BRICS RS-02-7 Ingólfsdóttir et al.: A Formalization of Linkage Analysis

BRICS
Basic Research in Computer Science

A Formalization of Linkage Analysis

Anna Ingólfsdóttir
Anders Lyhne Christensen
Jens Alsted Hansen
Jacob Johnsen
John Knudsen
Jacob Illum Rasmussen

BRICS Report Series **RS-02-7**
ISSN 0909-0878 **February 2002**

Copyright © 2002, Anna Ingólfsdóttir & Anders Lyhne Christensen & Jens Alsted Hansen & Jacob Johnsen & John Knudsen & Jacob Illum Rasmussen.
BRICS, Department of Computer Science
University of Aarhus. All rights reserved.

Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.

See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/02/7/`

A Formalization of Linkage Analysis

Anna Ingólfsdóttir   Anders Lyhne Christensen
Jens Alsted Hansen   Jacob Johnsen   John Knudsen
Jacob Illum Rasmussen

February 2000

Abstract

In this report a formalization of genetic linkage analysis is introduced. Linkage analysis is a computationally hard biomathematical method, which purpose is to locate genes on the human genome. It is rooted in the new area of bioinformatics and no formalization of the method has previously been established.

Initially, the biological model is presented. On the basis of this biological model we establish a formalization that enables reasoning about algorithms used in linkage analysis. The formalization applies both for single and multi point linkage analysis. We illustrate the usage of the formalization in correctness proofs of central algorithms and optimisations for linkage analysis.

A further use of the formalization is to reason about alternative methods for linkage analysis. We discuss the use of MTBDDs and PDGs in linkage analysis, since they have proven efficient for other computationally hard problems involving large state spaces.

We conclude that none of the techniques discussed are directly applicable to linkage analysis, however further research is needed in order to investigated whether a modified version of one or more of these are applicable.

Contents

List of Symbols (in order of appearance)

P	Pedigree $P = \langle F, N, father, mother \rangle$, see Definition 1.
$father$	Father function $father : N \rightarrow V$, see Definition 1.
$mother$	Mother function $mother : N \rightarrow V$, see Definition 1.
V	Set of individuals in a pedigree (P), $V = N \cup F$, see Definition 1.
F	Set of founders in a pedigree (P), $F \subseteq V$, see Definition 1.
N	Set of non-founders in a pedigree (P), $N \subset V$, see Definition 1.
$ancestor$	Ancestor function $ancestor : V \rightarrow 2^V$, see Definition 1.
G	Pedigree graph $G = \langle P, \rightarrow \rangle$, see Definition 2.
A	Set of alleles for a pedigree, see Definition 3.
M	A marker $M = \langle A_M, \pi_M \rangle$, see Definition 4.
A_M	The set of allelic states for the marker M, see Definition 4.
π_M	A function $\pi_M : A_M \rightarrow [0, 1]$, maps the allelic states of A_M to its allele frequency, see Definition 4.
$\mathcal{G}$	Genotype information $\mathcal{G} = \langle \mathcal{L}, astates \rangle$, see Definition 3.2.1.
$\mathcal{L}$	Set of genotyped individuals for a pedigree, see Definition 3.2.1.
$astates$	Allelic state function $astates : \mathcal{L} \rightarrow \{\{a_1, a_2\} \mid a_1, a_2 \in A_M\}$, see Definition 3.2.1.
v	Inheritance vector function: $v : N \times \{p, m\} \rightarrow \{p, m\}$, see Definition 6.
$\mathbf{v}$	The set of all inheritance vectors for a pedigree, see Section 3.2.2.
p	Parameter for the inheritance vector function, denotes the *paternal* allele, see Definition 6.
m	Parameter for the inheritance vector function, denotes the *maternal* allele, see Definition 6.
Z_M	A founder allele assignment, $Z_M : F \times \{p, m\} \rightarrow A_M$. for the marker M, see Definition 7.
F	Allele assignment function $\mathsf{F} : (N \times \{p, m\} \rightarrow \{p, m\}) \rightarrow ((V \times \{p, m\}) \rightarrow (F \times \{p, m\}))$, which given an inheritance vector gives the founder alleles received by an individual on the maternal or paternal side, see Definition 8.
F^v	$\mathsf{F}^v = \mathsf{F}(v)$, see Definition 8.

$P(v\|\mathcal{G})$	The probability of the inheritance vector v given some genotype information $\mathcal{G}$, see Section 3.2.2.
$P(\mathcal{G}\|v)$	The probability of some genotype information $\mathcal{G}$ given an inheritance vector v, see Section 3.2.2.
$P(Z_M)$	The probability of the founder allele assignment, see Section 3.2.2.
$\mathcal{F}_{\mathcal{G}}^{v}$	The set of all compatible founder allele assignments given the inheritance vector v, and the genotype information $\mathcal{G}$, see Section 3.2.2.
$f, f_1, f_2, \ldots$	Founder alleles.
$a_1, a_2, \ldots$	Allelic states.
ϖ	Denotes either p for paternal or m for maternal ($\varpi \in \{p, m\}$).
$f_{(n,\varpi)}$	The founder allele for founder $n \in F$, see Section 3.3.1.
$f_{\mathsf{F}^v(n,\varpi)}$	The founder allele inherited by $n \in V$ given the inheritance vector v, see Section 3.3.1.
$\varphi_{\mathcal{G}}^{v}$	Founder allele assignment formula for the inheritance vector v and the genotype information $\mathcal{G}$, see Theorem 1.
CC	The set of connected components for a graph.
$\mathcal{U}$	The set of unassigned founder alleles, see page 42.
$\mathcal{E}$	The set of ambiguously assigned founder alleles, see page 42.
$\mathcal{A}$	The set of (unambiguously) assigned founder alleles, see page 42.
θ_i	The recombination fraction between markers M_i and M_{i+1}, see Section 3.4.
$Ham(v, w)$	The Hamming distance between two inheritance vectors v and w, see Definition 11.

Chapter 1

Introduction

Linkage analysis is a well established method used to locate genes in the human genome. To be more exact, the gene, which is under investigation, is a gene that is responsible for some trait that an individual possesses. Examples of traits under investigation range from the more simple, such as blood type and eye color, to the more serious that might predispose an individual for a deadly disease.

Disease causing genes for some major diseases have already been discovered by performing linkage analysis, e.g. Parkinson's disease, obesity, and anxiety, [dNC01]. Although environmental factors influence the disease development on some of these diseases, it would be ignorant not to have their genetic compounds in mind for the following reasons, [dPGD01]:

- Their causes are not fully understood.
- Current treatments are of limited effectiveness.
- There is currently no means of tailoring treatment to the cause.
- Genetic diseases are getting an proportionally increased influence on peoples health in the western world, e.g. the percentage of childhood deaths in United Kingdom hospitals attributable to genetic causes have increased from 16.5% in 1914 to 50% in 1976, [JCBW99].

Linkage analysis is a statistical method, which might locate certain trait causing genes, based on some biological model. However, due to the complexity of current algorithms incorporated in linkage analysis, the analysis is only tractable on moderately sized data sets. Analysis of larger data sets bestows the analyst with an improved likelihood of locating the genes for the trait under investigation.

Our final goal is to enable a reduction of the linkage analysis complexity by applying modern computer science techniques. Currently, problems arise due to explicit representations of state spaces, which shows to be exponential in the number of individuals involved in the analysis. Dealing with exponential state spaces is one of the areas where formal methods applied in the field of verification has proven successful in the past decades, and the advances have often resulted in several orders of magnitude reductions in complexity. These formal methods include symmetry reduction, partial order reduction, abstraction, and compositionality.

To reach our final goal, we define three major sub goals. The first is the establishment of a formal model that enables reasoning about the correctness and complexity of different approaches to linkage analysis. Secondly, we analyse the strategies which seem fruitful to pursue further, in order to reduce the complexity of linkage analysis. The third and final sub goal is to construct a new method for linkage analysis which enables linkage analysis of larger data sets.

We have with this project moved into the relatively new area of bioinformatics, which spans a number of different sciences, such as biology, statistics, mathematics, and computer science. We are unaware of anyone who have previously attempted to formalize linkage analysis in a computer scientific context. Currently, several different software packages exist, which can perform linkage analysis. A brief overview of the major packages are given in Appendix A. However, we have not been able to find any formal framework, in which it is possible to reason about the correctness and complexity of existing algorithms. Furthermore, a formal framework would be convenient for evaluating new methods and for investigating the deeper theoretical nature of linkage analysis.

The work represented here is based on a collaboration with the Icelandic company deCODE Genetics (http://www.decode.is). deCODE Genetics is a company conducting research into inherited causes of common diseases. The research at deCODE is based on the Icelandic population for three major reasons, the original genetic pool is small, literature documents the family relations many generations back, and finally the juridical foundations have been established. deCODE currently employs more than 600 people divided into several departments ranging from the Research Laboratory and the Database Division, to the Statistics Department with which we have collaborated. The Statistics Department is, among other areas, responsible for developing software capable of doing statistical analysis of genetic data. Part of our project group has spent time working with Ph.D. Daniel Fannar Gudbjartsson, the creator of *Allegro*, which is the software package used at deCODE for linkage analysis (See Appendix A). The stay gave the group a chance to verify and extend our understanding of the methods currently used within the field of linkage analysis.

1.1 This report

We have investigated to what extent the area of linkage analysis is formalized in a computer scientific context. To our knowledge, no formal framework exist to describe linkage analysis. For this reason we develop such a framework, to be able to reason about the correctness of current and future algorithms. After establishing the formal framework, we attempt to determine the complexity of linkage analysis and based on this investigation we discuss approaches to reduce the complexity in the average case. More specifically, we investigate how well MTBDDs and PDGs work for symbolic representation of probability distributions in our problem domain.

This report is meant to be a self contained document. Thus readers with a mathematical, engineering, or computer science background should be able to read and understand it with little or no prior knowledge on linkage analysis and genetics. Readers that do not have this background might experience difficulties in understanding the more computer scientific aspects, although some introduction is given into selected areas in the appendices. In the back of this report we have provided a glossary (See page 107) of the most important genetic terms, for quick reference, and a list of symbols can be found on page v.

The structure of the report is as follows: The first part is an introduction to the biological model. This part is not exhaustive, however it provides a sufficient background for understanding the motivation behind the two following parts. In the second part we build a formal framework, for proving the correctness of the algorithms used in linkage analysis. In the final part of the report we state the properties which a new method should preserve, in order to produce the same results as well established methods. Furthermore, we discuss the performance of alternative strategies for the representation and computational tasks in linkage analysis.

1.2 Acknowledgments

First we would like to acknowledge Daniel Fannar Gudbjartsson, the creator of Allegro, for his help and guidance as well as other people in the Statistics Department, and Sverrir Þorvaldsson and Gisli Màsson, at deCODE for their patience and willingness to answer questions and give explanations.

We would also like to thank Professor of Computer Science Finn Verner Jensen and Assistant Professor of Computer Science Thomas Nielsen, Aalborg University for their help with Bayesian theory and hidden Markov models.

We would like to thank the following people for inspiration: Professor of Computer Science Kim G. Larsen and Associate Professor of Computer Science Luca Aceto, both from Aalborg University. Professor of Biotechnology Aalborg

University Karen G. Welinder, Assistant Professor Dennis Nilsson from the Mathematics Department at Aalborg University, Research Assistant Professor at BRICS Aarhus Christian N. Storm Pedersen, Professor of Computer Science at the University of Birmingham Martha Kwiatkowska and Ph.D. student at the University of Birmingham David Parker.

Chapter 2

Introduction to Human Genetics

This chapter introduces the reader to the terms and concepts of human genetics which are essential for understanding linkage analysis. Readers who are familiar with genetic concepts like alleles, mendelian inheritance, linkage analysis, and so on may want to skip this chapter. We remind the reader unfamiliar with any of these concepts, that there is a glossary on page 107, where it is possible to find a short explanation and a reference to many of the terms and concepts present in this chapter.

This introduction is based on various sources, such as, [SR99], [Ott99], and [KC97]. For further reading, we recommend the reader to see Section 2.7 for a short discussion of some literature.

The structure is as follows. We begin with a rather high level description of genetics, by introducing the two major concepts of chromosomes and DNA. This is followed by the introduction of alleles and allelic states, which describe the different forms a gene can assume. Two important concepts in inheritance, meiosis and recombination, are introduced. Furthermore, we introduce the concept of linkage analysis and finally summarize the chapter.

2.1 Chromosome

Inside each human cell there are 23 pairs of chromosomes, 22 pairs of *autosomes* common for both males and females, and one pair of sex-chromosome which differs between the two sexes. A chromosome consists of two identical sister *chromatides* - each a double helix DNA, *strand*, [SR99, page 30], thus a total of 92 DNA strands, constituting 46 chromosomes. During fertilization a fetus receives one chromosome from each parent (actually it is one chromatide, which later is transformed into a chromosome, but in this context can be thought of as a chromosome).

The complete set of genetic material is called *The Human Genome*.

2.2 DNA

The molecule that encodes genetic information of humans is deoxyribonucleic acid (DNA). DNA is a double helix molecule that consists of a sugar-phosphate backbone and four nitrogenous bases - adenine, thymine, cytosine, and guanine (A, T, C, and G), [SR99, Chapter 1]. The nitrogenous bases are also called *base pairs* (bps), as they always pair: A with T, and C with G. The base pairs are situated between the two helixes, thereby binding the two backbones together.

A combination of three bps are termed a *codon*[1]. Each codon specifies an amino acid or a stop code (see Figure 2-1). After a stop code, the bps can possibly specify a non-coding region, which for example could be hundreds of C-T combinations. Thus, 3 bps code a codon, codons code the sequence of amino acids, which again determines the genetic code for a single protein molecule (a gene), [Lan97, page 246]. *Proteins* function as enzymes, hormones, etc. and therefore influence many physiological and psychological *traits*, e.g. eye-color and schizophrenia. Thus, DNA is the recipe for proteins where the genetic code consists of nitrogenous bases interpreted in sequences of three. When the genetic code is translated to a protein, each codon specifies an amino acid. Amino acids are the building blocks of proteins. When a stop codon is reached the translation process is ended and the protein is finished and ready to perform its function, such as to signal other cells to start or stop the production of enzymes, in case the protein just built is a hormone.

Some traits are more dependent on outside stimuli than others, e.g. some traits might not show at all, due to the environmental factors, while other traits only show in the presence of one or more outside stimuli. For example, it is believed that a gene exists, that predisposes individuals for obesity[2]. However, the obesity trait is only shown in the case where plentiful nutrients have been available for an individual with the disease gene.

The sequence of base pairs encoding a single protein is called a *gene*, and the section or position on the chromosome harboring a gene is denoted as the *locus* (plural: loci) of the gene.

A well characterized locus can serve as a genetic *marker*, by well characterized we mean that the locus of the gene is known and that the different allelic states of the gene is known (allelic states will be explained shortly). Markers

[1] Also termed a *triplet*, [KC97, page 325].

[2] We learned that a group of statisticians and biologists are currently working on isolating the gene for this disease, during our stay at deCODE Genetics. Some results have already been released as news, [dNC01].

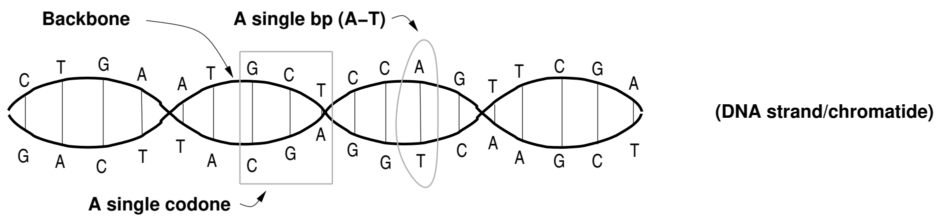

Figure 2-1: A subsection of a chromatide, which is also known as a (subsection of a) strand. The molecular structure is a DNA molecule. Notice the pairing A with T and C with G composing the 18 bps, that are coding 6 codons.

can be used as reference points on a chromosome and they can be analysed in order to determine the genotype of an individual at a given marker, [Gud00, page 9].

2.3 Alleles

An individual has two alleles at each locus of a gene, since one chromosome is received from each parent. An *allelic state* refers to a specific encoding (in codons) of a specific allele. The allelic states on the two chromosomes are not necessarily different, but they can be, hence the sequences of codons coding the alleles, and thereby genes, are different. An individual is said to be *homozygous* if the *maternal* and the *paternal* allelic states (the allele received from the mother and the allele received from the father) at a locus are identical. If the allelic states are different, the individual is said to be *heterozygous* at that locus.

Two alleles at a locus make up the *genotype* of that locus for a specific individual. If a person is heterozygous it is possible that only one of the alleles are *expressed*. In this case, the allele expressed is called *dominant*, while the allele not expressed is called *recessive*. If both alleles are expressed, they are said to be *codominant*. The expression of a genotype is called the *phenotype*.

Example 1 *As an example consider the human blood type. The gene for the human blood type is located on chromosome 9 at band q34, [Lan97, page 2] (band refers to a position on the chromosome). Three allelic states exists for blood type:* **A**, **B**, *and* **0**. *Each person has two alleles, however there exists only four blood types*[3] *(phenotypes):* **A**, **B**, **A/B**, *and* **0**, *even though there are are six possible combinations of allelic states:* **A/A**, **B/B**, **0/0**, **A/B**, **A/0**, *and* **B/0** *(note that the ordering is irrelevant). If an individual has the allelic*

[3]For simplicity we have omitted the fact, that a person can either be resus negative or resus positive.

states **A**/**A**, **B**/**B**, *or* **0**/**0** *the individual is homozygous. In the event that an individual has the allelic states* **A**/**0** *the individual is heterozygous, and for these alleles* **A** *is dominant and* **0** *recessive, since only the* **A** *allele is expressed, giving the person the* **A** *blood type, just as if the person had the genotype* **A**/**A**. **B** *is also dominant with respect to* **0**, *since a person will have blood type* **B** *if the person has either the genotype* **B**/**B** *or* **B**/**0**. *However,* **A** *and* **B** *are codominant, since an individual with the genotype* **A**/**B** *will have neither blood type just* **A** *or* **B**, *but* **A**/**B**. *The final blood type* **0** *is only expressed if a person has the genotype* **0**/**0**.

2.4 Meiosis

In this section we present the meiosis in a simplified version, however, sufficient for understanding the later chapters. Due to the complexity of this biological process we consider it out of scope to explain it in detail[4].

Meiosis is an essential process of the human reproduction. Abstractly, it is a cell division process that produces *gametes*, female egg cells or male sperm cells, which combine in reproduction and become the seed for a new individual. Meiosis is a process where a cell with one chromosome pair (a *diploid* cell) is divided into four cells with one chromatide in each (four *haploid* cells). This is illustrated in the left half (before the second ⇒) of Figure 2-2.

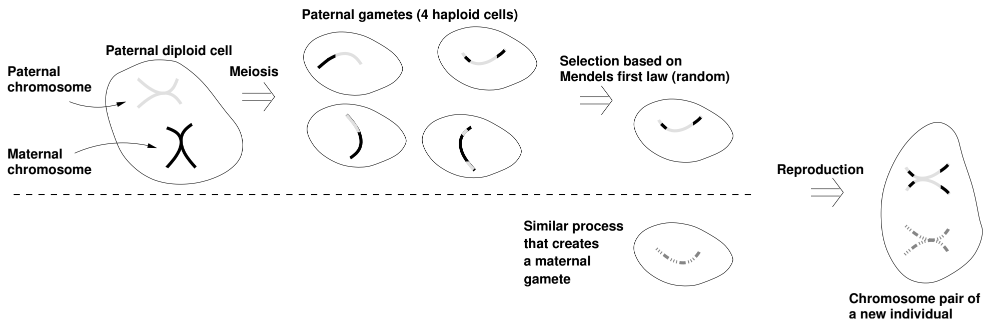

Figure 2-2: Meiosis illustrated for a single chromosome pair of a male and (partially) a female. The results of their meiosis is later combined in reproduction to form the chromosome pair of a new individual. Note that this is simplified, in reality there are 22 more chromosomes in each cell, to which a similar process takes place in parallel.

The probability that a given gamete is passed on to a child is assumed to be random. Hence, probability $\frac{1}{4}$ for each of the four combinations. A single

[4]See [KC97, Chapter 2] for a detailed description of meiosis.

gamete is randomly "selected" for reproduction (see the right half of Figure 2-2). Since a given allele is present in precisely two of the four gametes, the probability that allele being transmitted to an offspring is $\frac{1}{2}$ (from $2 \cdot \frac{1}{4}$). This hypothesis, that the source of inheritance for a single allele is equally likely to be either of the two grandparents, was stated by Gregor Johann Mendel in the 18'th century and is known as *Mendel's first law*, [KC97, Chapter 3].

The pair of alleles received from both the parents at a given locus constitute a child's genotype at that given locus. As an example see the pedigree in Figure 2-3.

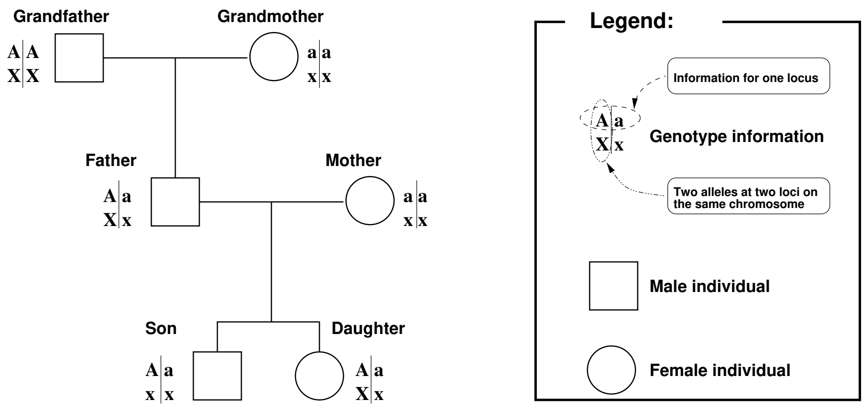

Figure 2-3: An example of a pedigree. The family is constituted by two grandparents, two parents, and two children. Bold letters next to the individuals indicate genotyped information, e.g. the daughter is $\mathbf{A}/\mathbf{a}$ at the first marker and $\mathbf{X}/\mathbf{x}$ at a second marker.

Recombination

Recombination refers to new combinations of genes. This phenomenon arises during meiosis when the two chromosomes of a pair, one inherited from the father and one from the mother, are (most probably) *recombined* to form *four* new unique chromatides, as depicted in Figure 2-2.

The cause of a recombination is the biological event called *crossover*[5]. A crossover involves physical breakage of the chromosome into one paternal and one maternal chromatide, and a joining of the paternal and maternal ends, [SR99, page 40], we have illustrated this in Figure 2-4. Note that we write $\mathbf{A}/\mathbf{a}$ when a person has the allelic states $\mathbf{A}$ and $\mathbf{a}$ at a given locus, whereas we write $\begin{array}{c|c}\mathbf{A}&\mathbf{a}\\\mathbf{X}&\mathbf{x}\end{array}$ when $\mathbf{A}$ and $\mathbf{X}$ reside on one chromosome while $\mathbf{a}$ and $\mathbf{x}$ reside on the other. When two alleles at two different loci reside on the same chromosome

[5] Also known as *crossing over*.

they are said to be *in phase*, otherwise they are *not in phase.* In the following example we assume to know which alleles reside on which chromosomes, hence the phases of the alleles are assumed to be known. When we do not know the phases of a set of alleles, as it is often the case for linkage analysis, the *phase is unknown.*

Example 2 *In the inheritance example of Figure 2-3 the son is a recombinant, whereas the daughter is a nonrecombinant, with respect to the two loci shown. In the example the son is the recombinant, since the alleles on the father's two chromosomes,* $\genfrac{}{}{0pt}{}{\mathbf{A}}{\mathbf{X}}$ *and* $\genfrac{}{}{0pt}{}{\mathbf{a}}{\mathbf{x}}$*, have recombined to form then new* $\genfrac{}{}{0pt}{}{\mathbf{A}}{\mathbf{x}}$ *combination, so* **A** *originates from a different chromosome than* **x**.

Meiosis assures that the number of chromosomes in a human individual is constant, since only one of the four gametes is passed on to a new individual. This gamete is then copied in order to constitute a full chromosome with two sister chromatides for the new individual. Furthermore, meiosis and recombination is the reason that the child's chromosomes, constituting a chromosome pair, are most likely not identical to any of the four chromosome sources of the two parents. Recombinations ensures diversity and uniqueness in a population.

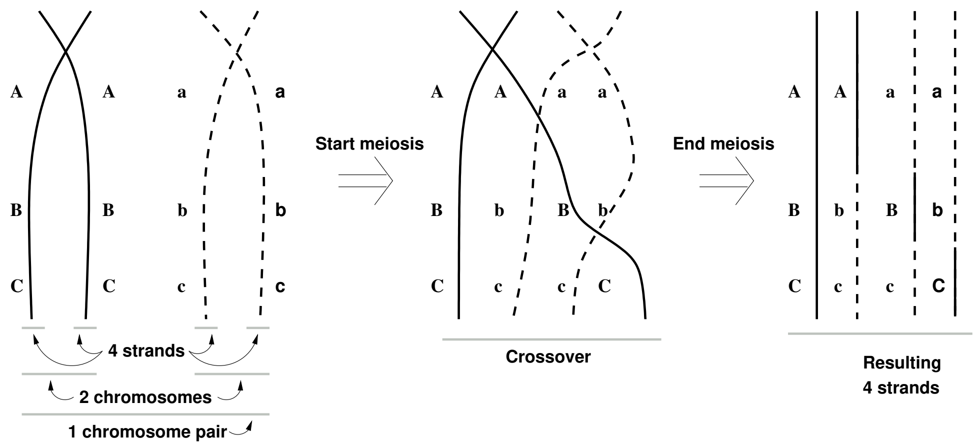

Figure 2-4: The figure illustrates how a crossover might occur on a chromosome, where three marker genes are residing. Hence this meiosis leads to the gametes $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$, $\{\mathbf{A}, \mathbf{b}, \mathbf{c}\}$, $\{\mathbf{a}, \mathbf{B}, \mathbf{c}\}$, and $\{\mathbf{a}, \mathbf{b}, \mathbf{C}\}$, where all but $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$ are recombinations, [Ott99, page 10]. The elements at the starting point is one chromosome pair, two chromosomes, which each consists of two strands (DNA molecules).

The probability that an odd number of crossovers occur between two loci is termed the *recombination fraction.* Note that if an even number of crossovers (including zero) occur between a pair of loci they are constructed from the same parental chromosome. Thus the individual inheriting this chromosome have inherited the genes at this pair of loci from the same ancestral origin.

Example 3 *Assume that a new chromatide is assembled from one end to the other. Assume that the first part of the chromatide is based on one of the paternal chromatides, a crossover occurs, hence the chromatide is now being constructed from a maternal chromatide. If a new crossover occurs (thus we have had two crossovers so far) the source of the assembly is now back to a paternal chromatide. Multiple switches can occur between the paternal and the maternal chromatides.*

In a sense, recombinations do not occur completely at random, [SR99, page 271]. The concept of *interference* states that in the immediate vicinity of a crossover, it is virtually impossible for another crossover to occur. Thus the recombination fraction remains very close to zero for some distance along the chromosome near a crossover. The recombination fraction between two loci is related to their physical distance. The mathematical relationship between physical distance and the recombination fraction is given by a *map function*. The most widely used map function is the Haldane map function, [Gud00, page 7]:

$$\begin{aligned} d &= -\frac{1}{2}ln(1-2\theta), \\ \theta &= \frac{1}{2}\left(1-e^{-2d}\right), \end{aligned}$$

where θ is the recombination fraction and d is the distance in Morgans. On average, approximately one recombination occur for each 10^8 bps. The distance between two loci is defined as 1 *Morgan* (M), if one recombination can be expected between them during meiosis (hence 1 M is roughly equivalent to 10^8 bps). Thus, there is a correspondence between the distance between two loci and the likelihood of a recombination. The greater the distance, the closer the recombination fraction approaches $\frac{1}{2}$, [Ott99, page 14]. Note that between two loci residing on different chromosome pairs the recombination fraction is $\frac{1}{2}$, as each meiosis on either chromosome are independent. Two loci are said to be *linked* if the recombination fraction between them is less than $\frac{1}{2}$, otherwise the two loci are *unlinked*. This is also an observation made by Mendel, in *Mendel's second law*, he states that unlinked, segregating gene pairs assort independently at meiosis, [Ott99, page 38]. We will refer to Mendel's first and second law as *Mendelian inheritance*[6].

[6] Often referred to as *the Mendelian laws*.

Note that the recombination fraction is not constant over the human genome with respect to physical distance. It can deviate along a chromosome and during female meiosis the recombination fraction is on average twice that of male meiosis. In the present study we assume that the recombination fraction is constant, throughout the genome and the same for both male and female[7].

2.5 Locating trait genes

The Human Genome project deals with sequencing the chromosomes so that the order of genes is established. However, when this work is completed, the task still remains to uncover which genes are responsible for what trait[8].

Doctors, physicians, psychologists, psychiatrist, etc. might suspect that a trait within their field is caused by some genetic mutation. The suspicion might arise from the fact that the trait is often carried by many in some families and by none in other families. If the trait causing gene(s) could be located, i.e. if the locus or loci could be identified, it is possible to infer the protein(s) that cause a given trait. This knowledge can be used for:

1. **Diagnosis:** If the locus and the allelic state which is responsible for some trait is known, a blood sample would be sufficient to diagnose a person, since the result of a sequencing of the persons DNA at the locus could be matched against the sequence of the trait causing allelic state. It would also be possible to assess the chance of a person developing a trait, by knowing the persons genotype at the trait locus or loci.

2. **Prescribe effective drugs:** Some people respond to some drugs, while others do not. If the responsiveness is dependent on one or more genes, then a determination of a persons allelic state for those genes can lead to the correct drug prescription.

3. **Cure:** If the proteins responsible for a trait are known, it might be possible to develop a drug which surpasses the effect of the proteins or the proteins themselves.

Two kinds of genetic studies exist in the literature. Both aim at locating genes causing traits: *Association studies* and *linkage analysis*, [Cur01]. In association studies the aim is to locate a trait gene by looking for marker alleles more frequent in a group of unrelated individuals carrying a trait, than to a group of people not carrying the trait. E.g. if two groups of 1000 individuals are studied, and 80% of the individuals in one group - the group of people

[7]Gudbjartsson makes the same assumptions in, [Gud00].

[8]Currently, there is quite a large amount of data available, [BO98], however the computational power is insufficient to process this data, [JIS93].

carrying the trait - carry a certain allele, while the same allele is only carried by 30% of individuals in the other group of people not carrying the trait, there is some evidence that the trait and the allele are close to one another (i.e. they are linked). Association studies are only useful when the trait causing gene is located very close to a marker gene, because it is based on a rather sharp division of the two groups (either they have the trait and the marker allele, or they do not have either).

In linkage analysis related people are studied and different pedigrees are studied independently. The aim in linkage analysis is to find the locus of one or more genes causing a trait, by looking at inheritance patterns of markers and comparing these with the inheritance patterns of the trait in question. By analysing how different markers are segregated compared to the set of effected individuals the location of the trait gene can be approximated. Linkage analysis is the focus of the remaining part of this report.

2.6 Summary

In this chapter we have introduced the reader to the biological background of this report, in order to give a basis for understanding the biological context of the rest of this report. We have explained the structure of a DNA molecules, and how they combine to the 23 chromosomes that encodes the genetic information of an individual. Furthermore, we introduced the reader to the process of genetic inheritance. How the alleles are inherited is based on some probability (Mendelian inheritance), which again is influenced by biological properties like recombination fraction, interference, etc. The main task of linkage analysis is to locate the gene on the human genome, which causes the trait of interest.

From now on we will abstract away from biological details. The focus is now on whole genes (markers), which is sufficient to describe linkage analysis formally. Another prime concept is that of pedigrees. As stated earlier, the glossary on page 107 can provide quick reference.

We now concentrate on the task of linkage analysis, thus locating the possible position of a trait causing gene on a single chromosome. The Mendelian laws of inheritance have a major role in linkage analysis. It is safe only to consider a single chromosome pair, since genes on different chromosome pairs assort independently.

2.7 Further reading

The literature is divided into two categories:

Biology orientated: This literature includes [KC97] and [SR99], which are written for, and by, biologists and geneticists. Both books focus on the physical and chemical processes, as well as cell anatomy. For a computer scientist, much of the information in these books is superfluous for understanding the biological model assumed for the algorithms presented in the following chapters. However, they can serve as an extensive reference.

Statistics orientated: This literature includes [Lan97] and [Ott99], which are written by people with a background in mathematics and/or statistics. All of the books we have read in this category have an introductory section containing a brief description of the biological model and terms used. We have found that this kind of literature are inadequate to the reader with a background in computer science, because the main topic of these books, the statistics models, outweighs the biological descriptions.

In our case the process of learning bioinformatics has largely been an iterative process from biology to statistics, back to biology, etc. This is mainly because we have found little literature on this topic, which was written from a computer scientific perspective.

We have found two references, that is not biology and statistics oriented. *Bioinformatics - A Practical Guide to the Analysis of Genes and Proteins*, [BO98], is largely a guide book for people interested in locating software and data for genetic analysis. It contains user guides for various genetics analysis software, as well as guides on how to find and extract data from public available databases. Another exception is *Faster Sequential Genetic Linkage Computations*, [JIS93]. This is probably the most computer science oriented paper, it incorporates two synthesis of the biological model and four implementation level speed ups of central algorithms. Something that is omitted in the paper is a discussion of the previously mentioned formal techniques, for speeding up computations. This is most probably because the paper is rather old (1993), compared to the development of these techniques within the computer scientific community.

Chapter 3

Linkage Analysis

The purpose of this chapter is to describe the concept of linkage analysis in a formal fashion. First, we motivate linkage analysis by giving an informal description of the concept, based on how it is performed. After this we formalize the underlying biological model, this includes formalizing pedigrees, genotype information, inheritance vectors, and founder allele assignments. The methods implemented in the software currently used to solve the linkage analysis problem consists of two algorithms, namely one to solve single point analysis and one to solve multi point analysis. After introducing the biological model we first focus on single point linkage analysis. We state the problem formally, followed by a formal description and proof of the correctness and complexity of the algorithms implemented in Genehunter and Allegro[1]. Since no previous framework has been established we prove these two applied algorithms. The purpose of multi point is to gain more accurate estimates of linkage. Our formalization is expanded in the description of multi point analysis. We describe multi point probability calculation as a Bayesian network and not as a hidden Markov model, as previously described in the literature, e.g. as presented in [LG87]. Finally, we describe two reductions, one that is applied in both Genehunter and Allegro and the other only in Allegro.

3.1 Introduction to Linkage Analysis

Linkage analysis is a process for constructing *genetic linkage maps*. A genetic linkage map is a description of how a number of loci on a chromosome relate in terms of genetic distance. Hence, a genetic linkage map consists of a number of loci for which the order and relative distance (recombination fraction[2]) are known.

[1] See Appendix A for a list of software used to analyse genetic data.
[2] See page 10.

The recent advances in molecular biology have allowed the construction of detailed genetic linkage maps based on molecular markers[3](from now on just marker). A marker is a polymorphic DNA or protein sequence deriving from a single chromosomal location, [SR99, page 551]. Historically, markers were limited to observable traits known to follow the mendelian[4] inheritance pattern, e.g. as blood groups. Today, however, certain highly polymorphic structures of the human DNA can be typed directly on large scale by automated equipment. These structures are called *microsatellites* which are di-, tri-, and tetraneucleotide repeats, and single nucleorite polymorphisms[5].

In the following we give a overview of how linkage analysis is performed, and which data is available.

3.1.1 Performing Linkage Analysis

The data needed in order to perform linkage analysis is:

- A genetic linkage map.
- A number of pedigrees.
- Information on the inheritance of markers.
- Trait status for some or all pedigree members.

Figure 3-1 depicts a pedigree with the information for one marker. The information is used to infer evidence that the inheritance pattern of the trait resembles the inheritance pattern of a marker. In Figure 3-1 the inheritance pattern of the trait follows the inheritance of the paternal allele of individual 1. The reason for this could be that the trait causing gene is situated close to the marker, i.e. they are linked which means recombination occurs less frequently than had they been unlinked.

Since crossovers occur, it is not always the case that the inheritance pattern of a trait follows exactly that of a marker. For this reason we need a measure of how closely the inheritance patterns of a trait and a marker resembles each other. Such measures are expressed through *scoring functions*. The *LOD score* is one such scoring function. It is out of the scope of this report to define any scoring functions, but for the sake of completeness Appendix B gives formal definitions of two commonly used scoring functions, the LOD and NPL scores.

Linkage analysis is performed for a number of pedigrees on a subset of markers across the genome. If the scoring between the trait and a marker indicates potential linkage, then that region is further analysed by including

[3]See page 6.

[4]See page 9 and 11.

[5]See [SR99, page 273] for a detailed description of the history, use, and typing of markers.

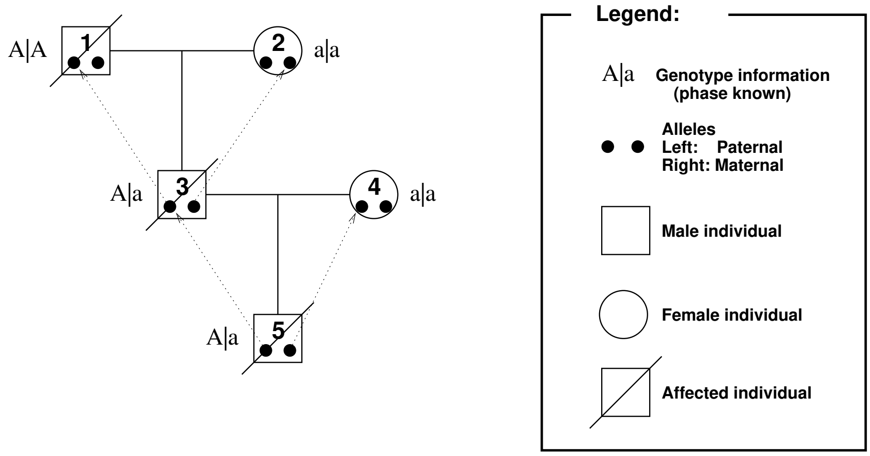

Figure 3-1: A pedigree with trait status and exact information on inheritance of alleles. Note that in reality the phase is often not known.

more markers from the genetic linkage map. Given that trait causing gene is located in the region under investigation, the new analysis estimates more precisely the location of the trait causing gene.

This process is iterated until no more markers are available in the proximity of the trait causing gene. The genetic linkage map is then updated by introducing a locus representing the trait causing gene.

Using exact knowledge of the inheritance patterns of markers, linkage analysis is straight forward, but in reality many pedigree members are not genotyped making it impossible to infer the precise pattern of inheritance. This and other problems are discussed in the following section.

3.1.2 Missing and Incomplete Data

As stated above, markers were originally limited to simple characteristic traits, such as blood type or eye color. Recent techniques have made it possible to determine the allelic states of microsatellites in human DNA. This method has only been used for a few decades, therefore genotype information is often only available for the youngest generations. Since pedigrees are partly constructed from written documents, such as church records, which contain no information about genetic markers, some individuals in the pedigree might not be genotyped at all. This is often the case for pedigree founders. Furthermore, the phase of the observed alleles for a marker cannot be determined.

Due to these problems, the inheritance pattern for each marker needs to be *estimated* on behalf of the genotyped individuals. Since the genotype information is insufficient and the phase is unknown, the precise inheritance pattern becomes impossible to infer. To illustrate the problem we introduce the concepts of *identical/identity by descent*(IBD) and *identical/identity by state*(IBS). Two individuals are IBD if they share inheritance of some founder allele. Two individuals are IBS if they have the same allelic state for some marker, these alleles need not be IBD.

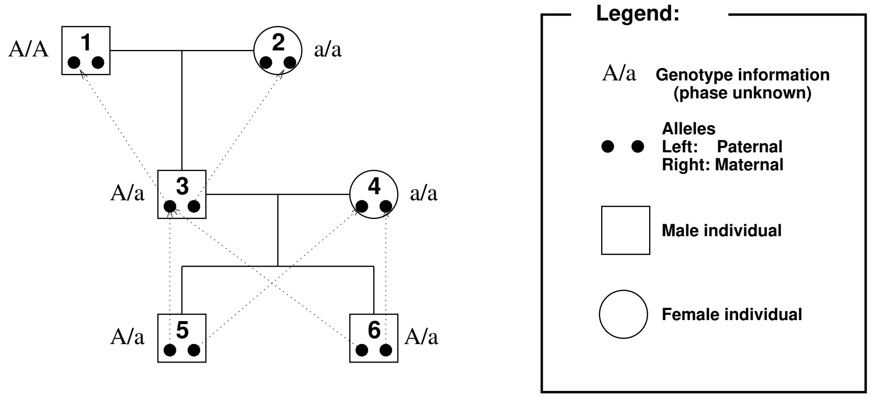

Figure 3-2: IBS vs. IBD. Individuals 5 and 6 are IBS, but since the mother is homozygous they have not necessarily inherited the same maternal allele. In fact in this case they are not IDB.

Figure 3-2 is a example of two siblings which are IBS, but not IBD. The individuals 5 and 6 share inheritance of their paternal alleles (A). The allelic states of their maternal alleles (a) are identical, but they are not identical by descent because of the pattern of inheritance depicted in the figure.

For the reasons given, the precise inheritance pattern is impossible to infer from the given data, that is, several inheritance patterns can give rise to the same observed genotype information. Some of these inheritance patterns might be more likely than others, thus we need to compute the individual probability of each. These computations are the topic of the rest of this chapter.

Each possible inheritance pattern can then be scored against the inheritance pattern of the trait, using for instance the LOD score. The contribution to the final score of the marker from each inheritance pattern is based on the probability of that pattern.

Computing the probabilities of possible inheritance pattern is performed in two steps, *single point analysis* and *multi point analysis*. These topics are covered in Sections 3.2 and 3.4, respectively.

3.2 Single Point Analysis

Single point analysis is the process of determining linkage between single markers and traits using only the genotype information available at each marker. The primary task is computing probability distributions of inheritance patterns at some locus given the available genotype information at that locus. This is called *single point probability computation*.

To compute the probability distribution over inheritance patterns, we introduce a formal framework for the biological concepts introduced in the preceding chapter. After the introduction of the formal framework we explain formally how the probability distribution is calculated.

3.2.1 Formal Model

Pedigree

In Section 2.4 the concept of a pedigree was introduced. In the following we formalize this concept.

Definition 1 (Pedigree) *A pedigree is a 4-tuple* $P = \langle F, N, father, mother\rangle$ *with a set of individuals (nodes)* $V = F \cup N$*, where* F *is the founder set and* N *is the non-founder set, and two functions:*

- $father : N \rightarrow V$.

- $mother : N \rightarrow V$, *satisfying the following constraints:*

- *A node cannot be both a founder and a non-founder, that is* $F \cap N = \emptyset$.

- *No node is both a father and a mother, that is:*

$$\begin{aligned} \forall n, n' \in N \text{ and } n'' \in V \;.\; mother(n) = n'' &\implies father(n') \neq n'' \text{ ; and} \\ \forall n, n' \in N \text{ and } n'' \in V \;.\; father(n) = n'' &\implies mother(n') \neq n''. \end{aligned}$$

- *A node is never its own ancestor, that is:*

$$\forall n \in V \;.\; n \notin ancestor(n),$$

  where $ancestor : V \rightarrow 2^V$ *is defined recursively as:*

  - *Any non-founder has its father and its mother as ancestors together with the ancestors of both parents, that is:*

$$\begin{aligned} \forall n \in N \;.\; ancestor(n) &= ancestor(father(n)) \\ &\cup ancestor(mother(n)) \\ &\cup father(n) \;\cup\; mother(n). \end{aligned}$$

– *Founders have no ancestors, that is:*

$$\forall n \in F \; . \; ancestor(n) = \emptyset.$$

The constraints on the pedigree are introduced to consistently describe the underlying biological model.

Definition 2 (Pedigree Graph) *The graph $G = \langle P, \rightarrow \rangle$ consisting of a pedigree P and a transition relation $\rightarrow \subset V \times N$, where:*

$$n \rightarrow n' \text{ iff } father(n') = n \text{ or } mother(n') = n,$$

is called the pedigree graph generated by $P = \langle F, N, mother, father \rangle$.

We use the relations $n \xrightarrow{father} n'$ and $n \xrightarrow{mother} n'$ to denote $father(n') = n$ and $mother(n') = n$, respectively. From this definition it follows that:

$$\rightarrow = \{(n, n') \mid (n, n') \in \xrightarrow{father} \vee (n, n') \in \xrightarrow{mother}\} = \xrightarrow{father} \cup \xrightarrow{mother} .$$

Remark: This implies the following relationship between ancestors and the $\rightarrow$ relation:

$$n \rightarrow^{+} n' \Leftrightarrow n \in ancestor(n'), \tag{3-1}$$

where $n \rightarrow^{+} n'$ denotes that there is a path in the pedigree graph of length at least 1 from n to n'.

Example 4 *Figure 3-4 is the representation of the example pedigree in Figure 3-3 under the formal framework. In this example the pedigree graph,* G, *has the following structure:*

$$\begin{aligned}
\mathsf{G} &= \langle P, \rightarrow \rangle, \text{ where:} \\
P &= \langle F, N, mother, father \rangle, \\
F &= \{1, 2, 4\}, \\
N &= \{3, 5\}, \\
mother(3) &= 2, \\
father(3) &= 1, \\
mother(5) &= 4, \text{ and} \\
father(5) &= 3.
\end{aligned}$$

Figure 3-4 illustrates how the individuals of the example above are related in terms of the $father$ and $mother$ functions.

Lemma 1 *A pedigree graph is a directed, acyclic graph (DAG).*

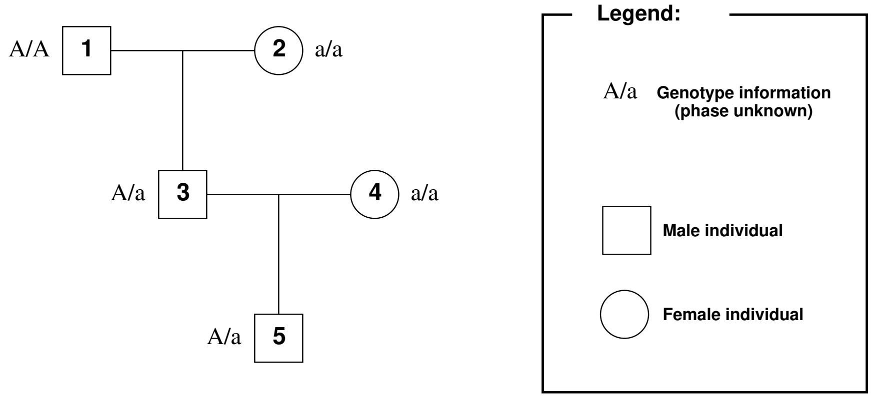

Figure 3-3: An example pedigree with genotype information for which the phase is unknown.

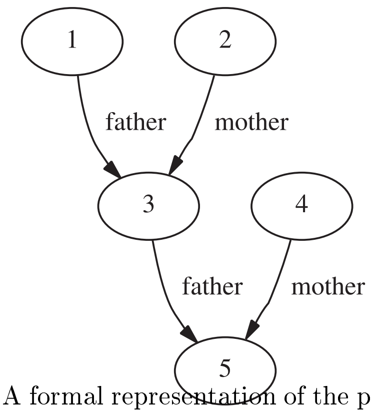

Figure 3-4: A formal representation of the pedigree in Figure 3-3. The transitions have been labeled to clarify whether a parent is *father* or *mother*.

Proof: The proof is by contradiction stating that a cycle dictates that some node would be its own ancestor. Assume that a pedigree graph $G = \langle P, \rightarrow \rangle$ contains a cycle consisting of the nodes $V' \subseteq V$, where:

$$\forall n \in V'.n \rightarrow^{+} n.$$

It follows directly from (3-1) on page 20, that $n \in ancestor(n)$, which contradicts Definition 1 of a pedigree. $\diamondsuit$

Allele

Definition 3 (Allele) *The set of alleles in a pedigree P is defined as:*

$$\mathsf{A} = \{(n, \varpi) |\ n \in V \ \ and\ \varpi \in \{p, m\}\}.$$

We call the subsets $\mathsf{A}_N = \{(n, \varpi)|\ n \in N \text{ and } \varpi \in \{p, m\}\}$ and $\mathsf{A}_F = \{(n, \varpi)|\ n \in F \text{ and } \varpi \in \{p, m\}\}$ the *set of non-founder alleles* and the *set of founder alleles*, respectively. (n, p) and (n, m) represent the paternal and maternal alleles.

Marker

Definition 4 (Marker) *A marker[6] M is defined as a 2-tuple $M = \langle A_M, \pi_M \rangle$, where:*

- $A_M = \{a_1, a_2, \ldots, a_k\}$ *is set of allelic states for the marker M, and*
- $\pi_M : A_M \rightarrow [0, 1]$ *is the allele frequency[7] function, such that:*
  - *The sum of all allele frequencies is 1, that is:*

$$\sum_{a \in A_M} \pi_M(a) = 1.$$

The allele frequencies are based on statistical information for a given population.

Notice the difference between the terms *allele* and *allelic state*. An allele is the term used for the section (also denoted position) of a chromosome representing a marker. An allele can have a number of states, depending on the allelic states for the marker. Thus, an allele can be thought of as a variable which can be assigned a value, that is, an allele can be assigned an allelic state.

[6] See page 6.

[7] A more appropriate term would be *allelic state frequency*, however to remain consistent with the current practice we call it *allele frequency*.

For the remaining part of single point analysis, we assume an underlying pedigree, and we always work with a single marker, which we call M. Thus, when we write A_M or π_M we refer to allelic states or the allele frequencies for the marker M.

Genotype information

Definition 5 (Genotype Information) *Genotype*[8] *information,* $\mathcal{G} = \langle \mathcal{L}, astates \rangle$*, for a gene, g, consists of a set of genotyped individuals $\mathcal{L}$ and a function astates, where:*

- $\mathcal{L} \subseteq V$*, and*
- $astates \,:\, \mathcal{L} \rightarrow \{\{a_1, a_2\} \mid a_1, a_2 \in A_M\}$.

Notice that the definition indicates that the phase is unknown since there is no reference to maternal and paternal alleles. If an individual is homozygous at a given locus, the *astates* function returns a set containing just one element.

Example 5 *The pedigree graph in Figure3-3 shows genotypic information for all individuals. Thus the set of genotyped individuals $\mathcal{L}$ is equal to the complete node set V. The allelic states of individual 5 are: $astates(5) = \{A, a\}$.*

Inheritance Vector

An inheritance vector[9] is a complete specification on how the alleles of founders are inherited by each non-founder. This is a formal way of describing inheritance patterns. We formalize the notion of *inheritance vectors* in the following definition:

Definition 6 (Inheritance Vector) *An inheritance vector v for some pedigree graph P is a function:*

$$v \,:\, N \times \{p, m\} \rightarrow \{p, m\}.$$

The inheritance vector is used to give information about the inheritance of the paternal, p, and maternal, m, alleles for every non-founder (See the example below).

[8] See Section 2.3 for the biological meaning of *genotypes* and *alleles*.

[9] The notion of *inheritance vectors* was introduced in [LG87] and further used in [KDRDL96].

Example 6 *In the running Example 4 on page 20, an inheritance vector could be:*

$$\begin{aligned} v(3,p) &= p, \\ v(3,m) &= m, \\ v(5,p) &= p, \text{ and} \\ v(5,m) &= p. \end{aligned}$$

This would result in the following inheritance pattern (see Figure 3-5):

Individual 3 has inherited his paternal allele from the father of individual 1 and his maternal allele from the mother of individual 2. The two alleles of individual 1 and the two alleles of individual 2 are founder alleles. *Hence, individual 3 has inherited the paternal allele of a founder, individual 1, and the maternal allele of another founder, individual 2. Note that the father of individual 5 (individual 3) does have parents in the pedigree. To determine which founder allele individual 5 has inherited, we use the information that the paternal allele is transmitted from the father's father. This means, that we can find the transmitted founder allele by searching the graph recursively using the inheritance information of individual 3. Thus the paternal allele of individual 5 is the same as the paternal allele of individual 3. Since $v(5,m) = p$ individual 5 has inherited the paternal allele of individual 4, who is a founder.*

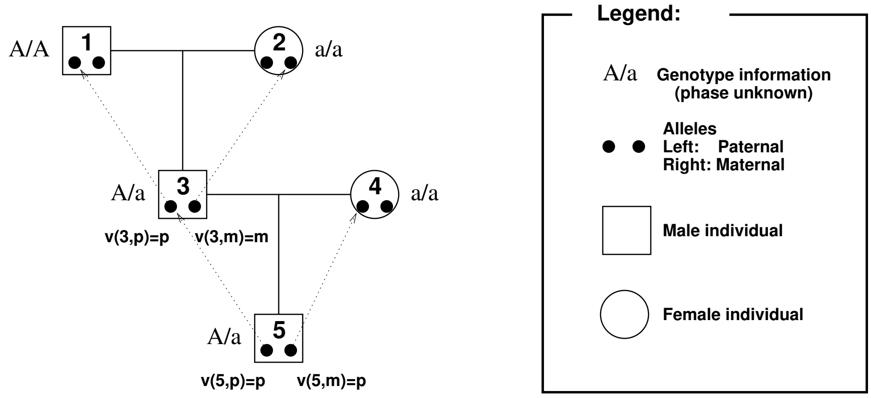

Figure 3-5: Shows how it is determined which founder alleles the non-founders receive given an inheritance pattern.

Founder Allele Assignment

We need a definition of a *founder allele assignment*, which assigns allelic states to all founder alleles. Such a definition is necessary in order to reason about allelic states of non-founders, since every non-founder has inherited his/her alleles from founders.

Definition 7 (Founder Allele Assignment) *An assignment of allelic states to founder alleles is a function:*

$$Z_M : F \times \{p, m\} \to A_M.$$

The founder allele assignment, Z_M, assigns allelic states to founders, whereas the inheritance vector, v, only gives information about the path of inheritance of alleles, not the exact allelic state.

Example 7 *Below we show the only founder allele assignment for the example:*

$$\begin{aligned} Z_M(1,p) &= A, \\ Z_M(1,m) &= A, \\ Z_M(2,p) &= a, \\ Z_M(2,m) &= a, \\ Z_M(4,p) &= a, \text{ and} \\ Z_M(4,m) &= a. \end{aligned}$$

This is because all founders are genotyped.

Assignment of Alleles to Non-founders

From the inheritance pattern, given by an inheritance vector, it is possible to determine which founder alleles a non-founder has inherited. This can be done by the function defined below:

Definition 8 *We define a function:*

$$\mathsf{F} : (N \times \{p, m\} \to \{p, m\}) \to ((V \times \{p, m\}) \to (F \times \{p, m\})),$$

which given an inheritance vector returns a recursive function that for each allele of each individual in the pedigree gives the founder allele inherited. That is:

$$\begin{aligned} \mathsf{F}^v(n,p) &= \begin{cases} (n,p) & : n \in F \\ \mathsf{F}^v(father(n), v(n,p)) & : \text{otherwise} \end{cases} \\ \mathsf{F}^v(n,m) &= \begin{cases} (n,m) & : n \in F \\ \mathsf{F}^v(mother(n), v(n,m)) & : \text{otherwise,} \end{cases} \end{aligned}$$

where F^v denotes $\mathsf{F}(v)$.

Example 8 *Using the inheritance vector, v, of Example 6, F^v assigns founder alleles to both non-founders according to:*

$$\begin{aligned}
\mathsf{F}^v(5,p) &= \mathsf{F}^v(father(5),p) && (\text{since } v(5,p)=p)\\
&= \mathsf{F}^v(3,p) = (father(3),p) && (\text{since } v(3,p)=p)\\
&= (1,p),\\
\mathsf{F}^v(5,m) &= (mother(5),p) = (4,p) && (\text{since } v(5,m)=p),\ \textit{and}\\
\mathsf{F}^v(3,m) &= (mother(3),m) = (2,m) && (\text{since } v(3,m)=m).
\end{aligned}$$

Compatibility

Kruglyak et. al. define the concept *v-compatibility* in [KDRDL96]. This term is an indication of whether the given inheritance vector is compatible with the genotypes observed. We formalize this notion below:

Definition 9 (Compatibility) *Let v be an inheritance vector and $\mathcal{G} = \langle \mathcal{L}, astates \rangle$ some genotype information. Then v is said to be compatible with $\mathcal{G}$ iff there exists a founder allele assignment Z_M such that the following condition is met:*

$$\forall n \in \mathcal{L} \,.\, \{a \mid a = Z_M(\mathsf{F}^v(n,p)) \ \vee\ a = Z_M(\mathsf{F}^v(n,m))\} = astates(n),$$

Z_M is then said to be a compatible founder allele assignment over v and $\mathcal{G}$.

Example 9 *The founder allele assignment of Example 7 is compatible with the inheritance vector of Example 6 and the genotype information of Example 5.*

Had the inheritance vector been changed such that $v(5,p) = m$ there would be no compatible founder allele assignment. This is because both individual 2 and 4 are genotyped with only the a allelic state and individual 5 is genotyped with one A, which could only have been inherited from individual 1.

3.2.2 Single Point Probability Computation

In the previous section we saw how some genotype information yields no compatible founder allele assignment for some inheritance vector. The main task in single locus probability calculation is exactly identifying compatible inheritance vectors and calculating their probability. For readers unfamiliar with basic probability theory we refer to Appendix C.

The probability of an inheritance vector given some genotype information is proportional to the sum of the probabilities of all compatible founder allele assignments (3-5). The probability of a founder allele assignment is given by the product of the allele frequencies[10] for all assigned founder alleles.

[10] See Section 3.2.1 for more on allele frequencies.

Formally, given an inheritance vector v and some genotype information $\mathcal{G}$, we are interested in computing $P(v|\mathcal{G})$, the probability of an inheritance vector, v, given some genotype information, $\mathcal{G}$. Applying Bayes' theorem we find:

$$P(v|\mathcal{G}) = \frac{P(\mathcal{G}|v) \cdot P(v)}{P(\mathcal{G})}.$$

However, since the inheritance vector for a pedigree graph is almost never uniquely determined[11] by the genotype information, each inheritance vector has a probability of denoting the true inheritance pattern.

If we let $\mathbf{v}$ denote all inheritance vectors for a pedigree we get:

$$P(\mathbf{v}|\mathcal{G}) \quad = \quad \frac{P(\mathcal{G}|\mathbf{v}) \cdot P(\mathbf{v})}{P(\mathcal{G})}, \tag{3-2}$$

where $P(\mathbf{v})$ denotes the probability distribution over all inheritance vectors. Formally, we say that $\mathbf{v}$ is a variable with states $\mathbf{v} = (v_1, v_2, ..., v_n)$, where each v_i is an inheritance vector. $P(\mathbf{v})$ denotes the probability distribution over these states, that is $P(\mathbf{v}) = (P(v_1), P(v_2), ..., P(v_n))$, where $\sum_{i=1}^{n} P(v_i) = 1$. $P(\mathbf{v})$ is called the ***probability distribution over inheritance vectors***. We use $P(v_i)$ for $P(\mathbf{v} = v_i)$ when the variable is understood. This notation is adopted from [Jen01] and [Gud00]. In (3-2) $P(\mathbf{v}|\mathcal{G})$ refers to the probability distribution $P(\mathbf{v}|\mathcal{G}) = (P(v_1|\mathcal{G}), \ldots, P(v_n|\mathcal{G}))$.

Notice that the denominator in (3-2) is an instance of genotype information, thus it is the same for all v. This means that the probability $P(\mathbf{v}|\mathcal{G})$ is proportional to $P(\mathcal{G}|\mathbf{v})P(\mathbf{v})$. We write this as:

$$P(\mathbf{v}|\mathcal{G}) \propto P(\mathcal{G}|\mathbf{v}) \cdot P(\mathbf{v}). \tag{3-3}$$

Since the inheritance is assumed to be Mendelian[12] the a priori probability of all inheritance vectors v is equal. That is:

$$\forall v_i \,.\, P(\mathbf{v} = v_i) = \frac{1}{|\mathbf{v}|}, \quad 1 \leq i \leq |\mathbf{v}|$$

where $|\mathbf{v}|$ denotes the number of states of $\mathbf{v}$. Under this assumption we deduct from (3-3) that:

$$P(\mathbf{v}|\mathcal{G}) \propto P(\mathcal{G}|\mathbf{v}). \tag{3-4}$$

As can be seen from (3-4) above, the probability of an inheritance vector v given some genotype information $\mathcal{G}$ is proportional to the genotype information given the inheritance vector. We use this result when computing the single point probability distribution.

[11] This is because the phase of the genotype information is usually unknown.

[12] See page 9 and 11 for more on Mendelian inheritance.

The computation of the probability of some genotype information given an inheritance vector is the sum of the probabilities of all compatible founder allele assignments. Let Z_M be a founder allele assignment of a pedigree then the probability of Z_M is defined as:

$$P(Z_M) = \prod_{n \in F} \pi_M(Z_M(n,p)) \cdot \pi_M(Z_M(n,m)),$$

where π_M is the probability function over allele frequencies. If $\mathcal{F}_{\mathcal{G}}^{v}$ is the set of all compatible founder allele assignments for some inheritance vector v and some genotype information $\mathcal{G}$, that is, $\mathcal{F}_{\mathcal{G}}^{v} = \{Z_M |\ Z_M$ is a compatible founder allele assignment over $\mathcal{G}$ and $v\}$ then the conditional probability is obtained by:

$$P(\mathcal{G}|v) \propto \sum_{Z_M \in \mathcal{F}_{\mathcal{G}}^{v}} P(Z_M). \tag{3-5}$$

This is a general formulation of single point probability for an inheritance vector. In the following section we will state and prove the correctness of two algorithms used to calculate the single point probability distribution over the set of inheritance vectors for a pedigree.

3.3 Algorithms for Single Point Probability Computation

In this section we present and prove the correctness of the two most recently developed algorithms for single point probability calculation. The first algorithm, Kruglyak, is used in the Genehunter software package, and was developed by Kruglyak et. al. and presented in [KDRDL96]. The second algorithm, FastTreeTraversal is used in the Allegro software package and it is presented in [Gud00]. We use the formal framework presented in Section 3.2.1. To our knowledge the correctness of the two algorithms has not previously been formulated and proven.

We wish to prove that the two algorithms do indeed find the single point probability distribution $P(\mathbf{v}|\mathcal{G})$[13]. The proof idea is to show, that the set of compatible founder allele assignments $\mathcal{F}_{\mathcal{G}}^{v}$ for an inheritance vector v can be described by a propositional formula, and that both algorithms implicitly construct correct and equivalent formulae for each inheritance for a pedigree.

Prior to the formulations and the proofs of the algorithms, we establish the notation needed to express the founder allele assignments in propositional logic.

[13] Actually, the algorithms find the distribution $P(\mathcal{G}|\mathbf{v})$, but since $P(\mathbf{v}|\mathcal{G}) \propto P(\mathcal{G}|\mathbf{v})$ the distribution is just normalized after an algorithm has terminated.

3.3.1 Founder allele assignment in propositional logic

In the following we build syntax and semantics of propositional formulae that allow us to reason about correctness of single point linkage analysis algorithms in terms of compatible founder allele assignments.

The semantics is build in such a way that we interpret propositional formulae in the sense of sets of compatible founder allele assignments. In this way we build a formula over some genotype information and an inheritance vector where the semantics is given as the set of compatible founder allele assignments in the pedigree.

The section is organized such that we first introduce the propositional variables, then we introduce the syntax and semantics thereof. Finally, we build a propositional formula over the genotype information and the inheritance vector and prove that the semantics of this formula is exactly the set of compatible founder allele assignments.

To be able to work with propositional logic we assume, that each founder allele $f \in F \times \{p, m\}$ in a pedigree is given by a vector of binary variables $f = (x_1, x_2, ..., x_u)$, where u is $\lceil \log_2(|A_M|) \rceil$. We need $\lceil \log_2(|A_M|) \rceil$ binary variables for founder alleles, since there are $|A_M|$ allelic states, and we need a unique, binary encoding for each of these, since we want to be able to assign an allelic state to each founder allele. We also assume that we have an enumeration on the allelic states in A_M so that each allele $a \in A_M$ has an unique vector representation of binary values $a = (\alpha_1, \alpha_2, \ldots, \alpha_u)$. For readability we abbreviate the expression:

$$x_1 = \alpha_1 \wedge x_2 = \alpha_2 \wedge ... \wedge x_u = \alpha_u,$$

by:

$$f_{n,\varpi} = a,$$

where $\varpi \in \{p, m\}$ and a is the binary representation of an allelic state, and $f_{n,\varpi}$ is the vector of binary variables representing some founder allele. We use $f_{\mathsf{F}^v(n,\varpi)}$ to denote the founder allele which individual $n \in V$ has inherited from the paternal (p) or maternal (m) side. Hence, $f_{\mathsf{F}^v(n,p)}$ is the binary representation of $\mathsf{F}^v(n, p)$, and $f_{\mathsf{F}^v(n,m)}$ is the binary representation of $\mathsf{F}^v(n, m)$.

The reason for explicitly describing the encoding of the propositions is to illustrate that we do not need to model explicitly that a founder allele can only be assigned one allelic state. This is guaranteed by the encoding. In the rest of this section we assume that this mutual exclusion is guaranteed.

Example 10 *If $A_M = \{a, b, c, d\}$ is the set of allelic states for the marker M, and we enumerate the allelic states in the order shown, they have the following binary vector representations: $a = [0, 0], b = [0, 1], c = [1, 0]$, and $d = [1, 1]$. We need 2 bits since there are 4 allelic states in A_M and $\lceil \log_2(|4|) \rceil = 2$.*

Now, assume that we have a pedigree with two founders, a mother (n_1) and a father (n_2) and one non-founder, their child (n_3). Since there are two founders, each with two alleles, we have four founder alleles. Each of these founder alleles, f_{1p}, f_{1m} for the mother (n_1), and f_{2p}, f_{2m} for the father (n_2), can be represented as a vector of binary variables such that:

$$\begin{aligned} f_{1p} &= [x_{1p1}, x_{1p2}], \\ f_{1m} &= [x_{1m1}, x_{1m2}], \\ f_{2p} &= [x_{2p1}, x_{2p2}], \ \textit{and} \\ f_{2m} &= [x_{2m1}, x_{2m2}]. \end{aligned}$$

Assume the following:

- *$astates(n_3) = \{a, d\}$. Thus the child has been genotyped with the two allelic states a and d, and assume that*
- *the inheritance vector v for the pedigree is defined as $v(n_3, m) = p$ and $v(n_3, p) = m$. Thus the child has inherited the paternal founder allele from its mother (f_{1p}) and the maternal founder allele from its father (f_{2m}).*

Since the child must have received either a or d from its mother and the other from the father, we can write this as:

$$(f_{1p} = a \ \textit{and} \ f_{2m} = d) \ \textit{or} \ (f_{1p} = d \ \textit{and} \ f_{2m} = a),$$

which is an abbreviation for the propositional expression:

$$\begin{aligned} \big((x_{1p1} = 0 \quad &\wedge \quad x_{1p2} = 0) \wedge (x_{2m1} = 1 \wedge x_{2m2} = 1)\big) \vee \\ \big((x_{1p1} = 1 \quad &\wedge \quad x_{1p2} = 1) \wedge (x_{2m1} = 0 \wedge x_{2m2} = 0)\big). \end{aligned}$$

We build propositional formulae, φ, for genotype information using the following syntax:

$$\begin{aligned} \varphi &:= \varphi \wedge \varphi \mid \psi \mid \mathtt{tt} \\ \psi &:= \vartheta \vee \vartheta \\ \vartheta &:= q \wedge q \end{aligned} \tag{3-6}$$

where q is a proposition of type $f = a$, such that $f \in F \times \{p, m\}$ and $a \in A_M$. The syntax is build such that genotype information is easily expressed, which becomes apparent as we construct formulae.

Let $a \in A_M$ be an allelic state and $f \in F \times \{p, m\}$ be a founder allele, then formulae, φ, build from the syntax of (3-6) are interpreted using the following semantics:

$$
\begin{aligned}
[\![\mathtt{tt}]\!] &= \{Z_M \mid Z_M \text{ is a founder allele assignment}\} \\
[\![f = a]\!] &= \{Z_M \mid\ Z_M(f) = a\} \\
[\![q_1 \wedge q_2]\!] &= [\![q_1]\!] \cap [\![q_2]\!] \\
[\![\vartheta_1 \vee \vartheta_2]\!] &= [\![\vartheta_1]\!] \cup [\![\vartheta_2]\!] \\
[\![\varphi_1 \wedge \varphi_2]\!] &= [\![\varphi_1]\!] \cap [\![\varphi_2]\!].
\end{aligned}
\tag{3-7}
$$

Two formulae φ_1 and φ_2 are said to be equivalent if they describe the same set of founder allele assignments. We write this as:

$$\varphi_1 \sim \varphi_2 \Leftrightarrow [\![\varphi_1]\!] = [\![\varphi_2]\!].$$

Furthermore, we say that a founder allele assignment, Z_M, satisfies a formula, φ, if Z_M is an element of the semantics of φ, that is:

$$Z_M \models \varphi \text{ iff } Z_M \in [\![\varphi]\!].$$

We now state an important theorem for proving single point linkage analysis algorithms for finding compatible founder allele assignments. The theorem states that from some genotype information and some inheritance vector over a pedigree, we can build a propositional formula from the syntax of (3-6) for which the semantics (3-7) is exactly the set of compatible founder allele assignments, $\mathcal{F}^v_{\mathcal{G}}$.

Theorem 1 (Allele Assignment Formula) *The set of compatible founder allele assignments $\mathcal{F}^v_{\mathcal{G}}$ over an inheritance vector v and some genotype information $\mathcal{G} = \langle \mathcal{L}, astates \rangle$ is the set described by the semantics of the propositional formula $\varphi^v_{\mathcal{G}}$:*

$$\varphi^v_{\mathcal{G}} \stackrel{\text{def}}{=} \bigwedge_{n \in \mathcal{L}} \big((f_{\mathsf{F}^v(n,p)} = a_1 \wedge f_{\mathsf{F}^v(n,m)} = a_2) \vee (f_{\mathsf{F}^v(n,p)} = a_2 \wedge f_{\mathsf{F}^v(n,m)} = a_1) \big), \tag{3-8}$$

where $\{a_1, a_2\} = astates(n)$. a_1 and a_2 are different if $astates(n)$ contains two elements (heterozygous) and identical if it contains one element (homozygous). In case of no genotyped individuals ($\mathcal{L} = \emptyset$) then $\varphi^v_{\emptyset} = \mathtt{tt}$, hence all founder allele assignments are compatible.

Proof: Recall that $\mathcal{F}^v_{\mathcal{G}}$ is defined as the set of all compatible founder allele assignments. By definition Z_M is compatible[14] with v and $\mathcal{G}$ iff:

[14] See page 26 for more on *compatibility*.

$$\forall n \in \mathcal{L} \,.\, \{a \mid Z_M(\mathsf{F}^v(n,p)) = a \text{ or } Z_M(\mathsf{F}^v(n,m)) = a\} = astates(n).$$

We can rewrite this and describe $\mathcal{F}^v_{\mathcal{G}}$ as set intersection, that is:

$$\mathcal{F}^v_{\mathcal{G}} = \bigcap_{n\in\mathcal{L}} \{Z_M \mid \{a \mid Z_M(\mathsf{F}^v(n,p)) = a \text{ or } Z_M(\mathsf{F}^v(n,m)) = a\} = astates(n)\}. \tag{3-9}$$

By definition the semantics of $\varphi^v_{\mathcal{G}}$ is given as:

$$[\![\varphi^v_{\mathcal{G}}]\!] = \bigcap_{n\in\mathcal{L}} [\![(f_{\mathsf{F}^v(n,p)} = a_1 \wedge f_{\mathsf{F}^v(n,m)} = a_2) \vee (f_{\mathsf{F}^v(n,p)} = a_2 \wedge f_{\mathsf{F}^v(n,m)} = a_1)]\!], \tag{3-10}$$

where $\{a_1, a_2\} = astates(n)$.

To complete the proof, what we need to show is that the inner part of (3-9) equals the inner part of (3-10), that is:

$$\begin{gathered}\{Z_M \mid \{a \mid Z_M(\mathsf{F}^v(n,p)) = a \text{ or } Z_M(\mathsf{F}^v(n,m)) = a\} = astates(n)\} \\ = \\ [\![(f_{\mathsf{F}^v(n,p)} = a_1 \wedge f_{\mathsf{F}^v(n,m)} = a_2) \vee (f_{\mathsf{F}^v(n,p)} = a_2 \wedge f_{\mathsf{F}^v(n,m)} = a_1)]\!].\end{gathered}$$

In the heterozygous case both formulae have only compatible founder allele assignments which assigns a_1 to $f_{n,p}$ and a_2 to $f_{n,m}$ or the other way around. In the homozygous case only founder allele assignments which assign the same allelic state a_1 to both $f_{n,p}$ and $f_{n,m}$ are compatible. Therefore no matter how individuals in the pedigree are genotyped, the semantics of $\varphi^v_{\mathcal{G}}$ describes exactly the set of compatible founder allele assignments. $\diamondsuit$

We say that the founder alleles in a sub-expression:

$$(f_{n,p} = a_1 \wedge f_{n,m} = a_2) \vee (f_{n,p} = a_2 \wedge f_{n,m} = a_1),$$

of $\varphi^v_{\mathcal{G}}$, where $a_1 \neq a_2$ are *ambiguously assigned*. In the case where $a_1 = a_2$ the sub-expression simplifies to:

$$f_{n,p} = a_1 \wedge f_{n,m} = a_1,$$

and we say that the founder alleles $f_{n,p}$ and $f_{n,m}$ are *unambiguously assigned*. In case $\varphi^v_{\mathcal{G}}$ does not contain a founder allele $f_{n,p}$, we say that the founder allele is *free*. A founder allele is free if no genotyped individual has inherited the allele, and the founder has not been genotyped.

3.3.2 Kruglyak's Algorithm for Single Point Probability Computation

The algorithm presented in [KDRDL96] is implemented in the software package Genehunter[15]. The algorithm takes a brute force approach to determine all the compatible founder allele assignments by building a graph for each inheritance pattern given by an inheritance vector. The brute force aspect refers to the fact, that the algorithm iterates over all possible inheritance vectors.

Let founder alleles be nodes in a graph, and let the edges be labeled with exactly two allelic states $a_1, a_2 \in A_M$. Let:

$$f_1 \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_2,$$

denote that the node representing founder allele f_1 and the node representing founder allele f_2 are connected by an edge labeled with the allelic states a_1 and a_2. The intuitive meaning of this notation is, that either $f_1 = a_1$ and $f_2 = a_2$, or $f_1 = a_2$ and $f_2 = a_1$.

Let $\mathbf{v}$ denote the set of all possible inheritance vectors for a given pedigree. Then for each $v \in \mathbf{v}$ a graph is build in two steps:

1. Initially alleles of genotyped founders, $n \in F$, are connected by an edge labeled by the genotype $astates(n)$, by the assignment $\mathsf{Graph} \leftarrow \mathsf{Graph}'$.

2. For all genotyped non-founders the two nodes representing the founder alleles inherited by the genotyped non-founder n are connected by an edge labeled with the genotype information for n. Hence, the nodes of the founder alleles $\mathsf{F}^v(n, m)$ and $\mathsf{F}^v(n, p)$ are labeled with $astates(n)$.

The algorithm takes a pedigree and genotype information on the pedigree as input. The output is the probability distributions over all inheritance vectors conditioned on the genotype information. We are also interested in analysing the output of each iteration, since every iteration is independent. The output of each iteration is a graph of founder allele nodes which are connected according to the genotype information. We analyse whether our interpretation of this graph represents all compatible founder allele assignment for the current inheritance vector. Furthermore, we argue for the correctness of probability calculations based on the graph.

We use the notation $a_1, a_2 \leftarrow astates(n)$ to indicate that a_1 and a_2 are assigned the allelic states of individual n. If n is heterozygous, a_1 and a_2 are assigned different allelic states with no specific ordering, and if n is homozygous both a_1 and a_2 are assigned the same allelic state. We use Graph $\leftarrow f_1 \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_2$ to denote that an edge connecting f_1 and f_2 labelled with a_1 and a_2 in added to Graph.

[15]See Appendix A for more on available genetic analysis software.

The pseudocode for the algorithm is given in Algorithm 3.3.1.

Algorithm 3.3.1: $\textsc{Kruglyak}(P, \mathcal{G})$

$\mathsf{Graph}' \leftarrow \emptyset$
for each $n \in \mathcal{L} \cap F$
do $\begin{cases} a_1, a_2 \leftarrow astates(n) (\text{the observed allelic states for } n) \\ \mathsf{Graph}' \leftarrow f_1 \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_2 \ (f_1 \text{ and } f_2 \text{ are the alleles of the founder } n) \end{cases}$
for each $v \in \mathbf{v}$
do $\begin{cases} \mathsf{Graph} \leftarrow \mathsf{Graph}' \\ \textbf{for each } n \in \mathcal{L} \cap N \\ \quad \textbf{do} \begin{cases} f_1, f_2 \leftarrow \mathsf{F}^v(n,p), \mathsf{F}^v(n,m) \ (\text{the founder alleles inherited by } n) \\ a_1, a_2 \leftarrow astates(n) \ (\text{the observed allelic state of } n) \\ \mathsf{Graph} \leftarrow \mathsf{Graph} \cup f_1 \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_2 \end{cases} \\ \textsc{CalculateProbability}(v, \mathsf{Graph}) \end{cases}$

First, we discuss the correctness of the algorithm, that is, that Kruglyak produces the full set of compatible founder allele assignments given an inheritance vector and some genotype information. Second, we analyse the complexity of CalculateProbability. Following this, we discuss the full complexity of the algorithm.

Correctness

We interpret graphs that are output after each iteration in Kruglyak by sets of founder allele assignments. The interpretation of a graph, Graph, is denoted *interpret*(Graph) and is defined as:

$$interpret(\text{Graph}) = \bigcap_{f_1 \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_2 \in \text{Graph}} interpret(f_1 \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_2),$$

where $interpret(f_1 \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_2)$ is defined according to the intuitive meaning as:

$$\begin{aligned} interpret(f_1 \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_2) &= \{Z_M \mid Z_M(f_1) = a_1 \text{ and } Z_M(f_2) = a_2 \ \} \\ &\cup \ \{Z_M \mid Z_M(f_1) = a_2 \text{ and } Z_M(f_2) = a_1\}. \end{aligned}$$

We formulate that the graph at each iteration of Kruglyak describes the full set of compatible founder allele assignments in the following theorem:

Theorem 2 (Kruglyak is equivalent to $\varphi^v_{\mathcal{G}}$) *For any iteration of* Kruglyak *with an inheritance vector, v, over some genotype information, $\mathcal{G}$, the interpretation of the graph, Graph, which is output is equal to $[\![\varphi^v_{\mathcal{G}}]\!]$, that is:*

$$interpret(Graph) = [\![\varphi^v_{\mathcal{G}}]\!]. \tag{3-11}$$

Proof: Since an edge is added to Graph for every genotyped individual in the pedigree we only need to show that:

$$\begin{aligned} \mathit{interpret}(f_1 \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_2) &= [\![(f_{\mathsf{F}^v(n,p)} = a_1 \wedge f_{\mathsf{F}^v(n,m)} = a_2) \\ &\quad \vee(f_{\mathsf{F}^v(n,p)} = a_2 \wedge f_{\mathsf{F}^v(n,m)} = a_1)]\!], \end{aligned}$$

for any genotyped individual n, where $a_1, a_2 \in \mathit{astates}(n)$, $f_1 = \mathsf{F}^v(n,p)$, and $f_2 = \mathsf{F}^v(n,m)$, that is, $f_1 \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_2$ is the edged added to Graph for the genotyped individual n. From the definition of $\mathit{interpret}(f_1 \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_2)$ (3-11) and the semantics of φ-formulae (3-10) the equality is easily shown. $\diamondsuit$

Before discussing the complexity of KRUGLYAK, we explain how the algorithm computes the probability of the individual inheritance vector based on the edges in Graph.

Probability

In this section we show that the probability computed by KRUGLYAK is correct. This gives us the information to reason about the full complexity of the algorithm.

We divide the set of founder nodes into two different sets. The set of founder nodes that are *free*, that is, a founder node which is not in Graph. In other words, a founder node that can be assigned any allelic state while still satisfying the constraints in Graph. The other set consists of the set of founder nodes which belong to some connected component in Graph, thus, nodes which have constraints on the possible assignments of allelic states.

If f_i is a free founder node, that is, f_i can be assigned any allelic state, then f_i has no influence on the probability: Consider a set of founder nodes in which all but f_i is uniquely assigned an allelic state, then all founder allele assignments, which assigns f_i any allelic state and the others their unique state, will be a solution to the constraints in Graph. The probability of these assignments is the sum of the probability of each assignment. Notice that the only thing that differs in the founder allele assignments is the allelic state assigned to f_i, and since the sum of all allele frequencies is 1 by definition, f_i has no influence on the final probability. This, of course, applies for all free founder nodes.

Actually, we can extend this reasoning to the set of founder nodes in Graph if we look at each connected component individually. Consider the case where Graph contains two connected components then any solution to the first component can be combined with any solution of the second component to form a complete solution. This implies that we need only sum the probability of the connected components separately and multiply the result to compute the probability for the inheritance vector.

We now explain this with a simple example.

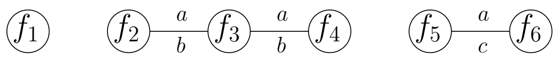

Figure 3-6: The six founder nodes of Example 11 and the edges between them.

Example 11 *Consider a set of 6 founder alleles* $\{f_1, f_2, f_3, f_4, f_5, f_6\}$ *and three possible allelic states* $\{a, b, c\}$*. We assume that* f_1 *is a free founder node and* $f_2 \underset{b}{\overset{a}{\longleftrightarrow}} f_3$*,* $f_3 \underset{b}{\overset{a}{\longleftrightarrow}} f_4$*, and* $f_5 \underset{c}{\overset{a}{\longleftrightarrow}} f_6$ *are the edges in Graph (See Figure 3-6). This gives two connected components that we need to consider:* $\{f_2, f_3, f_4\}$ *and* $\{f_5, f_6\}$*. Both components have two solutions. Let* $\pi(f_2 = a)$ *denote the allele frequency of* a $(\pi(a))$ *indicating that* f_2 *has been assigned the allelic state* a*. Since the probability of the genotype given the inheritance vector is proportional to the sum of all compatible founder allele assignments, we get the following product of probabilities, for the inheritance vector which Graph was constructed, by rearranging the components of the sum:*

$$\begin{aligned} P(Graph) = \quad & (\pi(f_1 = a) + \pi(f_1 = b) + \pi(f_1 = c)) \ \cdot \\ & (\pi(f_2 = a)\pi(f_3 = b)\pi(f_4 = a) + \pi(f_2 = b)\pi(f_3 = a)\pi(f_4 = b)) \ \cdot \\ & (\pi(f_5 = a)\pi(f_6 = c) + \pi(f_5 = c)\pi(f_6 = a)). \end{aligned}$$

Notice, this is the product of the probabilities of the solutions to the individual components. The contribution from a free founder node to the final product is a factor 1 and we can omit these in the computation.

Let CC denote the set of connected components in Graph and let $s \in X$ denote a solution to a connected component X, then the probability of all founder allele assignments is:

$$\prod_{X \in CC} \sum_{s \in X} P(s), \tag{3-12}$$

where $P(s)$ denotes the probability of the solution. Note that, $P(s)$ is only the product of allelic states assigned to founder nodes in X. If the solution set of a connected component is empty, we say the probability is zero. In this way the probability of the genotype information given the inheritance vector becomes zero, if no compatible founder allele assignment exists.

Before reasoning about the complexity of the computation we first argue that each connected component has at most 2 solutions, that is the alleles in the connected component can only be assigned in at most two different ways.

The reason is that assigning an allelic state to any founder node in a connected component uniquely assigns allelic states to all other founder nodes in the component. Consider two founder nodes f_i and f_j in a connected component. For f_i to be in the connected component at least one edge $f_i \xleftrightarrow[a_2]{a_1} f_k$ exists for some f_k. This indicates that f_i must be assigned the allelic state a_1 or a_2. Assigning i.e. a_1 to f_i forces the assignment of a_2 to f_k. This in turn forces any founder node connected to f_k by an edge with allelic states a_2, a_3 to be assigned a_3. Since f_j by definition is connected to f_i, f_j is also uniquely assigned an allelic state. Therefore, there can be at most two solutions to a connected component.

If, however, f_k was connected to some founder node by an edge with allelic states a_1, a_3, assigning a_1 to f_i would yield no solution, but assigning it a_2 would. This way a connected component can have two, one, or zero solutions.

Example 12 *Let a part of* Graph *consist of three nodes f_1, f_2 and f_3. Let them be connected by the following edges:*

$$f_1 \xleftrightarrow[a_2]{a_1} f_2 \text{ and } f_2 \xleftrightarrow[a_3]{a_2} f_3.$$

There is only one compatible assignment possible given the edges above. The assignment is: $f_1 = a_1, f_2 = a_2$, and $f_3 = a_3$. If f_1 is selected as the start node, it can be assigned to either a_1 or a_2, however if it is assigned to a_2, then f_2 is forced to be assigned to a_1. This assignment is incompatible with the given vector, since a_1 is not present on the edge $f_2 \xleftrightarrow[a_3]{a_2} f_3$.

Complexity

We now discuss the running time of the Kruglyak algorithm.

The dominating factor of the complexity is the iteration over all inheritance vectors ($\mathbf{v}$) for the pedigree in question. The size of $\mathbf{v}$ is given by: $2^{2 \cdot |N|}$, where $|N|$ denotes the number of non-founders.

Building Graph is linear in the number of genotyped individuals since we add exactly one edge for each genotyped individual. The set of genotyped individuals can be no bigger than the set of all individuals which is linear in the number of non-founders $|N|$.

Since each connected component in Graph can be treated separately and has at most two solutions, the complexity of CalculateProbability is linear in the number of connected components, which again is linear in the number of non-founders $|N|$.

This adds up to a total complexity of $O(|N| \cdot 2^{2 \cdot |N|})$. The complexity stated in [KDRDL96] is: $O(|N| \cdot 2^{2 \cdot |N| - |F|})$, where $|F|$ is the number of founders. This reduced complexity is obtained by using *founder reduction*, which is introduced in Section 3.4.4.

3.3.3 Single Point Probability Computation in Allegro

The FASTTREETRAVERSAL algorithm provided by Gudbjartsson in [Gud00] computes all compatible[16] founder allele assignments $\mathcal{F}_g^v$ for every inheritance vector $v \in \mathbf{v}$, for a pedigree, with some genotype information $\mathcal{G}$.
KRUGLYAK has two loops, the outer loop is a loop over all possible inheritance vectors and in the inner loop a graph is build. Gudbjartsson's algorithm FAST-TREETRAVERSAL has only one loop. Conceptually, the algorithm traverses a binary tree with depth $2 \cdot |N|$, where each path from the root to a leaf represents an inheritance vector. In reality, the tree is traversed recursively from the root in a depth-first fashion. In each recursive call the compatibility of the subset of inheritance vectors, sharing the path traversed so far, is checked. If this subset of the vectors turns out to have no compatible founder allele assignments, the traversal down the current branch is terminated, since no inheritance vector in the subset is compatible.

Initially, the whole set of inheritance vectors $\mathbf{v}$ is divided into four subsets:

$$\begin{aligned}
\mathbf{v_1} &= \{v \in \mathbf{v} \mid v(n_1, p) = p \text{ and } v(n_1, m) = p\}, \\
\mathbf{v_2} &= \{v \in \mathbf{v} \mid v(n_1, p) = p \text{ and } v(n_1, m) = m\}, \\
\mathbf{v_3} &= \{v \in \mathbf{v} \mid v(n_1, p) = m \text{ and } v(n_1, m) = p\}, \\
\mathbf{v_4} &= \{v \in \mathbf{v} \mid v(n_1, p) = m \text{ and } v(n_1, m) = m\},
\end{aligned}$$

for some non-founder n_1. Then $\mathbf{v_1}$ is divided into four subsets:

$$\begin{aligned}
\mathbf{v_{11}} &= \{v \in \mathbf{v_1} \mid v(n_2, p) = p \text{ and } v(n_2, m) = p\}, \\
\mathbf{v_{12}} &= \{v \in \mathbf{v_1} \mid v(n_2, p) = p \text{ and } v(n_2, m) = m\}, \\
\mathbf{v_{13}} &= \{v \in \mathbf{v_1} \mid v(n_2, p) = m \text{ and } v(n_2, m) = p\}, \\
\mathbf{v_{14}} &= \{v \in \mathbf{v_1} \mid v(n_2, p) = m \text{ and } v(n_2, m) = m\},
\end{aligned}$$

for some non-founder n_2. This recursive division into subsets continues until there are no more non-founders (meaning that there is only one inheritance vector in the subset), or all the vectors in a subset are found to be incompatible.

Practical experiments have shown, that this algorithm performs better than KRUGLYAK, which is due to its ability to determine incompatibility for a whole subset of inheritance vectors without testing each individual inheritance vector[17].

[16] See page 26 for more on *compatibility*.

[17] For a pedigree with 14 non-founders Allegro uses 7 seconds for the single point calculation (9.6% of the total running time), while Genehunter uses 1702 seconds (64.5% of its total running time), [Gud00, page 51].

Example 13 *Consider the pedigree with* $F = \{father, mother\}$, $N = \{son, daughter\}$ *and* $\mathcal{L} = \{father, son, daughter\}$ *shown in Figure 3-7.*

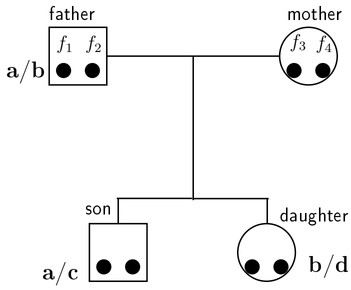

Figure 3-7: An example pedigree P with one genotyped founder and two genotyped non-founders.

In the first recursion the subset $\mathbf{v}'$ *is considered, where the subset of inheritance vectors for which* $v(son, p) = p$ *and* $v(son, m) = p$ *is checked. The founder alleles inherited by son for this subset of inheritance vectors:*

$$\mathsf{F}^{v'}(son, p) = f_1,$$

and

$$\mathsf{F}^{v'}(son, m) = f_3,$$

for any $v' \in \mathbf{v}'$. *Since there is no incompatibility yet, the traversal is continued. Now, the subset* $\mathbf{v}''$ *of* $\mathbf{v}'$ *for which* $v'(daughter, p) = p$ *and* $v'(daughter, m) = p$ *is considered in the next recursion. The compatibility for this subset is evaluated:*

Since $\mathsf{F}^{v''}(daughter, p) = \mathsf{F}^{v''}(son, p) = f_1$ *and* $\mathsf{F}^{v''}(daughter, m) = \mathsf{F}^{v''}(son, m) = f_3$, *the son and the daughter inherit the same founder alleles which is impossible, as the two children have been genotyped with different allelic states. Thus, the vectors in* $\mathbf{v}''$ *are incompatible, and a new subset* $\mathbf{v}'''$ *of* $\mathbf{v}'$, *where for which* $v'(daughter, p) = p$ *and* $v'(daughter, m) = m$, *is considered. When all the subsets, each with a different value for* $(v'(daughter, p), v'(daughter, m))$ *- there are four possible values for the pair:* $(p, p), (p, m), (m, p), (m, m)$ *- have been evaluated, another subset of* $\mathbf{v}$ *is selected. The subset selected, is the subset for which* $v'(son, p) = p$ *and* $v'(son, m) = m$. *This new subset is again divided into four new subset for the four different values of* $(v'(daughter, p), v'(daughter, m))$.

Implementation of Inheritance Vectors as Bit Vectors

In Allegro and in Genehunter the inheritance vector function is defined as a vector of bits (hence the name inheritance *vector*). This is done by enumerating the non-founders in a pedigree and by letting two bits denote the inheritance pattern for one non-founder. E.g. assume that we want to define an inheritance vector v for a pedigree with two non-founders n_1 and n_2. This can be done by a binary vector $\overrightarrow{b} = [b_1, b_2, b_3, b_4]$, where b_1 and b_2 denotes the inheritance pattern for n_1, and b_3 and b_4 denotes the inheritance pattern for n_2. The bits of $\overrightarrow{b}$ should be interpreted as described in the example below.

Example 14 *Let b_1 and b_3 represent the paternally inherited allele of n_1 and n_3, respectively, and let b_2 and b_4 represent the maternally inherited allele of n_1 and n_2, respectively. The value of the of the bits should be interpreted such that 0 corresponds to p and 1 corresponds to m.*

An instance of an inheritance vector for the pedigree with the two non-founders n_1 and n_2 could be described by the binary vector:

$$\overrightarrow{b} = [0, 0, 1, 0],$$

which corresponds to the inheritance vector function defined as:

$$\begin{aligned} v(n_1, p) &= p \text{ since } b_1 = 0, \\ v(n_1, m) &= p \text{ since } b_2 = 0, \\ v(n_2, p) &= m \text{ since } b_3 = 1, \text{and} \\ v(n_2, m) &= p \text{ since } b_4 = 0. \end{aligned}$$

An inheritance vector of the inheritance pattern in Figure 3-2 implemented as a bit vector would be $[0, 1, 0, 0, 0, 1]$ where the ordering is individual 3-5-6.

Hence, an instance of an inheritance vector can be described by a binary vector of length $2 \cdot |N|$, where $|N|$ is the number of non-founders. In this way pedigrees can be categorized by the number of bits required to describe an inheritance vector for the pedigree.

The FastTreeTraversal Algorithm

The pseudocode for the FastTreeTraversal is found in Algorithm 3.3.2. In the algorithm the variable n is assigned a non-founder by the method FindNextNonFounder, which returns non-founders in a top-down, breath-first fashion, so that when a genotyped non-founder is reached, the founder alleles inherited by that non-founder are the same for all the inheritance vectors in the subset v'' of v'. More specifically, we assume that we have an enumeration on the non-founders: $n_1, n_2, \ldots, n_{|N|}$, and for all indices $1 \leq i,\ j \leq |N|$ then:

$$n_i \in ancestor(n_j) \ \Rightarrow \ i < j. \tag{3-13}$$

FINDNEXTNONFOUNDER adds non-founders one by one according to this enumeration. Therefore, FINDNEXTNONFOUNDER only adds a non-founder n if all ancestors of n have already been added.

If we assume the bit vector representation described on page 39 and the enumeration described above, FASTTREETRAVERSAL recursively fixes larger and larger prefixes of the binary vector until either the prefix is the complete vector or until the all the inheritance vectors sharing the prefix are found to be incompatible.

In the **for**-loop the subset of inheritance vectors $\mathbf{v}'$ given as the parameter, is further subdivided into four subsets (one subset for each iteration), for which the inheritance pattern for the non-founder n is different. The four different inheritance patterns for n are:

$$\begin{aligned}
1)\ v''(n,m) &= p \text{ and } v''(n,p) = p,\\
2)\ v''(n,m) &= p \text{ and } v''(n,p) = m,\\
3)\ v''(n,m) &= m \text{ and } v''(n,p) = p, \text{ and}\\
4)\ v''(n,m) &= m \text{ and } v''(n,p) = m.
\end{aligned}$$

Algorithm 3.3.2: FASTTREETRAVERSAL($\mathbf{v}'$,$\mathcal{U},\mathcal{E},\mathcal{A}$)

$\mathbf{v}'$ is a set of inheritance vectors, $\mathcal{U},\mathcal{E},\mathcal{A}$ are the sets of founder alleles (and edges for $\mathcal{E}$)
as described in the text.

```
n ← FindNextNonFounder()
if an n is found
  then for (P, M) in {(p,p), (p,m), (m,p), (m,m)}
         do v'' ← {v ∈ v' | v(n,p) = P and v(n,m) = M}
            U' ← U
            E' ← E
            A' ← A
            if n ∈ L
              then (U', E', A') ← PartitionFounderAlleles(v'', n, U', E', A')
                   if (U', E', A') ≠ incompatible
                     then FastTreeTraversal(v'', U', E', A')
              else FastTreeTraversal(v'', U', E', A')
  else CalculateProbability(v', U, E, A)
```

If the individual n, for which $\mathbf{v}'$ is divided, has been genotyped (that is: $n \in \mathcal{L}$), the founder alleles are repartitioned by the method PARTITION-FOUNDERALLELES. After the founder alleles have been repartitioned the algorithm checks whether the new partitioning can generate a compatible founder allele assignment, e.g. whether any inheritance vector within the subset currently considered, can be compatible[18]. If a compatible founder allele assign-

[18]The check is actually performed in PARTITIONFOUNDERALLELES, where incompatibility is detected.

ment is possible given the new partitioning, a new recursive call is performed, otherwise the recursion down the current branch is terminated.

If $n \notin \mathcal{L}$ no repartitioning is performed, since the individual n does not add any further information to the compatibility of founder allele assignments, and the recursion is continued. The partitioning algorithm is presented in the following.

The PartitionFounderAlleles algorithm

The PartitionFounderAlleles algorithm (Algorithm 3.3.3) partitions the founder alleles into one, and only one, of three sets $\mathcal{A}$, $\mathcal{E}$ and $\mathcal{U}$, according to the inheritance pattern common to a subset of inheritance vectors $\mathbf{v}''$, see Figure 3-8. At first sight, the algorithm can appear rather complex, however much of the complexity is due to the large number of combinations of cases, allelic states, and the actions taken. In some of the cases shown the action depends on whether the individual n is homo- or heterozygous, however the differences in these actions are often quite trivial. When reading the algorithm keep in mind that when the algorithm is run it *only* performs one action (e.g. assigns allelic states to two founder alleles and moves them into the $\mathcal{A}$ set). The algorithm should be interpreted in the following way:

- Every case corresponds to one of the six possible combinations of membership for founder alleles f_1 and f_2 of the three sets $\mathcal{A}$, $\mathcal{E}$, and $\mathcal{U}$.
- In each case there are one or more tests on the allelic states of the founder alleles, and the allelic states of n, follow by an ':' after which the appropriate action is specified.
- A test like $f_1 \in \mathcal{A} \wedge f_2 \in \mathcal{E} \wedge \mathit{assigned}(f_1) = a_1 \wedge a_2 \in f_2 \xleftrightarrow[\beta]{\alpha} f_3$ (case 4), should be interpreted as: If one of the founder alleles f_1 is assigned, and the other f_2 is in edge, and the allelic state of f_1 matches one of the allelic states (a_1) of n, while the other allelic state (a_2) of n is one of the allelic states on the edge with f_2 then perform the action.
- An action like $f_2 \leftarrow a_1, f_3 \leftarrow a_2$ and $\mathcal{A} \leftarrow f_2, f_3$ (case 4) should be interpreted as: Assign the allelic state a_1 to f_2, and a_2 to f_3, and put both founder alleles into the set of assigned alleles ($\mathcal{A}$).

The intuitive meaning of each of the three sets is explained below:

$\mathcal{U}$: $\mathcal{U}$nassigned is the set of founder alleles which no genotyped individual has inherited. Hence, the founder alleles in this set are currently free.

$\mathcal{E}$: $\mathcal{E}$dged is a set of edges and founder alleles, for which an allele assignment is *ambiguous* by $\mathbf{v}''$, thus an edge $f_1 \xleftrightarrow[a_2]{a_1} f_2$ in the set, denotes that the two founder alleles f_1 and f_2 have been inherited by a genotyped individual, $n \in \mathcal{L}$, with the genotype information $astates(n) = \{a_1, a_2\}$. The founder alleles f_1 and f_2 are said to be ambiguously assigned, since there are two possible assignments for the two founder alleles: Either $f_1 = a_1$ and $f_2 = a_2$, or $f_1 = a_2$ and $f_2 = a_1$. $f_1 \xleftrightarrow[a_2]{a_1} f_2$ has the same interpretation as in KRUGLYAK.

$\mathcal{A}$: $\mathcal{A}$ssigned is the set of founder alleles that, *unambiguously*, has been assigned an allelic state for every inheritance vector in the subset.

The three sets $\mathcal{A}$, $\mathcal{E}$ and $\mathcal{U}$ will, before entering the FASTTREETRAVERSAL, be initialized such that the founder alleles of a genotyped founder are put into $\mathcal{E}$, if the founder is heterozygous (the founder alleles are ambiguously assigned). Otherwise, the founder alleles are moved into $\mathcal{A}$, if the founder is homozygous (unambiguously assigned). The initialization of the sets is not shown in the algorithm. The founder alleles of ungenotyped founders are put in the set $\mathcal{U}$.

PARTITIONFOUNDERALLELES($\mathbf{v}''$, n, $\mathcal{U}, \mathcal{E}, \mathcal{A}$) is called with: The subset of inheritance vectors $\mathbf{v}''$, such that the founder alleles inherited by the genotyped individual n can be determined, and with the three sets $\mathcal{U}, \mathcal{E}, \mathcal{A}$, which represent the current partition of founder alleles. PARTITIONFOUNDERALLELES returns the new partitioning based on current partitioning and the information of the genotyped individual n.

First PARTITIONFOUNDERALLELES finds the founder alleles of n, that is: $\mathsf{F}^{v''}(n, p)$ and $\mathsf{F}^{v''}(n, m)$ for any $v'' \in \mathbf{v}''$, which are represented by f_1 and f_2 in Algorithm 3.3.3, and the genotype information $astates(n)$ for n represented by a_1 and a_2. Now when the founder alleles and the genotype are known, the algorithm repartitions the founder alleles f_1 and f_2 based on which of the sets $\mathcal{U}, \mathcal{E}$ and $\mathcal{A}$ they are currently an element of. There are six cases as seen in Figure 3-8. Note, that a founder allele can only be a member of one set, e.g. when f_1 is put into the $\mathcal{A}$ set, it is removed from the set which it was previously in (e.g. $\mathcal{U}$).

The algorithm for PARTITIONFOUNDERALLELES is shown in Algorithm 3.3.3.

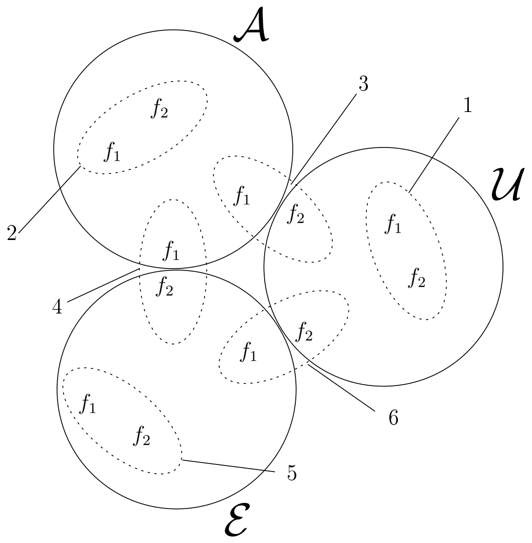

Figure 3-8: The three sets which founder alleles can be in during execution of FastTreeTraversal and the six cases of the PartitionFounderAlleles algorithm.

Algorithm 3.3.3: PartitionFounderAlleles($\mathbf{v}''$, n, $\mathcal{U}$, $\mathcal{E}$, $\mathcal{A}$)

$f_1, f_2 \leftarrow \mathsf{F}^{v''}(n,p)$ and $\mathsf{F}^{v''}(n,m)$ - for any $v'' \in \mathbf{v}''$ the founder alleles inherited by n
$a_1, a_2 \leftarrow astates(n)$ - the observed allelic states of n

$$\textbf{case } 1 : \begin{cases} f_1, f_2 \in \mathcal{U} \wedge a_1 = a_2 \ : \ f_1, f_2 \leftarrow a_1 \text{ and } \mathcal{A} \leftarrow f_1, f_2 \\ \textbf{or} \\ f_1, f_2 \in \mathcal{U} \wedge a_1 \neq a_2 \ : \ \mathcal{E} \leftarrow f_1 \xleftrightarrow[a_2]{a_1} f_2 \end{cases}$$

$$\textbf{case } 2 : \left\{ f_1, f_2 \in \mathcal{A} \wedge ((assigned(f_1) = a_1 \wedge assigned(f_2) = a_2) \ : \ \textbf{skip} \right.$$

$$\textbf{case } 3 : \left\{ f_1 \in \mathcal{A} \wedge f_2 \in \mathcal{U} \wedge assigned(f_1) = a_1 \ : \ f_2 \leftarrow a_2 \text{ and } \mathcal{A} \leftarrow f_2 \right.$$

$$\textbf{case } 4 : \begin{cases} f_1 \in \mathcal{A} \wedge f_2 \in \mathcal{E} \wedge assigned(f_1) = a_1 \wedge a_2 \in f_2 \xleftrightarrow[\beta]{\alpha} f_3 \ : \\ \qquad f_2 \leftarrow \alpha, f_3 \leftarrow \beta \text{ and } \mathcal{A} \leftarrow f_2, f_3 \end{cases}$$

$$\textbf{case } 5 : \begin{cases} f_1, f_2 \in \mathcal{E} \wedge a_1 = a_2 \ : \ f_1, f_2 \leftarrow a_1, f_3 \leftarrow a_3, f_4 \leftarrow a_4, \text{ and} \\ \qquad \mathcal{A} \leftarrow f_1, f_2, f_3, f_4, \text{ where } f_1 \xleftrightarrow[a_3]{a_1} f_3 \text{ and } f_2 \xleftrightarrow[a_4]{a_1} f_4 \\ \textbf{or} \\ f_1, f_2 \in \mathcal{E} \wedge a_1 \neq a_2 \wedge a_1, a_2 \in f_1 \xleftrightarrow[\beta_1]{\alpha_1} f_3, f_2 \xleftrightarrow[\beta_2]{\alpha_2} f_4 \ : \ SPLIT(f_1, f_2, a_1, a_2) \\ \textbf{or} \\ f_1, f_2 \in \mathcal{E} \wedge a_1 \neq a_2 \wedge a_1 \in f_1 \xleftrightarrow[\beta_1]{\alpha_1} f_3 \wedge a_2 \in f_2 \xleftrightarrow[\beta_2]{\alpha_2} f_4 \ : \\ \qquad f_1 \leftarrow \alpha_1, f_2 \leftarrow \alpha_2, f_3 \leftarrow \beta_1, f_4 \leftarrow \beta_2 \text{ and } \mathcal{A} \leftarrow f_1, f_2, f_3, f_4 \end{cases}$$

$$\textbf{case } 6 : \begin{cases} f_1 \in \mathcal{E} \wedge f_2 \in \mathcal{U} \wedge a_1, a_2 \in f_1 \xleftrightarrow[\beta]{\alpha} f_3 \ : \ SPLIT(f1, f2, a_1, a_2) \\ \textbf{or} \\ f_1 \in \mathcal{E} \wedge f_2 \in \mathcal{U} \wedge a_1 \in f_1 \xleftrightarrow[\beta]{\alpha} f_3 \wedge a_2 \notin f_1 \xleftrightarrow[\beta]{\alpha} f_3 \ : \\ \qquad f_1 \leftarrow a_1, f_2 \leftarrow \alpha, f_3 \leftarrow \beta \text{ and } \mathcal{A} \leftarrow f_1, f_2, f_3 \end{cases}$$

else $(\mathcal{U}, \mathcal{E}, \mathcal{A}) \leftarrow$ **incompatible**
return $(\mathcal{U}, \mathcal{E}, \mathcal{A})$

Below each of the six cases of PARTITIONFOUNDERALLELES are explained in text:

case 1: When both founder alleles are in $\mathcal{U}$, meaning that they are both free, and individual n is homozygous there is no ambiguity, and both founder alleles are assigned. Hence, both founder alleles f_1 and f_2 are assigned to the genotype of n and moved to $\mathcal{A}$. If n is heterozygous, on the other hand, the founder alleles inherited by n, f_1 and f_2, are moved to $\mathcal{E}$ with the edge: $f_1 \underset{a_2}{\overset{a_1}{\longleftrightarrow}} f_2$.

case 2: If both founder alleles are unambiguously assigned (the are both currently in $\mathcal{A}$) a check is performed to see whether the genotype of n is compatible with their current assignment. If the genotype of n is not compatible with the current assignment, it means that no inheritance vector in the subset of inheritance vectors $\mathbf{v}''$ can be compatible, and the traversal of the current branch is terminated ($\mathcal{F}_g^{\mathbf{v}''}$ is set to $\emptyset$).

case 3: If one founder allele, f_1, is unambiguously assigned (in $\mathcal{A}$) and the other, f_2, is unassigned (in $\mathcal{U}$), then if one of the allelic states, a_1, matches the assignment of f_1, then f_2 can be unambiguously assigned to a_2. Hence, f_2 is put into $\mathcal{A}$ with the assignment a_2. If the f_1 is not assigned with any of the allelic states of n, we can terminate the traversal due to the incompatibility.

case 4: If one of the founder alleles f_1 is unambiguously assigned (is in $\mathcal{A}$) and the other, f_2, is ambiguously assigned (is in $\mathcal{E}$), we need to check that: One of the allelic states of n, a_1, is the assignment of f_1, and that the other allelic state, a_2, observed for n is the same as one of the allelic states of the edge: $f_2 \underset{a_4}{\overset{a_3}{\longleftrightarrow}} f_3$. Hence, that either $a_2 = a_3$ or $a_2 = a_4$. If $a_2 = a_3$ then f_2 is assigned to a_2, thus moved to $\mathcal{A}$, and f_3 can also be unambiguously assigned to a_4. If $a_2 = a_4$ then $f_2 = a_2$ and $f_3 = a_3$. If $a_2 \neq a_3 \wedge a_2 \neq a_4$ the traversal down the current branch is terminated do to the incompatibility for all the vectors in $\mathbf{v}''$.

case 5: If both founder alleles, f_1 and f_2, are ambiguously assigned (they are both in $\mathcal{E}$), and if n is homozygous ($a_1 = a_2$), then both founder alleles are moved to $\mathcal{A}$, and the founder allele with an edge to f_1 and the founder allele with an edge to f_2, can be assigned. If n is heterozygous, then if both allelic states a_1 and a_2 are on both edges, we do a split, where two recursions down the current branch is made, one where f_1 is assigned to a_1 and f_1 to a_2, and the other where f_1 is assigned to a_2 and f_2 and a_1.

case 6: Finally, if one node f_1 is ambiguously assigned, hence is in $\mathcal{E}$, and another f_2 is unassigned (is in $\mathcal{U}$), and the edge with f_1 has both allelic states a_1 and a_2 we do a split. If only one of the allelic states is on the edge, we can assign f_1 and f_2, as well as the other founder allele on the edge, denoted by f_3 in the pseudocode.

FastTreeTraversal Complexity

The worst case complexity of the algorithm is:

1. When all possible inheritance vectors have compatible founder allele assignments, FastTreeTraversal will be called $2^{2\cdot|N|}$ times.
2. For every $n \in \mathcal{L}$ the two founder alleles of n, f_1 and f_2, are found. How fast this operation is depends on implementation, but if founder alleles are propagated recursively, for each individual we only have to lookup the founder alleles of the parents, and we assume this can be done in constant time.
3. For each $n \in \mathcal{L}$ the memberships of n's two founder alleles are found, however, since the founder alleles can be stored in an array, with either their assignment (ambiguous or unambiguous) or no assignment at all (meaning that they are in $\mathcal{U}$), this can be performed in constant time.

Hence, the worst time complexity of FastTreeTraversal is: $O(2^{2\cdot|N|})$.

Correctness

When FastTreeTraversal is called it adds one new non-founder from the set of all non-founders in a top-down, breath-first fashion defined by the enumeration of (3-13). For every non-founder added we enter a loop over all combinations of values for the alleles of the non-founder in the inheritance vector. As stated previously, this can be viewed as traversing a binary tree of depth $2 \cdot |N|$, where $|N|$ is number of non-founders.

This implies that at any point in the recursion, the inheritance vector is uniquely determined for the set of added non-founders, N'. At any point in the recursion when FastTreeTraversal is called with some subset of inheritance vectors, $\mathbf{v}'$, where N' is the set of non-founders added so far, we claim that the following relationship holds between N' and $\mathbf{v}'$:

$$N' = \{ \ n \in N \mid \forall \ v, w \in \mathbf{v}' \ . \ v(n, \varpi) = w(n, \varpi) \ \}, \tag{3-14}$$

where $\varpi \in \{p, m\}$.

If inheritance vectors were specified as bit vectors according to the enumeration of (3-13), the above is the same as stating that all inheritance vectors in $\mathbf{v}'$ have the same prefix for the individuals of N'.

Furthermore, since individuals are added such that all ancestors of an individual are added before the individual due to the enumeration of (3-13) we deduce from (3-14) that:

$$\forall\, v, w \in \mathbf{v}' \,.\, \mathsf{F}^v(n, \varpi) \;=\; \mathsf{F}^w(n, \varpi), \tag{3-15}$$

where $\varpi \in \{p, m\}$.

The algorithm takes action depending on if the individual is genotyped or not. If the genotype information on the pedigree in question is $\mathcal{G} = \langle \mathcal{L}, astates \rangle$, then based of N' we defined the amount of genotype information added at the current branch of recursion as $\mathcal{G}' = \langle \mathcal{L}', astates' \rangle$, where $\mathcal{L}' = \mathcal{L} \cap (N' \cup F)$. For every element, n, in the domain of $astates'$; $astates'(n) = astates(n)$. That is $\mathcal{G}'$ is genotype information only for founders and non-founder added so far.

A note on notation: We use $f_1, f_2, f_3, \ldots$ to denote founder alleles - the same notation as used in Algorithm 3.3.3 (PartitionFounderAlleles). However, in the construction of $\varphi^v_{\mathcal{G}}$, we let $f_1, f_2, f_3, \ldots$ represent the vector of binary variables for the founder alleles. Likewise, we use $a_1, a_2, a_3, \ldots$ to denote the alleles a genotyped individual, and as the corresponding vector of binary values in the construction of $\varphi^v_{\mathcal{G}}$. This is again to achieve mutual exclusion on assignments of founder alleles.

Before proving the correctness of FastTreeTraversal we explain the interpretation of the three sets $\mathcal{A}$, $\mathcal{E}$, and $\mathcal{U}$.

The interpretation of the sets is given as:

$$interpret(\mathcal{A}, \mathcal{E}, \mathcal{U}) = interpret(\mathcal{U}) \cap interpret(\mathcal{A}) \cap interpret(\mathcal{E}),$$

where:

$$\begin{aligned}
interpret(\mathcal{U}) &= \bigcap_{f_1 \in \mathcal{U}} \{Z_M \mid Z_M(f_1) \in A_M\}, \\
interpret(\mathcal{A}) &= \bigcap_{f_1 = a_1 \in \mathcal{A}} \{Z_M \mid Z_M(f_1) = a_1\}, \text{ and} \\
interpret(\mathcal{E}) &= \bigcap_{f_1 \xleftrightarrow[a_2]{a_1} f_2 \in \mathcal{E}} interpret(f_1 \xleftrightarrow[a_2]{a_1} f_2).
\end{aligned}$$

$interpret(f_1 \xleftrightarrow[a_2]{a_1} f_2)$ is the same as in (3-11) from Kruglyak. That is:

$$\begin{aligned}
interpret(f_1 \xleftrightarrow[a_2]{a_1} f_2) &= \{Z_M \mid Z_M(f_1) = a_1 \text{ and } Z_M(f_2) = a_2\ \} \\
&\cup\ \{Z_M \mid Z_M(f_1) = a_2 \text{ and } Z_M(f_2) = a_1\}.
\end{aligned}$$

This way, if there is no genotyped information available, all founder alleles are placed in $\mathcal{U}$ and never moved to a new set, and the set of compatible founder allele assignments is then the set of all founder allele assignments. Note that *interpret*($\mathcal{U}$) only states that the founder alleles in $\mathcal{U}$ are free.

Theorem 3 (Invariant on FastTreeTraversal) *Whenever* FastTreeTraversal *is called on* ($\mathbf{v}', \mathcal{A}', \mathcal{E}', \mathcal{U}'$)*, where N' and $\mathcal{G}'$ are the non-founders and genotype information added so far at the current branch of recursion the following invariant holds:*

$$\forall\, w \in \mathbf{v}'\ .\ interpret(\mathcal{A}', \mathcal{E}', \mathcal{U}') \;=\; [\![\varphi^{w}_{\mathcal{G}'}]\!], \tag{3-16}$$

before a new individual is added.

Proof: First we show that it is sufficient to show (3-16) for only one inheritance vector w in $\mathbf{v}'$. The genotype information $\mathcal{G}'$ is only defined on individuals of the set of non-founders added at the current branch of recursion, N', and founders. As claimed, (3-14) states that for all inheritance vectors $v, v' \in \mathbf{v}'$ then $v(n, \varpi) = v'(n, \varpi)$, with $n \in \mathcal{L}'$ and $\varpi \in \{p, m\}$. Furthermore, (3-15) states that for all genotyped individuals $n \in \mathcal{L}'$ we have $\mathsf{F}^{v}(n) = \mathsf{F}^{w}(n)$ which by Theorem 1 on allele assignment formulae in turn implies that:

$$[\![\varphi^{v}_{\mathcal{G}'}]\!] = [\![\varphi^{v'}_{\mathcal{G}'}]\!],$$

for any two inheritance vectors in $\mathbf{v}'$. Thus, it suffices to show that (3-16) holds for just one inheritance vector in $\mathbf{v}'$.

The proof is by induction on the number of individuals added at the current branch of recursion.

Base case:

Upon the first call of FastTreeTraversal (no individuals are added, $N' = \emptyset$), $\mathcal{A}', \mathcal{E}'$ and $\mathcal{U}'$ are initialized such that $\mathcal{A}'$ contains the founder alleles of homozygously genotyped founders, $\mathcal{E}'$ contains edges between heterozygously genotyped founders, and $\mathcal{U}'$ contains founder alleles of ungenotyped founders. Furthermore, $\mathbf{v}'$ is the set of all inheritance vectors. It is easily shown that the interpretation of $\mathcal{A}'$ and $\mathcal{E}'$ satisfies the invariant since the interpretation is only reductions due to *ambiguously* and *unambiguously* assigned founder alleles as shown in Section 3.3.1 and therefore $interpret(\mathcal{A}', \mathcal{E}', \mathcal{U}') = [\![\varphi^{w}_{\mathcal{G}'}]\!]$, for any $w \in \mathbf{v}'$.

Induction hypothesis:

We assume $interpret(\mathcal{A}', \mathcal{E}', \mathcal{U}') \;=\; [\![\varphi^{v'}_{\mathcal{G}'}]\!]$ to hold for any $v' \in \mathbf{v}'$ in the k-th recursion.

Induction step:

We now need to show that when the k-th recursion of FastTreeTraversal calls the $k+1$-th recursion, ($\mathbf{v}', \mathcal{A}', \mathcal{E}', \mathcal{U}'$) are formatted into ($\mathbf{v}'', \mathcal{A}'', \mathcal{E}'', \mathcal{U}''$) such that the following holds:

$$interpret(\mathcal{A}'', \mathcal{E}'', \mathcal{U}'') = [\![\varphi^{v''}_{\mathcal{G}''}]\!],$$

where $\mathcal{G}''$ is the genotype information added at the $k+1$-th branch of recursion and v'' is any inheritance vector in $\mathbf{v}''$.
We prove this in two steps: n is not genotyped ($n \notin \mathcal{L}$) and n is genotyped ($n \in \mathcal{L}$), where n is the individual added in the $k+1$-th recursion. In the following we assume v' to be an element of $\mathbf{v}'$ and v'' an element of $\mathbf{v}''$.

1. $n \notin \mathcal{L}$:
   if n is not genotyped, then $(\mathcal{A}', \mathcal{E}', \mathcal{U}') = (\mathcal{A}'', \mathcal{E}'', \mathcal{U}'')$ since no action is performed by FastTreeTraversal. This further implies that $\mathcal{G}' = \mathcal{G}''$ and therefore:

$$\begin{aligned} \mathit{interpret}(\mathcal{A}', \mathcal{E}', \mathcal{U}') &= \mathit{interpret}(\mathcal{A}'', \mathcal{E}'', \mathcal{U}'') \text{ and} \\ [\![\varphi_{\mathcal{G}'}^{v'}]\!] &= [\![\varphi_{\mathcal{G}''}^{v''}]\!], \end{aligned}$$

   and from the induction hypothesis it follows that:

$$\mathit{interpret}(\mathcal{A}'', \mathcal{E}'', \mathcal{U}'') = [\![\varphi_{\mathcal{G}''}^{v''}]\!].$$

2. $n \in \mathcal{L}$:
   If n is genotyped such that $\{a_1, a_2\} = \mathit{astates}''(n)$ then we know from Theorem 1 that:

$$\begin{aligned} [\![\varphi_{\mathcal{G}''}^{v''}]\!] &= [\![\varphi_{\mathcal{G}'}^{v'}]\!] \\ &\cap \big((f_{\mathsf{F}^{v''}(n,p)} = a_1 \wedge f_{\mathsf{F}^{v''}(n,m)} = a_2) \\ &\vee (f_{\mathsf{F}^{v''}(n,p)} = a_2 \wedge f_{\mathsf{F}^{v''}(n,m)} = a_1)\big). \end{aligned}$$

   From Kruglyak we know that $[\![\varphi_{\mathcal{G}'}^{v'}]\!]$ equals putting all genotype information into the edge set $\mathcal{E}$[19]. Therefore, it suffices to show $\mathit{interpret}(\mathcal{A}'', \mathcal{E}'', \mathcal{U}'')$ is the same as if all genotype information in $\mathcal{G}''$ had been put into $\mathcal{E}$. From the induction hypothesis this indicates that it is sufficient to show:

$$\begin{aligned} \mathit{interpret}(\mathcal{A}'', \mathcal{E}'', \mathcal{U}'') &= \mathit{interpret}(\mathcal{A}', \mathcal{E}', \mathcal{U}') \\ &\cap \mathit{interpret}(f_{\mathsf{F}^{v''}(n,p)} \xleftrightarrow[a_2]{a_1} f_{\mathsf{F}^{v''}(n,m)}), \end{aligned} \tag{3-17}$$

   to prove that:

$$\mathit{interpret}(\mathcal{A}'', \mathcal{E}'', \mathcal{U}'') = [\![\varphi_{\mathcal{G}''}^{v''}]\!].$$

   Since FastTreeTraversal moves founder alleles between the three sets using the PartitionFounderAlleles algorithm, we show that

[19] Recall that the edge set of Kruglyak and $\mathcal{E}$ have the same interpretation.

the interpretation of the three sets after the founder alleles have been moved corresponds to a simplification of:

$$interpret(\mathcal{A}', \mathcal{E}', \mathcal{U}') \cap interpret(f_{\mathsf{F}^{v''}(n,p)} \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_{\mathsf{F}^{v''}(n,m)}),$$

and thereby proving (3-17). We observe that FASTTREETRAVERSAL is only called if one of the cases in PARTITIONFOUNDERALLELES is satisfied (otherwise the subset of inheritance vectors v'' is rejected), therefore we need to prove that all cases in PARTITIONFOUNDERALLELES generate a simplification.

We will go through the six cases of PARTITIONFOUNDERALLELES one at a time and show that either one performs a simplification. We assume that we add the genotyped individual $n \in \mathcal{L}$, and we denote the founder alleles inherited by that individual as f_1 and f_2.

case 1: If both f_1 and f_2 were previously unassigned, then we either move the founder alleles into $\mathcal{A}''$ or $\mathcal{E}''$ depending on if the individual is homozygous or heterozygous, respectively. This corresponds to performing simplifications as described in Section 3.3.1.

case 2: If both f_1 and f_2 are already assigned with the genotype of n no action is performed, that is $interpret(f_{\mathsf{F}^{v}(n,p)} \overset{a_1}{\underset{a_2}{\longleftrightarrow}} f_{\mathsf{F}^{v}(n,m)})$ is already expressed in $interpret(\mathcal{A}', \mathcal{E}', \mathcal{U}')$ and the simplification corresponds to not including it.

case 3: If founder allele f_2 is unassigned and f_1 is already unambiguously assigned to a_1. Then we perform the following simplification of:

$$\begin{aligned} & \{Z_M \mid Z_M(f_1) = a\} \\ \cap \quad & (\{Z_M \mid Z_M(f_1) = a_1 \text{ and } Z_M(f_2) = a_2\ \} \\ \cup \quad & \{Z_M \mid Z_M(f_1) = a_2 \text{ and } Z_M(f_2) = a_1\}) \end{aligned}$$

to:

$$\{Z_M \mid Z_M(f_1) = a_1\} \cap \{Z_M \mid Z_M(f_2) = a_2\}.$$

Hence, f_2 can be unambiguously assigned, and is moved to $\mathcal{A}$.

case 4: If f_1 is unambiguously assigned a_1 and f_2 is ambiguously assigned with an edge to f_3 with allelic states a_2, a_3, we simplify:

$$\begin{aligned} & \{Z_M \mid Z_M(f_1) = a_1\} \\ \cap \quad & (\{Z_M \mid Z_M(f_2) = a_2 \text{ and } Z_M(f_3) = a_3\} \\ & \cup \{Z_M \mid Z_M(f_2) = a_3 \text{ and } Z_M(f_3) = a_2\}) \\ \cap \quad & (\{Z_M \mid Z_M(f_2) = a_1 \text{ and } Z_M(f_1) = a_2\} \\ & \cup \{Z_M \mid Z_M(f_2) = a_2 \text{ and } Z_M(f_1) = a_1\}), \end{aligned}$$

to:

$$\{Z_M \mid Z_M(f_1) = a_1\} \cap \{Z_M \mid Z_M(f_2) = a_2\} \cap \{Z_M \mid Z_M(f_3) = a_3\},$$

which corresponds to moving all three founder alleles into $\mathcal{A}''$.

case 5: If n is homozygous ($a_1 = a_2$), f_1 is connected to f_3 with allelic states a_1, a_3, and f_2 is connected to f_4 with allelic states a_1, a_4, we perform the following simplification from:

$$\begin{aligned} & \{Z_M \mid Z_M(f_1) = a_1\} \ \cap \ \{Z_M \mid Z_M(f_2) = a_1\} \\ \cap \quad & (\{Z_M \mid Z_M(f_1) = a_1 \text{ and } Z_M(f_3) = a_3\} \\ & \cup \{Z_M \mid Z_M(f_1) = a_3 \text{ and } Z_M(f_3) = a_1\}) \\ \cap \quad & (\{Z_M \mid Z_M(f_2) = a_1 \text{ and } Z_M(f_4) = a_4\} \\ & \cup \{Z_M \mid Z_M(f_2) = a_4 \text{ and } Z_M(f_4) = a_1\}), \end{aligned}$$

to:

$$\begin{aligned} & \{Z_M \mid Z_M(f_1) = a_1\} \cap \{Z_M \mid Z_M(f_2) = a_1\} \\ \cap \quad & \{Z_M \mid Z_M(f_3) = a_3\} \cap \{Z_M \mid Z_M(f_4) = a_4\}. \end{aligned}$$

If n is heterozygous ($a_1 \neq a_2$), f_1 is connected to f_3, and f_2 is connected to f_4, two situations arise: Either both a_1 and a_2 are present on the edge between f_1 and f_3 and the on edge between f_2 and f_4, or a_1 is present on one edge and a_2 is present on the other. In the latter case we can perform the simplification of:

$$\begin{aligned} & (\{Z_M \mid Z_M(f_1) = a_1 \text{ and } Z_M(f_2) = a_2\} \\ & \cup \{Z_M \mid Z_M(f_1) = a_2 \text{ and } Z_M(f_2) = a_1\}) \\ \cap \quad & (\{Z_M \mid Z_M(f_1) = a_1 \text{ and } Z_M(f_3) = a_3\} \\ & \cup \{Z_M \mid Z_M(f_1) = a_3 \text{ and } Z_M(f_3) = a_1\}) \\ \cap \quad & (\{Z_M \mid Z_M(f_2) = a_2 \text{ and } Z_M(f_4) = a_4\} \\ & \cup \{Z_M \mid Z_M(f_2) = a_4 \text{ and } Z_M(f_4) = a_2\}), \end{aligned}$$

to:

$$\begin{aligned} & \{Z_M \mid Z_M(f_1) = a_1\} \cap \{Z_M \mid Z_M(f_2) = a_2\} \\ \cap \quad & \{Z_M \mid Z_M(f_3) = a_3\} \cap \{Z_M \mid Z_M(f_4) = a_4\}. \end{aligned}$$

In the former case we cannot determine if f_1 is assigned a_1 or a_2 since we have:

$$\begin{aligned} & \{Z_M \mid Z_M(f_1) = a_1 \text{ and } Z_M(f_3) = a_2\} \\ \cup \quad & \{Z_M \mid Z_M(f_1) = a_2 \text{ and } Z_M(f_3) = a_1\}, \end{aligned}$$

and a similar situation between f_2 and f_4. In this situation FASTTREETRAVERSAL performs a split which corresponds to duplicating the current branch and assuming $f_1 = a_1$ in one copy and $f_1 = a_2$ in the other. This is an implementation decision made by Gudbjartsson to simplify the calculation of the probability. Since $SPLIT$ is implementation specific we do not go into further details about the $SPLIT$ operation.

case 6: Again two situations arise if f_1 is ambiguously assigned with an edge to f_3 and f_2 is unassigned. One case if both a_1 and a_2 are present on the edge between f_1 and f_3, and the other case if only one of a_1 or a_2 are present on the edge. If only one is present, say a_1, we perform the following simplification from:

$$\begin{array}{rl} & (\{Z_M \mid Z_M(f_1) = a_1 \text{ and } Z_M(f_2) = a_2\} \\ & \cup\, \{Z_M \mid Z_M(f_1) = a_2 \text{ and } Z_M(f_2) = a_1\}) \\ \cap & (\{Z_M \mid Z_M(f_1) = a_1 \text{ and } Z_M(f_3) = a_3\} \\ & \cup\, \{Z_M \mid Z_M(f_1) = a_3 \text{ and } Z_M(f_3) = a_1\}), \end{array}$$

to:

$$\{Z_M \mid Z_M(f_1) = a_1\} \cap \{Z_M \mid Z_M(f_2) = a_2\} \cap \{Z_M \mid Z_M(f_3) = a_3\}.$$

If both are present we perform a spilt as in **case 5**, considering both assignments.

Based on these results we can conclude that the interpretation of the three sets $\mathcal{A}', \mathcal{E}'$ and $\mathcal{U}'$ in FASTTREETRAVERSAL describes precisely the same set of founder allele assignments as $[\![\varphi_{\mathcal{G}'}^{v'}]\!]$, whenever FASTTREETRAVERSAL is called on $(\mathbf{v}', \mathcal{A}', \mathcal{E}', \mathcal{U}')$, where N' and $\mathcal{G}'$ are the non-founders and genotype information added so far at the current branch of recursion. Therefore the invariant holds. $\diamondsuit$

From Theorem 3 we deduct that when all individuals have been added ($N' = N$ and $\mathcal{G}' = \mathcal{G}$), which from (3-14) implies that $\mathbf{v}'$ contains only one inheritance vector v then:

$$interpret(\mathcal{A}', \mathcal{E}', \mathcal{U}') \;=\; [\![\varphi_{\mathcal{G}}^{v}]\!].$$

This states that for all inheritance vectors, when FASTTREETRAVERSAL computes the probability of the inheritance given the genotype information, then the interpretation of the three set $\mathcal{A}', \mathcal{E}', \mathcal{U}'$ is exactly the set of compatible founder allele assignment over v and $\mathcal{G}$.

Since FASTTREETRAVERSAL performs early termination on branches in the binary tree, we need to prove that only inheritance vectors which have the empty set of compatible founder allele assignments are rejected.

Theorem 4 (Rejection of Incompatible Inheritance Vectors) *If* PARTITIONFOUNDERALLELES *is called from* FASTTREETRAVERSAL *on* $(\mathbf{v}', n, \mathcal{A}', \mathcal{E}', \mathcal{U}')$ *and a genotyped individual* n *is added to* N' *(the set of added individuals at the current branch of recursion) then none of the cases of* PARTITIONFOUNDERALLELES *are satisfied iff:*

$$interpret(\mathcal{A}', \mathcal{E}', \mathcal{U}') \cap interpret(f_1 \xleftrightarrow[a_2]{a_1} f_2) = \emptyset,$$

where $\{a_1, a_2\} = astates(n)$ *and* f_1 *and* f_2 *are the founder alleles inherited by* n *according any vector in* $\mathbf{v}'$*.*

Proof: Recall that each case in PARTITIONFOUNDERALLELES corresponds to the founder alleles being in one of the cases in Figure 3-8. We say the a subset of inheritance vectors is rejected in a case if f_1 and f_2 are in the corresponding case in Figure 3-8, but none of the expressions in the corresponding case of PARTITIONFOUNDERALLELES are satisfied. Subsets are rejected in the following cases:

case 2: If both f_1 and f_2 are unambiguously assigned it must be so that the individual in the current iteration has been genotyped with allelic states similar to the allelic states which f_1 and f_2 are assigned, or else the simplification would be:

$$\begin{aligned} & \{Z_M \mid Z_M(f_1) = a_3\} \cap \{Z_M \mid Z_M(f_2) = a_4\} \\ \cap \quad & (\{Z_M \mid Z_M(f_1) = a_1 \text{ and } Z_M(f_2) = a_2\} \\ & \cup \{Z_M \mid Z_M(f_1) = a_2 \text{ and } Z_M(f_2) = a_1\}) \\ = \quad & \emptyset, \end{aligned}$$

where $\{a_1, a_2\} \neq \{a_3, a_4\}$. This also holds if exactly one of a_1 or a_2 equals a_3 or a_4. That is, there would be no compatible founder allele assignment.

case 3: Similar to **case 2**, but where only one of the founder alleles is unambiguously assigned.

case 4: If f_1 is unambiguously assigned a_3 and f_2 is ambiguously assigned a_4 or a_5 together with f_3. We have the following:

$$\begin{aligned} & \{Z_M \mid Z_M(f_1) = a_3\} \\ \cap \quad & (\{Z_M \mid Z_M(f_2) = a_4 \text{ and } Z_M(f_3) = a_5\} \\ & \cup \{Z_M \mid Z_M(f_2) = a_5 \text{ and } Z_M(f_3) = a_4\}) \\ \cap \quad & (\{Z_M \mid Z_M(f_1) = a_1 \text{ and } Z_M(f_2) = a_2\} \\ & \cup \{Z_M \mid Z_M(f_1) = a_2 \text{ and } Z_M(f_2) = a_1\}) \\ = \quad & \emptyset, \end{aligned}$$

if either $(a_1 \neq a_3)$ and $(a_2 \neq a_3)$, or $(a_1 = a_3)$ and $(a_2 \neq a_4)$ and $(a_2 \neq a_5)$, or $(a_2 = a_3)$ and $(a_1 \neq a_4)$ and $(a_1 \neq a_5)$, the set of compatible founder allele assignments equals the empty set, and we can safely conclude, that the all the vectors in the subset $\mathbf{v}'$ of inheritance vectors are incompatible.

case 5: If f_1 is connected to f_3, f_2 is connected to f_4, and just one of the edges has neither of the allelic states a_1 and a_2, then the subset of inheritance vectors are rejected, since the interpretation simplifies to the empty set of compatible founder allele assignments. That is:

$$\begin{aligned}
& (\{Z_M \mid Z_M(f_1) = a_1 \text{ and } Z_M(f_2) = a_2\} \\
& \cup \{Z_M \mid Z_M(f_1) = a_2 \text{ and } Z_M(f_2) = a_1\}) \\
\cap \quad & (\{Z_M \mid Z_M(f_1) = a_3 \text{ and } Z_M(f_3) = a_4\} \\
& \cup \{Z_M \mid Z_M(f_1) = a_4 \text{ and } Z_M(f_3) = a_3\}) \\
\cap \quad & (\{Z_M \mid Z_M(f_2) = a_5 \text{ and } Z_M(f_4) = a_6\} \\
& \cup \{Z_M \mid Z_M(f_2) = a_6 \text{ and } Z_M(f_4) = a_5\}) \\
= \quad & \emptyset,
\end{aligned}$$

since $a_1 \neq a_3, a_4, a_5, a_6$ and $a_2 \neq a_3, a_4, a_5, a_6$.

case 6: If f_1 is connected to f_3, f_2 is unassigned, and the edge has neither of the allelic states a_1 and a_2, then again the interpretation simplifies to the empty set of compatible founder allele assignments. This is similar to **case 5**.

$\diamondsuit$

Combining our knowledge from Theorems 3 and 4 we conclude that whenever FastTreeTraversal computes the probability of an inheritance vector then the interpretation of the three sets $\mathcal{A}', \mathcal{E}', \mathcal{U}'$ is exactly the set of compatible founder allele assignments. Furthermore, only inheritance vectors with no compatible founder allele assignments are terminated in the recursion, thus FastTreeTraversal finds all inheritance vector and all their compatible founder allele assignments. That is FastTreeTraversal is correct, since it for every inheritance vector finds the correct probability:

$$P(\mathbf{v}|\mathcal{G}),$$

where $\mathbf{v}$ is the set of all inheritance vectors and $\mathcal{G}$ is the available genotype information for the pedigree.

3.4 Multi Point Analysis

The previous section described the computation of the probability distribution over inheritance vectors based on the genotype information available at a single marker. Recall from page 15 that the genetic linkage map contains multiple markers across each chromosome. Furthermore, Section 2.4 states that the probability of crossover between two adjacent markers is given by the recombination fraction. For this reason, the probability distribution of inheritance vectors at one marker is affected by the probability distributions at the two adjacent markers. This means that all markers on a chromosome influence the probability distribution at all other markers on the same chromosome. The process of inferring information from all other markers is called *multi point probability computation*. Hence, the probability distributions over inheritance vectors for every marker given all available genotype information across the chromosome are computed.

Obviously, multi point analysis is a more accurate representation of the underlying biological model than single point analysis. This is due to the fact that all markers are used. As mentioned in [Gud00] some markers are more informative that others, that is, more genotype information is present at some markers or some markers have more possible allelic states.

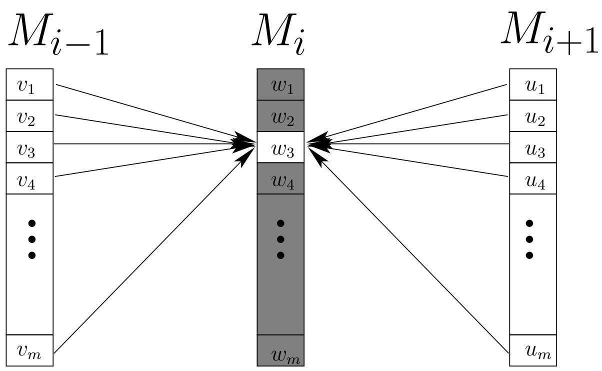

Figure 3-9: Depicting how a single marker is affected by the probability distribution of inheritance vectors from both adjacent markers.

By using the probability distribution obtained from the single point probability computation we can update the individual probabilities at every marker based on the contribution from neighboring markers. This process is depicted in Figure 3-9.

In the remaining part of this section we formalize multi point probability computation by extending the framework introduced in the single point probability computation.

3.4.1 Formal Model

Before we can define multi point analysis, we need to extend our framework with the notion of recombination fractions and Hamming distance.

Definition 10 (Recombination fraction) *The recombination fractions between any two adjacent markers on a chromosome is given by the function θ:*

$$\theta : \mathbb{N}_m \rightarrow [0, \frac{1}{2}],$$

where $\mathbb{N}_m = \{1, \ldots, m\}$ is the set of the first m natural numbers. m is the number of markers on the chromosome.

We use θ_i to denote $\theta(i)$. Intuitively θ_i denotes the probability of an odd number of crossovers[20] occurring between the i-th and $i+1$-th marker.

We introduce the notion of *Hamming distance* between inheritance vectors to reason about the number of odd crossovers that have occurred between two adjacent markers. For any given allele of an individual, the value of the inheritance vector distinguishes between maternal and paternal inheritance. If two adjacent markers on the same chromosome are inherited differently an odd number of crossovers have occurred between the them during meiosis. The Hamming distance describes exactly this difference between two vectors of equal length.

Definition 11 (Hamming Distance) *The Hamming distance between two inheritance vectors v and w over a pedigree $P = \langle F, N, mother, father \rangle$ is defined as:*

$$Ham(v, w) = \sum_{n \in N, \varpi \in \{p,m\}} d_n^{\varpi}; where \begin{cases} 1 : v(n, \varpi) \neq w(n, \varpi) \\ 0 : v(n, \varpi) = w(n, \varpi)) \end{cases}$$

3.4.2 Multi Point Probability Computation

The objective of multi point probability computation is to calculate a better probabilistic estimate of the inheritance pattern for each markers given all available genotype information based on the single point probability distribution. The method we describe is inspired by [LG87] were the problem is described by the means of a *hidden Markov model* (HMM)[21]. This approach is widely used due to the fact that complexity only grows linearly with number of markers introduced, but grows exponentially with size of the pedigree.

[20] See page 9 for more on crossover.

[21] See Appendix D for more on hidden Markov models.

We formalize the computation using the more general framework of *Bayesian Networks.*

Furthermore, we show that HMMs are too restrictive to represent the problem, but that an ordinary Bayesian network with structure similar to a HMM can be used.

Bayesian Networks

In this section we use Bayesian networks as a framework for computing the multi point probability distribution. This is consistent with the theory used in [Gud00] and [LG87]. The description of Bayesian theory used here is based on [Jen01].

Definition 12 (Bayesian Network) *A Bayesian network is a tuple* $\langle \mathcal{V}, E \rangle$ *where:*

- $\mathcal{V}$*: a set of variables each with a set of mutual exclusive states,*
- $E \subseteq \mathcal{V} \times \mathcal{V}$*: a set of directed edges between variables, and*
- *each variable* $A \in \mathcal{V}$ *with parents* $B_1, \ldots, B_n$ *has a (conditional) probability table* $P(A|B_1, \ldots, B_n)$ *attached.*

If $(B, A) \in E$ then the edge is directed from B to A and B is said to be the *parent* of A. We denote the states of a variable A by $A = (a_0, \ldots, a_k)$. When modeling a problem using Bayesian networks the directed edges should represent the causality of the problem domain. That is, there is a edge from B to A if the state of B effects the state of A

The joint probability distribution[22] over all variables in a Bayesian network is calculated using the *chain rule* of Bayesian networks:

Theorem 5 (Chain Rule) *Let BN be a Bayesian network over the variables* $\mathcal{V} = \{A_1, \ldots \mathrm{A}_n\}$*. Then the joint probability distribution* $P(\mathcal{V})$ *is given by:*

$$P(\mathcal{V}) = \prod_i P(A_i|pa(A_i)),$$

where $pa(A_i)$ *is the parent set of* A_i*.*

For the proof we refer to [Jen01, page 21].

Hidden Markov models belong to a special kind of Bayesian network, namely dynamic Bayesian networks[23]. The structure of a hidden Markov model

[22] If the universe, $\mathcal{V}$, consists of three variables A, B, and C then the joint probability $P(\mathcal{V}) = P(A, B, C)$, which is a 3-dimensional table of size $|A| \cdot |B| \cdot |C|$.

[23] Also referred to as *temporal Bayesian networks* in the literature. E.g. [Jen01].

as a dynamic Bayesian network is illustrated in Figure 3-10. Each step, t_i, is referred to as a *time slice* or *time step*. For every time slice i, the conditional probability table $P(O_i|S_i)$ is the same and all transitional probabilities between time slices are the same, that is, $P(S_i|S_{i-1}) = P(S_j|S_{j-1})$ for all i and j. This is consistent with the definition of [Jen01].

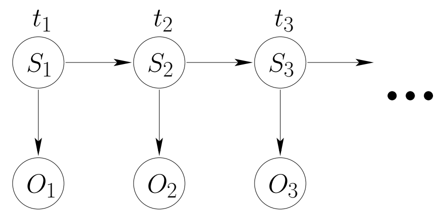

Figure 3-10: The structure of a hidden Markov model as a dynamic Bayesian network.

In the remaining part of this section we explain how multi point probability computation shares structure with hidden Markov models. Under some assumptions the problem can be described as a HMM. These assumptions state that the possible observable genotype information at each marker is the same, and between each marker the recombination fraction is the same. This is necessary in order to keep each conditional probability table identical over time slices (according to the definition of HMMs). We follow the construction introduced by Lander and Green in [LG87] and we show why the constructed Bayesian network is a HMM under these assumptions only.

As introduced by [LG87], the construction of the Bayesian network goes as follows. The parent variable of time slice i, $\mathbf{v}_i$, becomes a variable with states corresponding to the set of inheritance vectors. The child variable for the same time slice, $\mathcal{G}_i$, has states corresponding to the possible genotype information. In this way we keep the causality that the inheritance vectors change the probability of observable genotype information. The i-th probability distribution over inheritance vectors refers to the inheritance distribution for the i-th marker on the chromosome. The a priori probability table of $\mathbf{v}_i$ is the standard Mendelian inheritance distribution stating that, initially, all inheritance vectors have equal probability. The conditional probability table of $\mathbf{v}_j$ for $j > 1$ is given by:

$$P(\mathbf{v}_j = v|\mathbf{v}_{j-1} = w) = \theta_{j-1}^d \cdot (1 - \theta_{j-1})^{n-d},$$

where n is the length of the inheritance vector and d is the Hamming distance between v and w, and θ_{j-1} is the recombination fraction between marker M_{j-1} and M_j. This is because the contribution from adjacent markers is given as the number of crossovers that have occurred, and this is exactly given as the Hamming distance between two markers.

The corresponding structure is depicted in Figure 3-11. Notice that the structure is similar to that of a HMM of length m, where m is the number of markers, but since both $P(\mathbf{v}_j|\mathbf{v}_{j-1})$ and $P(\mathcal{G}_j|\mathbf{v}_j)$ differ at each time slice this is not a HMM. The parent node of each time slice is a variable representing

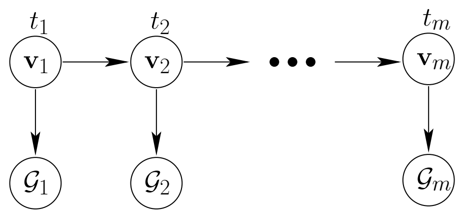

Figure 3-11: Multi point probability computation represented as a Bayesian network with similar structure to a hidden Markov model.

all inheritance vectors, whereas the child node is an instance of genotype information, that is a state of the genotype variable. This is not consistent with the definition of Bayesian networks, but we introduce it to symbolise that the genotype information is always observable, that is, we know the state of the genotype variable. This observation might be that no genotype information is available. Furthermore, to propagate evidence in the form of genotype information we need only $P(\mathcal{G}_i|\mathbf{v}_i)$. In this sense, the state space in each parent node remains the same, but the observations and probabilities change over each time slice.

Using this model we are able to compute the probability of all markers given the available genotype information. Let $\mathcal{G}_i$ denote the observed genotype information at locus i and let $\mathcal{G}_{all} = \{\mathcal{G}_i | 1 \leq i \leq m\}$ denote the set of all genotype information. We can then compute $P(\mathbf{v}_1, \ldots, \mathbf{v}_m|\mathcal{G}_{all})$ by applying Bayes' rule:

$$P(\mathbf{v}_1, \ldots, \mathbf{v}_m|\mathcal{G}_{all}) = \frac{P(\mathbf{v}_1, \ldots, \mathbf{v}_m, \mathcal{G}_{all})}{P(\mathcal{G}_{all})}.$$

Notice that the denominator is just the normalization factor and we need only calculate the numerator. By applying the chain rule for Bayesian networks the numerator is given as:

$$\begin{aligned} P(\mathbf{v}_1, \ldots, \mathbf{v}_m, \mathcal{G}_{all}) &= P(\mathbf{v}_1)P(\mathcal{G}_1|\mathbf{v}_1)P(\mathbf{v}_2|\mathbf{v}_1)P(\mathcal{G}_2|\mathbf{v}_2)P(\mathbf{v}_3|\mathbf{v}_2) \\ &\quad \cdots P(\mathbf{v}_m|\mathbf{v}_{m-1})P(\mathcal{G}_m|\mathbf{v}_m). \end{aligned}$$

Finding the conditional probability distributions for the individual $\mathbf{v}_i$ is just a matter of marginalizing out the other variables.

This approach is inefficient since $P(\mathbf{v}_1, \ldots, \mathbf{v}_m, \mathcal{G}_{all})$ has $2^{|\mathbf{v}| \cdot m}$ entries, where $|\mathbf{v}|$ is the number of inheritance vectors. By rewriting the numerator we get:

$$P(\mathbf{v}_1, \ldots, \mathbf{v}_m, \mathcal{G}_{all}) \propto P(\mathbf{v}_1|\mathcal{G}_1)P(\mathbf{v}_2|\mathbf{v}_1)P(\mathbf{v}_2|\mathcal{G}_2)P(\mathbf{v}_3|\mathbf{v}_2) \cdots P(\mathbf{v}_m|\mathbf{v}_{m-1})P(\mathbf{v}_m|\mathcal{G}_m),$$

since $P(\mathbf{v}_1)$ is the Mendelian probability distribution and from (3-4) in single point probability computation we have that $P(\mathcal{G}_i|\mathbf{v}_i) \propto P(\mathbf{v}_i|\mathcal{G}_i)$. So far we achieved no reduction in size, but the reduction becomes apparent when we want to calculate the probability of a single variable given the genotype information. The reason is that we can use the commutative and distributive properties of marginalization [Jen01, page 16], which means that to marginalize out a variable we need only consider the probability tables containing the variable. This enables us to define the *left-conditioned probability*, P_i^L, and the *right-conditioned probability*, P_i^R, recursively as:

$$P_{i+1}^L = P_i^L \cdot \sum_{\mathbf{v}_i} P(\mathbf{v}_i|\mathcal{G}_i)P(\mathbf{v}_{i+1}|\mathbf{v}_i), \quad 1 \leq i < m \text{ and} \qquad \text{(3-18)}$$

$$P_{i-1}^R = P_i^R \cdot \sum_{\mathbf{v}_i} P(\mathbf{v}_i|\mathcal{G}_i)P(\mathbf{v}_i|\mathbf{v}_{i-1}), \quad 1 < i \leq m, \qquad \text{(3-19)}$$

with base cases $P_1^L = 1 = P_m^R$. From this it follows that the probability of the variable $\mathbf{v}_i$ conditioned by all genotype information is:

$$P(\mathbf{v}_i|\mathcal{G}_{all}) \propto P_i^L \cdot P(\mathbf{v}_i|\mathcal{G}_i) \cdot P_i^R. \qquad \text{(3-20)}$$

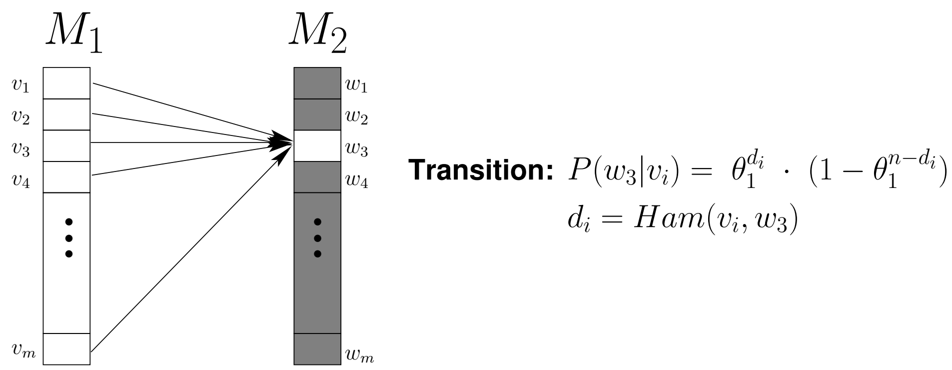

Figure 3-12: One step in the left-conditioned probability calculations.

Intuitively, we first calculate all the single point probabilities at each marker locus. We then transfer the updated probabilities one step at a time across the chromosome towards the marker for which we want to calculate $P(\mathbf{v}_i|\mathcal{G}_{all})$.

Figure 3-12 illustrates the calculation of the left-conditioned probability between marker 1 and 2. In the figure $\mathbf{v}_2 = w_3$ is updated according to all states of $\mathbf{v}_1$.

The idea behind multi point analysis is that we exploit the fact that few crossovers are most likely to occur between two markers, since the recombination fraction between two adjacent markers is always less than $\frac{1}{2}$. For instance, if we have observed the exact state of $\mathbf{v}_1$ as v_3, that is $P(\mathbf{v}_1 = v_3|\mathcal{G}_1) = 1$, then the probability of all inheritance vectors at $\mathbf{v}_2$ close[24] to v_3 get increased probability, whereas inheritance vectors far[25] from v_3 get decreased probability.

More precisely, the contribution from any state v_i of $\mathbf{v}_1$ is given by the probability of the inheritance vector at marker 1 conditioned on the genotype information at that locus, $P(v_i|\mathcal{G}_1)$, times the transition probability $P(w_3|v_i)$. The transition probability is given by the Hamming distance between the two inheritance vectors which indicates the number of odd crossovers that have occurred between two markers on a chromosome. The probability of d crossovers occurring between two markers with an inheritance vector of length n is $\theta_1^d \cdot (1-\theta_1)^{n-d}$.

We then sum over the contribution to w_3 of every vector v_i, which corresponds to marginalizing out $\mathbf{v}_1$. The sum in then multiplied with the conditional probability of w_3 given $\mathcal{G}_2$. This product is proportional to $P(w_3|\mathcal{G}_1, \mathcal{G}_2)$.

We compute this product for every inheritance vector at marker M_2 to get the full probability distribution, $P(\mathbf{v}_2|\mathcal{G}_1, \mathcal{G}_2)$[26]. The updated probability distribution at marker 2 can then be used to calculate the left-conditioned probability at marker 3 and so forth. The right-conditioned probability is calculated in at similar fashion.

Using this procedure we can compute $P(\mathbf{v}_i|\mathcal{G}_{all})$ for any $\mathbf{v}_i$. These values can then be used in the scoring functions, i.e. LOD score, to determine linkage between markers and traits.

3.4.3 Multi point calculation using Fourier Transforms

Kruglyak et. al. shows in [KL98] that the multi point step, hence calculating the left and right conditioned probability distributions, can be done using Fourier Transforms. Inheritance vectors encoded as binary vectors have some nice properties and since the transition probability between to inheritance vectors for adjacent markers only depends on their Hamming distance and the recombination fraction between the two markers. This has led to the use of Fast Fourier Transforms for the multi point step in both Genehunter and Allegro, for which the computational complexity is $O(|\mathbf{v}| \log |\mathbf{v}|)$ per convolution,

[24] Close means that the Hamming distance indicates that few crossovers have occurred.
[25] Far means that the Hamming distance indicates that many crossovers have occurred.
[26] Under the assumption that the result is normalized.

where $|\mathbf{v}|$ is the number of inheritance vectors. A convolution has to be done twice for each marker: Once for calculating the left conditioned probability and once for the right conditioned probability.

In [Gud00] Gudbjartsson shows how the algorithms for Fast Fourier Transforms can be rewritten for better utilisation of CPU cache and registers with speeds-up ranging from a factor of 1.9 to 5.3 depending on the data analysed, [Gud00, page 32-34]. The speed-ups are obtained through a reordering of the computations so that they take advantage of the cache memory, and through unrolling loops which reduces the amount of book-keeping.

For more on Fourier Transforms within the field of linkage analysis, see [KL98] and [Gud00].

3.4.4 Founder Reduction

One way of reducing the number of computations for the set of inheritance vectors for a pedigree is by applying the so called founder reduction, [Gud00, page 34]. The intuitive idea is that consistently changing the allele pointing to a founder, from paternal to maternal and vice versa for all children, yields an new inheritance vector with the same probability as the original vector. This is because we cannot distinguish between maternal and the paternal alleles of founders since the phase is unknown.

Notice that it is not a reduction in the number of inheritance vectors, rather an exploitation of symmetries in the probability distribution.

For the $\{p, m\}$ describing paternal and maternal inheritance we define inversion such that:

$$\overline{p} = m \text{ and } \overline{m} = p.$$

The founder reduction applies for any founder, but for convenience we just state the reduction for one male founder. The reduction for all founders is equivalent. Stated formally:

Theorem 6 (Founder Reduction) *Let $h \in F$ be a male founder in some pedigree $P = \langle F, N, father, mother\rangle$. Let C denote the set of children of h, that is $C = \{n \in N \mid father(n) = h\}$. Let v and w be two inheritance vectors, such that for any $n \in N$:*

$$\begin{aligned} w(n,p) &= \begin{cases} \overline{v(n,p)} : n \in C \\ v(n,p) : \textit{otherwise;} \end{cases} \\ w(n,m) &= v(n,m). \end{aligned}$$

Given some genotype information $\mathcal{G} = \langle \mathcal{L}, astates\rangle$ on the pedigree the following always holds:

$$P(v|\mathcal{G}) = P(w|\mathcal{G}). \tag{3-21}$$

Intuitively, the definition of v and w states that if a child of h has inherited h's paternal allele according to v, then that child has inherited h's maternal allele in w, and vice versa.

Proof: Prior to proving for multi point linkage analysis we prove that it holds for single point linkage analysis. The definition of v and w implies the following relationship between F^v and F^w:

$$\mathsf{F}^w(n,\varpi) = \begin{cases} (h,m) : \ \mathsf{F}^v(n,\varpi) = (h,p) \\ (h,p) : \ \mathsf{F}^v(n,\varpi) = (h,m) \\ \mathsf{F}^v(n,\varpi) : \ \text{otherwise,} \end{cases}$$

where $\varpi \in \{p, m\}$. Theorem 1 states that:

$$\varphi_{\mathcal{G}}^{v} \stackrel{\text{def}}{=} \bigwedge_{n\in\mathcal{L}} \big((f_{\mathsf{F}^v(n,p)} = a_1 \wedge f_{\mathsf{F}^v(n,m)} = a_2) \vee (f_{\mathsf{F}^v(n,p)} = a_2 \wedge f_{\mathsf{F}^v(n,m)} = a_1)\big),$$

where $a_1, a_2 \in astates(n)$. By definition $\varphi_{\mathcal{G}}^{w}$ is equal to $\varphi_{\mathcal{G}}^{v}$ with regard to all sub-expressions, except that any sub-expression:

$$(f_{h,\varpi} = a_1 \wedge f_{n,\varpi'} = a_2) \vee (f_{h,\varpi} = a_2 \wedge f_{n,\varpi'} = a_1),$$

in $\varphi_{\mathcal{G}}^{v}$ looks like:

$$(f_{h,\overline{\varpi}} = a_1 \wedge f_{n,\varpi'} = a_2) \vee (f_{h,\overline{\varpi}} = a_2 \wedge f_{n,\varpi'} = a_1),$$

in $\varphi_{\mathcal{G}}^{w}$. This means that any constraints on founder allele (h, ϖ) in $\varphi_{\mathcal{G}}^{v}$ is also a constraint on $(h, \overline{\varpi})$ in $\varphi_{\mathcal{G}}^{w}$.

This implies that for any founder allele assignment, Z_M, which satisfies $\varphi_{\mathcal{G}}^{v}$ that assigns the allelic state a to (h, ϖ), there exists a similar founder allele assignment with equal probability, Z'_M, which satisfies $\varphi_{\mathcal{G}}^{w}$ that assigns the allelic state a to $(h, \overline{\varpi})$. In other words:

$$\forall Z_M \in [\![\varphi_{\mathcal{G}}^{v}]\!] \ \exists Z'_M \in [\![\varphi_{\mathcal{G}}^{w}]\!] \ \forall n \in F. \ Z'_M(n,\varpi) = \begin{cases} Z_M(n,\overline{\varpi}) : \ n = h \\ Z_M(n,\varpi) : \ \text{otherwise,} \end{cases}$$

for a fixed h. The opposite is also true since we can apply the same argument using v as w and w as v. Furthermore, $P(Z_M) = P(Z'_M)$ since they assign exactly the same set of allelic states to founder alleles, which in turn implies that:

$$\sum_{Z_M \in [\![\varphi_{\mathcal{G}}^{v}]\!]} P(Z_M) = \sum_{Z'_M \in [\![\varphi_{\mathcal{G}}^{w}]\!]} P(Z'_M)$$

and from (3-5) we deduce:

$$P(v|\mathcal{G}) = P(w|\mathcal{G}). \tag{3-22}$$

Now we further prove that founder reduction also holds in the multi point case. To do this we use the fact that founder reduction holds in the single point case, that is we use that (3-22) holds. We consider two markers M_1 and M_2 with computed single point probabilities. We prove that transferring the probabilities from M_1 to M_2 preserves the equality between the probabilities of the two inheritance vectors v_2 and w_2, where v_2 and w_2 are inheritance vectors for marker M_2, and v_2 and w_2 are defined at v and w, respectively, in Theorem 6. Hence, we want to show that two inheritance vectors in the same equivalence class, due to the founder reduction at marker M_2, remain in the same equivalence class after we have propagated the evidence (genotype information) observed at marker M_1.

Assume that a vector, v_1, is some inheritance vector for marker M_1, and w_1 is defined in terms of v_1 as described in Theorem 6. Observe that $P(v_1|\mathcal{G}_1) = P(w_1|\mathcal{G}_1)$, and $P(v_2|\mathcal{G}_2) = P(w_2|\mathcal{G}_2)$ due to (3-22), where $\mathcal{G}_1$ and $\mathcal{G}_2$ are the genotype information for marker M_1 and M_2, respectively. We want to show that:

$$P(v_2|\mathcal{G}_1, \mathcal{G}_2) = P(w_2|\mathcal{G}_1, \mathcal{G}_2) \tag{3-23}$$

Let the recombination fraction between M_1 and M_2 be θ_1. Then contribution in terms of probability from v_1 to v_2 is given by $\theta_1^{d_v}(1-\theta_1)^{n-d_v}$, where $d_v = Ham(v_1, v_2)$ is the Hamming distance between the vectors and n is the length of an inheritance vector for the pedigree. To show that (3-23) holds we need to show that the contribution of v_1 and w_1 to the probability of v_2, is exactly the same as the contribution of v_1 and w_1 to w_2. To do this we prove that $Ham(v_1, v_2) = Ham(w_1, w_2)$ and that $Ham(v_1, w_2) = Ham(w_1, v_2)$, because this implies that v_2 and w_2 are updated with exactly the same probabilities given the observed genotype for M_1. We only prove this for $Ham(v_1, v_2) = Ham(w_1, w_2)$ since the opposite is similar.

The value w_1 and v_1 are identical for all (n, ϖ), except in the case where $\varpi = p$ and $father(n) = h$, in which case the value differs. Since the same applies for v_2 and w_2 we need only concentrate on Hamming distances between the inheritance vectors for children of h. For any (n, ϖ) if $v_2(n, \varpi) \neq v_1(n, \varpi)$ then, by the definition of w_2 and w_1 it follows that $w_2(n, \varpi) \neq w_1(n, \varpi)$. The same argument applies for parts of the inheritance vectors where $v_2(n, \varpi) = v_1(n, \varpi)$. In this case, by definition $w_2(n, \varpi) = w_1(n, \varpi)$, hence the Hamming distance between v_1 and v_2 is equal to the Hamming distance between w_1 and w_2.

The same can be shown for $Ham(v_1, w_2) = Ham(w_1, v_2)$ with the same arguments. For this reason, in the multi point step the founder reduction holds.

$\diamondsuit$

3.4.5 Founder Couple Reduction

In [Gud00] Gudbjartsson defines *founder couple reduction* as an additional way of reducing the number of computations performed during single and multi point linkage analysis. Intuitively, founder reduction states that we cannot distinguish between the maternal and paternal alleles of founders, that is, we can switch between their two alleles. Founder couple reduction states, intuitively, that we cannot distinguish the male and female founder of a founder couple. A founder couple is two founders who share offspring in the pedigree. We state this formally in the following theorem:

Theorem 7 (Founder Couple Reduction) *Let $h, h' \in F$ be the male and female founder, respectively, of a founder couple in some pedigree $P = \langle F, N, father, mother \rangle$. Let C denote the set of children of h and h', that is $C = \{n \in N \mid father(n) = h \text{ and } mother(n) = h'\}$. Let v and w be two inheritance vectors, such that for any $n \in N$:*

$$w(n, \varpi) = \begin{cases} v(n, \overline{\varpi}) : n \in C \\ \overline{v(n, \varpi)} : father(n) \in C \text{ or } mother(n) \in C \\ v(n, \varpi) : otherwise. \end{cases}$$

Given some genotype information $\mathcal{G} = \langle \mathcal{L}, astates \rangle$ on the pedigree where $h, h' \notin \mathcal{L}$ and that neither h nor h' has any children in V not in C, the following always holds:

$$P(v|\mathcal{G}) = P(w|\mathcal{G}). \tag{3-24}$$

We do not prove this, but refer the keen reader to [Gud00, page 35] for more information on founder couple reduction.

3.5 Summary

The first subject of this chapter was an introduction to linkage analysis. The purpose of linkage analysis is to locate the position of trait causing genes and update the genetic linkage map accordingly. The main problems of this process is the missing genotype information, but even with missing information it is often possible to consolidate the knowledge and infer information about location of the gene in question. Another aspect is the complexity of current methods which limit either the size of the pedigree of the number of markers in the analysis.

After the introduction we presented a formal framework that allowed us to reason about the concepts used in linkage analysis. This framework primarily consists of definitions and properties of the biological model presented in

Chapter 2. Both single point and multi point linkage analysis were described using the formal framework.

Two algorithms, KRUGLYAK and FASTTREETRAVESAL, for finding compatible founder allele assignments and probabilities of inheritance vectors were analysed for correctness and complexity using the formal framework. These algorithm were proved by defining interpretations of the output from the to algorithms in terms of founder allele assignments. These interpretations proved equals to the definition of compatibility introduced in the framework. The complexity of KRUGLYAK is $O(|N| \cdot 2^{2 \cdot |N|})$ and the complexity of FASTTREETRAVERSAL is $O(2^{2 \cdot |N|})$. Both algorithms can be reduced in complexity be using founder reduction and founder couple reduction.

The formal framework was extended to describe multi point linkage analysis. We found that multi point linkage analysis cannot, as described in the literature [LG87], be described as a hidden Markov model. Instead we argued that a Bayesian network which shares structure with hidden Markov models is a more accurate description of the problem.

Chapter 4

Towards an Improved Method

As we have documented, the current methods for linkage analysis are computationally heavy. The best known algorithm is the algorithm used in Allegro (See Section 3.3.3)[1]. The purpose of this chapter is to present a discussion of strategies for the development of a method, by which linkage analysis can be performed more efficiently. The strategies considered are inspired by classical concepts of computer science such as: Compositionality, abstraction, and symbolic representation.

A possibility for speeding up the computation is to approximate the probability distribution over inheritance vectors. Approximation is an important future work aspect pointed out in [Gud00]. The approximation approach opens a range of possibilities, but what in the end is needed is still a probability distribution that is consistent with the underlying biology, so linkage can be detected. Thus small deviations seem tolerable. However, approximation is out of scope for this project, because our interest is not in proving statistical properties, relative to the true probability distribution, of an approximation method, rather we wish to apply a computer scientific approach to linkage analysis. We believe that it is still possible to develop better methods, which produce exact results. Therefore we will *not* pursue approximation as a possible optimisation any further in this report.

Another interesting aspect is to establish the computational complexity of linkage analysis. If it turns out to be NP-complete, it makes little sense to look for polynomial-time solutions. Initially, it seems appealing to establish a reduction from a satisfiability-based problem to the problem of sorting out the incompatible inheritance vectors.

The structure of this chapter is as follows. First, we determine the relevant focus for approaching the problem. Then, before presenting possible

[1] A paper was recently been published that claims several orders of magnitude in improvements of time and memory requirements, relative to Genehunter version 2.0 (KRUGLYAK). The algorithm is implemented in Genehunter version 2.1, [MDK01].

optimisation strategies, a clear definition of how a method can be considered an optimisation relative to other methods, e.g. when an algorithm is correct and how to compare the complexities, is needed. We then investigate the theoretical hardness of linkage analysis. Strategies for improvements are then considered. Finally, we summarise the results of the chapter.

4.1 Level of Focus

Multi point linkage analysis might be improved on several levels. Optimisations and improvements can be done on everything from the code level to changing the biological model. Basically, we have divided the task into three main areas (see Figure 4-1). Our interpretation of the "biological model" is that it is a model, which is based purely on biological entities and their relations[2]. The "representation and algorithms" level focuses on representing the needed data and providing effective algorithms. The "source code" level refers to optimisations that are very low-level and focuses on implementing the details of the algorithms.

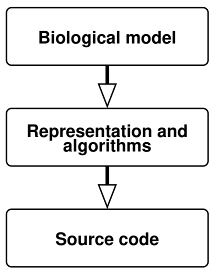

Figure 4-1: Tasks of establishing tools for linkage analysis.

Optimisations on the source code level could either be done by implementing the existing algorithm emphasising on code optimisations - or by optimising or distributing an existing implementation. However, neither of the two approaches seem appealing. Optimising on the source code level is infeasible, since it can only be expected to yield minor improvements and current compiler techniques already takes care of the most obvious optimisations. Distributing the computation does not yield improved algorithms, in terms of the work performed, but is a possibility for minimising the time used on each data set.

[2]Note that [JIS93] refers to the "representation and algorithm" level as the "biological level". We disagree with this classification, because we do not consider modifications of an algorithm or a data structure, to a different one with equivalent semantics, as tampering with the biological model. Our interpretation of the biological model is that it is a model, which is based purely on biological entities and their relations.

By modifying the biological model, on the other hand, one could hope to be able to develop a different set of algorithms, which have a lower complexity as the set of algorithms currently in practice. The biological model is widely accepted in the biological and statistics community, therefore we do not find it meaningful to find solutions on this level[3]. We aim at improving the multi point linkage analysis on some intermediate level, while maintaining the biological model[4]. This implies that the notions of inheritance vectors, recombination fractions, single and multi point probability distributions still apply. However, we still want to develop a data structure and a set of algorithms, by which we are able to reduce the computational complexity of multi point analysis.

Since we do not intend to change the biological model, a new method for doing the analysis should of course be correct in the sense that results obtained using the existing method and results obtained by a new method should be identical. We are now ready to describe the properties that an improved method should posses.

4.1.1 Methods Properties

The most fundamental result of new methods for linkage analysis should be probability distributions over inheritance vectors, given all observed genotype data, which must be suitable for further statistical analysis. What we mean by suitable for further statistical analysis, is that it must be possible to extract the necessary data for applying the scoring functions (see Appendix B), which again allows the potential detection of linkage.

The probability distributions produced by new methods, should not differ from the probabilities obtained by the left and right conditioned probabilities on page 60 (which again equals the distributions produced by Genehunter and Allegro). The reason for this is that they are defacto methods for obtaining the probability distributions and adhere to the biological model. What can be optimised are the time and space complexities that currently exist for determining these distributions. The best known current method is the one implemented in Allegro (see Section 3.3.3), the worst case complexity of Allegro is still exponential in the number of non-founders.

The software currently available, including Allegro, all use flat representations of probability distributions over inheritance vectors (arrays of floating point numbers indexed by the decimal values of the inheritance vectors). An improvement should include the single point probability computation to remedy this representation of probability distributions, otherwise a new method would have to do the left and right conditioned probability computation step

[3]We also consider the insight needed for tampering with the biological model out of scope for this project.

[4]Allegro uses approximately 75 % of the time in computing the multi point distributions, [Gud00].

faster than Fast Fourier Transform (FFT). However, improving the Fast Fourier Transformation technique is not within the scope of this project. Hence, we will require that a new method includes an improvement of the single point probability computation, and produces a more compact representation of the probability distributions, or else it is not possible to perform faster average multi point analysis than in exponential time.

To clarify the goals:

Definition 13 (Method Properties) *The properties adhered to by an improved method are the following (let $NEW_x(y)$ denote the result of calculating y with a new method for x):*

I. Single point probability computation:

$$NEW_{singlepoint}(\mathbf{v}_i) = \sum_{Z_M \in \mathcal{F}_M^v} P(Z_M).$$

II. Multi point probability computation:

$$NEW_{multipoint}(\mathbf{v}_i) = P_i^L \cdot P(\mathbf{v}_i|\mathcal{G}_i) \cdot P_i^R.$$

III. Extract statistical information: The information needed to compute LOD, NPL, and other scoring functions must be available.

IV. The complexity of some new cases of the new method should be better than the average complexity of the best known current method[5].

4.2 Hardness of Linkage Analysis

An important aspect of optimisations to computationally hard problems is reasonings about the complexity class of the problem. If the problem can be shown to be NP-complete it makes little sense to seek a P-time algorithm (unless P=NP). In this section we discuss approaches to establishing the complexity class, in which the general task of linkage analysis belongs.

As already mentioned, the term linkage analysis is a broad term covering several different aspects of locating trait causing genes. We therefore reduce our focus, in this section, to the task of determining the compatible inheritance vectors for a pedigree, P, with some genotype information, $\mathcal{G}$, for a single marker. We denote this problem as COMP. The reasoning behind this selection

[5]An establishment of the currently best known average complexity is out of scope in this project, because it requires detailed knowledge of the expected data. For instance, Allegro's running time depends significantly on how many subsets of inheritance vectors that are compatible. We regard an experimental comparison on realistic data sets as an adequate indicator for the effectiveness of different methods.

is that COMP is a central task in linkage analysis, whether it is single or multi point linkage analysis.

NP-complete problems are decision problems. COMP differs from these in the sense that it is not a decision problem, but a problem of determining a set of possible solutions to a given specification. We now define a language, for which membership testing is decidable, and which intuitively is close to COMP:

$$SET = \{\langle P, \mathcal{G}, W\rangle \mid W \text{ is the set of inheritance vectors, where for each there exists a compatible founder allele assignment for } P \text{ given } \mathcal{G}\}.$$

A key issue of a Turing Machine (TM) that decides SET, TM_{SET}, is the size of W. In its most simple form W is simply a list of all the compatible inheritance vectors, which is potentially exponentially large. A more sophisticated W could be a polynomial size encoding of the compatible inheritance vectors. In the most simple form a polynomial size encoding could be another pedigree P' and some genotype information on this pedigree $\mathcal{G}'$ of which exactly W is the compatible vectors. This encoding needs some sort of mapping from inheritance vectors in P' to inheritance vectors in P. This is just to illustrate that there might exist some simple polynomial encoding, but this does not bring us closer to a solution of COMP.

If W is polynomial size and the output of TM_{COMP} is P-time comparable to W the complexity of TM_{SET}, cannot be harder than a TM that solves COMP, TM_{COMP}, because we can construct a TM that uses TM_{COMP} to decide SET in P-time. That is we can construct a polynomial time reduction from SET to COMP (i.e. $SET \leq_P TM_{\text{COMP}}$). The specification of TM_{SET} is as follows:

TM_{SET} = "On input $\langle P, \mathcal{G}, W\rangle$.
1. Run TM_{COMP} on $\langle P, \mathcal{G}\rangle$.
2. Compare the result of **1.** with W.
2.1 If they are equal *accept*.
2.2 Otherwise *reject*."

Because of this reduction, membership testing of SET is no harder than a central task of the current algorithms, KRUGYLAK and FASTTREETRAVERSAL, which solves the COMP problem. This is because COMP is no easier than testing membership of SET, since we can use a solution to COMP to decide SET. A further implication is that if we can reduce some NP-complete problem to decide SET, then solving COMP is no easier than solving an NP-complete problem.

A slight modification of SET gives the following language:

$$CONS = \{\langle P,\, \mathcal{G}\rangle \mid \text{For some inheritance vector } v \text{ there exists a compatible founder allele assignment over } v \text{ for } P \text{ given } \mathcal{G}\},$$

where P is a pedigree and $\mathcal{G}$ is some genotype information. This is the language investigating whether $W \neq \emptyset$ in SET. i.e. whether there exists an inheritance vector that is compatible with P given $\mathcal{G}$. If some $\langle P, \mathcal{G}\rangle \notin CONS$ then the pedigree is not consistent (hence $CONS$) with the genotype information. That is, either the genotype information is invalid[6] or some parent information is not correct. In paternity and other genealogy studies this is a central issue.

We argue that $CONS \leq_P SET$ because of the following TM (TM_{CONS}):

TM_{CONS} ="On input $\langle P, \mathcal{G}\rangle$.
1. Run TM_{SET} on $<P,\, \mathcal{G},\, \emptyset>$.
1.1 If TM_{SET} accepts, *reject*.
1.2 If TM_{SET} rejects, *accept*".

Membership testing of $CONS$ can, stated in propositional logic, be thought of as a satisfiability problem. Intuitively, translating the requirements of $CONS$ membership to propositional logic seems fruitful, since satisfiability of propositional logic is a well studied area with many results, e.g. satisfiability of Horn clauses, disjunctive normal form (dnf), and conjunctive normal form with at most two literals (2cnf) is known to be decidable in polynomial time. If $CONS$ can be translated to propositional logic in any of these forms we know that $CONS$ is P-time decidable.

It seems possible to model inheritance dependencies of allelic states, dictated by inheritance vectors, in propositional logic as Horn clauses. For this reason we investigate the possibility of translating membership testing of $CONS$ to satisfiability of propositional logic. In the following we explain how to build propositional formulae for which satisfiability corresponds to testing membership of $CONS$.

Propositional Variables

For each non-founder $n \in N$, we introduce four propositional variables $y_m^{(n,p)}$, $y_m^{(n,m)}$, $y_p^{(n,p)}$, and $y_p^{(n,m)}$ representing the value of the inheritance vector, v. That is, $y_m^{(n,\varpi)}$ represents the truth assignment of $v(n,\varpi) = m$ and $y_p^{(n,\varpi)}$ represents the truth assignment of $v(n,\varpi) = p$, where $\varpi \in p, m$.

From Definition 8 we know that, given a founder allele assignment (Z_M) and an inheritance vector (v)[7], all individuals in the pedigree are, unam-

[6] This can be due to mutations in marker genes or genotyping errors.
[7] See definition 6.

biguously, assigned one maternal and one paternal founder allele given by $\mathsf{F}^v(n,\varpi)$. In other words we say that the allelic state of (n,ϖ) is determined by $Z_M(\mathsf{F}^v(n,\varpi))$. For each individual, $n \in V$, we introduce propositional variables, $\xi_a^{(n,\varpi)}$, for each possible allelic state $a \in A_M$, where A_M is the set of allelic states for the marker under consideration. Thus $\xi_a^{(n,\varpi)}$ represents whether the allelic state of (n,ϖ) is a or not.

Clauses for Testing $CONS$ Membership

As mentioned above, the determination of allelic states in the pedigree is based on the inheritance vector and the founder allele assignment. We now state these requirements dictated by Definition 8:

$$\begin{aligned}
y_m^{(n,m)} \wedge \xi_a^{(n,m)} &\rightarrow \xi_a^{(mother(n),m)}, \\
y_p^{(n,m)} \wedge \xi_a^{(n,m)} &\rightarrow \xi_a^{(mother(n),p)}, \\
y_m^{(n,p)} \wedge \xi_a^{(n,p)} &\rightarrow \xi_a^{(father(n),m)}, \text{ and} \\
y_p^{(n,p)} \wedge \xi_a^{(n,p)} &\rightarrow \xi_a^{(father(n),p)}.
\end{aligned}$$

These implications can be rewritten as Horn clauses, i.e.

$$y_m^{(n,m)} \wedge \xi_a^{(n,m)} \rightarrow \xi_a^{(mother(n),m)} \Leftrightarrow \neg y_m^{(n,m)} \vee \neg \xi_a^{(n,m)} \vee \xi_a^{(mother(n),m)}.$$

The left most part (relative to the biimplication) indicates: If there is inheritance from the maternal grandparent and the individual is assigned allelic state a, then the maternal allele of the mother must also have allelic state a[8]. This set of clauses determines the compatible patterns of inheritance. Notice that we need two propositional variables to reason about the ancestral origin of a single allele (one entry in the inheritance vector), that is, we cannot use $\neg y_m^{(n,m)}$ instead of $y_p^{(n,m)}$ since $\neg y_m^{(n,m)} \wedge \xi_a^{(n,m)} \rightarrow \xi_a^{(mother(n),p)}$ is not a Horn clause because of the two positive literals when rewriting the expression.

The above set of clauses needs to be joined with clauses for genotype information. A homozygous genotyped individual is easily expressed as two Horn clauses:

$$\text{Homozygous with } a : \xi_a^{(n,p)} \wedge \xi_a^{(n,m)}.$$

Unfortunately, representing heterozygous genotyped individuals turns out to be more difficult. The reason is that we need to state that each allele should have exactly one of the allelic states. In other words we need to state(for two alleles and the paternal assignment):

$$(\xi_{a_1}^{(n,p)} \vee \xi_{a_2}^{(n,p)}) \wedge (\neg \xi_{a_1}^{(n,p)} \vee \neg \xi_{a_2}^{(n,p)}).$$

[8] Conversely for the paternal case.

This is, however, not a set of Horn clauses (because of the two positive literals in the first clause), and we cannot state it as such. Actually, the same problem arises with the inheritance vector, since the above implications permits both $y_m^{(n,m)}$ and $y_p^{(n,m)}$ to evaluate to false, which is inconsistent with the biological model, because a single allele must be inherited from either a paternal or a maternal source, but not both.

The genotype information, however, can be stated as 2cnf as just shown. Furthermore, the described problem with inheritance vectors can also be stated as 2cnf in a similar fashion. In this way we have a combination of clauses both of which can be tested for satisfiability in polynomial time. This is unfortunately, to our knowledge, not the case for the combination of the two.

We have found no way of expressing genotype information as Horn clauses and no way of expressing inheritance patterns as 2cnf. Could genotype information be expressed as Horn clauses or inheritance patterns as 2cnf the we would know that $CONS$ was decidable in polynomial time.

Regarding disjunctive normal form then we can construct a propositional formula in dnf for the problem. This formula, however, would be exponential in the size of the pedigree since it would be a disjunctive listing of all possibilities which corresponds to explicitly expressing every possible pattern of inheritance.

To summarise, we have investigated the possibilities of performing membership testing of $CONS$ in polynomial time using propositional logic. No appropriate formulae have been established. This does not imply that a propositional formula does not exist and further investigation seems reasonable. We have shown the construction of Turing machines which indicate the relationship between COMP, SET, and $CONS$ depicted in Figure 4-2. We know that reducing some NP-complete problem to either SET of $CONS$ would imply that COMP is no easier to solve than a NP-complete problem. This would be a strong indication that no polynomial time algorithm exists for COMP.

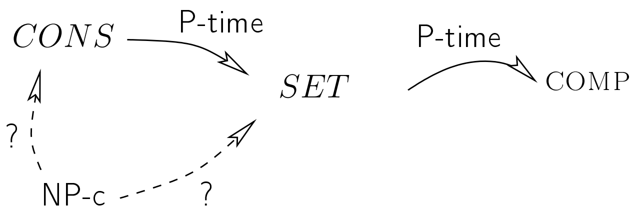

Figure 4-2: The known relationship between several decidability problems. The dashed lines represent interesting reductions, that can lead to the conclusion that COMP is at least as hard as NP-complete problems.

4.3 Symbolic Representation

We have in Section 3.3.1 shown that the problem of determining the set of compatible founder allele assignment, $\mathcal{F}^v_{\mathcal{G}}$, can be described in terms of propositional logic. Well known heuristics for handling propositional logic are the xDD structures. The fact that the problem of finding compatible founder allele assignments can be expressed using a xDD has made us look further into heuristic-based solutions capable of expressing the probability distribution over all inheritance vectors for a pedigree.

We have examined two data structures, namely the Multi Terminal Binary Decision Diagram (MTBDD), [KKNP01] and [KNPS99] and the Probabilistic Decision Graph (PDG)[9], [BM99]. Both of these are discussed in the context of linkage analysis in Section 4.3.1 and 4.3.2. Some familiarity with the xDD structures and operations on them is assumed through out the rest of this section.

4.3.1 Evaluating MTBDDs

A MTBDD is a graph based representation for functions of the type $f : \mathbb{B}^n \mapsto \mathbb{R}$. This is expressed in the following definition:

Definition 14 (MTBDD) *A MTBDD over the boolean variables $Vars = \{x_1 \ldots x_n\}$ with a fixed ordering $\prec$ is a rooted, directed, acyclic graph $M = \langle T, NT, var, child_0, child_1, value\rangle$, where T is the set of terminal nodes, NT is the set of non terminals, and four functions:*

- *$var : NT \rightarrow Vars$,*
- *$child_0 : NT \rightarrow NT \cup T$,*
- *$child_1 : NT \rightarrow NT \cup T$, and*
- *$value : T \rightarrow \mathbb{R}$.*

This seems to fit well with the computations introduced in Section 3.2.2, where we formulate the need to compute the probability for all compatible inheritance vectors. With MTBDDs we can express an inheritance vector as the path from the root node to a terminal node, if $value(t \in T) = 0$, the path expresses a incompatible vector. If $value(t \in T) \neq 0$ the terminal node expresses the probability for an inheritance vector given the genotype information, $P(v|\mathcal{G})$. In [KNPS99] the REDUCE operation is applied on a MTBDD reducing it in the same manner as BDDs are reduced. This implies that it

[9] [BM99] gives the name PDG to avoid another xDD acronym. i.e. a PDG could also be called a PDD.

might be possible to collapse some of the nodes. The following example shows how it is possible to construct a MTBDD representing the probabilities of all inheritance vectors of a pedigree with genotype information.

Example 15 *Given the pedigree of Figure 4-3 with genotyped individuals*[10] $\mathcal{L} = \{n1, n2\}$, *a marker* $M = \langle A_M, \pi_M \rangle$, *where the allele set is* $A_M = \{a, b, c, d\}$, *and* $\pi_M(a) = 0.5$, $\pi_M(b) = 0.2$, $\pi_M(c) = 0.2$ *and* $\pi_M(d) = 0.1$.

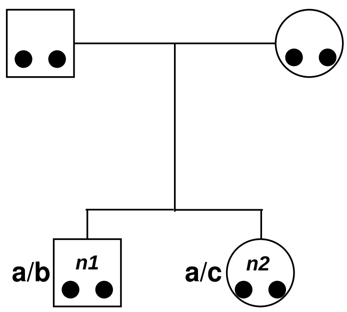

Figure 4-3: The example pedigree for the illustration of MTBDDs in the context of linkage analysis.

Figure 4-4 illustrates the inheritance vectors for this pedigree and genotype information in a MTBDD. The path from the root to each non-terminal represent the subset of inheritance vectors. E.g. the path $n1p \xrightarrow{0} n1m \xrightarrow{1} n2pn$ *represents the subset of inheritance vectors for which* $v(n1, p) = p$ *and* $v(n1, m) = m$. *A terminal represent the probability of all the vectors with a path leading to the terminal.*

The inheritance vector for which $v(n1, p) = p$, $v(n1, m) = p$, $v(n2, p) = m$ *and* $v(n2, m) = m$ *has the probability* $P(\mathcal{G}|v) = 0.0025$. *The alert reader would notice that* 16 *terminal nodes are present in the MTBDD only describing three different probabilities, which indicate a presence of some equivalence between the probabilities of some inheritance patterns. Applying the* REDUCE *operation on the MTBDD of Figure 4-4 produces the MTBDD of Figure 4-5, reducing the number of nodes from* 31 *to* 14.

The example shows promising results when the single point case is considered. However, results have indicated that the number of terminal nodes and thus, the number probability classes change for the multi point case. We have investigated this, by evaluating some of the debug information produced by Allegro. These debug files reveals information on intermediate results. The data that we have used is the right conditioned probabilities[11] for all markers

[10] See Section 3.2 for the formal definitions used in this example.

[11] See Section 3.4 for the definition of the left and right conditioned probabilities.

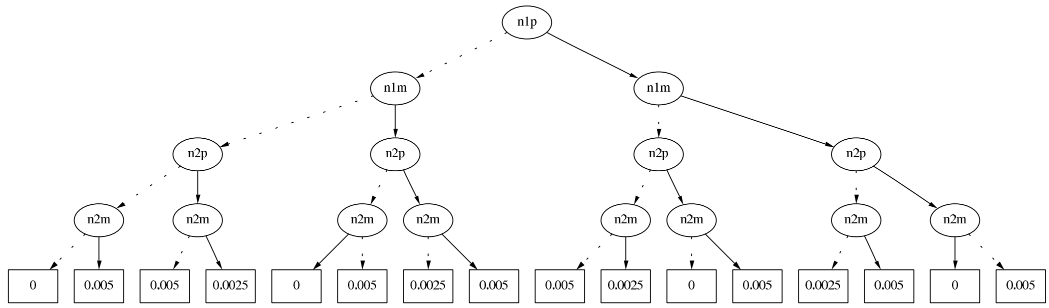

Figure 4-4: The possible inheritance vectors of the pedigree of Figure 4-3 illustrated in a MTBDD before REDUCE is applied. Each terminal node represent the probability $P(\mathcal{G}|v)$. The full lines represent that the parent node is set to "1" (m), and the dotted lines that the parent node is set to "0" (p). Note that $P(\mathcal{G}|v)$ is not normalised to get $P(v|\mathcal{G})$, since the values of the terminal nodes do not sum to 1.

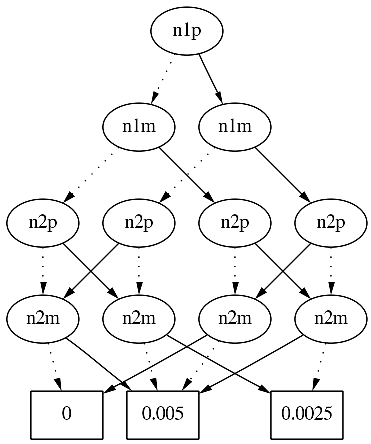

Figure 4-5: The possible inheritance patterns of the example pedigree in a reduced MTBDD. The full lines represent that the parent node is set to "1" (m), and the dotted lines that the parent node is set to "0" (p).

multiplied with the single point probability for the marker. Furthermore, the debug data has been post processed in order to create suitable graphs. Analysis of this data indicates, that the number of different probabilities grows over each maker in the multi point case, this is depicted in Figure 4-6.

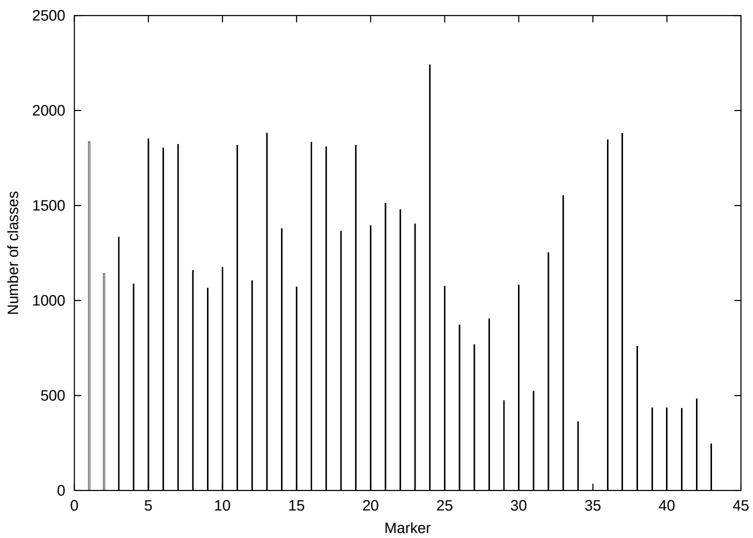

Figure 4-6: A graph showing the number of different probabilities for each marker in a 12-bit family (an inheritance pattern for the family can be described by an inheritance vector with a length of 12 bits).

Figure 4-6 shows the number of markers on the x-axis and the total number of probabilities at each marker on the y-axis. At marker 43 (right most bar) there are only a couple of hundred different probabilities and thus the same number terminal nodes. But at the final markers moving from right to left, the number of different probabilities is close to half the number of vectors in the state space. This is an unfortunate property that all of the pedigrees investigated share.

As stated, MTBDDs could be beneficial for the single point probability computations, and actually the FastTreeTraversal can be viewed as an implicit construction of the non-reduced MTBDD for each marker. The problem arises when calculating the conditional probabilities in the multi point analysis, as we have found no means of exploiting the compact representation of the MTBDD. Furthermore, multi point computation is based on the Hamming distance between inheritance vectors, which disables the exploitation of the equivalence classes, as each vector must be considered separately. Access time to an inheritance vector in a MTBDD is linear in the length of the vector opposed to the constant time access in the explicit representation, with vectors stored in an array.

4.3.2 Evaluating PDG's

Probabilistic Decision Graphs are generalisations of Probabilistic Decision Trees, which are defined as follows:

Definition 15 (Probabilistic Decision Tree) [BM99]
A probabilistic decision tree (PDT) of depth n is a tuple $D = \langle S, 0, 1, v \rangle$ where $S = \mathbb{B}^n$, 0 and 1 are respectively the left and right successor partial functions on S and $v : s \rightarrow [0,1]$, $v(\epsilon)$[12] $= 1$ and for every non-leaf node s, $v(s0)+v(s1) = 1$ and $P(sx) = P(s) \cdot v(sx)$, where the nodes $s0$ and $s1$ are child nodes of s and where sx is any child node of s.

PDTs do not solve any problems since they are exponential in the number of variables used. Therefore [BM99] defines the concept of PDGs following:

Definition 16 (Probabilistic Decision Graph) [BM99]
Let $D = \langle S, 0, 1, v \rangle$ be a PDT and let $\sim$ be a congruence relation on S defined as $s \sim s'$ if $v(s) = v(s')$ and both $s0 \sim s'0$ and $s1 \sim s'1$. The associated PDG is $G = \langle S/\sim, 0, 1, v \rangle$.

This shows that the $\sim$ relation utilises equivalence classes for compact representation which, just as MTBDDs, seems to enable us to create a symbolic representation. In [BM99], where PDTs and PDGs are presented, the authors state that PDGs can succeed in some situations where MTBDDs fail, more specifically that MTBDDs fail when the corresponding vectors and matrices do not have a lot of identical entries, but that PDGs can be used in case the domain has certain structural properties. As shown in the previous section it is exactly the case that we do not have many identical probabilities, but that we have some structure.

The PDG structure, however, does not seem to be directly applicable to the problem domain. There remain a lot of unanswered questions, such as:

- In [BM99] there is specification on an algorithm to convert a PDT to a PDG. We would somehow have to do this on the fly, so that we do not need the full PDT before reducing it to a PDG.
- The article introduces a way of representing $n \times n$ matrices using $nlog_2n$ space, if the matrix satisfies a certain constraint. The probability matrices between markers in multi point linkage analysis satisfy this constraint. This structure can be used to perform vector multiplication with

[12] The empty string ϵ denotes the entire $\mathbb{B}^n$.

matrices getting a PDG as result. The article, however, does not state a complexity of this operation, and a such needs to be established to evaluate the efficiency of PDG's.

- An efficient algorithm for multiplication of two PDGs is needed in the multipoint step in the left and right step. The authors of [BM99] do not sketch such an algorithm.

We cannot conclude whether the use of PDGs would yield an improved method. It depends on whether the above issues can be solved efficiently.

4.3.3 Future Heuristics

Neither the MTBDD nor the PDG are directly applicable to the problem of linkage analysis, although our evaluation of the two heuristics shows that further investigation into heuristic models could be fruitful. The research has shown that it is possible to reduce the state space using MTBDDs, but that this heuristic fail in the multi point analysis. Further investigation into heuristics might consider linkage analysis as a whole or go along tailoring a heuristic that is usable in both single and multi point analysis.

Furthermore, an investigation on the logic properties of the structure of a pedigree might reveal methods of decomposition. We now reason about exploiting this structural information.

4.4 Utilising the Structural Information of Pedigrees

As we have shown, MTBDDs do not seem to yield promising results, and PDGs will only work in the case where efficient algorithms for converting a PDT to a PDG, and multiplication of two PDGs exist. The reason for their inability to work right of the box could be that they are too general for our problem domain, and that the probability of an inheritance vector given some genotype information, $P(v|\mathcal{G})$, cannot directly be related to individual subsets of the pedigree. That is, the probability of an inheritance pattern defined on the pedigree with the set of N non-founders, $v : N \times \{p, m\}$, is not always determined by its values for $N' \subset N$ independent of $N'' \subset N$ where $|N'|, |N''| > 0$ and $N' \cap N'' = \emptyset$, because for some inheritance probability distribution $v \in \mathbf{v}$, n' and n'', might inherit the same founder allele, where $n' \in N'$ and $n'' \in N''$.

In this section we show how the probability distribution of inheritance patterns, $\mathbf{v}$, for a pedigree can be recursively partitioned into subset $\mathbf{v_1} \cup \mathbf{v_2} \cup \mathbf{v_3} \cup ... \cup \mathbf{v_n} = \mathbf{v}$ in which all the inheritance patterns in a subset $\mathbf{v}_i$ share some *inheritance dependencies*. We formalize the notion of *inheritance dependence* and show how the structure of a pedigree might be exploited to obtain a smaller space complexity, than that of the current applications, such as Allegro and Genehunter.

Dependencies in Pedigrees

The algorithms in Allegro and Genehunter already utilise some of the structural information of the pedigrees analysed by using the *founder reduction*[13], furthermore the algorithms in Allegro also use the *founder couple reduction*[14], however we believe that the structure of the pedigree can be exploited even further to reduce the complexity of the probability calculations.

Definition 17 (Inheritance Independence) *Given an inheritance pattern $v \in \mathbf{v}$ and a pedigree P, where N are the non-founders of P, two disjoint subsets N' and N'' of N, are inheritance independent iff:*

$$\begin{aligned} \mathcal{I}_v^{N'} \cap \mathcal{I}_v^{N''} &= \emptyset, \ and \quad (4\text{-}1) \\ \mathcal{L} \cap \{\ f \mid (f, \varpi) \in \mathcal{I}_v^{N'} \ and \ (f, \varpi') \in \mathcal{I}_v^{N''} \varpi, \varpi' \in \{m,p\}\} &= \emptyset, \quad (4\text{-}2) \end{aligned}$$

where $\mathcal{I}_v^N$ is a set of founder alleles:

$$\mathcal{I}_v^N = \{\ f \mid f = \mathsf{F}^v(n,m) \ or \ f = \mathsf{F}^v(n,p) \ for \ some \ n \in N\}. \quad (4\text{-}3)$$

Hence, N' and N'' are independent if none of the individuals in N' inherit any of the founder alleles inherited by the individuals in N'' (4-1), and that no common founder of the two sets are genotyped (4-2).

If two subsets of N are not independent they are said to be inheritance dependent.

Theorem 8 (Decomposition Based on Inheritance Independence) *Let N' and N'' be non-empty disjoint subsets of N, where N is the set of non-founders for some pedigree. Let $N' \cup N'' = N$. Let $\mathbf{v}_{\mathcal{I}}$ be the subset of $\mathbf{v}$ where N' and N'' are inheritance independent. Then the single point probability of an inheritance pattern $v \in \mathbf{v}_{\mathcal{I}}$ given some genotype information $\mathcal{G}$ is given by:*

$$P(v|\mathcal{G}) = P(v_{N'}|\mathcal{G}) \cdot P(v_{N''}|\mathcal{G}), \quad (4\text{-}4)$$

where, $v_{N'} : N' \times \{p,m\} \rightarrow \{p,m\}$ and $v_{N''} : N'' \times \{p,m\} \rightarrow \{p,m\}$ is defined as:

$$\begin{aligned} \forall \quad & n \in N'.v_{N'}(n,\varpi) = v(n,\varpi) \\ \forall \quad & n \in N''.v_{N''}(n,\varpi) = v(n,\varpi), \\ \text{where} \quad & \varpi \in \{p,m\}. \end{aligned}$$

[13] See Section 3.4.4 for more on *founder reduction*.

[14] See Section 3.4.5 for more on *founder couple reduction*.

Proof: Since none of the individuals in N' inherit a founder allele that some individual in N'' inherit, and since no common founder is genotyped by definition of inheritance independence, we can see that the Graph produced by the KRUGLYAK algorithm (See Section 3.3.2), will contain at least two connected components. One containing the founder alleles (or nodes) inherited by the individuals in N' and the other containing those of the individuals in N''. As shown in (3-12) on page 36 the following holds:

$$P(Graph) = \prod_{X \in CC} \sum_{s \in X} P(s),$$

where CC denotes the set of connected components and $P(s)$ denotes the probability of the solution to a connected component. Hence, (4-4) must hold. $\diamondsuit$

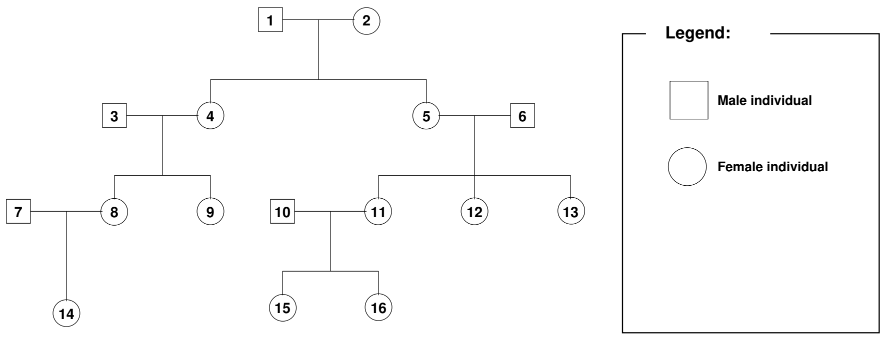

Figure 4-7: An example pedigree `ex1 f1` distributed with the Allegro software package.

Example 16 *This example is meant to give the reader an intuitive idea about under which conditions two sub-pedigrees are inheritance independent. Consider the pedigree in Figure 4-7. It contains two sub-pedigrees, one where individual 3 and individual 4 are considered as founders, and one where individual 5 and individual 6 are a considered as founders. These two sub-pedigrees need only to be considered in a unified whole, when a genotyped individual from each of the sub-pedigrees inherit the same founder allele. In the pedigree it is only when* $v(4,m) = v(5,m)$ *or* $v(4,p) = v(5,p)$*, that the two sub-pedigrees need to be considered as a whole. Assume that the non-founders 4 and 5 have not been genotyped, then even if* $v(4,m) = v(5,m)$ *it is only in the event that* $(v(8,m) = m \vee v(9,m) = m) \wedge (v(11,m) = m \vee v(12,m) = m \vee v(13,m) = m)$[15]*,*

[15] Minimum one child of individual 4 *and* minimum one child of individual 5 inherit the maternal allele from the maternal side.

that they are dependent. Hence, only in the event that a child in both sub-pedigrees actually receives the allele IBD from individual 4 and individual 5, respectively. The case where $v(4,p) = v(5,p)$ is symmetrical.

For the inheritance patterns in the set $\mathbf{v}_{\mathcal{I}}$ where the two sub-pedigrees are independent in the example pedigree found in Figure 4-7, the single point probability is the product of the probability of the inheritance patterns for the sub-pedigree below individual 4 and the probability of the inheritance patterns for the sub-pedigree below individual 5[16].

From Theorem 8 it follows that the single point probability distributions for the inheritance patterns $\mathbf{v}_{\mathcal{I}} \subset \mathbf{v}$, for which two subsets of non-founders are inheritance independent, can be represented as two probability distributions instead of one. This reduces the space complexity, since $\mathbf{v}_{\mathcal{I}}$ can be represented in $O(2^{b'})$ space, using a flat representation, for the sub-pedigree with the non-founders in N' plus $O(2^{b''})$ space for the non-founders in N'', where b' and b'' are the number of bits needed to represent the two sub-pedigrees. If the subset $\mathbf{v}_{\mathcal{I}}$ of $\mathbf{v}$ was represented in one distribution it would require $O(2^{b'+b''})$ space instead of $O(2^{b'} + 2^{b''})$ space.

This could open for a more compact representation for parts of the state space, where subsets of a pedigree are independent. The principle of decomposition can also be applied recursively to independent subsets. How well this works will of course depend on the pedigree structure. We consider the idea of decomposition based on inheritance dependence a strategy which might lead to an improved algorithm for linkage analysis. However, this requires:

1. A data structure which can represent the state space symbolically with respect to the different inheritance dependencies for different subsets of the state space.

2. A multi point probability computation algorithm to work efficiently with such a structure.

4.5 Summary

In this chapter different approaches for improving algorithms for linkage analysis have been discussed. We have established our level of focus for major improvements as the "representation and algorithms" level. Furthermore, the requirements that future algorithms must uphold are stated, based on the formalization. We have investigated the more theoretical nature of the linkage analysis task and we have found no way of describing the satisfiability-based task of determining the set of compatible inheritance vectors in a single form

[16] This is only true in the event that neither individual 1 nor 2 have been genotyped.

that is P-time solvable. Furthermore, we have presented two reductions that would imply that the task of determining the compatible inheritance vectors is not easier than solving a NP-complete problem, if some reduction can be established to testing membership of two languages in these reductions. Methods for making a more compact representation of the state space were analysed as well as a method for exploiting inheritance independence to decompose the pedigree. We found that MTBDDs are inappropriate because of the high number of equivalence classes in the multi point probability computation. Furthermore, either the PDG structure must be adapted to our problem domain or a set of efficient algorithms need to be developed. We formalized inheritance independence, that might reduce the complexity of both single and multi point linkage analysis, this also depends on whether an efficient structure and a set of efficient algorithms can be developed. This is also subject for further research. Thus, none of the techniques considered in this chapter seem to be directly applicable to linkage analysis, but might serve as an inspiration for a new customised method for linkage analysis.

Chapter 5

Conclusion

As computer scientists we have moved into a new area, namely the field of bioinformatics. More specifically, we have worked with the area of linkage analysis, which is a recognised way to build genetic linkage maps. By locating genes responsible for traits, within the human genome, companies such as deCODE Genetics hope to be able to develop diagnostics products, drugs, or even cures for diseases. Due to the amount of computation required to extract the statistical data, on which evidence of linkage is based, only pedigrees of a moderate size are feasible today for analysis.

The bioinformatics field spans a variety of sciences. Originally, genetic analysis was performed by applying methods of biology and statistics, but as the technology for extracting genetic information has improved, the amount of available data has introduced a need for effective algorithms. However, currently, no formal framework in which to reason about the correctness and complexity of the algorithms of linkage analysis exist.

Today's literature on linkage analysis display quite a diversity in terms of notation and definitions. Since further developments within bioinformatics, and especially linkage analysis, require people with different backgrounds to communicate and cooperate, the lack of a formal framework makes it hard for these people to express themselves precisely. We have build a formal framework in which it is possible to express and reason about linkage analysis precisely. The formal model presented here could hopefully be the formal model which will serve as a common base for the community, as it is true to the widely adopted biological model.

Within our framework we have been able to prove and express the correctness of two algorithms for single point probability computation, namely KRUGLYAK and FASTTREETRAVERSAL - the two algorithms used in the software packages Genehunter and Allegro, respectively. We have also identified that the correct framework in which to describe multi point analysis is Bayesian theory and not the more specialised HMMs.

We have reasoned about the complexity class of linkage analysis and found no single P-time solvable form that can express the task of finding the compatible inheritance vectors. We have made some initial attempts to investigating whether the task of finding the compatible inheritance vectors is NP-complete. However, more research is needed to determine the complexity class of linkage analysis and its sub-tasks.

We have considered two common data structures for symbolic representation to represent probability distributions over inheritance vectors, namely MTBDDs and PDGs. Both data structures do not seem to work well for our problem and would require some modifications in order to reduce the complexity of linkage analysis. MTBDDs seem applicable in single point analysis but fail in multi point analysis, due to diversity of the probabilities in the distribution over inheritance vectors. PDGs seem more promising for representing the intermediate results, but many unanswered questions needs to be resolved.

Now that we have a formal framework, we are able to reason about new algorithms. Until now, we have not been able to find an existing data structure which will reduce the complexity of single and multi point analysis in practice. A natural future work aspect is to try to develop a customised data structure for linkage analysis, and maybe to base this structure on inheritance dependencies. Another future research direction is investigating the deeper theoretical nature of the complexity of linkage analysis. Finally, we mention approximation as possible future research, since approximations to problems, known to be computationally hard, in some cases yield usable results.

Appendix A

Linkage Analysis Tools

This appendix contains descriptions of some the programs available for computation on genetic data and is based on [Ott99] and [oGAS00].

Using computers in linkage analysis is not new. Initially computers were used to store hand calculated LOD score, however as soon as 1955 LOD scores were calculated by vacuum tube based computers. Computers has traditionally been exploited as powerful calculators and as storage media, but due to vast amounts of genetic data, researchers turn to more (computer) scientific approaches like data mining, [Ram01].

In the following the most popular programs for linkage analysis and related genetic computations are presented in alphabetical ordering.

Allegro

Allegro is a tool for multi point linkage analysis, that is able to perform both parametric linkage analysis and analysis based on allele sharing models. The program provides features like estimation of total number of recombinations among markers, it computes posterior IBD sharing probabilities and is able to reconstruct haplotypes. Literature and source code of Allegro was kindly made available to us by deCODE Genetics, and provided insight to the computations needed for linkage analysis.
Allegro is written in C++ by Daniel F. Gudbjartsson, Kristjan Jonasson, Augustine Kong, Michael L. Frigge, deCODE Genetics, Inc. Iceland.

ASPEX

ASPEX is a package of programs for performing multi point exclusion mapping of affected sibling pair data for discrete traits. The tool is able to use allele frequencies to reconstruct missing information, and is tailored for data sets where parents are missing, but additional typed children may be used to

reconstruct and phase the parents. ASPEX estimates multi point marker distance from sib pair data. It can do transmission disequilibrium testing and is able to verifying the degree of relatedness of individuals within a family.
The package is written in ANSI C by Professor Neil Risch, Stanford.
The source code for ASPEX, and pre-compiled binaries are available at:
ftp://lahmed.stanford.edu/pub/aspex

CRI-MAP

CRI-MAP allows rapid, largely automated construction of genetic linkage maps, generates LOD tables, and detects data errors. The tool was originally designed to handle codominant loci scored on pedigrees "without missing individuals", such as nuclear families, but can now be used on general pedigrees, and some disease loci.
CRI-MAP is implemented by Professor Phillip Green, Washington University.
Version 2.8 of CRI-MAP is distributed as source code in the language C and is available at:
http://compgen.rutgers.edu/multimap/crimap/

Genehunter

Genehunter provides a wide range of tools for performing linkage and disequilibrium analyses. The package is able to perform multi point analysis on moderate size pedigree, computing LOD and NPL scores. Genehunter can be used to do sib pair analysis. In addition, tools are available to search for association or disequilibrium in addition to linkage. Genehunter provide calculations involving dozens of markers, even in pedigrees with inbreeding and marriage loops. Additionally, the multi point inheritance information allows the reconstruction of maximum-likelihood haplotypes for all individuals in the pedigree and information content mapping which measures the fraction of the total inheritance information extracted from the marker data.
Genehunter has a C source code and is developed by Leonid Kruglyak, Mark Daly, Mary Pat Reeve-Daly, Eric Lander Whitehead Center for Genome Research, MIT.
Version 2 was released 1998 and is available at:
http://www.fhcrc.org/labs/kruglyak/Downloads/

HOMOZ

HOMOZ is a program for rapid multi point mapping of recessively inherited disease genes in nuclear families including homozygosity mapping. It includes an efficient algorithm for computation of multi point LOD scores in small pedigrees including those with inbreeding loops and missing genotype information.

HOMOZ is developed in C by Leonid Kruglyak, Mark Daly and Eric Lander, Whitehead Center for Genome Research, MIT.
HOMOZ can be downloaded at:
ftp://ftp-genome.wi.mit.edu/distribution/software/homoz/

LINKAGE

The core of the LINKAGE package is a series of programs for maximum likelihood estimation of recombination rates, calculation of LOD score tables, and analysis of genetic risks. The analysis programs are divided into two groups. The first group can be used for general pedigrees with marker and disease loci. Programs in the second group are for three-generation families and codominant marker loci, and are primarily intended for the construction of genetic maps from data on reference families. LINKAGE is capable of making two-point and multi point analysis. LINKAGE is a Pascal program of Dr. Mark Lathrop, Centre National de Genotypage, Evry Cedex, France.
Version 5.5 available at:
ftp://linkage.rockefeller.edu/software/linkage/

LIPED

The LIPED program carries out genetic linkage analysis. It estimates the recombination fraction, by calculating pedigree likelihoods for various assumed values of the recombination fraction. The program is able to handle age of onset data. Only two loci can be handled at a time.
LIPED is written in Fortran IV (1974) by Professor Jurg Ott, Laboratory of Statistical Genetics, Rockefeller University.
Latest version is from June 1995 and is available at:
ftp://linkage.rockefeller.edu/software/liped

Loki

Loki analyses a quantitative trait observed on large pedigrees using Markov chains, Monte Carlo, multi point linkage, and segregation analysis. The trait may be determined by multiple loci.
Loki is implemented in C by Postdoctoral Researcher Dr Simon Heath, University of Washington.
Loki Version 2.3 was released November 2000 and is available at:
ftp://ftp.u.washington.edu/pub/user-supported/pangaea/PANGAEA/Loki

MAP+

The MAP+ program is written to construct high resolution linkage maps. MAP+ requires pairwise sex specific LOD scores, and a trial map containing

trial locations for all the loci to be included in the analysis.
MAP+ is Fortran based and implemented by Dr. A. Collins, J. Teague and Professor N.E. Morton, University of Southampton.
MAP+ was released in 1996 and is available at:
http://cedar.genetics.soton.ac.uk/pub/PROGRAMS/map+

MAPMAKER

The MAPMAKER software package is consisting of two parts MAPMAKER/EXP and MAPMAKER/QTL. MAPMAKER/EXP performs full multi point linkage analysis and estimates recombination fractions for dominant, recessive, and codominant markers. MAPMAKER/QTL is a companion program to MAPMAKER/EXP which allows mapping of genes controlling polygenetic quantitative traits in F2 inter-crosses and BC1 back-crosses relative to a genetic linkage map.
MAPMAKER is programmed in C by Eric Lander, Ph. D., Director of Whitehead Center for Genome Research, MIT.
Version 3.0 of the tool from 1992 is available at:
ftp://ftp-genome.wi.mit.edu/distribution/software/mapmaker3

MENDEL

MENDEL does genetic analysis of human pedigree data under models involving a small number of loci. MENDEL is useful for segregation analysis, linkage calculations, genetic counseling, allele frequency estimation, and related kinds of problems. MENDEL is able to handle looping pedigrees efficiently, which tends to be problematic in other tools.
MENDEL is implemented in Fortran 77 by Professor Kenneth Lange, UCLA.
For registration and download visit:
http://www.biomath.medsch.ucla.edu/faculty/klange/software.html

Pedigree Analysis Package(PAP)

PAP may be used for segregation analysis, variance components analysis, linkage analysis, measured genotype analysis, or genetic model fitting. The genetic model may contain any number of loci and alleles. Phenotypes may be simulated assuming any model available in PAP. Linkage analysis on four loci.
PAP comprises a set of Fortran 77 programs produced by Associate Professor Sandra J. Hasstedt, Department of Human Genetics, University of Utah.
Revision 4.0 from March 1994 is available at:
ftp://ftp.genetics.utah.edu/pub/software/pap

VITESSE

VITESSE is a software package that computes likelihoods. VITESSE uses the algorithms of set-recoding and fuzzy inheritance to reduce the number of genotypes needed for exact computation of the likelihood, which accelerates the calculation. It also represents multi locus genotypes locus-by-locus to reduce the memory requirements.
VITESSE is developed in ANSI C by Assistant Professors Jeffrey O'Connell and Associate Professor Dan E. Weeks at the University of Pittsburgh.
Version 1.0 published in 1995 is available at:
ftp://watson.hgen.pitt.edu/pub/vitesse/

Appendix B

LOD and NPL score

A scoring function is a measure of the likelihood of linkage between a marker and a trait, for which the locus of the marker is known, but for which the locus or loci of the trait causing gene(s) are unknown. Note that when one is looking for the genetic cause of a trait a number of pedigrees are used.

It is not within the scope of this project to go into detail with the scoring functions, as they rely the statistical part of linkage analysis which is not our focus. We will, however, briefly describe two commonly used scoring functions: The *Log Likelihood of Odds* (LOD) score, and the *Non-Parametric Linkage* (NPL) score.

B.0.1 LOD score

The LOD score is a measure of how plausible an observed set of data is given a model, [Ott99, page 38]. It is the logarithm of the odds of the observed data if linkage is assumed (recombination fraction $\theta < \frac{1}{2}$) compared to the *null-hypothesis* (no linkage) where the recombination fraction θ is equal to $\frac{1}{2}$. Formally the LOD score is, [Ott99, page 39]:

$$\mathrm{LOD}(\theta) = \log_{10}\Big(\frac{L(\theta)}{L(\frac{1}{2})}\Big), \tag{B-1}$$

where L is the likelihood of a hypothesis H given some observation data F, defined as: $L(H) = P(F|H)$. The odds in favor one hypothesis H_1 versus H_2 are expressed by the likelihood ratio:

$$R = \frac{L(H_1)}{L(H_2)}.$$

Maximum likelihood estimation is used to find a θ which maximises the LOD-score for all markers (See [Ott99, Chapter 3.3] for more on the statistics of LOD scores and maximum likelihood estimation). The probability of each inheritance vector $v \in \mathbf{v}$ is used in the expression for the score (See [Gud00] for a more detailed presentation of the LOD score computation).

B.0.2 NPL score

The NPL score does not depend on a hypothesis (or a parameter) as the LOD score does. It depends on a *non-parametric scoring function* S such as S_{pair} or S_{all} defined by Kruglyak et. al in [KDRDL96]. Z_i is the standardized from S_i (the score for family i):

$$Z_i = \frac{S_i - \mu_i}{\sigma_i},$$

where μ_i is the mean of S_i, and σ_i is the standard deviation.

A formal definition of NPL is given by Gudbjartsson in [Gud00]:

$$\text{NPL} = \frac{\sum_i \gamma_i \bar{Z}_i}{\sqrt{\sum_i \gamma_i^2}},$$

where $\sum_i$ is the sum over all families, γ_i is a family specific weight and $\bar{Z}_i$ is the expectation of Z_i conditioned on the observed genotype data. Hence:

$$\bar{Z}_i = \sum_{\mathbf{v_i}} P(\mathbf{v_i}|\mathcal{G}_i) Z_i(\mathbf{v_i}),$$

where $\mathbf{v_i}$ is the set of inheritance vectors for family i at a marker and $\mathcal{G}_i$ is the genotype information for the family at the marker.

The reader should note that the every element of the probability distribution $P(\mathbf{v}|\mathcal{G})$ is used in both scoring functions.

Appendix C

Basic Probability Theory

Here we present an introduction to the probability theory which should suffice for understanding the probability theory used in this report. We start with some basic axioms and develop further theory. This theory is based on [SF00] and [Jen01].

The most basic concepts of probability theory is that of random variables and events. A *random variable* has, in our context, a finite set of *states*, which is the outcome of a particular experiment. In most cases we only write $P(a)$ as the probability of a variable A being in state a, instead of $P(A = a)$, when variable A is clear from the context. Furthermore, we abbreviate that variable A is in state a as the event a.

Definition 18 (Basic Probability Axioms) *We denote the probability of a given event a (a state) as $P(a)$, which is a value in the interval $[0, 1]$. Thus $P : a \rightarrow [0, 1]$. The following basic axioms are preserved by these probabilities:*

- *If $P(a) = 1$ for an event a, then event a is certain.*
- *If $P(a) = 0$ for an event a, then event a is impossible.*
- *If events a and b are mutually exclusive then $P(a \vee b) = P(a) + P(b)$.*

Now consider the case where the probability of an event a is conditioned by some other event b. For instance whether or not the roads are slippery (a) is conditioned by whether or not it is freezing (b). We term the *conditioned probability* of event a by b equals x as $P(a|b) = x$. If event a is *independent* of b then $P(a) = P(a|b)$.

Generally if the event a, conditioned by n other events, has the probability x, we write $P(a|e_1, e_2 ... e_n) = x$, where e_i denotes the i'th event under which a is conditioned. Note that $P(a|b, c) = P(a|c, b)$.

The joint probability $P(a, b)$, which states the probability of both a and b occurring, is given by the so-called *fundamental rule* (denoted so by [Jen01]) for probability calculus:

$$P(a,b) = P(a|b) \cdot P(b),$$

although this is known as the fundamental rule, it should be considered an axiom. We will not justify it here, see [Jen01] for a justification.

If two events are independent but not mutually exclusive, the joint probability is given by:

$$P(a,b) = P(a) \cdot P(b),$$

Bayes' theorem relates two conditional events. The basis is that, from the fundamental rule, we know that: $P(a,b) = P(a|b)P(b)$ and $P(a,b) = P(b|a)P(a)$, this leads to the well known *Bayes' rule*:

$$P(b|a) = \frac{P(a|b) \cdot P(b)}{P(a)}.$$

We now present a small example, illustrating the probability concepts introduced in this appendix.

Example 17 *Consider a bowl experiment. We will draw objects that vary in shape (round or cubic) and color (red or blue) from a bowl, the amount of each is depicted in Table C-1.*

	red	blue
round	2	5
cubic	4	7

Table C-1: The shape and color of the 18 different objects in the bowl.

The basic probability of drawing a round object and drawing a red object is given as:

$$P(round) = \tfrac{7}{18}, \text{ and}$$
$$P(red) = \tfrac{6}{18}.$$

The probability of the object turning out to be red, after we already know it is a round object, is given by the conditional probability:

$$P(red|round) = \tfrac{2}{7}.$$

If we want to compute the general probability of drawing a red and round object, we can use the fundamental rule:

$$P(red, round) = P(red|round) \cdot P(round) = \frac{2}{7} \cdot \frac{7}{18} = \frac{1}{9}.$$

As an example of applying Bayes' rule, lets assume that we cannot directly determine the probability of an object turning out to be round, given that the object is red: $P(round|red)$. Then if we know the probabilities of $P(red|round)$, $P(round)$ and $P(red)$ Bayes' rule can be applied:

$$P(round|red) = \frac{P(red|round) \cdot P(round)}{P(red)} = \frac{\frac{2}{7} \cdot \frac{7}{18}}{\frac{6}{18}} = \frac{1}{3}.$$

Whether or not the fundamental rule or Bayes' rule is applied, it is possible to determine the probability of every element in the probability distribution $P(color, shape)$, given in Table C-2.

	$color = red$	$color = blue$
$shape = round$	$\frac{1}{9}$	$\frac{5}{18}$
$shape = cubic$	$\frac{2}{9}$	$\frac{7}{18}$

Table C-2: The probability distribution $P(color, shape)$.

From the $P(color, shape)$ distribution it is possible to calculate the $P(color)$ distribution by marginalizing shape out, by the following expression:

$$P(color) = \sum_{shape} P(color, shape), \tag{C-1}$$

which denotes that the states of shape is marginalized out of $P(color, shape)$. This results in the probability distribution $P(color) = (\frac{1}{3}, \frac{2}{3})$, which is notation for $P(color = red) = \frac{1}{3}$ and $P(color = blue) = \frac{2}{3}$.

We end this section on probability theory, by giving the general method for marginalizing a variable out of a joint probability distribution.

Definition 19 (Marginalization) *We marginalize a variable B, with states $b_1, ..., b_m$, out of a joint distribution $P(A, B)$, where A have the states $a_1, ..., a_n$ by:*

$$P(A) = \sum_{j=1}^{m} P(a_i, b_j), \tag{C-2}$$

where $1 \leq i \leq n$ and $1 \leq j \leq m$.

Appendix D

Markov Models

In this appendix we describe a model for probabilistic transition systems. The systems have the property that all nodes have outgoing edges to all other nodes and the probability of these edges sum to one. The model is called *Discrete Time Markov Chains* (DTMC).

Definition 20 (Discrete Time Markov Chains) *A discrete time Markov chain over some finite alphabet Σ is a 4-tuple $\langle S, p_0, \mathcal{T}, L\rangle$ with:*

- *S: finite set of states,*
- *$p_0 : S \rightarrow [0,1]$: probability distribution over initial states,*
- *$\mathcal{T} : S \times S \rightarrow [0,1]$: probability function assigning probabilities to edges, and*
- *$L : S \rightarrow \Sigma$: function assigning an element of Σ to each state, such that:*
  - *the sum of the probabilities of all outgoing edges from a node $s \in S$ is one, that is:*

$$\forall s \in S \ . \ \sum_{s' \in S} \mathcal{T}(s, s') = 1, \ \textit{and}$$

  - *the sum of all start probabilities is one:*

$$\sum_{s \in S} p_0(s) = 1.$$

It should be noted that sometimes in the literature (e.g. [HJ90]) DTMCs are defined as having *one* distinct start state as opposed to a probability distribution over all states. However, both definitions are equally expressive. Using a single start state would be equivalent to some state having probability 1 in

p_0. Vice versa, using a single start state to describe a probability distribution can be accomplished by introducing a new state as the start state with transitions to all other state having the probabilities described in p_0.

We introduce a series of examples to give the reader a more intuitive understanding of the introduced models. We expand the same example as needed to explain the different models.

Example 18 *Let C be a biased coin with the probability p for ending up heads and $1-p$ for ending up tails. This can be described as the DTMC over $\Sigma = \{h, t\}$ with two states H and T where $L(H) = h$ and $L(T) = t$. The probability for either state being the initial state is $p_0(H) = p$ and $p_0(T) = 1-p$. Figure D-1 is a graphical representation of the DTMC.*

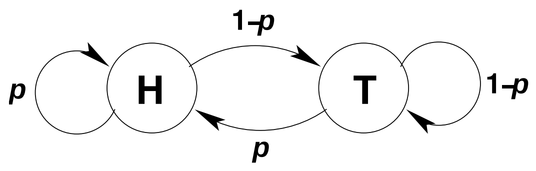

Figure D-1: A biased coin represented as a discrete time Markov chain.

The reason for the name Markov is, that the transition system preserves the *Markov property* which states that *the future is independent of the past given the present*. Formally, this can be stated as:

$$P(S_{i+1}|S_1, \ldots, S_i) = P(S_{i+1}|S_i).$$

Using Example 18 the Markov property states given that we are in state H the probability of flipping heads does not depend on whether we flipped heads of tails previously. Actually, in this simple model the sequence never matters, but all DTMCs preserve the Markov property.

Hidden Markov Models

This section describes *hidden Markov models* (HMMs) which is a generalisation of DTMCs. When in a state, s, in a DTMC, we say that the symbol $L(s) \in \Sigma$ is observable in that state. In the more general hidden Markov model, the observable symbol is a probability distribution over the symbols in the alphabet. If a symbol a is observable in a state s with probability p we say that s emits a with probability p.

Definition 21 (Hidden Markov Model) *A hidden Markov model over some finite alphabet Σ is a 4-tuple $\langle S, p_0, \mathcal{T}, P_L \rangle$ with:*

- *S: finite set of states,*

- $p_0 : S \to [0,1]$*: probability distribution over initial states,*

- $\mathcal{T} : S \times S \to [0,1]$*: probability function assigning probabilities to edges, and*

- $P_L : S \times \Sigma \to [0,1]$*: function stating probability of each state emitting each symbol in* Σ*, such that:*

  – *the sum of probabilities of all outgoing edges from a state* $s \in S$ *is one, that is*
  $$\forall s \in S \;.\; \sum_{s' \in S} \mathcal{T}(s, s') = 1, \text{ and}$$

  – *the sum of all start probabilities is one,*
  $$\sum_{s \in S} p_0(s) = 1, \text{ and}$$

  – *the probabilities of emitting the symbols of* Σ *in a state* $s \in S$ *sum to one, that is:*
  $$\forall s \in S \;.\; \sum_{a \in \Sigma} P_L(s, a) = 1.$$

Example 19 *Continuing the with the biased coin of Example 18 we can describe the situation as a single state HMM. This state* C *emits either* h *or* t *with probability* p *and* $1 - p$*, respectively. This model is illustrated in Figure D-2.*

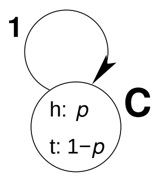

Figure D-2: The coin of Example 18 as a hidden Markov model.

We expand the previous example to two biased coins to introduce the framework for explaining multi point linkage analysis. The first coin $C1$ has probability p and $1 - p$ for emitting heads and tails, respectively. The second coin $C2$ has q and $1 - q$ as probability for the same emissions. The probability for starting in state $C1$ is r, thus $1 - r$ for $C2$. The transition from $C1$ to $C2$ has probability m and n is the probability of the transition from $C2$ to $C1$. The information is gathered in Figure D-3.

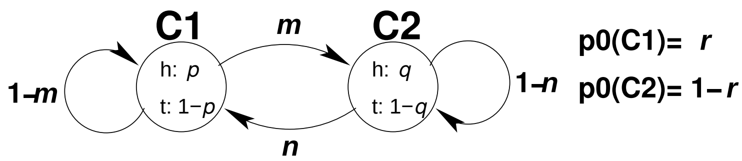

Figure D-3: A hidden Markov model describing two biased coins.

Example 20 *An example of a simple use of the model is given here. We could state the question: What is the chance of flipping heads followed by tails. To come up with an answer we need to examine all possibilities for the first outcome being heads and the second tails and then summing their probabilities. One instance is starting in $C1$ flipping heads and then using $C2$ to flip tails. The probability of this is $r \cdot p \cdot m \cdot (1-q)$. The four paths leading to this and their probabilities are listed below.*

First coin	*Second coin*	*Probability*
$C1$	$C1$	$r \cdot p \cdot (1-m) \cdot (1-p)$
$C1$	$C2$	$r \cdot p \cdot m \cdot (1-q)$
$C2$	$C1$	$(1-r) \cdot q \cdot n \cdot (1-p)$
$C2$	$C2$	$(1-r) \cdot q \cdot (1-n) \cdot (1-q)$

Another use could be to calculate the most probable outcome of two flips which of course depends on the bias. The model can also be used to calculate the probability of eventually flipping heads.

For more information on examining properties of Markov chains we refer to [HJ90].

Bibliography

[BM99] Marius Bozga and Oded Maler. On the Representation of Probabilities over Structured Domains. In *Computer Aided Verification*, pages 261–273, 1999.

[BO98] Andreas D. Baxevanis and B.F. Francis Ouellette. *Bioinformatics - A practical guide to the analysis of genes and proteins.* Wiley-Interscience, 1998.

[Cur01] Dave Curtis. Fundamental concepts in genetics, http://www.mds.qmw.ac.uk/statgen/dcurtis/lectures/introgen.html, October 2001.

[dNC01] deCode News Center. http://www.decode.com/news/releases/, November 2001.

[dPGD01] deCode Products Gene Discovery. http://www.decode.com/products/gd/, November 2001.

[Gud00] Daniel Fanner Gudbjartsson. Multipoint Linkage Analysis based on Allele Sharing Models, 2000.

[HJ90] Hans Hanson and Bengt Jonsson. A logic for reasoning about time and reliability. *SICS Research Report*, 1990.

[JCBW99] Lynn B. Jorde, John C. Carey, Michael J. Bamshad, and Raymond L. White. *Medical Genetics.* Mosby, 1999.

[Jen01] Finn V. Jensen. *Bayesian Networks an Decision Graphs.* Springer Verlag New York inc., 2001.

[JIS93] Robert W. Cottingham Jr., Ramana M. Idury, and Alejandro A. Schaffer. Faster sequential genetic linkage computations. *American journal of human genetics*, 53:252–263, 1993.

[KC97] William S. Klug and Michael R. Cummings. *Concepts of Genetics 5th edition.* Prentice Hall, 1997.

[KDRDL96] Leonid Kruglyak, Mark J. Daly, Mary Pat Reeve-Daly, and Eric S. Lander. Parametric and Nonparametric Linkage Analysis: A Unified Multipoint Approach. *Am. J. Hum. Genet.*, 58:1347–1363, 1996.

[KKNP01] Joost-Peiter Katoen, Marta Kwaitkowska, Gethin Norman, and David Parker. Faster and symbolic CTMC model checking. page 18, 2001.

[KL98] Leonid Kruglyak and Eric S. Lander. Faster multipoint linkage analysis using fourier transforms. *Journal of Computational Biology*, 5(1):7, 1998.

[KNPS99] Marta Z. Kwiatkowska, Gethin Norman, David A. Parker, and Roberto Segala. Symbolic model checking of concurrent probabilistic systems using MTBDDs and simplex. Technical Report CSR-99-1, 1999.

[Lan97] Kenneth Lange. *Mathematical and Statistical Methods for Genetic Analysis*. Springer, 1997.

[LG87] Eric S. Lander and Philip Green. Construction of Multilocus Genetic Linkage Maps in Humans. *Proc. Natl. Acad. Sci.*, 84:2363–2367, 1987.

[MDK01] Kyriacos Markianos, Mark J. Daly, and Leonid Kruglyak. Efficient multipoint linkage analysis through reduction of inheritance space. *American journal of human genetics*, 68:963–977, 2001.

[oGAS00] Alphabetic List of Genetic Analysis Software. http://linkage.rockefeller.edu/soft/, November 2000.

[Ott99] Jurg Ott. *Analysis of Human Genetic Linkage,Third Edition*. The Johns Hopkins University Press, 1999.

[Ram01] Jens Ramskov. Gener fra fem hunderede skandinaviske familier under lup. *Ingeniøren*, 46:10, 2001.

[SF00] Chris Stricker and Rohan Fernando. Analysis of QTL-Effects in Animal Breeding,
http://meishan.ansci.iastate.edu/rohan/notes-dir/qtl.pdf,
November 2000.

[SR99] Tom Strachan and Andrew P. Read. *Human Molecular Genetics 2*. Wiley-Liss, 1999.

List of Figures

List of Tables

Glossary

Allele: Term used for the (maybe unknown) gene that is inherited from the father and the mother respectively. A single allele for each gene locus (allele) is inherited separately from each parent. An allele can be in different allelic states. Page 7.

Allelic state: A specific gene can be in different allelic states. Page 7.

Base pair: A base pair is constructed from two nitrogenous bases of the four A, T, C and G. A always pair with T and C always pair with G. Page 6.

Chromatide: A double helix DNA strand, humans have 92 chromatides in all. Page 5.

Chromosome: Humans have 23 pairs of chromosomes. Each chromosome consists of two chromatides. Page 5.

Codominant: When both alleles at a locus is phenotypically expressed. Page 7.

Codon: A codon is constituted of 3 base pairs and can either specify an amino acid or a stop code in the coding of genes. Page 6.

Crossover: The event where genetic material exchange between two chromosomes that pair during meiosis. Page 9.

DNA Strand: See chromatide. Page 5.

DNA: Abbreviation for Deoxyribonucleic acid. The molecular structure that encodes genetic information. Page 6.

Dominant: An Allele is dominant over another allele if the other allele is recessive (at a given locus). If the allelic state of the allele is not expressed the allele is recessive. Page 7.

Expressed: An allele is expressed if it is expressed in the phenotype. Page 7.

Gene: The sequence of codons that encode a single protein. Page 6.

Genotype: Refers to the allelic states of the two alleles at some locus. These states constitute a persons genotype at that locus. Page 7.

Heterozygous: An individual is heterozygous if the maternal and paternal gene at a specific locus are different. Page 7.

Homozygous: An individual is homozygous if the maternal and paternal gene at a specific locus are the same. Page 7.

IBD: Identical By Descent. Two individuals who have inherited a common allele. Page 18.

IBS: Identical By State. Two individuals with the same allelic state for some marker. The alleles need not be identical by descent. Page 18.

Interference: Occurrence of one crossover highly reduces the likelihood of another in its vicinity. Page 11.

LOD score: Likelihood of Odds (LOD) is a scoring function which determines the likelihood of two loci being linked given a recombination fraction versus the likelihood that they are unlinked. Page 16.

Linked: Denotes that the recombination is less than $\frac{1}{2}$ between two loci. Page 11.

Locus: The position on the chromosome where a certain gene is located. Page 6.

Map function: The function that specifies the relationship between physical distance and recombination fraction. Page 11.

Marker: An identifiable physical location on a chromosome whose inheritance can be observed. Page 6.

Maternal: Refers to the allele received from the mother at a specific locus. Page 7.

Meiosis: A cell division process where haploid cells are divided to form diploid germ cells. In our context some of the cells are later combined with other germ cells to form the genome of a new individual. Page 8.

Mendelian inheritance: In our context, the two laws that influence the pattern of inheritance for genes. Page 11.

Morgan: Denotes a distance where one recombination can be expected on average. Page 11.

Paternal: Refers to the allele received from the father at a specific locus. Page 7.

Phase: Refers to the relationship between two alleles at different loci. If they are in phase they are residing on the same chromosome. Page 10.

Phenotype: The observable characteristic of an individual. Page 7.

Protein: Important molecules in the human body that influence many traits. Page 6.

Recessive: When an allele is only phenotypically expressed in the homozygous state. Page 7.

Recombination fraction: The probability that an odd number of recombinations occur between two loci. Page 10.

Recombination: The process by which offspring receives a combination of linked genes different from that of either parent. This occurs by a crossover between parental chromatides during meiosis. Page 9.

Scoring Function: A measure of resemblance between inheritance patterns of a trait and a marker. Page 16.

Strand: See chromatide. Page 5.

Trait: An observable attribute of an individual. Page 6.

Triplet: See codon. Page 6.

Unlinked: Denotes that the recombination is $\frac{1}{2}$ between two loci. Page 11.

bp: See Base pair. Page 6.

Recent BRICS Report Series Publications

RS-02-7 **Anna Ingólfsdóttir, Anders Lyhne Christensen, Jens Alsted Hansen, Jacob Johnsen, John Knudsen, and Jacob Illum Rasmussen.** ***A Formalization of Linkage Analysis*****. February 2002. vi+109 pp.**

RS-02-6 **Luca Aceto, Zoltán Ésik, and Anna Ingólfsdóttir.** ***Equational Axioms for Probabilistic Bisimilarity (Preliminary Report)*****. February 2002. 22 pp.**

RS-02-5 **Federico Crazzolara and Glynn Winskel.** ***Composing Strand Spaces*****. February 2002. 30 pp.**

RS-02-4 **Olivier Danvy and Lasse R. Nielsen.** ***Syntactic Theories in Practice*****. January 2002. 34 pp. This revised report supersedes the earlier BRICS report RS-01-31.**

RS-02-3 **Olivier Danvy and Lasse R. Nielsen.** ***On One-Pass CPS Transformations*****. January 2002. 18 pp.**

RS-02-2 **Lasse R. Nielsen.** ***A Simple Correctness Proof of the Direct-Style Transformation*****. January 2002.**

RS-02-1 **Claus Brabrand, Anders Møller, and Michael I. Schwartzbach.** ***The `<bigwig>` Project*****. January 2002. 36 pp. This revised report supersedes the earlier BRICS report RS-00-42.**

RS-01-55 **Daniel Damian and Olivier Danvy.** ***A Simple CPS Transformation of Control-Flow Information*****. December 2001.**

RS-01-54 **Daniel Damian and Olivier Danvy.** ***Syntactic Accidents in Program Analysis: On the Impact of the CPS Transformation*****. December 2001. 41 pp. To appear in the** ***Journal of Functional Programming*****. This report supersedes the earlier BRICS report RS-00-15.**

RS-01-53 **Zoltán Ésik and Masami Ito.** ***Temporal Logic with Cyclic Counting and the Degree of Aperiodicity of Finite Automata*****. December 2001. 31 pp.**

RS-01-52 **Jens Groth.** ***Extracting Witnesses from Proofs of Knowledge in the Random Oracle Model*****. December 2001. 23 pp.**

RS-01-51 **Ulrich Kohlenbach.** ***On Weak Markov's Principle*****. December 2001. 10 pp.**