Basic Research in Computer Science

On Counting the Number of Consistent Genotype Assignments for Pedigrees

Jiří Srba

BRICS Report Series **RS-05-28**

ISSN 0909-0878 **September 2005**

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/05/28/`

On Counting the Number of Consistent Genotype Assignments for Pedigrees

Jiří Srba*

BRICS**
Department of Computer Science, Aalborg University
Fredrik Bajersvej 7B, 9220 Aalborg East, Denmark
srba@brics.dk

Abstract. Consistency checking of genotype information in pedigrees plays an important role in genetic analysis and for complex pedigrees the computational complexity is critical. We present here a detailed complexity analysis for the problem of counting the number of complete consistent genotype assignments. Our main result is a polynomial time algorithm for counting the number of complete consistent assignments for non-looping pedigrees. We further classify pedigrees according to a number of natural parameters like the number of generations, the number of children per individual and the cardinality of the set of alleles. We show that even if we assume all these parameters as bounded by reasonably small constants, the counting problem becomes computationally hard (#P-complete) for looping pedigrees. The border line for counting problems computable in polynomial time (i.e. belonging to the class FP) and #P-hard problems is completed by showing that even for general pedigrees with unlimited number of generations and alleles but with at most one child per individual and for pedigrees with at most two generations and two children per individual the counting problem is in FP.

1 Introduction

Pedigrees are fundamental structures used in genetics. A pedigree describes family relations among generations of individuals. Genealogists study pedigrees in connection with the genotype information associated to the individuals at a particular locus. A genotype of a given individual is a pair of alleles in its genome (allele is one of the possible forms a gene may have). Due to different reasons, for a given pedigree the genotype information of its individuals can be known only partially. In order to complete the missing genotype information or to filter out erroneous input data, genealogists need to verify that the given partial information is consistent with the classic Mendelian laws of inheritance (see e.g. [6]),

* The author is supported in part by the research center ITI, project No. 1M0021620808

** **B**asic **R**esearch in **C**omputer **S**cience,
Centre of the Danish National Research Foundation.

which means that every individual in the pedigree has to inherit exactly one allele from each of its parents. This process is called *consistency checking* and as argued in [10], in many real-life cases a manual consistency check is very difficult, time-consuming and sometimes unsuccessful. For an accessible overview of further biological aspects we refer the reader to [1].

To the best of our knowledge only algorithmic issues of consistency and likelihood checking for pedigrees have been studied in the literature so far (see e.g. [7, 11, 13, 2, 1]). In this paper we shall focus on a more general problem of *counting* the total number of complete genotype assignments consistent with the input data. This approach can provide a deeper insight and generalize the algorithms already developed for pure consistency checking. Moreover, knowing the total number of complete genotype assignments consistent with the input data can answer several additional questions. For example the fact that the number of assignments is 1 tells us that the missing information can be uniquely reconstructed from the available data. On the other hand, knowing that there are too many possibilities how to interpret the input data indicates that more genotype sampling is needed in order to reduce uncertainty.

Our contribution. We introduce characterization of pedigrees according to a number of natural parameters that describe their shapes. Apart for the standard notion of looping/non-looping pedigrees we further distinguish the number of generations, number of children per individual and the cardinality of the set of alleles. We describe a polynomial time algorithm that counts the number of complete genotype information for a given partial genotype data in non-looping pedigrees. We use this result to show that the counting problems for general pedigrees with at most 2 generations and 2 children per individual and for pedigrees with at most 1 child per individual are also solvable in polynomial deterministic time. We complete the results by demonstrating two parsimonious reductions (i.e. reductions that preserve the number of solutions) from #Bpos-2Sat to the counting problems for pedigrees with (i) 3 generations, 2 children per individual and 2 alleles, and (ii) 2 generations, 3 children per individual and 2 alleles. Together with an obvious containment in #P this proves #P-completeness of the problems.

Related work. For the case of pure consistency checking the following results are known. The problem for non-looping pedigrees is decidable in polynomial time using a *genotype elimination* algorithm proposed by Lange and Goradia [8] and further optimized and extended by O'Connell and Weeks [11], and Du and Hoeschele [2]. For general pedigrees there is a recent work by Aceto et al. [1] showing that consistency checking is NP-complete for pedigrees with marriage loops. They prove the result by reduction from 3SAT, however, their reduction is not parsimonious (does not preserve the number of solutions). It also works only for pedigrees with at least 5 generations, 3 children per individual and 3 alleles but under the assumption that there is either a complete or no knowledge about the genotype of single individuals. Another related result is NP-completeness of marginal probability and maximum likelihood by Piccolboni and Gusfield [13].

As discussed in [1], although the problems are closely connected to consistency checking, they cannot be used to imply hardness results for our problem.

2 Basic Definitions

2.1 Pedigrees and Genotype Information

In order to reason about pedigrees and the genotype information that they contain, we need to introduce a formal model. Several formalizations of the notion of pedigree have been presented in the literature on computational genetics (see e.g. [1, 9, 13]). The definition that we provide is equivalent to the ones mentioned above.

Definition 1 (Pedigree). *A* pedigree *is a triple* $P = (M, F, \phi)$ *where*

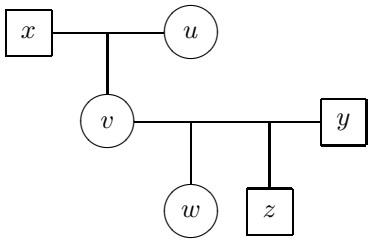
- M *and* F *are finite disjoint sets of* male*, resp.* female*, individuals,*
- $\phi : M \times F \longrightarrow 2^{(M \cup F)}$ *is a function called* family function *satisfying:*
  1. $\phi(f) \cap \phi(f') = \emptyset$ *for all* $f, f' \in M \times F$ *such that* $f \neq f'$*,*
  2. *the transitive closure of the* parental relation $\prec \subseteq (M \cup F) \times (M \cup F)$ *is irreflexive, where* $\prec$ *is defined by* $u \prec v$ *iff there is a* $w \in M \cup F$ *such that* $u \in \phi(v, w)$ *if* $v \in M$*, or* $u \in \phi(w, v)$ *if* $v \in F$*.*

We define a set of families *in* P *given by* ϕ *as* $\mathcal{F}(\phi) \stackrel{\text{def}}{=} \{f \in M \times F \mid \phi(f) \neq \emptyset\}$*. Let us also define* $p(f) \stackrel{\text{def}}{=} \{u, v\}$ *for any family* $f = (u, v) \in \mathcal{F}(\phi)$*; we call* u *and* v *the* parents *in the family* f*.*

Here is an informal explanation of the definition. Given a male $u \in M$ and a female $v \in F$, $\phi(u, v)$ is the set of all children they have. Condition 1. says that every child belongs to exactly one family and condition 2. guarantees that no individual can be its own ancestor.

The maximal elements from $M \cup F$ w.r.t. $\prec$ are called *founders* of the pedigree. Individuals that are not founders are called *non-founders*. The length of a longest chain (counting the number of nodes) w.r.t. $\prec$ is called the number of *generations*. The *set of children* of an individual $u \in M \cup F$ is defined by $\cup_{f \in \mathcal{F}(\phi), u \in p(f)} \phi(f)$ and the number of *children per individual* is the largest cardinality of this set over all individuals in the pedigree.

Example 1. Let $M \stackrel{\text{def}}{=} \{x, y, z\}$ and $F \stackrel{\text{def}}{=} \{u, v, w\}$. We define a pedigree P by $\phi(x, u) \stackrel{\text{def}}{=} \{v\}$ and $\phi(y, v) \stackrel{\text{def}}{=} \{w, z\}$. In all other cases $\phi(\_, \_) \stackrel{\text{def}}{=} \emptyset$. This is graphically depicted as follows (male individuals are represented by squares, female individuals by circles and families with parental relation by lines).



The founders of the pedigree are the individuals x, u and y. The pedigree has 3 generations and 2 children per individual. □

For each individual $u \in M \cup F$ we define a *community* $C(u)$ as a collection of all families where u is a parent, i.e., $C(u) \stackrel{\text{def}}{=} \{f \in \mathcal{F}(\phi) \mid u \in p(f)\}$. *Maximal community size* is the largest cardinality over all communities in the pedigree, i.e., $\max_{u \in M \cup F} |C(u)|$.

Remark 1. In any pedigree, the number of children per individual is at least the maximal community size.

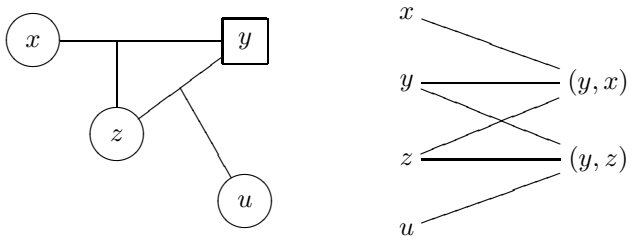
We shall now introduce formal definitions of a mating graph and (non-)looping pedigrees. The following definitions are equivalent to the ones in [11] and [1].

Definition 2 (Mating Graph and Connected Pedigrees). *Let $P = (M, F, \phi)$ be a pedigree. We define an undirected bipartite graph $G(P) \stackrel{\text{def}}{=} (M \cup F, \mathcal{F}(\phi), \leftrightarrow)$, also called the* mating graph *of P, by stating that for all $u \in M \cup F$ and for all $f \in \mathcal{F}(\phi)$ we have $\{u, f\} \in \leftrightarrow$ iff $u \in p(f)$, or $u \in \phi(f)$. We say that a pedigree P is* connected *iff $G(P)$ is a connected graph.*

From now on we shall consider only non-empty and connected pedigrees. All results presented in the paper can be extended also to unconnected pedigrees in a straightforward manner.

Definition 3 ((Non-)Looping Pedigree). *We say that a pedigree P is* looping *if there is a loop in the mating graph $G(P)$. Otherwise we call P a* non-looping *pedigree.*

Example 2. Consider the following pedigree (on the left) where $\phi(y, x) = \{z\}$ and $\phi(y, z) = \{u\}$.



Its mating graph (on the right) contains a loop $y, (y, x), z, (y, z), y$. This is an example of a so-called *inbreeding* loop. Another type of loop is demonstrated in the proof of Theorem 3 and it is called a *marriage* loop. □

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. Let $\mathcal{A}$ be a finite and non-empty set of *alleles*. A particular genotype information associated to every individual is represented by an element from $\mathcal{A}^2$ modulo the least equivalence on $\mathcal{A}^2$ satisfying $xy \equiv yx$ (the order of alleles in a genotype does not play any role).

Definition 4 ((Partial) Genotype Information). *Let $P = (M, F, \phi)$ be a pedigree. A* (partial) genotype information *for P is a function $\mathcal{G} : M \cup F \longrightarrow 2^{\mathcal{A}^2}$ that associates a set of possible genotype data to the individuals in the pedigree, s.t. $\mathcal{G}(u) \neq \emptyset$ for all $u \in M \cup F$.*

The intuition is that $\mathcal{G}(u) = \{AB\}$ means that the genotype of the individual u is known to be exactly AB. If e.g. $\mathcal{G}(u) = \{AB, AC\}$ then we have only a partial information about the individual u (we know that its genotype can be either AB or AC). If only one allele (let us say A) from the pair is know then we model it by $\mathcal{G}(u) = \{AX \mid X \in \mathcal{A}\}$. In case that nothing is known about the genotype of u then $\mathcal{G}(u) = \mathcal{A}^2$.

Definition 5 (Specialization). *Let $\mathcal{G}$ and $\mathcal{G}'$ be two partial genotype information. We say that $\mathcal{G}$* specializes into *$\mathcal{G}'$ iff $\mathcal{G}'(u) \subseteq \mathcal{G}(u)$ for all $u \in M \cup F$.*

Definition 6 (Complete Genotype Information). *A genotype information $\mathcal{G}$ is called* complete *if $|\mathcal{G}(u)| = 1$ for every $u \in M \cup F$, i.e., every individual is assigned exactly one genotype.*

Verifying the 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 e.g. [6]). The law of segregation implicitly defines the following constraint on consistent genotype assignments:

Each individual inherits exactly one allele from both 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. Given two genotypes AB and CD, we define $zygote(AB, CD) \stackrel{\text{def}}{=} \{AC, AD, BC, BD\}$ as the set of all possible combinations of the given genotypes. Note that due to the commutativity introduced on $\mathcal{A}^2$ we get that e.g. $zygote(AB, AB) = \{AA, AB, BB\}$ and $zygote(AA, AB) = \{AA, AB\}$.

Definition 7 (Consistent Genotype Information).

1. *A complete genotype information $\mathcal{G}$ is* consistent *if for all $(u, v) \in \mathcal{F}(\phi)$ such that $\mathcal{G}(u) = \{AB\}$, $\mathcal{G}(v) = \{CD\}$ for some $A, B, C, D \in \mathcal{A}$ and for all $w \in \phi(u, v)$ it is the case that $\mathcal{G}(w) \subseteq zygote(AB, CD)$.*
2. *A partial genotype information $\mathcal{G}$ is* consistent *if there is a complete consistent genotype information $\mathcal{G}'$ such that $\mathcal{G}$ specializes into $\mathcal{G}'$.*

Let P be a pedigree and $\mathcal{G}$ a genotype information for P. By $\#(P, \mathcal{G})$ we denote the number of complete and consistent genotype information into which $\mathcal{G}$ specializes (or simply the number of solutions). A natural algorithmic problem is that of computing $\#(P, \mathcal{G})$ and we call it the *counting problem* for a pedigree P and a genotype information $\mathcal{G}$.

2.2 Counting Problems and Complexity Classes FP and #P

The complexity classes FP and #P for counting problems play a similar role as the complexity classes P and NP in case of decision problems. Let R be a polynomially balanced ($(x, y) \in R$ implies $|y| \leq |x|^k$ for some constant $k > 0$) and polynomial time decidable binary relation [12]. The counting problem #R is for a given input x to count how many different y there are such that $(x, y) \in R$. To provide the answer to such a problem is generally considered as a hard computational task. #P is the class of all such problems. Alternatively, #P can be defined as the class of functions that can be computed by counting the number of accepting paths of a polynomial time nondeterministic Turing machine. On the other hand, the complexity class FP is the class of functions that are computable by a deterministic Turing machine in polynomial time (also called the class of feasible functions, i.e., those solvable by computers). We can easily notice that FP $\subseteq$ #P and it is widely conjectured that the inclusion is strict.

In order to show #P-hardness of a counting problem, we often use the notion of *parsimonious reduction*. Let #R and #S be two counting problems. We say that there is a parsimonious reduction from #R to #S if there is a polynomial time transformation f from the instances of #R to the instances of #S which preserves the number of solutions, i.e., for all x we have that $|\{y \mid (x, y) \in R\}| = |\{y \mid (f(x), y) \in S\}|$.

Remark 2. This notion of reduction is little too restrictive so sometimes one defines that #R reduces to #S if there is a polynomial time algorithm for #R given an oracle that solves #S. Nevertheless, all the reductions presented in this paper are parsimonious.

As noted in [12] p. 439: "Even in cases in which the decision problem is polynomial, counting the solutions may be highly nontrivial." An example of such a problem is e.g. counting the number of perfect matchings in a bipartite graph. This is a #P-complete problem, while the decision variant of the problem is in P. On the other hand, showing that a counting problem is in FP immediately gives a polynomial time algorithm for the corresponding decision problem. Hence proving that the counting problem for a certain subclass of pedigrees (e.g. non-looping pedigrees) is in FP provides a stronger claim than only showing that the decision problem of consistency checking is in P.

3 Pedigrees with the Counting Problem in FP

In this section we demonstrate that for an arbitrary non-looping pedigree P and a given genotype information $\mathcal{G}$, the number $\#(P, \mathcal{G})$ can be computed in polynomial time on a deterministic Turing machine. Hence we generalize the result by Lange and Goradia [8] where they showed that the decision version of the problem is solvable in polynomial time using a genotype elimination algorithm. In our approach, we provide a different solution which exploits dynamic

programming and enables us to count (and list if necessary) the total number of complete and consistent genotype information. We also show how to count in polynomial time the number of consistent pedigree assignments for pedigrees with 2 generations and maximal community size 2, and for pedigrees with at most one child per individual.

Let us consider a pedigree $P = (M, F, \phi)$ with a partial genotype information $\mathcal{G}$ over the alleles from $\mathcal{A}$. When counting $\#(P, \mathcal{G})$ we will use the techniques of dynamic programming and store the intermediate results in the following table.

$$T : (M \cup F) \times \mathcal{A}^2 \to \mathbb{N}$$

We shall often denote a table element $T(u, XY)$ where $u \in M \cup F$ and $XY \in \mathcal{A}^2$ by $T^u(XY)$. The main idea of the algorithm is that we shall process all families (and the corresponding individuals) of the pedigree in a particular order such that the number assigned to the table position $T^u(XY)$ stands for the number of solutions in a subpedigree that was already processed and is connected to u, all under the assumption that the genotype information of u is fixed to XY.

The procedure **initialize** in Figure 1 initializes the table T for all individuals from $M \cup F$.

Let $(x, y) \in \mathcal{F}(\phi)$, $u \in M \cup F$ s.t. $u \leftrightarrow (x, y)$ and $XY \in \mathcal{G}(u)$. The function **update** in Figure 1 returns the number of solutions in the already processed subpedigree connected (in the mating graph) to the individual u, under the assumption that the genotype information for u is fixed to XY and that the table T^v is fully computed for all the individuals v connected to the family (x, y) except for u.

Finally, the function **count** in Figure 1 computes the number $\#(P, \mathcal{G})$ where the notion of a mediator for $f \in \mathcal{F}(\phi)$ and $Z \subseteq \mathcal{F}(\phi)$ is defined as follows: an individual $u \in M \cup F$ is a *mediator* for f w.r.t. Z iff $u \leftrightarrow f$ and there is some $f' \in Z$ such that $f \neq f'$ and $u \leftrightarrow f'$. In other words a mediator is an individual that connects two different families in the mating graph. The function **count** first initializes the table T to its initial values and creates a set Z, which represents the set of families to be processed. It then removes the families from Z one by one in a particular order which ensures that the table T^u for a mediator u can be easily computed. Finally, when Z contains only one family, the final number of solutions is computed and returned.

Theorem 1. *The counting problem for non-looping pedigrees is in FP.*

Proof. (Sketch) It is easy to see that the algorithm runs in polynomial time (the return sums are for clarity reasons written in their unfolded forms which are not necessarily polynomial but they can be computed by first making sums for each child and then multiplying the results together). We have to argue that for a given pedigree P and a genotype information $\mathcal{G}$ the function **count**$(P, \mathcal{G})$ returns the number $\#(P, \mathcal{G})$. The requirement that P is non-looping (and connected) ensures that we can always select a family $f \in Z$ with exactly one mediator u with respect to Z. Moreover, whenever a value is assigned to $T^u(XY)$ the following assertion holds: "$T^u(XY)$ is the number of solutions in the subpedigree

initialize =
for all $v \in M \cup F$ **do**
  for all $XY \in \mathcal{A}^2$ **do**
    $T^v(XY) := \begin{cases} 1 & \text{if } XY \in \mathcal{G}(v) \\ 0 & \text{otherwise} \end{cases}$
  end for
end for

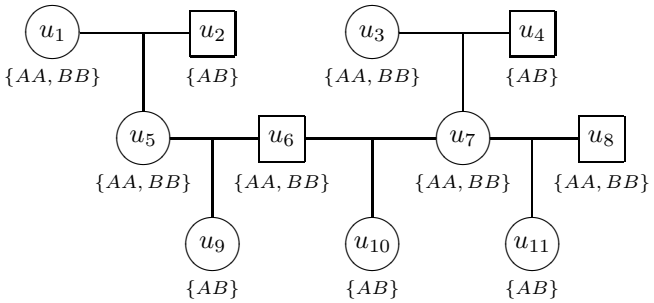
update$((x,y),\, u,\, XY)$:int =
if $u = x$ **then**
  *** u is a male parent in the family ***
  let $\{w_1, \ldots, w_\ell\} = \phi(x,y)$ be all children in the family (x,y)
  return $\sum_{\substack{AB \in \mathcal{G}(y) \\ X_1Y_1,\ \ldots,\ X_\ell Y_\ell \in zygote(XY, AB)}} T^y(AB) \cdot T^{w_1}(X_1Y_1) \cdot \cdots \cdot T^{w_\ell}(X_\ell Y_\ell)$
else if $u = y$ **then**
  *** u is a female parent in the family ***
  let $\{w_1, \ldots, w_\ell\} = \phi(x,y)$ be all children in the family (x,y)
  return $\sum_{\substack{AB \in \mathcal{G}(x) \\ X_1Y_1,\ \ldots,\ X_\ell Y_\ell \in zygote(AB, XY)}} T^x(AB) \cdot T^{w_1}(X_1Y_1) \cdot \cdots \cdot T^{w_\ell}(X_\ell Y_\ell)$
else
  *** $u \in \phi(x,y)$ is a child in the family ***
  let $\{w_1, \ldots, w_\ell\} = \phi(x,y) \setminus \{u\}$ be all the children except for u
  return $\sum_{\substack{AB \in \mathcal{G}(x) \text{ and } CD \in \mathcal{G}(y) \\ \text{s.t. } XY \in zygote(AB, CD) \\ X_1Y_1,\ \ldots,\ X_\ell Y_\ell \in zygote(AB, CD)}} T^x(AB) \cdot T^y(CD) \cdot T^{w_1}(X_1Y_1) \cdot \cdots \cdot T^{w_\ell}(X_\ell Y_\ell)$
end if

count$(P = (M, F, \phi),\ \mathcal{G} : M \cup F \to 2^{\mathcal{A}^2})$:int =
$Z := \mathcal{F}(\phi)$
initialize
while $|Z| > 1$ **do**
  select $f \in Z$ s.t. f has exactly one mediator u w.r.t. Z
  for all $XY \in \mathcal{G}(u)$ **do**
    $T^u(XY) := T^u(XY) \cdot$**update**$(f,\, u,\, XY)$
  end for
  $Z := Z \setminus \{f\}$
end while
let $f = (x,y)$ where $\{f\} = Z$;
let $\{w_1, \ldots, w_\ell\} = \phi(x,y)$ be all children in the family (x,y)
return $\sum_{\substack{AB \in \mathcal{G}(x) \text{ and } CD \in \mathcal{G}(y) \\ X_1Y_1,\ \ldots,\ X_\ell Y_\ell \in zygote(AB, CD)}} T^x(AB) \cdot T^y(CD) \cdot T^{w_1}(X_1Y_1) \cdot \cdots \cdot T^{w_\ell}(X_\ell Y_\ell)$

Fig. 1. Algorithm for computing the number $\#(P, \mathcal{G})$

generated by u and the families in $\mathcal{F}(\phi) \setminus Z$ (together with their children) that are connected to u in the mating graph". □

Example 3. We shall demonstrate the algorithm for counting the number of solutions on the following pedigree.



The pedigree consists of 11 individuals $u_1, \ldots, u_{11}$ and 5 families (u_2, u_1), (u_4, u_3), (u_6, u_5), (u_6, u_7) and (u_8, u_7). The genotype information for each individual is depicted in the picture (e.g. $\mathcal{G}(u_5) = \{AA, BB\}$ and $\mathcal{G}(u_2) = \{AB\}$).

During the call of the function **count** we first for all individuals initialize the table T to either 0 or 1 according to the given genotype information. Next we start eliminating the families in the pedigree. Let us assume that the first selected family in the while-loop is (u_2, u_1) and the mediator u is equal to u_5. As u_5 is a child in the family (u_2, u_1) we compute $T^{u_5}(AA) = 1 \cdot 1 = 1$ and $T^{u_5}(BB) = 1 \cdot 1 = 1$ according to the third case in the function **update**. The family (u_2, u_1) is then removed from Z. Next assume that the second selected family is (u_6, u_5) with a mediator u_6 and we compute $T^{u_6}(AA) = T^{u_6}(BB) = 1 \cdot 1 = 1$ by using the first case in the function **update**. The family (u_6, u_5) is removed from Z. Assume that the third selected family is (u_4, u_3) where u_7 is the mediator and we compute $T^{u_7}(AA) = T^{u_7}(BB) = 1 \cdot 1 = 1$. The family (u_4, u_3) is removed from Z. Now we can select e.g. the family (u_8, u_7) with the mediator u_7 and compute $T^{u_7}(AA) = T^{u_7}(BB) = 1 \cdot 1 = 1$ while removing (u_8, u_7) from Z. Finally, only the family (u_6, u_7) remains in Z and we return the final value

$$T^{u_6}(AA) \cdot T^{u_7}(BB) \cdot T^{u_{10}}(AB) + T^{u_6}(BB) \cdot T^{u_7}(AA) \cdot T^{u_{10}}(AB) = 1 + 1 = 2.$$

Indeed, there are exactly two possibilities for the assignment of a genotype to u_1 and this uniquely determines the assignments in the rest of the pedigree. In particular, one can see that u_3 has to be assigned exactly the same genotype as u_1 in order to preserve consistency. □

Theorem 2. *If a pedigree $P = (M, F, \phi)$ has at most one child per individual then it is non-looping.*

Proof. By contradiction assume that there is a loop in the mating graph $G(P)$, i.e., there are individuals $u_1, \ldots, u_n \in M \cup F$ (for some $n \geq 2$) and families

$f_1, \ldots, f_n \in \mathcal{F}(\phi)$ such that $f_{i-1} \leftrightarrow u_i \leftrightarrow f_i$ for all i, $2 \leq i \leq n$, and $f_n \leftrightarrow u_1 \leftrightarrow f_1$, graphically:

$$u_1 \leftrightarrow f_1 \leftrightarrow u_2 \leftrightarrow f_2 \leftrightarrow u_3 \leftrightarrow f_3 \leftrightarrow \cdots u_n \leftrightarrow f_n \leftrightarrow u_1.$$

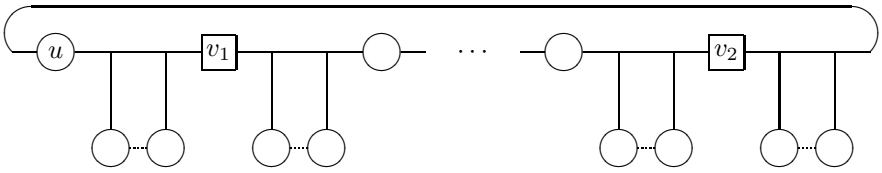
Without loss of generality we may assume that all u_i and f_i are distinct. Let us consider the individual u_1 and the connected families f_1 and f_n. As u_1 has at most one child and f_1 and f_n are distinct, we know that either (i) $u_1 \in p(f_n) \cap \phi(f_1)$ or (ii) $u_1 \in \phi(f_n) \cap p(f_1)$. We analyze only case (i); case (ii) is symmetric. Since $u_1 \in \phi(f_1)$ then necessarily $u_2 \in p(f_1)$, otherwise both parents in f_1 would have two distinct children u_1 and u_2. Similarly, since $u_2 \in p(f_1)$ then $u_2 \in \phi(f_2)$, otherwise u_2 would have two distinct children. By induction we can easily establish that $u_i \in \phi(f_i) \cap p(f_{i-1})$ for all i, $2 \leq i \leq n$. This implies that $u_1 \prec u_2 \prec \cdots \prec u_n \prec u_1$, which is a contradiction with condition 2. of Definition 1. □

Corollary 1. *The counting problem for pedigrees with at most one child per individual is in FP.*

Let us now consider pedigrees with 2 generations only and the maximal community size at most 2. We shall demonstrate that the counting problem for this subclass is also in FP by applying the observations about "loop-breakers" from [5].

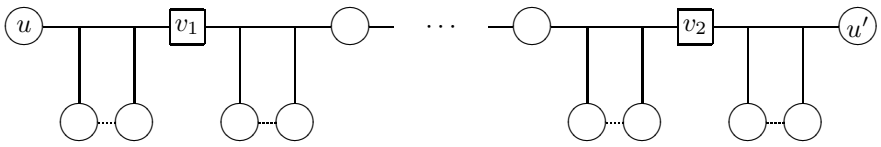
Theorem 3. *The counting problem for pedigrees with* 2 *generations and maximal community size* 2 *(and an arbitrary number of alleles) is in FP.*

Proof. Let $P = (M, F, \phi)$ be a pedigree satisfying the requirements of the theorem. As mentioned in Section 2, we assume that P is connected. Either, there is no loop in P and then we can apply directly Theorem 1. Otherwise, we claim that there is exactly one loop and the pedigree looks like in the following picture (without loss of generality all children are assumed to be females; every family can have an arbitrary number of children).



This is because every connected (undirected) graph with all nodes of degree at most 2 either contains no loop (if two of its nodes have degree 1) or exactly one loop with all the nodes on the loop (application of the walk-technique from the Eulerian path algorithm [3]).

We transform the pedigree P into a non-looping $P' = (M, F \cup \{u'\}, \phi')$ by adding a new female individual u' such that $\phi'(x, y) \stackrel{\text{def}}{=} \phi(x, y)$ for all $x \in M$ and $y \in F \setminus \{u, u'\}$, $\phi'(v_1, u) \stackrel{\text{def}}{=} \phi(v_1, u)$ and $\phi'(v_2, u') \stackrel{\text{def}}{=} \phi(v_2, u)$. The construction is depicted in the following picture.



Let $\mathcal{G}$ be a given genotype information. The following algorithm counts the number of complete consistent assignments in P into which $\mathcal{G}$ specializes in polynomial time (note that because P' in non-looping the number $\#(P', \mathcal{G}')$ can be computed in polynomial time due to Theorem 1).

count(P, $\mathcal{G}$):int =
sum := 0
for all $XY \in \mathcal{G}(u)$ **do**
  let $\mathcal{G}'(u) \stackrel{\text{def}}{=} \mathcal{G}'(u') \stackrel{\text{def}}{=} \{XY\}$ and
  $\mathcal{G}'(v) \stackrel{\text{def}}{=} \mathcal{G}(v)$ for all $v \in (M \cup F) \setminus \{u, u'\}$
  sum := sum + $\#(P', \mathcal{G}')$
end for
return sum

□

Corollary 2. *The counting problem for pedigrees with* 2 *generations and at most two children per individual (and an arbitrary number of alleles) is in FP.*

Proof. Directly from Theorem 3 and Remark 1. □

4 Pedigrees with #P-Complete Counting Problem

In this section we shall argue that the counting problems in all other pedigrees except for those considered in Section 3 are computationally hard. We will demonstrate #P-hardness (with respect to parsimonious reduction) for pedigrees with 3 generations, 2 children per individual and 2 alleles, and for pedigrees with 2 generations, 3 children per individual and 2 alleles. Even for general pedigrees one can easily see that the problems are in #P, which together with the hardness results implies #P-completeness. This completes the full picture of the computational complexity of counting problems for pedigrees.

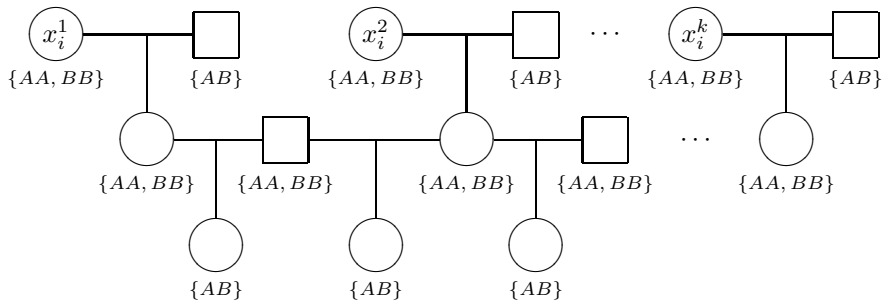
Let us consider the #Bpos-2Sat counting problem [4]. We are given two disjoint sets of variables $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_m\}$ and a formula $C_1 \wedge C_2 \wedge \ldots \wedge C_k$ where for every ℓ, $1 \leq \ell \leq k$, the clause C_ℓ is of the form $x_i \vee y_j$ such that $1 \leq i \leq n$ and $1 \leq j \leq m$. Counting the number of satisfying truth assignments of such a formula is a #P-complete problem [4]. Note that the corresponding decision problem is trivial as any #Bpos-2Sat formula is satisfiable.

We shall reduce #Bpos-2Sat to the counting problem for pedigrees of particular shapes such that the reduction preserves the number of solutions (i.e. it is a parsimonious reduction). In our reductions we shall use only two alleles $\mathcal{A} \stackrel{\text{def}}{=} \{A, B\}$ such that AA represents *true* and BB represents *false*.

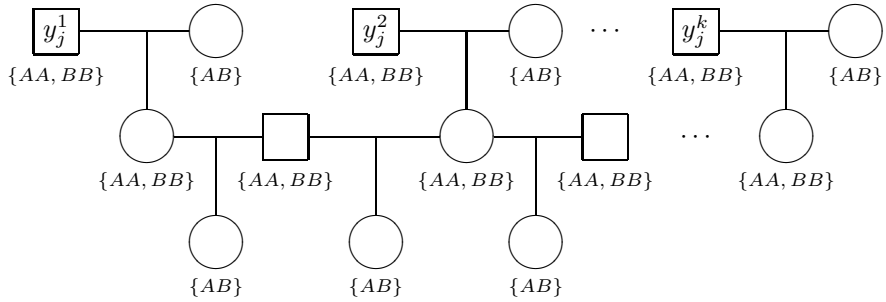
Theorem 4. *The counting problem for pedigrees with* 3 *generations,* 2 *children per individual and* 2 *alleles is #P-complete.*

Proof. (Sketch) Containment in #P is easy. We shall argue for #P-hardness of the problem. Let $C_1 \wedge C_2 \wedge \ldots \wedge C_k$ be a given instance of #Bpos-2Sat. We shall construct a pedigree P as follows.

First, for every variable x_i, $1 \leq i \leq n$, we create k of its copies by constructing the following pedigree part.

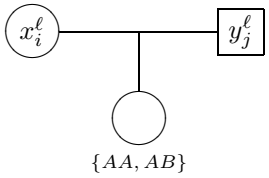


Next we create a similar structure for every variable y_j, $1 \leq j \leq m$.



As argued in Example 3, we can now see that the pedigrees for x_i (resp. y_j) have exactly two solutions so that $x_i^1, \ldots, x_i^k$ (resp. $y_j^1, \ldots, y_j^k$) can simultaneously take the assignment AA or BB, hence representing the truth value true or false.

Now for every ℓ, $1 \leq \ell \leq k$, such that $C_\ell \equiv x_i \vee y_j$ we will add the following pedigree part where the individuals x_i^ℓ and y_j^ℓ are identified with the corresponding nodes in the pedigrees above.

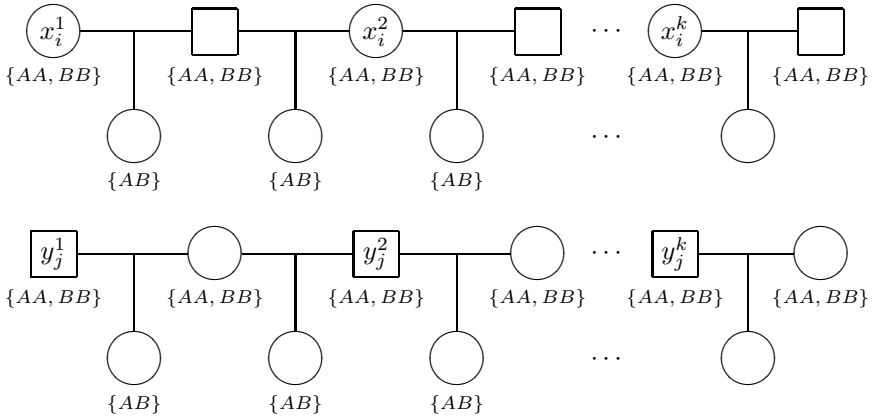


We can now easily verify that at least one of x_i^ℓ and y_j^ℓ has to be set to true (i.e. takes the value AA) in order to achieve a consistent assignment. It is a routine

exercise to check that the number of satisfying truth assignments of the formula is the same as the number of complete genotype information for the constructed pedigree. The construction ensures that the pedigree has 3 generations, at most 2 children per individual and uses only 2 alleles. □

Theorem 5. *The counting problem for pedigrees with* 2 *generations,* 3 *children per individual and* 2 *alleles is #P-complete.*

Proof. (Sketch) As in the previous proof, the reduction goes from #Bpos-2Sat. We modify the way in which we generate the truth assignments. We shall use only 2 generation pedigrees at the expense of 3 children per individual.



It is easy to see that $x_i^1, \ldots, x_i^k$ (resp. $y_j^1, \ldots, y_j^k$) can simultaneously take the assignment AA or BB and after adding the pedigree parts for all clauses C_ℓ as in the construction above, the reduction preserves the number of solutions. The final pedigree has 2 generations and at most 3 children per individual. As before, the containment in #P is easy. □

5 Conclusion

We have studied counting problems for genotype assignments in pedigrees and found a delicate borderline between tractable and intractable instances of the problem. The following table summarizes the main results achieved in the paper.

type	# generations	# children	# alleles	complexity
non-looping	∞	∞	∞	in FP (Thm. 1)
looping	∞	∞	1	in FP (trivial)
looping	1	∞	∞	in FP (trivial)
looping	∞	1	∞	in FP (Cor. 1)
looping	2	2	∞	in FP (Cor. 2)
looping	3	2	2	#P-complete (Thm. 4)
looping	2	3	2	#P-complete (Thm. 5)

The table provides a complete characterization of the computational complexity of counting problems with respect to the selected parameters. Moreover, the hardness results use only marriage loops [13] and contain no inbreeding loops.

In [1] it is shown that consistency checking for general pedigrees with 2 alleles only is in P, provided that for each individual we either know precisely his/her genotype information or we know nothing at all, i.e., for every individual u we have either $|\mathcal{G}(u)| = 1$ or $\mathcal{G}(u) = \mathcal{A}^2$. It would be interesting to see whether the counting problem is in FP under this restriction. The future research will also focus on investigating possible ways of tackling the counting problem for pedigrees with loops (here some of the ideas from [11] seem to be applicable) and on the complete complexity characterization of the consistency checking problems.

Acknowledgments. I would like to thank Luca Aceto and Anna Ingólfsdóttir for introducing me into the area and for drawing my attention to the studied problem.

References

1. Luca Aceto, Jens A. Hansen, Anna Ingólfsdóttir, Jacob Johnsen, and John Knudsen. The complexity of checking consistency of pedigree information and related problems. *Journal of Computer Science and Technology*, 19(1):42–59, 2004.
2. F.-X. Dua and I. Hoeschele. A note on algorithms for genotype and allele elimination in complex pedigrees with incomplete genotype data. *Genetics*, 156:2051–2062, 2000.
3. H. Fleischner. Eulerian graphs and related topics. *Annals of Discrete Mathematics*, 45(1), 1990.
4. M.O. Ball J.S. Provan. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.
5. K. Lange K and R.C. Elston. Extensions to pedigree analysis i. likehood calculations for simple and complex pedigrees. *Human Heredity*, 25(2):95–105, 1975.
6. William S. Klug and Michael R. Cummings. *Concepts of Genetics*. Prentice Hall, 5th edition, 1997.
7. K. Lange and T. Goradia. An algorithm for automatic genotype elimination. *American Journal of Human Genetics*, 40:250–256, 1987.
8. Kenneth Lange and Tushar Madhu Goradia. An algorithm for automatic genotype elimination. *American Journal of Human Genetics*, 40:250–256, 1987.
9. Jing Li and Tao Jiang. Efficient rule-based haplotyping algorithms for pedigree data [extended abstract]. In *Proceedings of RECOMB'03, April 10–13, 2003, Berlin, Germany*, pages 197–206. ACM, 2003.
10. Jeffrey R. O'Connell and Daniel E. Weeks. Pedcheck: A program for identification of genotype incompatibilities in linkage analysis. *American Journal of Human Genetics*, 63:259–266, 1998.
11. Jeffrey R. O'Connell and Daniel E. Weeks. An optimal algorithm for automatic genotype elimination. *American Journal of Human Genetics*, 65:1733–1740, 1999.
12. Ch.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, New York, 1994.

13. Antonio Piccolboni and Dan Gusfield. On the complexity of fundamental computational problems in pedigree analysis. *Journal of Computational Biology*, 10(5):763–773, 2003.

Recent BRICS Report Series Publications

RS-05-28 **Jiří Srba.** ***On Counting the Number of Consistent Genotype Assignments for Pedigrees*****. September 2005. 15 pp. To appear in FSTTCS '05.**

RS-05-27 **Pascal Zimmer.** ***A Calculus for Context-Awareness*****. August 2005. 21 pp.**

RS-05-26 **Henning Korsholm Rohde.** ***Measuring the Propagation of Information in Partial Evaluation*****. August 2005. 39 pp.**

RS-05-25 **Dariusz Biernacki and Olivier Danvy.** ***A Simple Proof of a Folklore Theorem about Delimited Control*****. August 2005. ii+11 pp. To appear in** ***Journal of Functional Programming*****. This version supersedes BRICS RS-05-10.**

RS-05-24 **Małgorzata Biernacka, Dariusz Biernacki, and Olivier Danvy.** ***An Operational Foundation for Delimited Continuations in the CPS Hierarchy*****. August 2005. iv+43 pp. To appear in the journal** ***Logical Methods in Computer Science*****. This version supersedes BRICS RS-05-11.**

RS-05-23 **Karl Krukow, Mogens Nielsen, and Vladimiro Sassone.** ***A Framework for Concrete Reputation-Systems*****. July 2005. 48 pp. This is an extended version of a paper to be presented at ACM CCS'05.**

RS-05-22 **Małgorzata Biernacka and Olivier Danvy.** ***A Syntactic Correspondence between Context-Sensitive Calculi and Abstract Machines*****. July 2005. iv+39 pp.**

RS-05-21 **Philipp Gerhardy and Ulrich Kohlenbach.** ***General Logical Metatheorems for Functional Analysis*****. July 2005. 65 pp.**

RS-05-20 **Ivan B. Damgård, Serge Fehr, Louis Salvail, and Christian Schaffner.** ***Cryptography in the Bounded Quantum Storage Model*****. July 2005. 23 pp.**

RS-05-19 **Luca Aceto, Willem Jan Fokkink, Anna Ingólfsdóttir, and Bas qLuttik.** ***Finite Equational Bases in Process Algebra: Results and Open Questions*****. June 2005. 28 pp.**

RS-05-18 **Peter Bogetoft, Ivan B. Damgård, Thomas Jakobsen, Kurt Nielsen, Jakob Pagter, and Tomas Toft.** ***Secure Computing, Economy, and Trust: A Generic Solution for Secure Auctions with Real-World Applications*****. June 2005. 37 pp.**