Basic Research in Computer Science

BRICS RS-03-17 Aceto et al.: The Complexity of Checking Consistency of Pedigree Information and Related Problems

The Complexity of Checking Consistency of Pedigree Information and Related Problems

Luca Aceto
Jens Alsted Hansen
Anna Ingólfsdóttir
Jacob Johnsen
John Knudsen

BRICS Report Series **RS-03-17**

ISSN 0909-0878 **March 2003**

Copyright © 2003, Luca Aceto & Jens Alsted Hansen & Anna Ingólfsdóttir & Jacob Johnsen & John Knudsen. 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/03/17/`

The Complexity of Checking Consistency of Pedigree Information and Related Problems

Luca Aceto[1], Jens A. Hansen[1], Anna Ingólfsdóttir[1,2], Jacob Johnsen[1], and John Knudsen[1]

[1] **BRICS** (**B**asic **R**esearch in **C**omputer **S**cience), Centre of the Danish National Research Foundation, Department of Computer Science, Aalborg University, Fr. Bajersvej 7E, 9220 Aalborg Ø, Denmark, luca@cs.auc.dk, alsted@cs.auc.dk, annai@cs.auc.dk, johnsen@cs.auc.dk, johnk@cs.auc.dk

[2] deCODE Genetics, Sturlugata 8, 101 Reykjavík, Iceland, annai@decode.is

Abstract. Consistency checking is a fundamental computational problem in genetics. Given a pedigree and information on the genotypes (of some) of the individuals in it, the aim of consistency checking is to determine whether these data are consistent with the classic Mendelian laws of inheritance. This problem arose originally from the geneticists' need to filter their input data from erroneous information, and is well motivated from both a biological and a sociological viewpoint. This paper shows that consistency checking is NP-complete, even with focus on a single gene and in the presence of three alleles. Several other results on the computational complexity of problems from genetics that are related to consistency checking are also offered. In particular, it is shown that checking the consistency of pedigrees over two alleles, and of pedigrees without loops, can be done in polynomial time.

AMS SUBJECT CLASSIFICATION (1991): 68Q25, 92D10.
CR SUBJECT CLASSIFICATION (1991): F.2.2, J.3.
KEYWORDS AND PHRASES: Consistency checking, pedigrees, genotypes, NP-completeness, satisfiability, polynomial time complexity, critical genotypes.

1 Introduction

A paradigmatic problem from the field of genetics in which the use of algorithmic techniques is by now widespread, and is embodied in software tools like Allegro [9], Genehunter [16], Merlin [1] and Pedcheck [20], is that of linkage analysis. *Linkage analysis* is a well established, statistical method used to relate genes in the human genome to some biological trait that an individual possesses. Example traits that may be investigated range from simple ones like blood type and eye colour to those that may predispose an individual for a disease. Genes causing major diseases (e.g., Parkinson's disease, obesity and anxiety) have already been discovered using this technique [5].

In order to track the inheritance of genetic traits, geneticists use structures called pedigrees. A *pedigree* describes the family relations amongst a collection of individuals, and usually comes equipped with (possibly partial) information on their genotypes—i.e., on the pairs of alleles at a locus in their genome. (An *allele* is one of the possible

forms a gene may have.) Pedigrees are the subject of algorithmic analysis via methods like linkage analysis.

A computational problem that is closely related to that of linkage analysis is *consistency checking*. Given a pedigree and information on the genotypes (of some) of the individuals in it, the aim of consistency checking is to determine whether these data are consistent with the classic Mendelian laws of inheritance (see, e.g., the reference [15] and Sect. 2). If it turns out that the inheritance of the genotypes in the pedigree is in conflict with the Mendelian laws of inheritance, then the pedigree and the information on the genotypes are *inconsistent*. If no such conflict arises, then the data are *consistent*.

The problem of consistency checking arose originally from the geneticists' need to filter their input data from erroneous information, because inconsistent data are undesirable. According to [25, p. 496], it is essential that all Mendelian inconsistencies be eliminated prior to linkage analysis as "a few inopportunely placed errors, if ignored, can tremendously affect evidence for linkage." Furthermore, as reported in [20], in many real-life cases the manual identification of inconsistencies can be very difficult, time consuming, and sometimes unsuccessful. It would therefore be most helpful to have automatic tool support for this task.

Another motivation for consistency checking is its applicability in determining family relationships. A DNA test is very useful in genealogical investigations, paternity issues and criminal investigations. For instance, the point of paternity issues has recently been brought up in the Danish press [6], where it is claimed that up to 10% of the Scandinavian population have wrong paternal information, and that the interest for such investigations is growing rapidly. This could be accelerated by the growing possibilities in society for performing DNA tests. According to [10], it is now possible to get a 10 marker fingerprint of a chromosome for approximately 300 euro by utilizing a DNA sampling kit at your own home. Thus it seems that genealogy studies based on genetic data have the possibility of becoming widely accessible in the near future. Of course, these studies must be based on consistent genealogical data to be meaningful.

A more philosophical motivation for consistency checking lies in the possibility of achieving a deeper understanding of the history of different species. A large scale effort as the Human Genome Diversity Project (see [12]) has an interest in assuring that the family relationships amongst the genetic data used for their analysis are correct. In their case, it is important that the individuals that are picked for DNA sampling are indeed offspring from the original population of a geographical area. Note that it is not only with respect to humans that the verification of family relationships is relevant. For instance, it is important that a horse breeder has the ability to document the family relationships of his/her horses, and that a botanist is certain of the family relationship of plants used in an experiment. Basically all living organisms are based on DNA, and can thus be subjected to consistency checking.

Hence, consistency checking is a well motivated problem from both a biological and a sociological viewpoint. Another issue is whether it is computationally feasible. The aim of this paper is to show that consistency checking is NP-complete *even if we focus on genotype information for a single gene*, and thus that the existence of consistency checking algorithms that have polynomial worst case complexity is unlikely—cf., e.g., the claim by O'Connell and Weeks that their "new genotype-elimination algorithm

is guaranteed to detect every Mendelian inconsistency efficiently and quickly" [21, pp. 1739–1740]. To the best of our knowledge, this is a new result in both computer science and genetics.

After discussing some of the biological background for our work (Sect. 2), we propose a simple formal model for pedigrees and associated genotype information, argue that this model is in agreement with the one used in the genetics literature, and use it to formalize the consistency checking problem with focus on a single gene (Sect. 3). The consistency checking problem is shown to be NP-complete in Sect. 4, even in the presence of *three* alleles. Our proof of NP-hardness for this problem is based on a reduction from 3SAT (a classic NP-complete problem—see, e.g., [23, Propn. 9.2, p. 183]), and uses pedigrees with loops. As stated in [21, p. 1733], likelihood computations on, and consistency checking of, pedigrees with loops continue to pose daunting computational challenges. This is confirmed by the use of looping pedigrees in our NP-completeness proof, and by the fact that pedigrees without loops can be checked for consistency in polynomial time (Thm. 2). (Note, however, that the loops that arise in our constructions are of the kind geneticists call "marriage loops" [24], and not loops arising from inbreeding.) Moreover, since we wish to use our result to infer the hardness of consistency checking in genetically meaningful situations, we offer a discussion of the reasonableness from a genetic viewpoint of our encoding of 3SAT in terms of pedigrees and genotype information. Sect. 6 presents results on the computational complexity of three problems from the genetics literature that are closely related to consistency checking. In particular, we show that checking consistency of pedigrees over *two* alleles is in P (Thm. 4). On the other hand, checking consistency of phase known genotype information, and deciding whether a pedigree has k critical genotypes (with $k \geq 0$) are both NP-complete (Thms. 3 and 5). The final section of the paper (Sect. 7) is devoted to some concluding remarks.

Related Work As previously mentioned, linkage analysis is a statistical method used to relate genes in the human genome to some biological trait that an individual possesses. Like this method, other pedigree analysis techniques involve calculations with probability distributions describing, e.g., the likelihood of gene transmission from one generation to the next. The study [24] investigates the structural complexity of two problems whose solution is part and parcel of many statistical pedigree analysis methods, viz. the calculation of the so-called *marginal probability*, and that of computing the so-called *maximum likelihood*. The decision problems associated with both of these computational tasks are shown NP-hard in *op. cit.* even for pedigrees without inbreeding loops, and with focus on a single gene.

There is a close connection between our NP-completeness result for the consistency checking problem and the NP-hardness results from [24], but neither set of results implies the other. For instance, a pedigree with genotype information is consistent if, and only if, the maximum likelihood for that pedigree is positive. The consistency checking problem can therefore be reduced to an instance of the decision version of the maximum likelihood problem. However, this does not yield our NP-completeness result as a corollary. Moreover, we focus on consistency checking, an apparently very basic problem in genetics, with a purely combinatorial flavour, that does not involve any likelihood computations.

It is also interesting to look at similarities and differences in the proofs of the NP-completeness result for consistency checking we offer here (see Thm. 1), and of Thms. 5 and 9 in [24]. Both sets of results use reductions from 3SAT. The reductions are, however, very different, and, at first sight, somewhat at odds with one another. In our proof of Thm. 1, we focus on a single gene with three alleles. The use of three alleles in this proof of NP-hardness of the consistency checking problem is most likely necessary because, as stated in Thm. 4, checking consistency of pedigrees over *two* alleles is in P. The reduction employed in [24] instead uses only two alleles, and the threshold value on, e.g., the maximum likelihood plays a crucial role in the proofs of the NP-hardness results offered *ibidem*. Indeed, the pedigrees over two alleles generated by the reductions employed there are *always* consistent, as would be detected by the algorithm on which the proof of our Thm. 4 is based.

In an effort to accelerate likelihood calculations, geneticists have proposed genotype elimination algorithms. The aim of these algorithms is to identify, and eliminate, those genotypes that are not consistent with the observed phenotype information in the pedigree. The first algorithm for genotype elimination was proposed by Lange and Goradia in [18], where it was shown that the algorithm is correct for genotype elimination over non-looping pedigrees, but fails to detect all superfluous genotypes for inbred pedigrees. An algorithm for genotype elimination that is correct also in the presence of loops in pedigrees has been offered by O'Connell and Weeks in [21].

Genotype elimination algorithms may be used to detect Mendelian inconsistencies and critical genotypes in pedigrees—see, e.g., the proof of Thm. 2, where we make use of the aforementioned algorithm by Lange and Goradia to argue that pedigrees without loops can be checked for consistency in polynomial time. This makes them suitable as pre-processing steps in algorithms that assume that the input genotype data be consistent. An example of such a use of genotype elimination algorithms is presented in [19], where the authors propose a rule-based, iterative, heuristic algorithm, the *block extension algorithm*, for the so-called *Minimum-Recombinant Haplotype Configuration Problem*. Although this problem is shown to be NP-hard in *op. cit.*, the encouraging preliminary experimental results given in that reference seem to indicate that the block extension algorithm performs rather well in practice under the assumption that its input data are consistent. As we show in this paper, however, checking the consistency of the input data is itself computationally hard. The reference [19] also offers a polynomial time algorithm for haplotype reconstruction *without recombination*; this algorithm assumes input data with no missing genotypes, whose consistency can be checked in linear time in the number of non-founders of the input pedigree. (See the proof of Thm. 1.)

Finally, we remark that bioinformatics seems to be a rich mine of NP-completeness results. In particular, the literature presents several such results in the field of protein folding—see, e.g., those obtained independently by Berger and Leighton in [2], and Crescenzi *et. al.* in [4]. Both these references show that the protein folding problem in the two-dimensional hydrophobic-hydrophilic model is NP-complete by reduction from the Hamiltonian cycle problem, and contain pointers to related studies.

2 Biological Preliminaries

It is well understood that we inherit genetic material from our ancestors. The idea of inheriting traits was discovered by Gregor Mendel (1865). Arguably, Mendel's main contribution to modern genetics was the Mendelian laws of inheritance. Since these principles guide the development of the formal model presented in Sect. 3, we now present Mendel's laws, which specify the concept known as *Mendelian inheritance*, verbatim from [15], and discuss their impact on our work. For further background information on the concepts from genetics on which our work is based, we refer the reader to [11, Appendix A].

> **Unit factors in pairs:** *Genetic characters are controlled by unit factors that exist in pairs in individual organisms.*
>
> The unit factor, as Mendel describes it, is today known as a *gene*. Genes occur in pairs, a paternal and a maternal allele, which reside on each of the chromosomes constituting a chromosome pair. This law implies that the genotype of an individual should always be considered as a pair.
>
> **Dominance/Recessiveness:** *When two unlike unit factors responsible for a single character are present in a single individual, one unit factor is dominant over the other, which is said to be recessive.*
>
> Dominance and recessiveness refer to phenotype, and are not considered further in our biological model. We assume that it is always the individual's genotype, and not its phenotype, that is considered known. That is, it is always the specific alleles we use, and never an abstraction of the trait they code.
>
> **Segregation:** *During the formation of gametes, the paired unit factors separate and segregate randomly so that each gamete receives one or the other with equal likelihood.*
>
> The principle of segregation states that any combination of alleles, which form a genotype, should be considered equally likely to occur. This means that all possible combinations of the paternal and maternal alleles are possible.
>
> **Independent Assortment:** *During gamete formation, segregating pairs of unit factors assort independently of each other.*
>
> Independent assortment states that each gene in a chromosome is inherited independently of all other genes. Since the rest of this paper focuses on a single locus, this principle is irrelevant for our developments. It is worth mentioning, however, that, unlike the previously mentioned principles, independent assortment is no longer believed to be unconditionally true. In fact, the primary aim of methods like linkage analysis is to determine whether there are *fragments* of the genome that are inherited in a pattern that is unlikely to occur purely by chance.

The Mendelian laws of inheritance have been chosen by many researchers in computational genetics as the starting point in their investigations (see, e.g., the references [8,21,22]). Furthermore, a large number of traits are today known to be caused by single gene disorders [13].

Pedigrees In order to track the inheritance of genetic traits, geneticists use structures called *pedigrees*. In our setting, we shall always assume that a pedigree has some (possibly incomplete) genotype information associated with it. A pedigree consists of individuals and their family relations. (See Fig. 1-1 for an example of a pedigree.) Each individual has an associated genotype, which we write just below the individual, that consists of two of the alleles for the gene under consideration. A basic relation amongst individuals is the *nuclear family*, which consists of two parents and their common offspring.

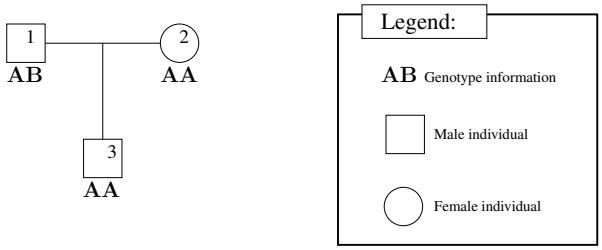

Fig. 1-1: Example of a pedigree.

The genotype of each individual in a pedigree is either known through a genotyping process, or it is a set of genotypes which can be inferred from the Mendelian laws of inheritance. As the principle of segregation states, an individual inherits one allele from each parent. *Genotype phase* refers to the heredity of each allele of the genotype for an individual, that is, whether a given allele is inherited from the paternal or maternal side. In general, by observing a chromosome pair, it is not possible to say which component is inherited paternally or maternally. We have chosen to treat the genotype of each individual as phase unknown, irrespective of the knowledge of that of its ancestors, unless otherwise stated. When describing genotypes, we only write one of the equivalent genotypes (e.g., $\mathbf{AB}$ is equivalent to $\mathbf{BA}$).

Consistency Checking A pedigree with associated genotype information is *consistent* when all observed or inferred genotypes are possible according to the Mendelian laws of inheritance [22].

There can be several reasons for inconsistencies in a pedigree and its genotype information. For instance, a family relationship could be misspecified, or there could be errors in the genotyping process or mutation. Generally it is not possible to determine the source of error; it is simply established that the given genotype information is inconsistent with the pedigree under investigation. By way of example, consider the pedigree shown in Fig. 1-2. (In this pedigree, and in what follows, whenever an individual has no genotype information attached to it, then no genotype information is available for him/her.) This pedigree is inconsistent in two places. One of the inconsistencies is due to the fact that it is impossible that individual 5 could have inherited the $\mathbf{C}$ allele from either of her parents. To observe the other inconsistency, it is necessary to reason about

more than two generations in the pedigree. The inconsistency appears because individual 7 cannot have inherited his **C** allele from individual 3, because individual 3 inherits her alleles from parents that do not have a **C** allele.

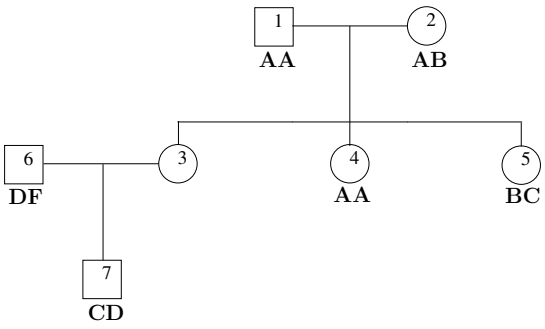

Fig. 1-2: An inconsistent pedigree.

The example we have just presented does not capture the complexities that arise when dealing with large pedigrees. Each test is simple, but considering that multiple individuals can have different possible genotypes, and that the possible genotypes of some individuals must be inferred by analyzing several generations, it should be clear that it can be a daunting task to analyze pedigrees by hand. As pointed out in Sect. 1, geneticists maintain that looping pedigrees present some special problems. A *loop* in a pedigree is a sequence of arcs that starts and ends in the same individual [21]. (See Def. 2 for a formal definition.) An example of a consistent, looping pedigree is depicted in Fig. 1-3. We invite the reader to find suitable genotypes for the ungenotyped individuals in it.

3 Formalizing Gene Inheritance and Mendelian Consistency

As already mentioned in Sect. 2, a pedigree is a fundamental structure used in genetics. In order to reason about pedigrees and the genotype information that they contain, we need a formal model for them. Several formalizations of the notion of pedigree have been presented in the literature on computational genetics. (See, e.g., [19,24].) We now proceed to present the models for pedigrees and their associated genotype information adopted in this study, and then use these models to formalize the consistency checking problem.

Definition 1 (Pedigree). *A* pedigree *consists of a 4-tuple* $P = \langle V, F, \mathbf{p}, \mathbf{m} \rangle$ *where:*

- V *is a finite, non-empty set of* members *of the pedigree (ranged over by* u, v*),*
- $F \subseteq V$ *is the set of* founders*,*

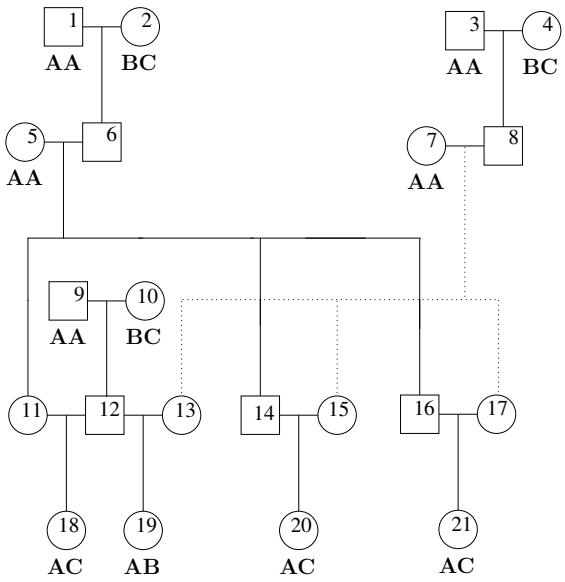

Fig. 1-3: A consistent looping pedigree. The dotted line also represents a parents-child relationship.

– $\mathbf{p}$,$\mathbf{m} : V \setminus F \longrightarrow V$ *are the* paternal *and* maternal functions*, respectively, where*

$$\mathbf{p}(V \setminus F) \cap \mathbf{m}(V \setminus F) = \emptyset$$

(that is, nobody can be both a mother and a father), and
– *the transitive closure of the binary relation obtained as the union of the graphs of* $\mathbf{p}$ *and* $\mathbf{m}$ *is irreflexive (that is, a member of the pedigree is never its own ancestor).*

The set $N = V \setminus F$ *is usually referred to as the set of* non-founders *of the pedigree.*

Remark 1. Note that the set of founders in a pedigree is always non-empty.

Note that, since the model specifies the sex of an individual only implicitly via the paternal and maternal functions, the sex of a "leaf" in a pedigree (i.e., of an individual without offspring) is not specified. In our examples and constructions, the sex of individuals in a pedigree without offspring will be chosen arbitrarily, as it is immaterial in consistency checking. Our pictorial representation of pedigrees (with associated genotype information) is borrowed from the genetics literature, and has already been introduced in Fig. 1-1. That figure represents a pedigree, consisting of a single nuclear family, whose founders are individuals 1 and 2, who are respectively the father and the mother of individual 3.

For the sake of precision, we now offer a formal definition of loop in a pedigree. The following definition is based on that in [21].

Definition 2 (Looping Pedigree). *Let $P = \langle V, F, \mathbf{p}, \mathbf{m} \rangle$ be a pedigree. Two distinct members u and v of the pedigree are said to* mate *if they have an offspring in common—that is, if there is a non-founder v' of P such that $\{\mathbf{p}(v'), \mathbf{m}(v')\} = \{u, v\}$. Such a v' is a* child *of u and v.*

The mating graph *associated with P is the undirected graph G_P whose set of nodes includes V, and contains* mating nodes *$M_{u,v}$ for every pair (u, v) of members of P that mate. The edges in such a graph are those that connect members u and v that mate to the mating node $M_{u,v}$, and those that connect such a mating node to the common children of u and v.*

A loop *in G_P is a non-empty path consisting of* distinct *edges that starts and ends in the same node.*

Finally, we say that a pedigree P is looping *(or* has a loop*) if its associated mating graph G_P contains a loop.*

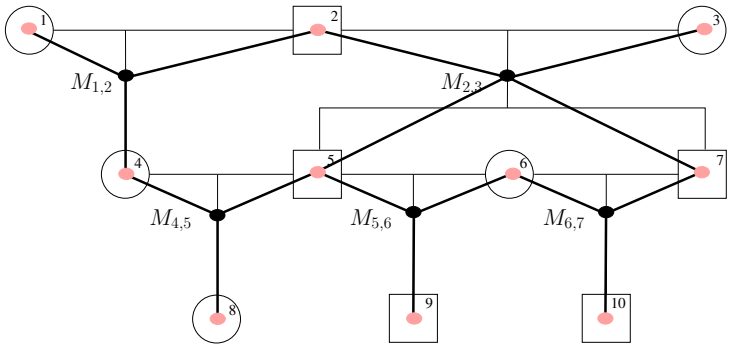

Fig. 1-4: A pedigree illustrated as done throughout this paper, and its associated mating graph as defined in Def. 2. The black dots are the mating nodes and the grey dots are "person nodes".

An example of a looping pedigree is given in Fig. 1-4, together with its associated mating graph. One of the loops in that pedigree is due to inbreeding, and arises because individuals 4 and 5 mate, and have a common ancestor. Another is a so-called marriage loop, and stems from the matings between individual 6 and the two brothers 5 and 7.

Consistency checking of a pedigree is based on its associated genotype information; intuitively, the pedigree defines the structure of the family relationships that are being modelled, and the genotype information is the data which must be consistent with the structure. We now present a formal genotype model. In what follows, it is always assumed that instances of this genotype model are in the context of a specific gene and pedigree. We also assume a fixed, finite and non-empty set $\mathcal{A}$ of *alleles* ranged over by $\mathbf{A}$, $\mathbf{B}$, etc.

In what follows, $\mathrm{Two}(\mathcal{A})$ denotes the family of non-empty subsets of $\mathcal{A}$ that contain no more than two alleles. As described below, an element of $\mathrm{Two}(\mathcal{A})$ will be used to represent a genotype over the set of alleles $\mathcal{A}$.

Definition 3 (Genotype Information). *Let $P = \langle V, F, \mathbf{p}, \mathbf{m}\rangle$ be a pedigree. A* genotype information *for P is a partial function $\mathcal{G} : V \hookrightarrow \mathrm{Two}(\mathcal{A})$ that associates a genotype to (some of) the members of the pedigree. The domain, $\mathrm{dom}(\mathcal{G})$, of the function is referred to as the* set of genotyped members *of the pedigree. The genotype information $\mathcal{G}$ is* complete *if $\mathrm{dom}(\mathcal{G}) = V$.*

Let $\mathcal{G}$ and $\mathcal{G}'$ be two genotype information. We say that $\mathcal{G}'$ extends *$\mathcal{G}$ if $\mathrm{dom}(\mathcal{G})$ is included in $\mathrm{dom}(\mathcal{G}')$, and $\mathcal{G}$ and $\mathcal{G}'$ coincide over $\mathrm{dom}(\mathcal{G})$.*

Remark 2. In the above definition, a genotype information may be seen as assigning an *unordered* pair of alleles to members of the pedigree. This indicates that the phase of the alleles is unknown. If a pedigree member is *homozygous* at a given locus in its genome, i.e., the two alleles at that locus coincide, the function $\mathcal{G}$ returns a singleton set.

In the literature on genetics, and in our pictorial representation of pedigrees, the genotype $\{\mathbf{A},\mathbf{B}\}$ is given as the string $\mathbf{AB}$ (or $\mathbf{BA}$). In particular, the genotype $\{\mathbf{A}\}$ is given as $\mathbf{AA}$. In the remainder of this paper, we shall use these notations interchangeably without further explanations.

Considering consistency for a specific gene amounts to checking whether the pedigree and the genotype information are consistent according to the Mendelian law of segregation (see page 5). The law of segregation implicitly defines the following constraint on consistent genotype assignments:

Each individual must inherit precisely one allele from each of its parents.

Our order of business will now be to formalize this constraint, and what it means that a genotype information is consistent with respect to a pedigree.

Definition 4 (Consistent Genotype Information). *Let $P = \langle V, F, \mathbf{p}, \mathbf{m}\rangle$ be a pedigree.*

1. *A complete genotype information $\mathcal{G}$ for P is* consistent *with P if, whenever $v \in N$:*
   (a) if $\mathcal{G}(v) = \{\mathbf{A},\mathbf{B}\}$, then either $\mathbf{A} \in \mathcal{G}(\mathbf{p}(v))$ and $\mathbf{B} \in \mathcal{G}(\mathbf{m}(v))$, or $\mathbf{B} \in \mathcal{G}(\mathbf{p}(v))$ and $\mathbf{A} \in \mathcal{G}(\mathbf{m}(v))$;
   (b) if $\mathcal{G}(v) = \{\mathbf{A}\}$, then $\mathbf{A}$ is contained in both $\mathcal{G}(\mathbf{p}(v))$ and $\mathcal{G}(\mathbf{m}(v))$.
2. *A genotype information for P is* consistent *with P if it can be extended to a complete, consistent genotype information for P.*

A genotype $\mathcal{G}(v)$ for a non-founder in a pedigree that satisfies the conditions of Def. 4(1) is often referred to as a *possible zygote* for the genotype pair $\{\mathcal{G}(\mathbf{p}(v)), \mathcal{G}(\mathbf{m}(v))\}$—see, e.g., [18, p. 252].

4 Consistency Checking is NP-complete

In what follows, CONS will denote the consistency checking problem for genes with an arbitrary number of alleles. We shall use nCONS to refer to the consistency checking problem for a gene with n possible alleles, for some positive integer n. Our aim in the remainder of this section will be to show the following result:

Theorem 1. *The problems nCONS ($n \geq 3$) and CONS are NP-complete.*

Remark 3. The proviso in the statement of the above theorem that the number of alleles n be larger than, or equal to, three is most likely necessary. In fact, in the presence of a single allele, there is only one complete genotype information, viz. that which assigns the only allele to each member of the pedigree, and that is consistent. Hence, in that case, each genotype information is consistent with respect to every pedigree. Moreover, as will be shown in Thm. 4, the problem 2CONS is decidable in polynomial time.

To prove Thm. 1, we shall first show that CONS, and thus nCONS for every n, is in NP. We then show that 3CONS, and therefore CONS and nCONS for every $n \geq 3$, is NP-hard.

It is not too hard to see that CONS is in NP. To this end, given any pedigree P with genotype information $\mathcal{G}$, it is sufficient to exhibit a certificate that is verifiable in polynomial time. The certificate for an instance of problem CONS is a complete and consistent genotype information $\mathcal{G}^c$ that extends $\mathcal{G}$ in the sense of Def. 3. To check the consistency of $\mathcal{G}^c$ we only have to make sure that the conditions in Def. 4(1) are satisfied for each non-founder of the pedigree. This only takes constant time for each non-founder, and thus the whole consistency check takes linear time in the number of non-founders of the pedigree. Note that the complexity of this consistency check is independent of the number of possible alleles, which shows that nCONS is in NP for every n.

Our order of business will now be to show that 3CONS, and thus CONS, is NP-hard. Note that this is a strong indication that the structural complexity of consistency checking does *not* depend on the number of alleles for a gene, if that number is at least three. We shall stress the importance of this constant number of alleles from a genetic viewpoint later in this section. Our NP-hardness proof for 3CONS is by reduction from 3SAT. The central idea of the proof is to build a pedigree with associated genotype information from a 3SAT instance in such a way that the structure of the pedigree together with the genotype information mimic the variables and clauses of the input 3SAT instance as closely as possible. The constructed pedigree with genotype information is consistent if, and only if, the 3SAT instance it models is satisfiable.

We recall, for the sake of clarity, that 3SAT is the special case of the satisfiability problem for boolean formulae in which the input formulae are in *conjunctive normal form*, and all of their clauses (i.e., disjunctions of literals) have exactly three literals—where a literal is either a variable or a negated variable. Our aim, in the remainder of this section, is to offer a polynomial time reduction from 3SAT to 3CONS. In fact, it is not too hard to see that, without loss of generality, we can restrict ourselves to considering boolean formulae in conjunctive normal form whose clauses have the form $x \vee y$, $\overline{x} \vee \overline{y}$, $x \vee y \vee z$, or $\overline{x} \vee \overline{y} \vee \overline{z}$, for some distinct variables x, y, z. Indeed, any 3SAT instance can be brought into that form in the following four steps:

1. Remove all clauses containing complementary literals (as they evaluate to true). If all clauses are removed in this step, then the original formula is satisfiable.
2. Replace multiple occurrences of the same literal within a single clause with a single occurrence of the same literal (as $l \vee l = l$, for every literal l).
3. If a clause consists of a single literal, then

(a) remove all clauses that contain this literal (as it must be assigned the value true) and
(b) remove all occurrences of its negation in other clauses (as they have to be assigned the value false).

If all clauses are removed in step 3a above, then the original formula is satisfiable. If some clause reduces to the empty clause in step 3b, then we know that there is no assignment that can satisfy the clause, and the formula is not satisfiable.

4. Finally, we put every clause in the formula into one of the forms $x \vee y$, $\overline{x} \vee \overline{y}$, $x \vee y \vee z$, or $\overline{x} \vee \overline{y} \vee \overline{z}$, for some distinct variables x, y, z. This can be done by introducing dummy variables. For instance, a clause of the form $\overline{x} \vee y \vee z$ is replaced with $(\overline{x} \vee \overline{p}) \wedge (y \vee z \vee p)$, for some fresh variable p. (We use a different variable p for each clause.) The complete set of reduction rules used in this step may be found in Table 1-1.

	2 literals	3 literals
0 negations	$x \vee y$ (no reduction)	$x \vee y \vee z$ (no reduction)
1 negation	$\overline{x} \vee y \rightarrow (\overline{x} \vee \overline{p}) \wedge (y \vee p)$	$\overline{x} \vee y \vee z \rightarrow (\overline{x} \vee \overline{p}) \wedge (y \vee z \vee p)$
2 negations	$\overline{x} \vee \overline{y}$ (no reduction)	$\overline{x} \vee \overline{y} \vee z \rightarrow (\overline{x} \vee \overline{y} \vee \overline{p}) \wedge (z \vee p)$
3 negations		$\overline{x} \vee \overline{y} \vee \overline{z}$ (no reduction)

Table 1-1: The rules for step 4 in the transformation of 3SAT instances. We pick a fresh variable p for each clause to be reduced.

It is clear that any instance of 3SAT can be rewritten to the form described above in polynomial time, and that the resulting formula is satisfiable if, and only if, so was the original one.

We are now ready to present our reduction from 3SAT to 3CONS. Let ϕ be an instance of 3SAT. In light of the above discussion, we may assume that ϕ is in conjunctive normal form, and that its clauses have one of the forms $x \vee y$, $\overline{x} \vee \overline{y}$, $x \vee y \vee z$, or $\overline{x} \vee \overline{y} \vee \overline{z}$, for some distinct variables x, y, z. Furthermore, we assume a fixed total ordering on the variables, and that the variables always appear in clauses in an order that is compatible with it. The construction of a pedigree P_ϕ with associated genotype information $\mathcal{G}_\phi$ from a formula ϕ proceeds in the following three steps:

1. Make variable gadgets for each of the variables in ϕ.
2. Make clause gadgets for each of the clauses in ϕ.
3. Combine the variable gadgets with the clause gadgets, and output the resulting pedigree.

In the construction outlined below, the genotype information $\mathcal{G}_\phi$ will be explicitly described in stepwise fashion as we show how P_ϕ is built.

We start by describing the construction of the variable gadgets. In our construction, we shall make use of three alleles, denoted by $\mathbf{A}$, $\mathbf{F}$ and $\mathbf{T}$. The alleles $\mathbf{T}$ and $\mathbf{F}$ are intended to play the role of "true" (denoted by T) and "false" (denoted by F) in the

3SAT problem. The third allele $\mathbf{A}$ is an auxiliary dummy allele used for controlling possible inheritance patterns.

For each variable x that occurs in ϕ we construct the pedigree P_x thus:

$$P_x = \langle V_x, F_x, \mathbf{p}_x, \mathbf{m}_x \rangle \ ,$$

where

$$\begin{aligned} V_x &= \{f_x, m_x, v_x, s_x\} \\ F_x &= \{f_x, m_x, s_x\} \\ \mathbf{p}_x(v_x) &= f_x \quad \text{and} \\ \mathbf{m}_x(v_x) &= m_x \ . \end{aligned}$$

The genotype information $\mathcal{G}_\phi$ assigns genotype $\mathbf{AA}$ to both m_x (the *mother* of v_x) and s_x (the *spouse* of v_x), and genotype $\mathbf{TF}$ to f_x (the *father* of v_x). The genotyped pedigree P_x is depicted in Fig. 1-5.

The pedigree P_x consists of three genotyped members, and one ungenotyped individual v_x. The genotype of v_x can, however, be partly inferred by the Mendelian laws, and has the form $x\mathbf{A}$, where the "allelic variable" x takes either the value $\mathbf{F}$ or $\mathbf{T}$. This is indicated by $x\mathbf{A}$ on the figure. Moreover, the allele x associated with the individual v_x is the only possible origin of a $\mathbf{T}$ or $\mathbf{F}$ allele that can be inherited further from the inheritance point of P_x. We shall refer to individual v_x in Fig. 1-5, as the *variable individual for* x. The illustration on the left of P_x in Fig. 1-5 shows how the variable gadgets are depicted in larger pedigrees.

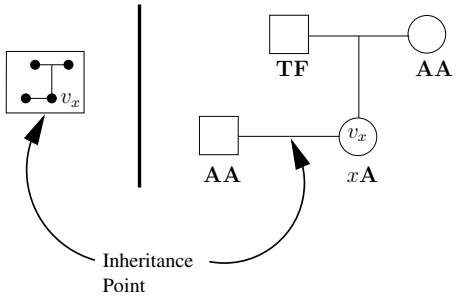

Fig. 1-5: The variable gadget P_x that is used in the proof showing that 3CONS is NP-complete.

The next step in the reduction is to construct a clause gadget P_γ, for each clause γ in the formula ϕ. As we have already pointed out, there are only four different types of clauses we need to consider, and each leads to a different type of clause gadget. The clause gadgets for clauses γ of the form $x \vee y$ and $\overline{x} \vee \overline{y}$ (respectively, $x \vee y \vee z$ and $\overline{x} \vee \overline{y} \vee \overline{z}$) have the same pedigree structure P_γ, but the genotype information $\mathcal{G}_\phi$

assigns a different genotype to the one individual in P_γ without offspring. In each pedigree P_γ, we shall use c_γ to denote this single "leaf", and f_γ and m_γ to stand for its father and mother, respectively. If the clause γ contains three literals, the pedigree P_γ also contains individuals gf_γ and gm_γ, who are, respectively, the maternal grandfather and grandmother of c_γ. The paternal and maternal functions $\mathbf{p}_\gamma$ and $\mathbf{m}_\gamma$ encode the family structure that we have just described—that is:

$$\mathbf{p}_\gamma(u) = \begin{cases} f_\gamma & \text{if } u = c_\gamma \\ gf_\gamma & \text{if } u = m_\gamma \text{ and } \gamma \text{ contains three literals} \end{cases}$$

$$\mathbf{m}_\gamma(u) = \begin{cases} m_\gamma & \text{if } u = c_\gamma \\ gm_\gamma & \text{if } u = m_\gamma \text{ and } \gamma \text{ contains three literals.} \end{cases}$$

In what follows, we shall write V_γ for the set of individuals of the pedigree P_γ.

The only new genotyped individual in P_γ is its leaf c_γ. The genotype $\mathcal{G}_\phi(c_\gamma)$ is **TA** if γ contains only positive literals, and **FA** otherwise.

The four different types of clause gadgets are depicted in Fig. 1-6, where we also show how the clause gadgets will be linked to the variable gadgets in the construction of the pedigree P_ϕ. The genotype information associated with the leaves of these pedigrees is used to code constraints on the values of the variables in a satisfying assignment for the original clauses. For instance, the leaves of the pedigrees associated with the clauses containing only positive literals have genotype **TA** to represent the fact that one of the variables in that clause must be assigned the truth value true in every satisfying assignment.

Having constructed a variable gadget for each variable and a clause gadget for each clause occurring in ϕ, we combine these gadgets, and output the resulting pedigree P_ϕ. The pedigree $P_\phi = \langle V_\phi, F_\phi, \mathbf{p}_\phi, \mathbf{m}_\phi \rangle$ is built thus:

- the set V_ϕ of members of P_ϕ is the union of the V_x's (with x a variable occurring in ϕ) and of the V_γ's (with γ a clause of ϕ);
- the set F_ϕ of founders of P_ϕ is the union of the F_x's (with x a variable occurring in ϕ);
- the functions $\mathbf{p}_\phi : V_\phi \setminus F_\phi \to V_\phi$ and $\mathbf{m}_\phi : V_\phi \setminus F_\phi \to V_\phi$ are obtained by extending the paternal and maternal functions for the pedigrees P_x and P_γ thus:

$$\mathbf{p}_\phi(u) = \begin{cases} s_x & \text{if } u = f_\gamma, \text{ and the first variable of } \gamma \text{ is } x \\ s_y & \text{if } u = m_\gamma, \text{ and } \gamma = x \vee y \text{ for some variable } x \\ s_y & \text{if } u = gf_\gamma, \text{ and } \gamma = x \vee y \vee z \text{ for some variables } x, z \\ s_z & \text{if } u = gm_\gamma, \text{ and } \gamma = x \vee y \vee z \text{ for some variables } x, y \end{cases}$$

$$\mathbf{m}_\phi(u) = \begin{cases} v_x & \text{if } u = f_\gamma, \text{ and the first variable of } \gamma \text{ is } x \\ v_y & \text{if } u = m_\gamma, \text{ and } \gamma = x \vee y \text{ for some variable } x \\ v_y & \text{if } u = gf_\gamma, \text{ and } \gamma = x \vee y \vee z \text{ for some variables } x, z \\ v_z & \text{if } u = gm_\gamma, \text{ and } \gamma = x \vee y \vee z \text{ for some variables } x, y. \end{cases}$$

The pedigree P_ϕ is depicted in Fig. 1-7.

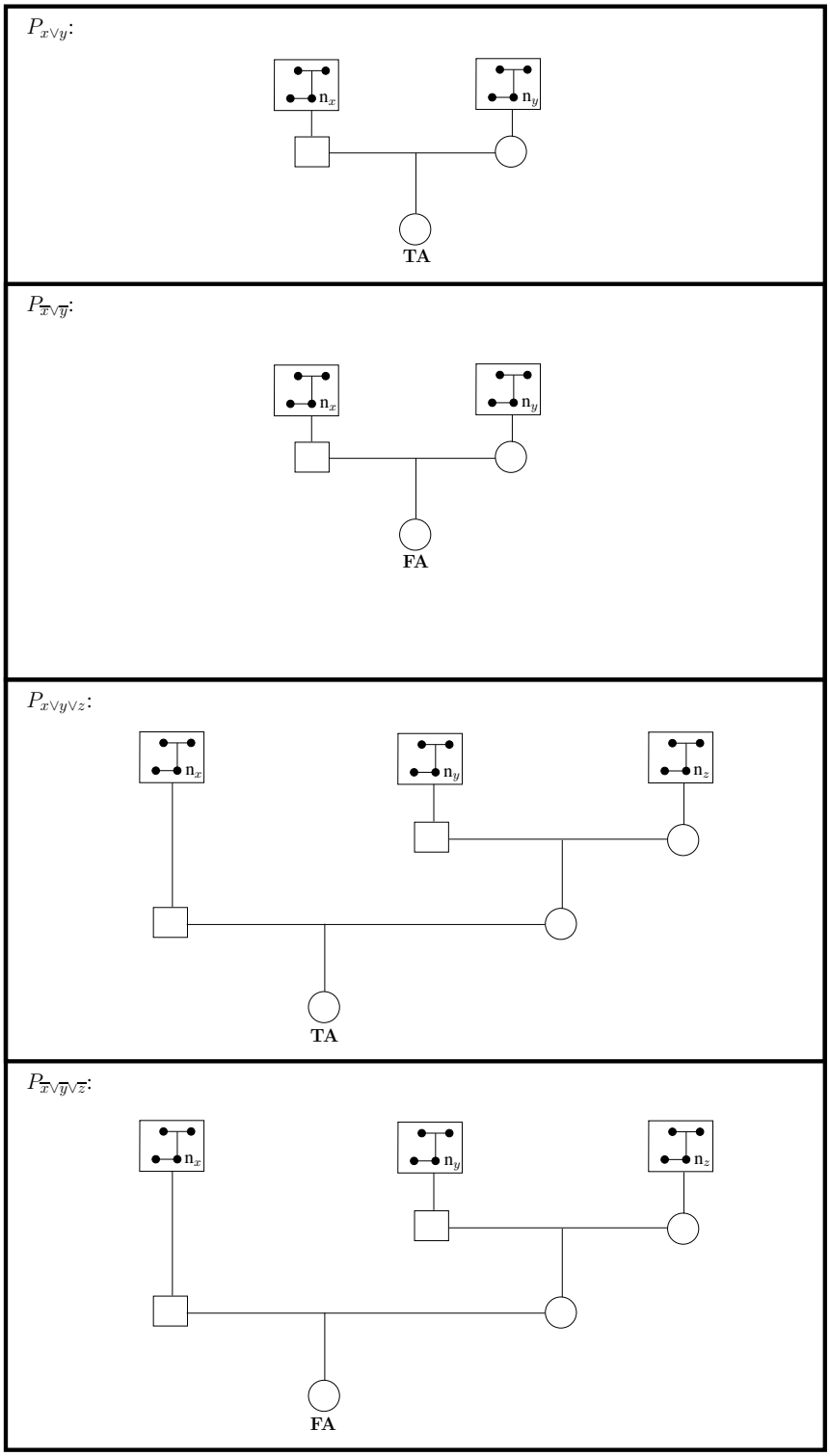

Fig. 1-6: The pedigrees constructed for the four basic clause types along with their connections with the appropriate variable gadgets. Notice the symmetry between $P_{x\vee y}$ and $P_{\overline{x}\vee\overline{y}}$, and $P_{x\vee y\vee z}$ and $P_{\overline{x}\vee\overline{y}\vee\overline{z}}$, respectively.

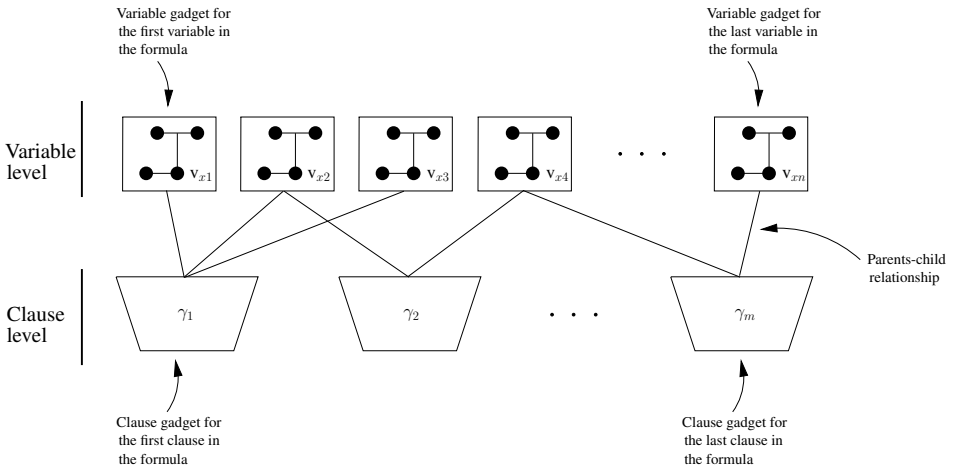

Fig. 1-7: The general form of the pedigree constructed in the reduction from 3SAT. Notice that there exists a one to one correspondence between the number of variables and clauses in ϕ, and the number of variable gadgets and clause gadgets, respectively, in the constructed pedigree.

Example 1. Let ϕ be the formula

$$(x \vee u) \wedge (\overline{y} \vee \overline{u}) \wedge (x \vee y) \wedge (\overline{x} \vee \overline{y}) \ . \tag{1-1}$$

The pedigree produced from this formula by the construction described above is depicted in Fig. 1-8.

The following result states the correctness of our construction of P_ϕ from a 3SAT formula ϕ.

Proposition 1. *A 3SAT formula ϕ is satisfiable if, and only if, the genotype information $\mathcal{G}_\phi$ is consistent with P_ϕ.*

Proof. Throughout the proof, for a boolean formula ϕ and an assignment ρ of truth values to its variables, we use $\boldsymbol{\rho(\phi)}$ to stand for the allele corresponding to the truth value $\rho(\phi)$—that is,

$$\boldsymbol{\rho(\phi)} = \begin{cases} \mathbf{T} & \text{if } \rho(\phi) = T \\ \mathbf{F} & \text{if } \rho(\phi) = F. \end{cases}$$

We are now ready to prove the two implications separately.

– 'ONLY IF' IMPLICATION. Let ρ be an assignment of truth values to the variables occurring in ϕ. We define the canonical extension $\mathcal{G}_\phi^\rho$ associated with ρ of the geno-

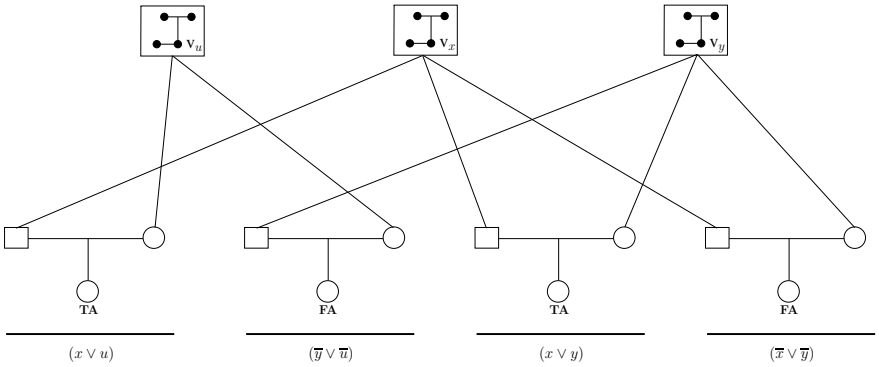

Fig. 1-8: The pedigree for the formula (1-1). Note that formula (1-1) is satisfiable, and that the above pedigree is consistent.

type information $\mathcal{G}_\phi$ thus:

$$\mathcal{G}^\rho_\phi(u) = \begin{cases} \mathcal{G}_\phi(u) & \text{if } u \in \mathrm{dom}(\mathcal{G}_\phi) \\ \boldsymbol{\rho(x)}\mathbf{A} & \text{if } u = v_x \text{ for some variable } x \\ \boldsymbol{\rho(x)}\mathbf{A} & \text{if } u = f_\gamma \text{ for some clause } \gamma \text{ whose first literal is } x \text{ or } \overline{x} \\ \boldsymbol{\rho(y)}\mathbf{A} & \text{if } u = m_\gamma, \text{ where } \gamma = x \vee y \text{ or } \gamma = \overline{x} \vee \overline{y} \text{ for some } x \\ \boldsymbol{\rho(y)}\mathbf{A} & \text{if } u = gf_\gamma, \text{ where } \gamma = x \vee y \vee z \text{ or } \gamma = \overline{x} \vee \overline{y} \vee \overline{z} \\ & \text{for some variables } x, z \\ \boldsymbol{\rho(z)}\mathbf{A} & \text{if } u = gm_\gamma, \text{ where } \gamma = x \vee y \vee z \text{ or } \gamma = \overline{x} \vee \overline{y} \vee \overline{z} \\ & \text{for some variables } x, y \\ \boldsymbol{\rho(y \vee z)}\mathbf{A} & \text{if } u = m_\gamma \text{ where } \gamma = x \vee y \vee z \text{ for some variable } x \\ \boldsymbol{\rho(y \wedge z)}\mathbf{A} & \text{if } u = m_\gamma \text{ where } \gamma = \overline{x} \vee \overline{y} \vee \overline{z} \text{ for some variable } x. \end{cases}$$

Note that the genotype information defined above is complete for the pedigree P_ϕ, and extends $\mathcal{G}_\phi$.

We now proceed to prove that if $\rho(\phi) = T$, then $\mathcal{G}^\rho_\phi$ is consistent with P_ϕ. Assume, to this end, that $\rho(\phi) = T$. Then $\rho(\gamma) = T$ for each clause γ of ϕ. In particular, for each clause of ϕ containing only positive (respectively, negative) literals there is a variable that occurs in it that is set to T (respectively, F) by ρ. To show that $\mathcal{G}^\rho_\phi$ is consistent with P_ϕ we consider each non-founder u in P_ϕ, and argue that $\mathcal{G}^\rho_\phi(u)$ is a possible zygote of the genotypes assigned to its parents by $\mathcal{G}^\rho_\phi$. Below, we only present the details for three selected cases. The remaining ones are similar, and we leave the details to the reader.

- CASE $u = v_x$, FOR SOME VARIABLE x OCCURRING IN ϕ. In this case $\mathcal{G}^\rho_\phi(u) = \boldsymbol{\rho(x)}\mathbf{A}$ is one of the possible zygotes of $\mathbf{TF}$ and $\mathbf{AA}$, that are the genotypes of u's parents, no matter what $\rho(x)$ is.
- CASE $u = m_\gamma$, FOR SOME CLAUSE γ OF ϕ THAT CONTAINS THREE LITERALS. Assume that the variables y and z occur in the second and the third

literal in γ, respectively. By the definition of $\mathcal{G}^{\rho}_{\phi}$, the parents of u have genotype $\boldsymbol{\rho(y)}\mathbf{A}$ and $\boldsymbol{\rho(z)}\mathbf{A}$. It is a simple matter to see that both $\boldsymbol{\rho(y \vee z)}\mathbf{A}$ and $\boldsymbol{\rho(y \wedge z)}\mathbf{A}$ are possible zygotes of $\boldsymbol{\rho(y)}\mathbf{A}$ and $\boldsymbol{\rho(z)}\mathbf{A}$.

- CASE $u = c_\gamma$, FOR SOME CLAUSE γ OF ϕ CONTAINING THREE POSITIVE LITERALS. Since γ contains only positive literals, say $\gamma = x \vee y \vee z$, then $\mathcal{G}^{\rho}_{\phi}(c_\gamma) = \mathbf{TA}$. As $\rho(\gamma) = T$, we have that either $\rho(x) = T$ or $\rho(y \vee z) = T$. By the definition of $\mathcal{G}^{\rho}_{\phi}$, it follows that either $\mathcal{G}^{\rho}_{\phi}(f_\gamma) = \mathbf{TA}$ or $\mathcal{G}^{\rho}_{\phi}(m_\gamma) = \mathbf{TA}$. Since the individual c_γ can inherit the $\mathbf{A}$ allele from either of its parents, we infer that $\mathcal{G}^{\rho}_{\phi}(c_\gamma)$ is a possible zygote of $\mathcal{G}^{\rho}_{\phi}(f_\gamma)$ and $\mathcal{G}^{\rho}_{\phi}(m_\gamma)$, which was to be shown.

– 'IF' IMPLICATION. Assume that the genotype information $\mathcal{G}_\phi$ is consistent with P_ϕ. This means that there is a complete, consistent genotype information $\mathcal{G}^{c}_{\phi}$ for P_ϕ that extends $\mathcal{G}_\phi$. We shall show how to construct from it a satisfying assignment ρ for ϕ.

Note, first of all, that, because of the way $\mathcal{G}_\phi$ is defined over variable gadgets, $\mathcal{G}^{c}_{\phi}$ must assign either $\mathbf{TA}$ or $\mathbf{FA}$ to each variable individual v_x. Hence the genotype information $\mathcal{G}^{c}_{\phi}$ determines an assignment ρ of truth values to the variables occurring in ϕ as follows:

$$\rho(x) = \begin{cases} T & \text{if } \mathcal{G}^{c}_{\phi}(v_x) = \mathbf{TA} \\ F & \text{if } \mathcal{G}^{c}_{\phi}(v_x) = \mathbf{FA}. \end{cases}$$

We now argue that this ρ is indeed a satisfying assignment for ϕ. To this end, it is sufficient to show that ρ satisfies each of the clauses of ϕ. This we now proceed to prove by considering each of the possible forms a clause γ of ϕ may take.

- CASE $\gamma = x \vee y$. Since $\mathcal{G}^{c}_{\phi}$ is consistent with P_ϕ, we have that $\mathcal{G}^{c}_{\phi}(f_\gamma)$ is contained in $\{\boldsymbol{\rho(x)}\mathbf{A}, \mathbf{AA}\}$, and that $\mathcal{G}^{c}_{\phi}(m_\gamma)$ is contained in $\{\boldsymbol{\rho(y)}\mathbf{A}, \mathbf{AA}\}$. Moreover, as $\mathcal{G}^{c}_{\phi}(c_\gamma)$ is $\mathbf{TA}$, either $\mathcal{G}^{c}_{\phi}(f_\gamma)$ or $\mathcal{G}^{c}_{\phi}(m_\gamma)$ must be equal to $\mathbf{TA}$. Because of the way P_ϕ is built, this can only happen if either $\mathcal{G}^{c}_{\phi}(v_x) = \mathbf{TA}$ or $\mathcal{G}^{c}_{\phi}(v_y) = \mathbf{TA}$. This yields that $\rho(x \vee y) = T$, which was to be shown.
- CASE $\gamma = \overline{x} \vee \overline{y} \vee \overline{z}$. Since $\mathcal{G}^{c}_{\phi}$ is consistent with P_ϕ, we have that:
  1. $\mathcal{G}^{c}_{\phi}(f_\gamma)$ is contained in $\{\boldsymbol{\rho(x)}\mathbf{A}, \mathbf{AA}\}$,
  2. $\mathcal{G}^{c}_{\phi}(gf_\gamma)$ is contained in $\{\boldsymbol{\rho(y)}\mathbf{A}, \mathbf{AA}\}$,
  3. $\mathcal{G}^{c}_{\phi}(gm_\gamma)$ is contained in $\{\boldsymbol{\rho(z)}\mathbf{A}, \mathbf{AA}\}$,
  4. $\mathcal{G}^{c}_{\phi}(m_\gamma)$ is contained in $\{\boldsymbol{\rho(y)}\mathbf{A}, \boldsymbol{\rho(z)}\mathbf{A}, \boldsymbol{\rho(y)\rho(z)}, \mathbf{AA}\}$, and
  5. either $\mathcal{G}^{c}_{\phi}(f_\gamma) = \mathbf{FA}$ or $\mathcal{G}^{c}_{\phi}(m_\gamma)$ is contained in $\{\mathbf{FA},\mathbf{FF},\mathbf{FT}\}$.

  If $\mathcal{G}^{c}_{\phi}(f_\gamma) = \mathbf{FA}$ holds, then, by item 1 above and the way P_ϕ was built, it follows that $\mathcal{G}^{c}_{\phi}(v_x) = \mathbf{FA}$. Hence, by the definition of ρ, we have that $\rho(x) = F$. We may therefore conclude that ρ satisfies γ.

  If $\mathcal{G}^{c}_{\phi}(m_\gamma)$ is contained in $\{\mathbf{FA},\mathbf{FF},\mathbf{FT}\}$, then either $\mathcal{G}^{c}_{\phi}(gf_\gamma)$ or $\mathcal{G}^{c}_{\phi}(gm_\gamma)$ equals $\mathbf{FA}$. Again, we have that either $\mathcal{G}^{c}_{\phi}(v_y) = \mathbf{FA}$ or $\mathcal{G}^{c}_{\phi}(v_z) = \mathbf{FA}$. Hence, by the definition of ρ, it follows that either $\rho(y)$ or $\rho(z)$ equals F. We may therefore conclude that ρ satisfies γ.

The proofs for the other two cases follow similar lines, and are therefore omitted.

This completes the proof of the proposition. □

Since the pedigree P_ϕ can be constructed in polynomial time from the formula ϕ, the proposition above allows us to conclude that 3CONS is NP-hard, and the proof of Thm. 1 is now complete.

4.1 Discussion

The reference [11] offers, amongst other things, an alternative reduction from 3SAT to CONS that uses a number of alleles that is linear in the number of variables in the input 3SAT instance. This reduction, albeit possibly conceptually simpler than the one presented here, is not reasonable from a genetic standpoint because the maximum number of alleles that can be expected for a gene is roughly 100 [14]. Furthermore, although inbreeding does occur often in the animal kingdom, it would still be critical from a genetic perspective if our modelling of 3SAT using pedigrees were based on severe inbreeding, and we have striven to avoid this problem in our constructions. (The loops that arise in the pedigrees resulting from our reductions are called *marriage loops* in the pedigree literature—see, e.g., [24]—, and are natural in real life pedigrees.) The question is whether our other modelling assumptions are fair in light of the biological knowledge on consistency checking, e.g., the amount and form of genotype information in the real world. We now discuss these aspects by analyzing the pedigree structure and the genotyped individuals produced by the reduction outlined above.

Number of Offspring The points were a large number of children from a single couple can occur in our construction are the inheritance points of the variable individuals. Every occurrence of the same variable implies that a child is constructed from the inheritance point. Theoretically, the reduction requires an arbitrary number of children from a single couple. Although it can be argued from a complexity theoretic perspective that it is equally complex to check for the satisfiability of formulae in conjunctive normal form where at most three occurrences of a single variable are allowed (see, e.g., [23, Propn. 9.3]), and thus that three children per couple are enough to reduce 3SAT to 3CONS, it is also possible to argue strongly on the subject from a biological viewpoint. Two arguments can be brought forth, the first regarding an expansion of the structure, and the second regarding the gender of the variable individuals. First, we have argued in [11] that it is possible to model a variable gadget in such a way that no more than fifteen children are needed in the reduction. Second, it is theoretically possible for male individuals to have a large number of children, but with different women (the women still have an upper bound on the number of offspring). This naturally requires that the variable individuals be males.

Genotype History The aforementioned reduction from 3SAT to 3CONS requires genotype information five generations back. In the case of species with a lifespan of up to five years this seems a reasonable assumption. But for humans and animals with a long lifespan, it is doubtful whether such data exist. Although it can be argued that it is just a matter of time before such genotype history does exist for humans, we have shown in [11] that it is possible to perform a reduction where there is only genotype information for the individuals in the youngest generation of a pedigree, at the price of using a larger number of dummy alleles.

5 The Complexity of Consistency Checking Non-looping Pedigrees

As already remarked in Sect. 1, our reduction from 3SAT to 3CONS employs looping pedigrees. The following result, which seems to be folklore in the literature on computational genetics, offers strong evidence that this is most likely necessary.

Theorem 2. *Checking the consistency of non-looping pedigrees can be performed in polynomial time.*

Our order of business will now be to prove the above theorem. In our proof, we shall make use of the algorithm for genotype elimination proposed by Lange and Goradia in [18]. Since we shall present a complexity analysis of that algorithm in what follows, we offer a slight adaptation of the algorithm from *op. cit.* in Table 1-2. (The only difference between the version of the algorithm presented in Table 1-2, and that from [18, pp. 251–252] is that, in step A, for each pedigree member we list all of the genotypes compatible with the genotype assignment $\mathcal{G}$, rather than those compatible with his/her phenotype.)

INPUT: A pedigree $P = \langle V, F, \mathbf{p}, \mathbf{m} \rangle$ with a genotype information $\mathcal{G}$ for it.
OUTPUT: A mapping $\mathcal{G}' : V \rightarrow 2^{\mathrm{Two}(\mathcal{A})}$ such that:

- $\mathcal{G}'(v)$ is either empty or equals $\{\mathcal{G}(v)\}$, for every $v \in \mathrm{dom}(\mathcal{G})$, and
- for every $v \in V$, it holds that $g \in \mathcal{G}'(v)$ if, and only if, $\mathcal{G}^c(v) = g$ for some complete and consistent genotype information $\mathcal{G}^c$ that extends $\mathcal{G}$.

LANGE-GORADIA ALGORITHM: On input P with associated genotype information $\mathcal{G}$ proceed as follows:

A. For each pedigree member v, set $\mathcal{G}'(v)$ to $\{\mathcal{G}(v)\}$, if $v \in \mathrm{dom}(\mathcal{G})$, and to $\mathrm{Two}(\mathcal{A})$, otherwise.

B. For each nuclear family:

1. Consider each mother-father genotype pair.
   (a) Determine which zygote genotypes can result (according to the rules in Def. 4(1)).
   (b) If each child in the nuclear family has one or more of these zygote genotypes among his current list of genotypes, then save the parental genotypes. Also save any child genotype matching one of the created zygote genotypes.
   (c) If any child has none of these zygote genotypes among his current list of genotypes—i.e., is incompatible with the current parental pair of genotypes—take no action to save any genotypes.
2. For each person v in the nuclear family, exclude from $\mathcal{G}'(v)$ any genotypes not saved during step 1 above.

C. Repeat step B until no more genotypes can be excluded.

Table 1-2: The Lange-Goradia Genotype Elimination Algorithm.

The correctness of the algorithm for non-looping pedigrees has been shown in [18, pp. 254–255]. The proof relies upon the observation that two nuclear families in a non-looping pedigree that share some individual have *exactly* one member in common.

In light of the following result, the Lange-Goradia algorithm for genotype elimination can be used to decide the consistency of a non-looping pedigree with associated genotype information.

Lemma 1. *Assume that P is a non-looping pedigree with associated genotype information $\mathcal{G}$. Then $\mathcal{G}$ is consistent for P if, and only if, the set $\mathcal{G}'(v)$ returned by the Lange-Goradia algorithm on input $(P, \mathcal{G})$ is non-empty for every member v of P.*

Proof. Suppose that $\mathcal{G}$ is consistent for P. This means that there is a complete, consistent genotype information $\mathcal{G}^c$ for P that extends $\mathcal{G}$. By the post-condition of the Lange-Goradia algorithm, the genotype $\mathcal{G}^c(v)$ is contained in $\mathcal{G}'(v)$ for every member v of P. It follows that $\mathcal{G}'(v)$ is non-empty for every member v of P, which was to be shown.

Conversely, assume that the set $\mathcal{G}'(v)$ returned by the Lange-Goradia algorithm on input $(P, \mathcal{G})$ is non-empty for every member v of P. Lange and Goradia show in [18, pp. 254–255] how to build a complete, consistent genotype information $\mathcal{G}^c$ such that $\mathcal{G}^c(v)$ is contained in $\mathcal{G}'(v)$ for every member v of P. By the post-condition of the Lange-Goradia algorithm, $\mathcal{G}'(v)$ is either empty or equals $\{\mathcal{G}(v)\}$, for every $v \in \mathrm{dom}(\mathcal{G})$. By our assumption, we have that $\mathcal{G}'(v)$ equals $\{\mathcal{G}(v)\}$, for every $v \in \mathrm{dom}(\mathcal{G})$. It follows that the constructed genotype information $\mathcal{G}^c$ extends $\mathcal{G}$, completing the proof. □

To finish the proof of Thm. 2, it is therefore sufficient to argue that the Lange-Goradia algorithm has polynomial worst-case complexity. (Apparently, no such complexity analysis is available in the genetics literature [17].) To this end, let us assume that the input pedigree consists of n members, and that m of them are founders. We shall prove that the worst-case time complexity of the Lange-Goradia algorithm is polynomial in n and m. (In fact, the algorithm is also polynomial in the cardinality of the allelic alphabet $\mathcal{A}$, but the precise upper bound depends on the data structures used to implement the list of sets of genotypes associated with the pedigree.)

To this end, in light of step B of the algorithm, we begin by providing an upper bound on the number of nuclear families that may exist in a pedigree with n members and m founders. Such an upper bound on the number of nuclear families is $n - m$. This follows because every non-founder in the pedigree can appear as a child in *exactly one* nuclear family.

Remark 4. The $n - m$ upper bound on the number of nuclear families can actually be achieved, as witnessed by the pedigree

$$P_m = \langle \{1, \ldots, 2m-1\}, \{1, m+1, \ldots, 2m-1\}, \mathbf{p}, \mathbf{m} \rangle \ , \tag{1-2}$$

where $m > 0$, and $\mathbf{p}(i) = i - 1$ and $\mathbf{m}(i) = m + i - 1$, for every $i \in \{2, \ldots, m\}$.

This pedigree, that is depicted in Fig. 1-9, has m male members (one of whom is a founder), $m-1$ female members (all of whom are founders) and $m-1$ nuclear families.

We are now in a position to offer an upper bound on the worst-case complexity of the Lange-Goradia algorithm. Step A in the algorithm can be performed in time $O(n)$.

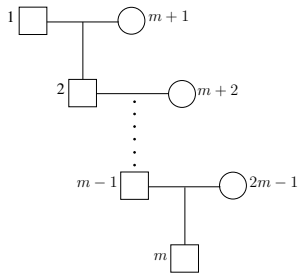

Fig. 1-9: The pedigree P_m.

Step B involves a loop that goes through all of the nuclear families in the pedigree. We saw above that there are at most $n - m$ nuclear families to consider. For each nuclear family, step 1a takes constant time, whereas steps 1b and 1c can be performed in linear time in the number of children in the nuclear family under consideration. Since every non-founder of the pedigree appears as child in exactly one nuclear family, it follows that step B can be performed in time $O(n - m)$. Since step B can be performed at most $\frac{k^2+k}{2} \cdot n$ times, where k is the size of the allelic alphabet, we can therefore conclude that the Lange-Goradia algorithm runs in time $O(n(n - m))$. This completes the proof of Thm. 2.

6 Further Results

In this section we discuss briefly three new problems related to CONS motivated by the underlying biology, and study their computational complexity.

6.1 Tolerance to Critical Genotypes

A *critical genotype* is genotype information on an individual that, if removed, would make an inconsistent pedigree with genotype information consistent. Assume that it is revealed that some application of pedigrees with genotype information is tolerant to a specific number, say k, of critical genotypes in the genotype information. We denote the problem of deciding whether there are k critical genotypes in a CONS instance as kCRIT. For example, the inconsistent pedigree depicted in Fig. 1-2 is in 1CRIT because removal of genotype information from individual 1 or 2 results in a consistently genotyped pedigree. Note that 0CRIT is just the CONS problem. We shall now show that:

Theorem 3. *In the presence of at least three alleles, kCRIT is NP-complete for every $k \geq 0$.*

Proof. Observe, first of all, that kCRIT is in NP for every $k \geq 0$. This follows because one can construct a nondeterministic Turing machine that, given a pedigree P, genotype

information $\mathcal{G}$ for P, and a non-negative integer k, first guesses k genotype assignments to be removed from $\mathcal{G}$, then guesses a complete extension $\mathcal{G}'$ of the resulting genotype information, and finally proceeds to check, in time that is linear in the number of non-founders in P, if $\mathcal{G}'$ is consistent for P.

To complete the proof, it therefore suffices only to show that 3CONS can be reduced to kCRIT in polynomial time, for every $k \geq 0$. This is immediate if $k = 0$ because, as remarked earlier, 0CRIT is just the CONS problem.

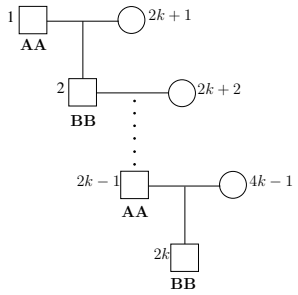

Fig. 1-10: The pedigree P_{2k}.

Assume now that k is positive, and that $\mathbf{A}$ and $\mathbf{B}$ are distinct alleles. Consider a pedigree P with associated genotype information $\mathcal{G}$. Build an instance of kCRIT by taking the disjoint union of the pedigrees P and P_{2k} (see, equation 1-2 in Remark 4 for the definition of P_{2k}), and extending the genotype information $\mathcal{G}$ to P_{2k} by assigning genotype $\mathbf{AA}$ to all of the male individuals i in P_{2k} with i odd, and genotype $\mathbf{BB}$ to all of the male individuals i in P_{2k} with i even. (For the sake of clarity, Fig. 1-10 depicts the resulting pedigree with genotype information. Note that this genotype assignment for P_{2k} is inconsistent.)

We claim that genotype information $\mathcal{G}$ is consistent for P if, and only if, the pedigree and associated genotype information resulting from the above construction have k critical genotypes.

Indeed, if $\mathcal{G}$ is consistent for P, then the genotype information obtained by removing the genotype $\mathbf{AA}$ from k members of P_{2k} is consistent for the disjoint union of the pedigrees P and P_{2k}. Conversely, assume that the pedigree and associated genotype information resulting from the above construction have k critical genotypes. Since the genotype information for P_{2k} can only be made consistent by removing at least k genotypes, we may therefore conclude that $\mathcal{G}$ is consistent for P. □

Remark 5. The polynomial time reduction from 3CONS to kCRIT ($k > 0$) used in the above proof produces a "disconnected" pedigree. It is, however, not too too hard to modify it so that it yields a "connected" pedigree. To this end, consider a pedigree P with associated genotype information $\mathcal{G}$. Assume, without loss of generality, that P has a female member u without offspring. We build a pedigree P' and associated genotype information $\mathcal{G}'$ as done in the above proof, but we add a new male founder, called 0,

whose genotype is $\mathbf{BB}$. Individual 0 is the father in P' of individual 1 in P_{2k}, and u is his mother.

Claim. $\mathcal{G}$ is consistent with P if, and only if, $(\mathcal{G}', P')$ is contained in kCRIT.

To see that the above claim holds, assume, first of all, that $\mathcal{G}$ is consistent with P. This means that there is a consistent, complete extension $\mathcal{G}^c$ of $\mathcal{G}$ over P. Consider the genotype information $\mathcal{G}''$ obtained from $\mathcal{G}'$ by removing the genotype $\mathbf{AA}$ from k members of P_{2k}. This genotype information is consistent for P'. In fact, assuming that $\mathcal{G}^c(u) = \mathbf{CD}$, say, the complete extension of $\mathcal{G}''$ that agrees with $\mathcal{G}^c$ over P, and assigns genotype $\mathbf{BB}$ to all of the members of P_{2k}, apart from 1 that has genotype $\mathbf{BC}$, is easily checked to be consistent with P'.

Conversely, assume that $(\mathcal{G}', P')$ is contained in kCRIT. As in the proof of Thm. 3, since the genotype information for P_{2k} can only be made consistent by removing at least k genotypes, we may readily conclude that $\mathcal{G}$ is consistent for P.

6.2 Consistency Checking with Two Alleles

According to [26, p. 274], *single nucleotide polymorphisms* are utilized markers where two alleles exist. Consistency checking of such data amounts to the problem 2CONS. A relevant question is whether 2CONS is also NP-complete or whether it is polynomial time decidable. Three is often a "magic number", when it comes to the structural complexity of a computational problem. For instance, 3COLORING and 3SAT are NP-complete, while 2COLORING and 2SAT are polynomial time decidable (see, e.g., [23, pp. 185 and 198]). The same holds for consistency checking of pedigrees in light of the following result:

Theorem 4. *The problem 2CONS is decidable in polynomial time.*

Proof. Let $\mathcal{G}$ be a genotype information for a pedigree P over an allelic alphabet $\mathcal{A}$ of cardinality two, say $\mathcal{A} = \{\mathbf{A},\mathbf{B}\}$. Assume, for use in our complexity analysis, that P has n members and m founders. We present a polynomial time algorithm to check whether $\mathcal{G}$ is consistent for P.

The algorithm consists of the following three steps:

1. Compute the set of members of P that must be assigned genotype $\mathbf{AA}$ or $\mathbf{BB}$ in each complete, consistent extension of $\mathcal{G}$. Report the inconsistency of $\mathcal{G}$ if some of these members is wrongly genotyped by $\mathcal{G}$. Otherwise, assign the appropriate genotype to all of the members in these sets.
2. Check whether P with the resulting genotype information contains a child with genotype $\mathbf{AA}$ that has a parent with genotype $\mathbf{BB}$, or a child with genotype $\mathbf{BB}$ that has a parent with genotype $\mathbf{AA}$. If such a child is found, then report the inconsistency of $\mathcal{G}$.
3. Check the consistency of each sub-pedigree consisting of a child and its two parents all of whose members are fully genotyped in the extension of $\mathcal{G}$ produced by step 1 of the algorithm. If any of these sub-pedigrees has inconsistent genotype information, then report the inconsistency of $\mathcal{G}$. Otherwise report that $\mathcal{G}$ is consistent for P.

We now argue that the above algorithm decides 2CONS, and has polynomial time worst-case complexity.

The first step in the algorithm amounts to computing the sets $\text{Must}_{\mathbf{AA}}$ and $\text{Must}_{\mathbf{BB}}$, which are the least sets satisfying the following conditions:

- If v is a member of P and $\mathcal{G}(v) = \mathbf{CC}$, where $\mathbf{C} \in \{\mathbf{A}, \mathbf{B}\}$, then $v \in \text{Must}_{\mathbf{CC}}$; and
- If v is a member of P, and $\mathbf{p}(v), \mathbf{m}(v)$ are contained in $\text{Must}_{\mathbf{CC}}$, where $\mathbf{C} \in \{\mathbf{A}, \mathbf{B}\}$, then $v \in \text{Must}_{\mathbf{CC}}$.

These sets can be computed in time $O(n^2)$. Moreover, a simple induction on the definition of the sets $\text{Must}_{\mathbf{AA}}$ and $\text{Must}_{\mathbf{BB}}$ shows that:

Claim. If v is contained in $\text{Must}_{\mathbf{CC}}$, where $\mathbf{C} \in \{\mathbf{A}, \mathbf{B}\}$, then each consistent genotype information that extends $\mathcal{G}$ must assign genotype $\mathbf{CC}$ to v.

Thus the algorithm reports genotype inconsistencies correctly in his first step.

Steps 2 and 3 of the algorithm can be carried out in time $O(n - m)$, and clearly also report genotype inconsistencies correctly. The overall worst-case running time of the algorithm is thus $O(n^2)$. Its correctness follows from the following:

Claim. Assume that the algorithm presented above reports that $\mathcal{G}$ is consistent for P. Let $\mathcal{G}'$ be the extension of $\mathcal{G}$ that is generated by step 1 of the algorithm. Then the complete genotype information $\mathcal{G}'^c$ that extends $\mathcal{G}'$ by assigning genotype $\mathbf{AB}$ to each member of P that is not contained in the domain of $\mathcal{G}'$ is consistent for P.

This claim can be shown by analyzing all the possible forms the genotype information $\mathcal{G}'$ may take on a sub-pedigree of P consisting of a child and its two parents—where, in light of step 3 of the algorithm, we need only consider the case in which at most two individuals in the sub-pedigree under consideration are in the domain of $\mathcal{G}'$.

- If none of the individuals under consideration is genotyped by $\mathcal{G}'$, then assigning genotype $\mathbf{AB}$ to all of them is consistent, and we are done.
- If only one of the individuals under consideration is genotyped by $\mathcal{G}'$, then two cases can arise:
  - exactly one of the parents is genotyped, or
  - the child is genotyped.

  If the child is genotyped, then its genotype is a possible zygote of $\mathbf{AB}$ and $\mathbf{AB}$, that are the genotypes assigned to its parents by $\mathcal{G}'^c$. Indeed, every member of $\text{Two}(\mathcal{A})$ is a possible zygote of $\mathbf{AB}$ and $\mathbf{AB}$.
  If one of the parents is genotyped, then observe that $\mathbf{AB}$, the genotype assigned to the child by $\mathcal{G}'^c$, is a possible zygote of $\mathbf{AB}$ and each member of $\{\mathbf{AA},\mathbf{BB},\mathbf{AB}\}$.
- If exactly two of the individuals under consideration are genotyped by $\mathcal{G}'$, then, by symmetry, we can limit ourselves to considering the following two sub-cases:
  - the two parents are genotyped by $\mathcal{G}'$, or
  - one of the parents and the child are genotyped by $\mathcal{G}'$.

If the two parents are genotyped by $\mathcal{G}'$, then we can assume that they are not homozygous—i.e., that it is not the case that they both have genotype $\mathbf{AA}$ or genotype $\mathbf{BB}$—, or else their child would have been genotyped at step 1 of the algorithm. It is now tedious, but not hard, to check that $\mathbf{AB}$, the genotype assigned to the child by G'^c, is a possible zygote of each pair of non-homozygous genotypes over alleles $\mathbf{A}$ and $\mathbf{B}$.

If one of the parents and the child are genotyped by $\mathcal{G}'$, then we can assume that it is not the case that one of them has genotype $\mathbf{AA}$ and the other has genotype $\mathbf{BB}$, or else the genotyped pedigree would have been deemed to be inconsistent at step 2 of the algorithm. A tedious, but not hard, case analysis now suffices to check that the genotype assigned to the child by $\mathcal{G}'$ is a possible zygote of $\mathbf{AB}$, the genotype assigned to the ungenotyped parent by $\mathcal{G}'^c$, and that of the parent genotyped by $\mathcal{G}'$.

This completes the proof of the claim, and that of the theorem. □

Remark 6. The pedigree depicted in Fig. 1-2, modified so that individual 5 has genotype $\mathbf{AA}$, shows that the algorithm used in the above proof is incorrect in the presence of more than two alleles. That genotyped pedigree would pass steps 1–3 in the algorithm, but is inconsistent.

In light of Thms. 2 and 4, one can argue that 3CONS is indeed the simplest consistency checking problem that is still intractable. In fact, restricting our attention to genes over two alleles or to pedigrees without loops yields algorithmic problems that can be solved in polynomial time.

6.3 Phase Known Consistency Checking

In this paper, we have considered consistency checking in a phase unknown setting—that is, when it is not possible, by observing a chromosome pair, to say which component is inherited paternally or maternally. We now briefly turn our focus to the task of consistency checking where the phase of the genotype information is known. The motivation for this type of investigation is that it is sometimes possible to infer the identity of the parent from whom some allele originated (and thereby also the origin of the other allele).

Definition 5. *A* phase known genotype information *for a pedigree $P = \langle V, F, \mathbf{p}, \mathbf{m}\rangle$ is a partial function $\mathcal{G}^p : V \hookrightarrow \mathcal{A} \times \mathcal{A}$. The genotype information $\mathcal{G}^p$ is* complete *if* $\mathrm{dom}(\mathcal{G}^p) = V$.

A complete, phase known genotype information $\mathcal{G}^p$ for a pedigree P is consistent *with P if whenever $v \in N$ and $\mathcal{G}^p(v) = (\mathbf{A}, \mathbf{B})$, then $\mathbf{A}$ is one of the components of $\mathcal{G}^p(\mathbf{p}(v))$, and $\mathbf{B}$ is one of the components of $\mathcal{G}^p(\mathbf{m}(v))$.*

A phase known genotype information is consistent *with P if it can be extended to a complete and consistent phase known genotype information for P.*

As it is common in the genetics literature, in what follows we shall write $\mathbf{A}|\mathbf{B}$ for the ordered pair $(\mathbf{A}, \mathbf{B})$.

Let PCONS be the problem of checking the consistency of a pedigree with phase known genotype information.

Theorem 5. *In the presence of at least four alleles, PCONS is NP-complete.*

Proof. Since PCONS is easily seen to be in NP, we need only show that this problem is NP-hard. To this end, in light of Thm. 1, it is sufficient to argue that every 3CONS instance can be reduced in polynomial time to a PCONS instance over four alleles that is consistent if, and only if, so was the original 3CONS instance.

Recall that a CONS instance consists of a pedigree $P = \langle V, F, \mathbf{p}, \mathbf{m} \rangle$ and a genotype information $\mathcal{G}$ for it. We assume, without loss of generality, that $\mathcal{A}$ is a totally ordered set of three alleles that does not contain the allele $\mathbf{A}$, and write the members of sets in $\mathrm{Two}(\mathcal{A})$ in an order that is consistent with the total order. We assume, furthermore, that all "leaves" in P are male individuals. From P and $\mathcal{G}$, we build a PCONS instance consisting of a pedigree $P' = \langle V', F', \mathbf{p}', \mathbf{m}' \rangle$ and a phase known genotype information $\mathcal{G}^p : V' \hookrightarrow (\mathcal{A} \cup \{\mathbf{A}\}) \times (\mathcal{A} \cup \{\mathbf{A}\})$ thus:

1. The set V' of members of P' includes V. Moreover, it contains individuals u_i (with $i \in \{1, 2, 3, 4\}$) for every $u \in \mathrm{dom}(\mathcal{G})$;
2. F', the set of founders of P', is equal to $F \cup \{u_1, u_2 \mid u \in \mathrm{dom}(\mathcal{G})\}$;
3. the functions $\mathbf{p}'$ and $\mathbf{m}'$ are given by:

$$\mathbf{p}'(v) = \begin{cases} \mathbf{p}(v) & \text{if } v \in V \setminus F \\ u & \text{if } u \text{ is a male individual, and } v \in \{u_3, u_4\} \\ u_{i-2} & \text{if } u \text{ is a female individual, and } v \in \{u_3, u_4\} \end{cases}$$

$$\mathbf{m}'(v) = \begin{cases} \mathbf{m}(v) & \text{if } v \in V \setminus F \\ u & \text{if } u \text{ is a female individual, and } v \in \{u_3, u_4\} \\ u_{i-2} & \text{if } u \text{ is a male individual, and } v \in \{u_3, u_4\} \end{cases} .$$

4. The phase known genotype information $\mathcal{G}^p : V' \hookrightarrow (\mathcal{A} \cup \{\mathbf{A}\}) \times (\mathcal{A} \cup \{\mathbf{A}\})$ is defined as follows:

$$\mathcal{G}^p(v) = \begin{cases} \mathbf{A}|\mathbf{A} & \text{if } v \in \{u_1, u_2\} \text{ for some } u \in \mathrm{dom}(\mathcal{G}) \\ \mathbf{B}|\mathbf{A} & \text{if } v = u_3 \text{ for some } u \in \mathrm{dom}(\mathcal{G}), \text{ and } \mathcal{G}(u) = \{\mathbf{B},\mathbf{C}\} \\ \mathbf{C}|\mathbf{A} & \text{if } v = u_4 \text{ for some } u \in \mathrm{dom}(\mathcal{G}), \text{ and } \mathcal{G}(u) = \{\mathbf{B},\mathbf{C}\} \\ \mathbf{B}|\mathbf{B} & \text{if } v \in \{u_3, u_4\} \text{ for some } u \in \mathrm{dom}(\mathcal{G}), \text{ and } \mathcal{G}(u) = \{\mathbf{B}\} \\ \text{undefined} & \text{otherwise} \end{cases} .$$

The idea underlying the construction presented above is depicted in Fig. 1-11. There we present the pedigreee transformation applied to a genotyped male individual in P. Note that, in the PCONS instance P', all of the individuals in V are ungenotyped. On the other hand, as exemplified in Fig. 1-11, all of the dummy individuals u_i ($i \in \{1, \ldots, 4\}$) associated with a $u \in \mathrm{dom}(\mathcal{G})$ are genotyped in such a way that the only consistent phase known genotypes that may be assigned to u in P' are ordered pairs whose components are the alleles in $\mathcal{G}(u)$.

We shall now prove that $\mathcal{G}$ is consistent for P if, and only if, $\mathcal{G}^p$ is consistent for P' in a phase known setting.

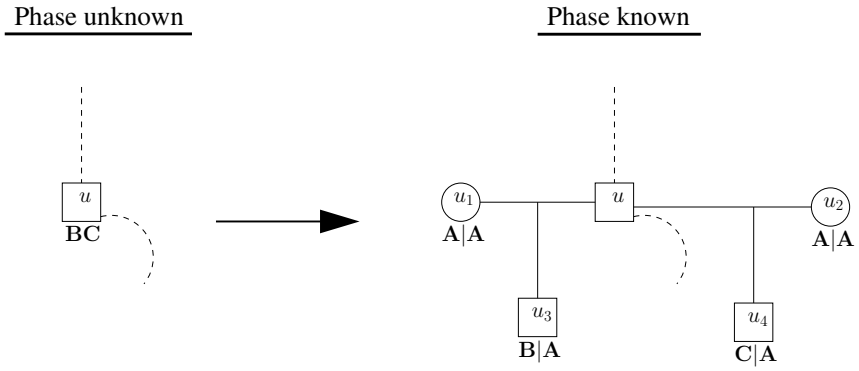

Fig. 1-11: The scheme for reducing a pedigree with genotype information in a phase unknown setting to a phase known setting. This illustrates the reduction for a single male individual. The arrow denotes the reduction, and the dashed lines indicate potential other family relationships.

Assume, first of all, that $\mathcal{G}$ is consistent for P. Then there is a consistent, complete genotype information $\mathcal{G}^c$ that extends $\mathcal{G}$. Our order of business is to construct a consistent, complete extension $\mathcal{G}^{p,c}$ of $\mathcal{G}^p$ from $\mathcal{G}^c$. This we do by defining $\mathcal{G}^{p,c}(u)$ for each $u \in V$ as follows:

- if $\mathcal{G}^c(u) = \{\mathbf{B}\}$, then we set $\mathcal{G}^{p,c}(u) = \mathbf{B}|\mathbf{B}$;
- if $\mathcal{G}^c(u) = \{\mathbf{B},\mathbf{C}\}$ and $u \in V \setminus F$, then, since $\mathcal{G}^c$ is consistent, we have that, say, $\mathbf{B}$ and $\mathbf{C}$ are contained in $\mathcal{G}^c(\mathbf{p}(u))$ and $\mathcal{G}^c(\mathbf{m}(u))$, respectively. In that case, we set $\mathcal{G}^{p,c}(u) = \mathbf{B}|\mathbf{C}$. To resolve any ambiguity, we use, if necessary, the total order on $\mathcal{A}$ to pick the smallest allele as the paternally inherited one in $\mathcal{G}^{p,c}(u)$. For instance, if $\mathcal{G}^c$ assigns genotype $\{\mathbf{B},\mathbf{C}\}$ to u and both of its parents, and $\mathbf{B}$ is smaller than $\mathbf{C}$ with respect to the total order on the allelic alphabet, then we stipulate that $\mathcal{G}^{p,c}(u) = \mathbf{B}|\mathbf{C}$;
- if $\mathcal{G}^c(u) = \{\mathbf{B},\mathbf{C}\}$ and $u \in F$, then we arbitrarily set $\mathcal{G}^{p,c}(u) = \mathbf{B}|\mathbf{C}$.

Note that the genotype assignment $\mathcal{G}^{p,c}$ sets $\mathcal{G}^{p,c}(u)$ to either $\mathbf{B}|\mathbf{C}$ or $\mathbf{C}|\mathbf{B}$ for the individual u in the phase known pedigree in Fig. 1-11.

We now argue that $\mathcal{G}^{p,c}$ is consistent. To this end, it is sufficient to check the consistency conditions given in Def. 5 for each $u \in V \setminus F$. (Note that, for each such u, it holds that $\mathbf{p}'(u) = \mathbf{p}(u)$ and $\mathbf{m}'(u) = \mathbf{m}(u)$.) We distinguish two cases, depending on the form $\mathcal{G}^{p,c}(u)$ takes.

- CASE $\mathcal{G}^{p,c}(u) = \mathbf{B}|\mathbf{B}$. By the definition of $\mathcal{G}^{p,c}$, we have that $\mathcal{G}^c(u) = \{\mathbf{B}\}$. Since $\mathcal{G}^c$ is consistent for the pedigree P, allele $\mathbf{B}$ is contained in both $\mathcal{G}^c(\mathbf{p}(u))$ and $\mathcal{G}^c(\mathbf{m}(u))$. By the definition of $\mathcal{G}^{p,c}$, it follows that $\mathbf{B}$ is a component of both $\mathcal{G}^{p,c}(\mathbf{p}'(u))$ and $\mathcal{G}^{p,c}(\mathbf{m}'(u))$, which was to be shown.
- CASE $\mathcal{G}^{p,c}(u) = \mathbf{B}|\mathbf{C}$. By the definition of $\mathcal{G}^{p,c}$, we have that $\mathcal{G}^c(u) = \{\mathbf{B},\mathbf{C}\}$, $\mathbf{B} \in \mathcal{G}^c(\mathbf{p}(u))$ and $\mathbf{C} \in \mathcal{G}^c(\mathbf{m}(u))$. By the definition of $\mathcal{G}^{p,c}$, it follows that $\mathbf{B}$ is

a component of $\mathcal{G}^{p,c}(\mathbf{p}'(u))$, and $\mathbf{C}$ is a component of $\mathcal{G}^{p,c}(\mathbf{m}'(u))$, which was to be shown.

Conversely, assume that $\mathcal{G}^p$ is consistent for P'. Then there is a consistent, complete extension $\mathcal{G}^{p,c}$ of $\mathcal{G}^p$ over P'. Because of the way P' and $\mathcal{G}^p$ were built, the restriction of the genotype assignment $\mathcal{G}^{p,c}$ to V agrees with $\mathcal{G}$ over $\mathrm{dom}(\mathcal{G})$, when order in the genotype assignments is ignored. It is moreover consistent with P in a phase unknown setting. □

7 Concluding Remarks

The results in this paper show that certain basic combinatorial problems in pedigree analysis, viz. consistency checking and determining whether a genotyped pedigree has some number of critical genotypes, are NP-complete, even if we focus on a single gene with a fixed, small number of alleles. It follows that these problems are most likely computationally intractable. It would be most interesting, however, to develop heuristic algorithms for these problems, and evaluate their efficiency on real-life and/or randomly generated data. In particular, we plan to develop and evaluate algorithms for consistency checking based upon *Binary Decision Diagrams* [3] and various available SAT-solvers and tautology checkers—see, e.g., the reference [7] for a survey. We believe that the experimental evaluation of these algorithms would be of value, because consistency checking routines, like genotype elimination ones, may be used as pre-processing steps in algorithms for, e.g., linkage analysis and haplotype reconstruction [19].

From a theoretical viewpoint, we conjecture that the problem of computing the number of complete consistent extensions of a genotype information for a pedigree is $\#P$-complete [27]—i.e., it is as hard as counting the number of satisfying assignments of a boolean formula. It would also be interesting to study the complexity of approximation algorithms for computing the number of critical genotypes in a pedigree. We leave an in-depth study of these problems as future work.

Acknowledgments We thank Mogens Nielsen for the initial inspiration on a possible reduction showing that consistency checking is NP-complete, and Emmanuel Fleury for fruitful discussions on the topic of this paper, and comments on its draft versions. We are most grateful to Dan Gusfield for his comments on the connections between our respective contributions, and for bringing the reference [19] to our attention. Kenneth Lange offered prompt and informative replies to our enquiries related to extant complexity analyses for the Lange-Goradia algorithm from [18]. We are indebted to Tao Jiang for making his paper [19] available to us. Any remaining infelicities are solely our responsibility.

The work reported in this paper was partly carried out while Luca Aceto was an invited professor at Reykjavík University, and Anna Ingólfsdóttir was at Iceland Genomics Corporation. Both authors thank these institutions for their hospitality and the excellent working conditions.

References

1. G. R. ABECASIS, S. S. CHERNY, W. O. COOKSON, AND L. R. CARDON, *Merlin: Rapid analysis of dense genetic maps using sparse gene flow trees*, Nature Genetics, 30 (2002), pp. 97–101.
2. B. BERGER AND T. LEIGHTON, *Protein folding in the hydrophobic-hydrophilic (hp) model is NP-complete*, Journal of Computational Biology, 5 (1998), pp. 27–40.
3. R. BRYANT, *Graph-based algorithms for boolean function manipulation*, IEEE Trans. Comput., C-35 (1986), pp. 677–691.
4. P. CRESCENZI, D. GOLDMAN, C. PAPADIMITRIOU, A. PICCOLBONI, AND M. YANNAKAKIS, *On the complexity of protein folding (extended abstract)*, in Proceedings of the thirtieth annual ACM Symposium on Theory of Computing, ACM Press, 1998, pp. 597–603.
5. DECODE NEWS CENTER, November 2001. `http://www.decode.com/news/releases/`.
6. J. DOHRMANN, *Mænd vil vide, hvem de er far til*, Nordjyske, (2002). In Danish.
7. J. GU, P. W. PURDOM, J. FRANCO, AND B. W. WAH, *Algorithms for the satisfiability (SAT) problem: a survey*, in Satisfiability problem: theory and applications (Piscataway, NJ, 1996), vol. 35 of DIMACS Ser. Discrete Math. Theoret. Comput. Sci., Amer. Math. Soc., Providence, RI, 1997, pp. 19–151.
8. D. F. GUDBJARTSSON, *Multipoint Linkage Analysis Based on Allele Sharing Models*, PhD thesis, Institute of Statistics and Decision Sciences, Duke University, 2000.
9. D. F. GUDBJARTSSON, K. JONASSON, AND C. A. KONG, *Fast multipoint linkage calculation with Allegro*, Nature Genetics, 20 (2000), pp. 12–13.
10. M. HAMER, *Back to your roots*, New Scientist, 2334 (2002), pp. 33–36.
11. J. A. HANSEN, J. JOHNSEN, AND J. KNUDSEN, *Computational complexity of consistency checking*, Master's thesis, Department of Computer Science, Aalborg University, June 2002. Available at `http://www.cs.auc.dk/~luca/PAPERS/hjk02.ps.gz`.
12. HUMAN GENOME DIVERSITY PROJECT, March 2002. `http://www.stanford.edu/group/morrinst/hgdp.html`.
13. L. B. JORDE, J. C. CAREY, M. J. BAMSHAD, AND R. L. WHITE, *Medical Genetics*, Mosby, 1999.
14. C. B. JØRGENSEN, Tuesday, 9 April 2002. Personal communication.
15. W. S. KLUG AND M. R. CUMMINGS, *Concepts of Genetics*, Prentice Hall, 5th ed., 1997.
16. L. KRUGLYAK, M. J. DALY, M. P. REEVE-DALY, AND E. S. LANDER, *Parametric and nonparametric linkage analysis: A unified multipoint approach*, American Journal of Human Genetics, 58 (1996), pp. 1347–1363.
17. K. LANGE, Monday, 28 October 2002. Personal communication.
18. K. LANGE AND T. M. GORADIA, *An algorithm for automatic genotype elimination*, American Journal of Human Genetics, 40 (1987), pp. 250–256.
19. J. LI AND T. JIANG, *Efficient rule-based haplotyping algorithms for pedigree data [extended abstract]*, in Proceedings of RECOMB'03, April 10–13, 2003, Berlin, Germany. To appear.
20. J. R. O'CONNELL AND D. E. WEEKS, *Pedcheck: A program for identification of genotype incompatibilities in linkage analysis*, American Journal of Human Genetics, 63 (1998), pp. 259–266.
21. ———, *An optimal algorithm for automatic genotype elimination*, American Journal of Human Genetics, 65 (1999), pp. 1733–1740.
22. J. OTT, *Analysis of Human Genetic Linkage*, The Johns Hopkins University Press, 3rd ed., 1999.
23. C. H. PAPADIMITRIOU, *Computational Complexity*, Addison Wesley, 1995.

24. A. PICCOLBONI AND D. GUSFIELD, *On the complexity of fundamental computational problems in pedigree analysis*, Tech. Rep. CSE-99-8, Computer Science Department, University of California, Davis, September 1999. Revised version to appear in the *Journal of Computational Biology*.
25. E. SOBEL, J. C. PAPP, AND K. LANGE, *Detection and integration of genotyping errors in statistical genetics*, American Journal of Human Genetics, 70 (2002), pp. 496–508.
26. T. STRACHAN AND A. P. READ, *Human Molecular Genetics 2*, Wiley-Liss, 1999.
27. L. G. VALIANT, *The complexity of computing the permanent*, Theoret. Comput. Sci., 8 (1979), pp. 189–201.

Recent BRICS Report Series Publications

RS-03-17 Luca Aceto, Jens Alsted Hansen, Anna Ingólfsdóttir, Jacob Johnsen, and John Knudsen. *The Complexity of Checking Consistency of Pedigree Information and Related Problems*. March 2003. 31 pp. This paper supersedes BRICS Report RS-02-42.

RS-03-16 Ivan B. Damgård and Mads J. Jurik. *A Length-Flexible Threshold Cryptosystem with Applications*. March 2003. 19 pp.

RS-03-15 Anna Ingólfsdóttir. *A Semantic Theory for Value–Passing Processes Based on the Late Approach*. March 2003. 49 pp.

RS-03-14 Mads Sig Ager, Dariusz Biernacki, Olivier Danvy, and Jan Midtgaard. *From Interpreter to Compiler and Virtual Machine: A Functional Derivation*. March 2003. 36 pp.

RS-03-13 Mads Sig Ager, Dariusz Biernacki, Olivier Danvy, and Jan Midtgaard. *A Functional Correspondence between Evaluators and Abstract Machines*. March 2003. 28 pp.

RS-03-12 Mircea-Dan Hernest and Ulrich Kohlenbach. *A Complexity Analysis of Functional Interpretations*. February 2003. 70 pp.

RS-03-11 Mads Sig Ager, Olivier Danvy, and Henning Korsholm Rohde. *Fast Partial Evaluation of Pattern Matching in Strings*. February 2003. 14 pp. To appear in Leuschel, editor, *ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation*, PEPM '03 Proceedings, 2003.

RS-03-10 Federico Crazzolara and Giuseppe Milicia. *Wireless Authentication in χ-Spaces*. February 2003. 20 pp.

RS-03-9 Ivan B. Damgård and Gudmund Skovbjerg Frandsen. *An Extended Quadratic Frobenius Primality Test with Average and Worst Case Error Estimates*. February 2003. 53 pp.

RS-03-8 Ivan B. Damgård and Gudmund Skovbjerg Frandsen. *Efficient Algorithms for gcd and Cubic Residuosity in the Ring of Eisenstein Integers*. February 2003. 11 pp.

RS-03-7 Claus Brabrand, Michael I. Schwartzbach, and Mads Vanggaard. *The METAFRONT System: Extensible Parsing and Transformation*. February 2003. 24 pp.