link relation) since these anchor id's are unique now.

```
create table req_dest(
  rid        number unique,
  filename   varchar(256),
  name       varchar(128)
);

create table prov_link(
  pid        number unique,
  sid        number,
  rid        number
);
```

These tables contain information about links and link destinations which could not yet be resolved because the destinations have not been registered yet. This may have two reasons: either we have dangling links or the document with the destination anchor just has not yet been registered.

```
create table pupil(
  puid        number unique,
  account    varchar(32) unique,
  name       varchar(256)
);
```

The `pupil` table consists of three columns: The identifying number `puid`, the (also unique) `account` name, and the (real) `name`.

```
create table performance(
  puid       number,
  did        number,
  when       date
);
```

In the `performance` table it is recorded which pupil (`puid`) has learned which document (`did`) at which date.

B.2 Sequences

```
create sequence did;
create sequence sid;
create sequence aid;
create sequence lid;
create sequence puid;
```

This sequence `did` contains the consecutive identification number of the document. The sequences `sid` and `aid` are use for identifying source and destination anchors, respectively, the `lid` identifies links. The `puid` sequence finally is used for the identification of pupils.

The fields `rid` and `pid` in the tables `req_dest` and `prov_link` describing still unresolved links share the sequences `aid` and `lid`, respectively, with the corresponding fields of the tables `dest` and `link` describing already resolved links.

Appendix C

Programs building RATH

The following list contains the programs of which RATH is built.

C.1 Directly executable programs

`generate-db` This tool is used to scan in HTML files writing an enriched copy into another directory and, in parallel, registering (also non–HTML) files and links in the database. For details on its usage, see Section 6.4. Internally, the programs `scan` and `add2rath` are used to scan and register HTML files or just to register non–HTMLfiles, respectively.

`init-db` This tool generates all tables used in the Oracle database. If tables exist, they are deleted first and newly (i. e. empty) created afterwards. The script does not need any parameters.

> **Warning:**
> **Using this script in system already up and running might result in data inconsistencies between Oracle and the `httpd` user and password administration, i. e. users might still exist for the WWW daemon without having a database entry.**

!

C.2 CGI scripts

Besides the tools mentioned with each of the CGI scripts all of them use the `cgiutils`, `cgiparse`, and/or `htadm` programs which belong to the WWW server. The first of these programs creates an HTTP header, the second program parses parameters for the CGI scripts, and the last one is used for maintaining the password files of the WWW server. Those scripts which use Oracle also call the internal script `oracle` setting some shell environment variables.

C.2.1 CGI scripts for user purposes

`ITS` This script is the interface between the `httpd` and the `filter` program which filters links from the enriched `HTML` files which should not be visible to the pupil.

`evaluate` The `evaluate` script evaluates the pupil's answers to the exercises. Currently this is done by the script itself but for more sophisticated courses that task might be delegated to a separate tool. The registration of correctly answered exercises in the database is done by the internal tool `answered`.

C.2.2 CGI scripts for administrative purposes

`progress` This script queries performance data about a pupil by directly calling the Oracle `sqlplus` interpreter.

`useradd, userdel, passwd` These scripts evaluate WWW forms using `cgiparse`, update the password file using `htadm`, and (if appropriate) update the database information using the internal tools `register` (for adding new users) or `deregister` (for deleting existing users)

C.3 Internal tools

For all of these programs, the main source file is a Pro C file, i. e. the main source file contains code to be analyzed by the Oracle Pro C/C++ precompiler first.

`add2rath` This program is used to register non–`HTML` files in the database. It has two parameters: the filename (including path information), and a date string to be printed into the logfile.

`scan` The `scan` program registers, scans, and enriches `HTML` files. The file itself and all anchors are registered in the database. An enriched version is created to facilitate the later filtering. It gets the filename (with path information) as parameter. This program works as a pipe, i. e. it additionally gets the file itself piped into its standard input and prints the enriched version to its standard output.

`filter` The filter program takes a file (specified by the second parameter) from the directory of enriched `HTML` files and filters the links contained in the `HTML` file according to the user's (specified by the first parameter) knowledge.Like `scan`, this program works as a pipe, i. e. it additionally gets the file itself piped into its standard input and prints the enriched version to its standard output.

`answered` The `answered` program stores information about solved problems in the database. Pages containing test problems should not be registered as learned after presentation to the pupil (as teaching lessons) but after the pupil has given the correct answer. This is done by `answered`. The program gets two parameters: the pupil's account and the name of the file containing the exercise (including path information).

`register` A new user specified by the parameters account and real user name is added to the database.

`deregister` This program deletes a pupil's entries from the database. It has one parameter, the pupil's account.

`oracle` This script is called by the CGI scripts to set shell environment variables needed by Oracle.

Appendix D

Structure of teaching material

Our selection of teaching materials in the field of elementary stochastics and probability theory is based on the work of Held (1993, chapter 8). Held constructed several problems about drawing balls from an urn. As a structure for this field, we use Model 1 of his Investigation II. However, we omitted his component d which represents different wordings for the problems. Additionally we slightly changed the structure in some points where it seemed necessary to us. The necessity for this changes is founded in the fact that Held focused on test problems while we focus on teaching lessons.

D.1 The structure derived from the work by Held

D.1.1 Demands

In his Investigation II, Held has identified the following demands (or skills; the titles of the corresponding `HTML` files are specified in brackets):

1. Knowledge that, in general, Laplace probabilities are computed as the ratio between the number of *convenient* events and the number of *possible* events.
   [Laplace–probabilities]

2. Ability to determine the number of possible events.
   [Events]

3. Ability to determine the number of convenient events if one ball is drawn.
   [Determining the number of convenient events]

4. Ability to determine a convenient event if one ball is drawn, or if the sample for which the probability has to be computed consists of equally coloured balls.
   [Drawing multiple balls with a common property]

5. Knowledge that if an outcome like "exactly/at least n balls are of colour x" is asked for, all possible sequences of drawings are convenient events.
   [Events containing elementary events with different properties]

6. Knowledge that $p(A \cup B) = p(A) + p(B)$, for two disjoint events A and B.
   [Addition of events and probabilities]

7. Knowledge that $p(A \cap B) = p(A) \cdot p(B)$, for two events A and B that are (stochastically) independent.
   [Multiplication of events and probabilities]

8. Knowledge that the probability of drawing a ball of a specific colour is not equal to 0.5 if there are different numbers of balls of different colours in the urn.
   [Different proportions of properties]

9. Knowledge that drawing *without* replacement reduces both, the total numbers of balls in the urn as well as the numbers of balls that have the same number as the drawn ball.
   [Drawing without replacement]

10. Knowledge that drawing *at least* a number of certain balls includes the — not explicitly stated — results of drawing more balls of the certain kind. [Generalized description of events]

We regard these demands as teaching contents. In our prototype, each of these demands will be represented by one teaching unit consisting of a teaching lesson and several examples.

D.1.2 Demand structure

Using the assignments of demands to the components of the three attributes a, b, and c introduced by Held, we obtain the structure of demands illustrated in Fig. D.1.

D.1.3 Problem structure

Regarding the 18 problems used by Held in his empirical investigation and neglecting his wording component d, we obtain six classes or problems (*notions* in the terminology of Doignon and Falmagne). This classification is specified in Table D.1. The resulting problem structure is shown in Fig. D.2. This problem structure results in the knowledge space shown in Fig. D.3.

Table D.1: Problem classes after omitting wording component

Problem class	Problems included	Attributes
A	14, 16, 18	a_2, b_3, c_3
B	12, 15, 17	a_2, b_2, c_2
C	8, 10, 13	a_1, b_3, c_1
D	2, 6, 9	a_2, b_1, c_1
E	4, 7, 11	a_1, b_2, c_1
F	1, 3, 5	a_1, b_1, c_1

D.1.4 Combined structure of teaching contents and test problems

Combining teaching lessons and training/test problems, we obtain the structure shown in Fig. D.4. This structure should be directly usable for an educational hypertext. In Fig. D.4, encircled numbers represent teaching materials (corresponding to the demands) while ensquared capital letters represent the training problems. We use the problems as a test if the pupil has understood the lessons selected so far and, therefore, require correct solutions to these problems before allowing the pupil to continue in the course. As a consequence, we also obtain some prerequisite relations between demands which are not specified in the demand structure in Figure D.1, e. g. the demands 7 and 8 are prerequisites of demand 10 in the combined structure but not in the original demand structure.

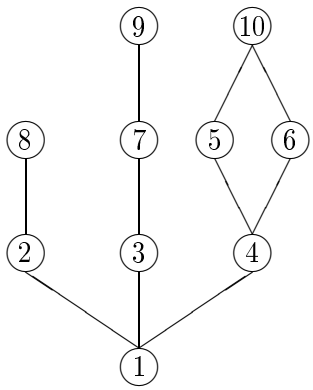

Figure D.1: Demand structure based on Held's Model 1

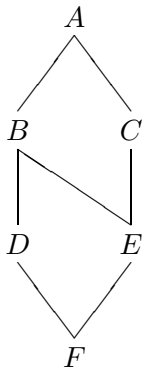

Figure D.2: Problem structure based on Held's Investigation II

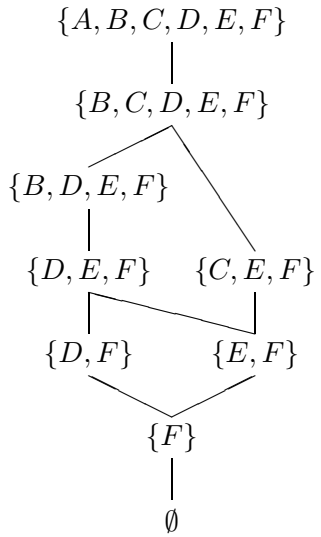

Figure D.3: Knowledge space of the problem structure from Fig. D.2

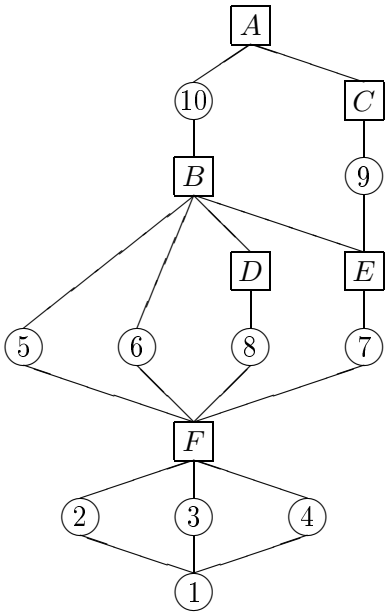

Figure D.4: Combined structure of teaching contents and training problems

D.2 A modified structure for teaching purposes

D.2.1 Changes in the structure

In the process of writing the teaching material for the stochastic course the problem arose that the structure derived from Held's work is not sufficient in two points: First, it seemed reasonable to us that pupils should learn about event spaces and convenient events first before learning the rule to compute Laplace probabilities. The second point was the fact that concepts like random experiment, result, or event were not dealt with at all in the framework of Held's investigations. These points have been identified by us as an additional Demand 0. These two points result in a slightly changed combined structure shown in Figure D.5 Please note, that these modifications in the demand structure have no effects at all in the problem structure from Figure D.2.

D.2.2 Learning paths

Combining Fig. D.5 with the knowledge space shown in Fig. D.3 we obtain the possible learning paths (Falmagne, 1993; Albert and Hockemeyer, 1997) shown in Fig. D.6. In this figure, the numbers next to the arrows denote the lessons (i. e. demands in the terminology of Held) which have to be learned in order to make a learning step from one knowledge state to the next one.

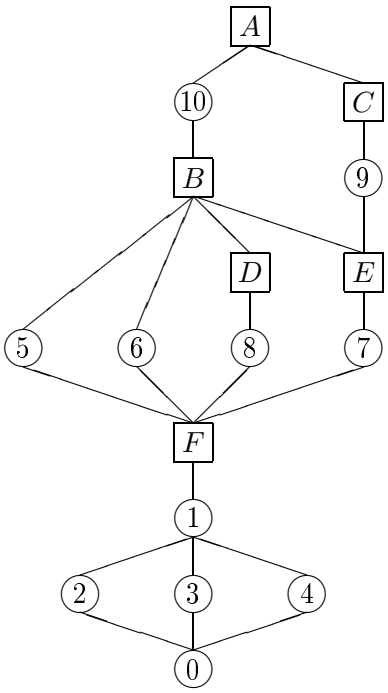

Figure D.5: Modified structure of teaching contents and training problems

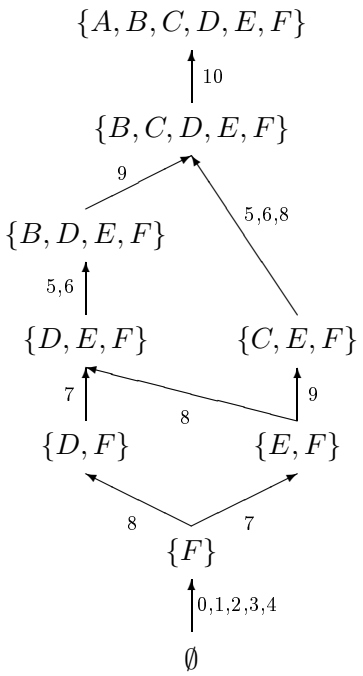

Figure D.6: Learning paths in the knowledge space

Appendix E

Illustrative teaching material

The following text contains the teaching material of the RATH prototype. This material was composed together with Gerhard Hermann using the results of prior research by Held (1993; Held and Albert, 1995). In the hypertext version, each section constitutes an own document.

E.1 Elementary Events

Experiments which may have different results under the same conditions are called *random experiments.* Throwing a dice, for example, may result in different numbers between one and six.

A random experiment may have a finite number of possible results ω_1, ω_2,..., ω_n ($n \in \mathbb{N}$). The set $\Omega = \{\omega_1,\omega_2, \ldots,\omega_n\}$ is called the *space of results.*

E.1.1 Example 1: Drawing balls from an urn

Condition: An urn contains two balls, one is white and the other is green.
Random experiment: Drawing one ball.
Determine the number of results.

Solution:
Possible Results:
ω_1 = "white ball"
ω_2 = "green ball"

Space of results:
$\Omega = \{\omega_1, \omega_2\} \Longrightarrow$ **The number of results is $|\Omega| = 2$.**

E.1.2 Example 2: Throwing a dice

Condition: A dice usually has six sides which have one to six points.
Random experiment: Throwing the dice and reading the number of points

on its upper side.
Determine the number of results.

Solution:
ω_1 = "There is one point on the dice's upper side."
ω_2 = "There are two points on the dice's upper side."
ω_3 = "There are three points on the dice's upper side."
ω_4 = "There are four points on the dice's upper side."
ω_5 = "There are five points on the dice's upper side."
ω_6 = "There are six points on the dice's upper side."

Space of results:
$\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6\}$
$\Longrightarrow$ **The number of results is $|\Omega| = 6$.**

E.1.3 Example 3: Taking a floppy disk from a box

Condition: A box contains three floppies. One of them is defective.
Random experiment: Taking one floppy from the box.
Determine the number of results.

Solution:
Possible Results:
ω_1 = "Intact floppy"
ω_2 = "Defective floppy"

Space of results:
$\Omega = \{\omega_1, \omega_2\} \Longrightarrow$ **The number of results is $|\Omega| = 2$.**

E.2 Events

Let Ω be a space of results. Any subset $A \subseteq \Omega$ is called *event*. The set of all events is the set of all subsets (or the *power set* $\wp(\Omega)$) of Ω. It is called *event space*. Note that the empty set $\emptyset$ and the space of results Ω are also subsets of the event space Ω and, therefore, events.

An event A has happened if the result of the experiment is an element of A and, vice versa, A has not happened if the result is not an element of A.

Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$, $n \in \mathbb{N}$, be a space of results. The events $\{\omega_i\}$, $i = 1, 2, \ldots, n$, are called *elementary events* of the event space $\wp(\Omega)$. For a given event $A \in \wp(\Omega)$, the elementary events $\{\omega_i\}$ that fulfil the condition $\omega_i \in A$ are called *convenient for* A.

E.3 Determining the number of convenient events

Let Ω be a set of possible results for a random experiment, and let A be an event described by "the result has the property Φ". Then the event A contains all those results $\omega \in \Omega$ that fulfil the specified condition, i. e. that have the property Φ.

E.3.1 Example 1: Drawing balls from an urn

An urn contains two green balls and two white balls.
A green ball is drawn.
Determine the number of convenient elementary events.

Solution:
Event A: "A green ball is drawn."

Convenient elemtary events for the event A are all green balls:
$A = \{green, green\} \implies |A| = 1$
The number of convenient elementary events is 1.

E.3.2 Example 2: Throwing a dice

A dice is thrown.
On its upper side, it shows an even number of points.
Determine the number of convenient elementary events.

Solution:
Event A: "The number of points on the upper side is even."

Convenient elementary events for the event A are the even numbers between one and six:
$A = \{2, 4, 6\} \implies |A| = 3$
The number of convenient elementary events is 3 .

E.3.3 Example 3: Taking a floppy disk out of a box

In a box there are two floppies. One is intact, the other is defective.
An intact floppy is taken out of the box.
Determine the number of convenient elemtary results

Solution:
Event A: "An intact floppy is taken out of the box."

Convenient elementary events for the event A are all intact floppies:
$A = \{i\} \implies |A| = 1$
The number of convenient elementary results is 1 .

E.4 Drawing multiple balls with a common property

Let Ω be a space of results with m elements with property ψ and n elements with property ϕ, and let the experiment be the drawing of k balls. For an event $A =$ "all k balls have the property ψ", all elementary events consisting of k balls with property ψ are convenient events, independent of the sequence of the k elements.

E.4.1 Example 1: Drawing balls from an urn

An urn contains one white and three green balls. Two balls are drawn from the urn, each ball is immediately put back into the urn (*drawing with replacement*). The drawn balls have a different colour. Determine the number of convenient elementary events.

Solution:
We denote the possible results of a single draw by "white" and "green", respectively. All pairs of different balls are convenient elementary events. Consequently, we obtain the set

$$A = \{(\text{green}, \text{white}), (\text{white}, \text{green})\}$$

of convenient elementary events. **The number of convenient elementary events is $|\mathbf{A}| = \mathbf{2}$.**

E.4.2 Example 2: Throwing a dice

A dice is thrown twice. Only the numbers 5 and 6 occur. Determine the number of convenient elementary events.

Solution:
Event A: "Only the numbers 5 and 6 occur. "

Convenient elementary events for the event A are the sequences $(5,5)$, $(5,6)$, $(6,5)$, $(6,6)$:
$A = \{(5,5), (5,6), (6,5), (6,6)\} \implies |A| = 4$
The number of convenient elementary events is 4 .

E.5 Laplace–probabilities

Theorem by Laplace: If Ω is a finite set of elementary events with an equally distributed probability then the probability $P(A)$ for an arbitrary event $A \subseteq \Omega$ can be computed as the ratio

$$P(A) = \frac{|A|}{|\Omega|} = \frac{\text{Number of convenient elementary events}}{\text{Total number of elementary events}}$$

Random experiments which fulfil both conditions

1. Ω is a finite set of elementary events ω: $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$ with $n < \infty$.

2. All n elementary events $\{\omega_1\}, \{\omega_2\}, \ldots, \{\omega_n\}$ are equally probably, i. e. $P(\{\omega_1\}) = \cdots = P(\{\omega_n\}) = p$.

are called *Laplace experiments*.

E.5.1 Example 1: Drawing balls from an urn

An urn contains four balls in the colours white, red, blue, and green. One ball is drawn from the urn. This ball is white. Compute the probability for this event.

Solution:
Possible elementary events are all four balls: $\Omega = \{\text{white}, \text{red}, \text{blue}, \text{green}\}$. The convenient elementary event is the white ball: $A = \{\text{white}\}$. As a result, we obtain

$$P(A) = \frac{|A|}{|\Omega|} = \frac{1}{4}.$$

E.5.2 Example 2: Taking floppy disks out of a box

A box contains two floppies. One is intact, the other is defective. One floppy is taken out of the box, This floppy is defective. Compute the probability for this event.

Solution:
We have two possible elementary events: $\Omega = \{i, d\}$, and we have one convenient elementary event: $A = \{d\}$. As a result, we obtain

$$P(A) = \frac{|A|}{|\Omega|} = \frac{1}{2}.$$

E.6 Events containing elementary events with different properties

Let Ω be a space of results containing m elements with property ψ and n elements with property ϕ. Let the event A contain v ($v < m$) elements of property ψ and w ($w < n$) elements with property ϕ. All sequences of possible results containing v elements of property ψ and w elements of property ϕ are convenient elementary events.

E.6.1 Example 1: Drawing balls from an urn

An urn contains one white and three green balls. Two balls are drawn with replacement from the urn. Exactly one of the balls is green. Determine the number of convenient results.

Solution:
Convenient elementary events are all pairs of balls containing exactly one green ball:

$$A = \{(\text{green}, \text{white}), (\text{white}, \text{green})\} \implies |A| = 2.$$

Consequently, the number of convenient elementary events is 2.

E.6.2 Example 2: Taking floppy disks out of a box

A box contains four floppy disks. Two of the floppies are intact, the other two are defective. Two floppies are taken out of the box with replacement. Exactly one floppy is intact. Determine the number of convenient results.

Solution: All elementary events containing exactly one intact floppy are convenient:

$$A = \{(\text{defective}, \text{intact}), (\text{intact}, \text{defective})\} \implies |A| = 2.$$

Consequently, the number of convenient elementary events is 2.

E.7 Addition of events and probabilities

For two disjoint events A_1, A_2, we obtain

$$P(A_1 \cup A_2) = P(A_1) + P(A_2).$$

The events $A_1, A_2, \ldots, A_n \subseteq \Omega$ are called *pairwise disjoint* if any two different events A_i, A_K $(1 \leq i, k \leq n$ and $i \neq k)$ are disjoint. For any sequence $A_1, A_2, \ldots, A_n$ of pairwise independent events, we obtain

$$P(A_1 \cup A_2 \cup \cdots \cup A_n) = P(A_1) + P(A_2) + \cdots + P(A_n).$$

E.7.1 Example: Drawing balls from an urn

An urn contains four balls of the colours white, red, green, and blue. One ball is drawn from the urn. Compute the probability that this ball is red or blue.

Solution:
The space of result is $\Omega = \{\text{white}, \text{red}, \text{green}, \text{blue}\}$. For each of the balls the

probability to be drawn is $\frac{1}{|\Omega|} = \frac{1}{4}$. Since the results 'red' and 'blue' exclude each other, the probability of drawing a red or a blue ball can be computed as a sum:

$$P(\text{red} \cup \text{blue}) = P(\text{red}) + P(\text{blue}) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

E.8 Multiplication of events and probabilities

Two events A_1 and A_2 are *(stochastically) independent* if the probability that event A_1 happens does not depend on the occurrence of the event A_2 and vice versa.

For stochastically independent events A_1, A_2 the following *multiplication rule* can be used to compute the combined probability:

$$P(A_1 \cap A_2) = P(A_1) \cdot P(A_2).$$

E.8.1 Example: Drawing balls from an urn

An urn contains three white balls and three green balls. Subsequently, two balls are drawn from the urn and put back. Compute the probability that both balls are green.

Solution:
We regard the event A ="both balls are green" as a sequence of two single events A_1 ="the first ball is green" and A_2 ="the second ball is green". Since the drawn ball is put back before the second ball is drawn, these single events A_1 and A_2 are independent. Therefore, the probability $P(A)$ can be computed as a product of the probabilities $P(A_1)$ and $P(A_2)$. For these single probabilities, we obtain $P(A_1) = P(A_2) = \frac{1}{2}$ and, consequently, for the combined probability, $P(A) = P(A_1) \cdot P(A_2) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$.

E.9 Different proportions of properties

If an urn contains balls of different colours in different proportions, the probability of drawing a ball of a specific colour can be computed as

$$P(\textit{Specific colour}) = \frac{\textit{Number of balls with "specific colour"}}{\textit{Total number of balls in the urn}}.$$

E.9.1 Example 1: Drawing balls from an urn

An urn contains three white balls and four green balls. One ball is drawn from the urn. Compute the probability that the ball is green.

Solution:
Altogether, there are seven balls in the urn. Consequently, we obtain

$$P(\text{green}) = \frac{\#\text{ green}}{\#\text{ balls}} = \frac{4}{7}.$$

E.10 Drawing without replacement

So far, all drawing experiments were performed with replacements. If, in a sequence of drawings, balls already drawn are not put back into the urn before drawing the next ball, the number of balls available in the next drawing is reduced. The single drawings can then be considered to be stochastically independent.

E.10.1 Example: Drawing balls from an urn

In an urn there are three green balls and two white balls. Two balls are successively drawn from the urn *without* replacement. Compute the probability that both balls are green.

Solution:
There is one convenient event: $A = \{(\text{green}, \text{green})\}$. We split up the experiment and the convenient event into two separate experiments and events, respectively. The convenient events then have the form $A_1 = A_2 = \{\text{green}\}$.

In the first drawing, we have five balls in the urn. Three of them are green. We compute the probability of drawing a green ball as

$$P(A_1) = P(\text{green}) = \frac{3}{5}.$$

Afterwards, we have two white balls and two green balls in the urn, i. e. four balls altogether. For the second drawing, we obtain

$$P(A_2) = P(\text{green}) = \frac{2}{4} = \frac{1}{2}.$$

There is no further dependence between the two drawings than the alteration of the numbers of balls. Since this alteration has already been considered, we can regard the two drawings as stochastically independent and, therefore, use the multiplication rule:

$$P(A) = P(\{(\text{green}, \text{green})\}) = P(A_1) \cdot P(A_2) = \frac{3}{5} \cdot \frac{1}{2} = \frac{3}{10}.$$

E.11 Generalized descriptions of events

If we have a random experiment which is constructed as a sequence of several single experiments then we have to distinguish between events of the form "exactly k of the n single results have the property Φ" and events of the form "at least (or at most) k of the n single results have the property Φ". In the latter form, we have to consider the cases of $k, k+1, \ldots, n$ (or $1, 2, \ldots, k$, respectively) convenient single results.

E.11.1 Example: Drawing balls from an urn

An urn contains two white balls and three green balls. Two balls are drawn with replacement. Compute the probability of drawing at least one white ball.

Solution:
All events with at least one white ball are convenient:

$$A = \{\underbrace{(\text{white}, \text{green})}_{A_1}, \underbrace{(\text{white}, \text{white})}_{A_2}, \underbrace{(\text{green}, \text{white})}_{A_3}\}.$$

For the three partial events, we obtain the probabilities

$$\begin{aligned} P(A_1) &= \frac{2}{5} \cdot \frac{3}{5} = \frac{6}{25} \\ P(A_2) &= \frac{2}{5} \cdot \frac{2}{5} = \frac{4}{25} \\ P(A_3) &= \frac{3}{5} \cdot \frac{2}{5} = \frac{6}{25} \end{aligned}$$

Since the three partial events are mutually exclusive, we can compute the final probability $P(A)$ as the sum of the three single probabilities:

$$P(A) = P(A_1) + P(A_2) + P(A_3) = \frac{6}{25} + \frac{4}{25} + \frac{6}{25} = \frac{16}{25}$$

E.12 Exercises

E.12.1 Exercise 1

An urn contains five white and five black balls. One ball is drawn from the urn. The drawn ball is black. Compute the probability of this event.

E.12.2 Exercise 2

An urn contains five yellow and five black balls. Two balls are drawn from the urn successively. Drawing is performed with replacement. The drawn balls are yellow. Compute the probability of this event.

E.12.3 Exercise 3

An urn contains ten white and six black balls. One ball is drawn from the urn. The drawn ball is white. Compute the probability of this event.

E.12.4 Exercise 4

An urn contains five white and five black balls. Two balls are drawn from the urn successively. Drawing is performed without replacement. The drawn balls are black. Compute the probability of this event.

E.12.5 Exercise 5

An urn contains six black and four white balls. Three balls are drawn from the urn successively. Drawing is performed with replacement. Exactly two of the drawn balls are black. Compute the probability of this event.

E.12.6 Exercise 6

An urn contains five black and four white balls. Three balls are drawn from the urn successively. Drawing is performed without replacement. At least two of the drawn balls are white. Compute the probability of this event.