YALE UNIVERSITY
Department of Computer Science

ROOMMATE STABILITY LEADS TO MARRIAGE: THE STRUCTURE OF THE STABLE ROOMMATE PROBLEM

Dan Gusfield

YALEU/DCS/TR-435
November 1985

Table of Contents

Roommate Stability Leads to Marriage:

The Structure of the Stable Roommate Problem

Dan Gusfield

Department of Computer Science, Yale University

Abstract

The stable roommate problem is a generalization of the well-known stable marriage problem. In the marriage problem n *men* and n *women* are matched into n man-woman pairs to achieve a certain type of "stability", while in the roommate assignment problem, individual gender is unspecified, and any two people may form a pair. It is well known that every instance of the stable marriage problem has a stable matching, and one can be found quickly, but in the roommate case, there are instances of the problem which permit no stable assignments. An instance of the stable roommate problem is called "solvable" if there is at least one stable assignment.

In this paper we show that every solvable instance of the stable roommate problem contains an embedded instance of the stable marriage problem which completely captures the set and structure of the stable roommate assignments. In particular, the 2n people in a solvable roommate instance can be partitioned into two equal sized sets called, say, men and women, and a set of preference lists (men ranking women, and women ranking men) can be extracted from the roommate lists, so that the set of stable assignments in the roommate problem is the same as the set of stable marriages in the derived stable marriage problem. In addition to proving the existence of the above partition, we give an $O(n^4)$ algorithm to explicitly obtain it and the reduction.

The theoretical and algorithmic consequences of this partition, and algorithm to find it, are manifest. First, the structure of the set of all stable assignments for a given solvable roommate instance is exactly that of the stable marriage problem: the assignments form a finite distributive lattice under the relation of dominance (male or female). Second, numerous derivative roommate problems which generalize solved stable marriage problems can be efficiently solved. For, after the reduction to stable marriage, the set of all stable roommate assignments for a given solvable instance can be enumerated in O(n) time per assignment, as in [G]. As another example, if p(A,i) denotes the position in i's list of i's mate in assignment A, then the problem of finding the optimal stable roommate assignment A to minimize $\sum_i p(A,i)$ can be solved in $O(n^4)$ time, by reducing the roommate instance to a marriage instance, and then solving the optimal stable marriage problem in $O(n^4)$ time, as in [ILG]. Third, there are many interesting, even surprising, theorems about stable marriage and derivative problems, which can now be generalized to the stable roommate problem and proved via the reduction to stable marriage. Finally, the partition theorem has immediate application in a game theoretic model of market exachanges [Q,R].

It is tempting to assert social implications for the partition result. We leave such speculation for others.

1. Introduction

The stable *roommates* problem is a well known problem of matching 2n people into disjoint pairs to achieve a certain type of stability. The input to the problem is a set of 2n preference lists, one for each person i, where person i's list is a rank ordering (most preferred first) of the 2n-1 people other than i. A roommate assignment A is a pairing of the 2n people into n disjoint pairs. Assignment A is said to be *unstable* if there are two people who are not paired together in A, but who each prefer the other to their respective mates in A; such a pair is said to *block* assignment A. An assignment which is not unstable is called *stable*. An instance of the stable roommates problem is called *solvable* if there is at least one stable assignment. It is known [GS, L, K, PTW] that there are unsolvable instances of the stable roommate problem; the problem of finding an efficient algorithm to determine if an instance is solvable was proposed by Knuth [K] and only recently solved by R. Irving [I].

The stable roommates problem is closely related to and is a generalization of another well-known problem, the stable *marriage* problem. In the stable marriage problem, the 2n people consist of n men and n women, and each pair is constrained to consist of a man and a women. Each man ranks only the women and each woman ranks only the men, and an assignment in this problem is called a *marriage* (from here on, the word "assignment" will be used only for roommate assignment). A marriage M is unstable if there is a man and a woman who are not married to each other in M, but who mutually prefer each other to their respective mates in M. In contrast to the stable roommates problem, it is well known [GS] that every instance of the stable marriage problem is solvable, i.e. has at least one stable marriage.

It is easy to reduce the stable marriage problem to the stable roommates problem. In this paper we consider the opposite direction, proving a very surprising result. We show that any solvable instance of the stable roommate problem reduces to an embedded instance of the stable marriage problem. In particular, the 2n people in a solvable roommate instance can be partitioned into two equal sized sets, called men and women, and a set of preference lists for a marriage problem between the men and women can be obtained, so that the set of stable assignments in the roommate problem is the same as the set of stable marriages in the derived stable marriage problem. In fact, the following simple rule produces such an instance of the stable marriage problem: for each pair of people (i,j), delete i from j's roommate list, and j from i's roommate list, if and only if (i,j) is a pair in no stable roommate assignment; given the partition, the resulting set of lists defines the desired stable marriage instance.

In addition to proving the existence of the above partition, we give an $O(n^4)$ algorithm to explicitly obtain the partition and to identify those pairs which are in no stable assignments. Hence in $O(n^4)$ time we can reduce a solvable instance of the stable roommates problem to an instance of the stable marriage problem with the same number of people, or fewer. The

theoretical and algorithmic consequences of the existence of this partition, and algorithm to find it, are manifest. First, the structure of the set of all stable assignments for a given solvable instance is exactly that of the stable marriage problem, i.e. the assignments form a finite distributive lattice under the relation of dominance [K]. Second, numerous derivative roommate problems which generalize solved stable marriage problems can be defined and efficiently solved. For example, the problem of enumerating all the stable roommate assignments for a given instance can be solved in $O(n^4)$ time (for the reduction to stable marriage) plus $O(n)$ time, per marriage, to enumerate each stable marriage (assignment), as in [G]. As another example, if $p(A,i)$ denotes the position in i's list of i's mate in assignment A, then the problem of finding the optimal roommate assignment A to minimize $\sum_i p(A,i)$ can be solved in $O(n^4)$ time by reducing the roommate instance to a marriage instance, and then solving the optimal stable marriage problem in $O(n^4)$ time, as in [ILG]. Third, the partition result has immediate applications in the game theoretic model of market exchange, as in [Q,R]. Finally, the reduction leads immediately to the formulation and proof of many theorems about roommates which generalize known results for the stable marriage problem and derivatives of it.

Figure 1 shows the three stable assignments in an eight person example, and shows a partition and an embedded stable marriage problem with the same three stable marriages.

The main result and algorithm in this paper are obtained by a close examination of Irving's algorithm [I] which finds a stable assignment if there is one, and else reports that no stable assignment exists. Hence we will begin by describing algorithm I.

2. Algorithm I and its execution tree D

2.1. Algorithm I

Algorithm I successively deletes entries from preference lists until either each person has only one entry on its list, or until someone has no entries. In the first case, the entries specify a stable roommate assignment, and in the second case, there are no stable assignments. The algorithm is divided into two phases. In phase one, entries are removed from lists, but no stable assignments are affected, i.e. if j is removed from i's list, then (i,j) is a pair in no stable assignment. In phase 2, the removed entries may affect stable assignments, but the invariant is maintained at each iteration, that *if* there is a stable assignment in the lists before the current iteration of removals, *then* there is a stable assignment in the lists resulting from the iteration of removals. Hence if any list becomes empty in either phase, there can be no stable assignment. Before describing the algorithm, we need the following definitions.

Definition: The current set of lists at any point in the algorithm is called a *table*.

Definition: Let e_i denote a person. At any point in the algorithm, h_i will denote the current head of person e_i's list, and s_i will denote the current second entry on e_i's list.

Definition: At any point in algorithm I, a person e_i is said to be ***semi-engaged*** to h_i if and only if e_i is the bottom entry in h_i's list. A person who is not semi-engaged is called ***free***. A person may alternate between being free and semi-engaged.

Note that semi-engagement is not a symmetric relation. However, if everyone is semi-engaged, then the set of list heads is a permutation of the 2n people. We now describe algorithm I.

Phase 1 of algorithm I iterates the following:

```
If there is an empty list, then terminate algorithm I; there is no
stable assignment.

Else, if everyone is semi-engaged, then go to phase 2.

Else, pick an arbitrary free person e_i, and execute the following
operations for each person k who is ranked below e_i on h_i's
list in T: remove k from h_i's list, and remove h_i from k's list.
```

Note that throughout phase 1, person i is on j's list if and only if j is on i's list. Hence in a step where e_i becomes semi-engaged to h_i, if there is a person p who is semi-engaged to h_i just before that step, then p is (automatically) not semi-engaged to h_i after that step; This follows since at the start of the step, p must be below e_i on h_i's list, so during that step, p is removed from h_i's list, and h_i is removed from p's list. After the step, p might be free, or it might have become (automatically) semi-engaged to the new head of its list.

The set of lists at the end of phase 1 is called the *phase 1 table*. It is proved in [I] that if j is missing from i's list in the phase 1 table, then there are no stable assignments which pair i to j. Hence if some list in the phase 1 table is empty, there are no stable assignments. Otherwise, when phase 1 terminates with everyone semi-engaged, j is the head of i's list if and only if i is the bottom of j's list, and so the set of head entries of the phase 1 table are a permutation of the 2n people.

Figure 2a shows the phase 1 table of the example from figure 1.

Phase 2

Throughout phase 2 all the people remain semi-engaged, although who they are semi-engaged to may change. Hence at any point in phase 2, j is the head of i's list if and only if i is the bottom of j's list. It will also be true that i is on j's list if and only if j is on i's list. Phase 2 starts with the phase 1 table and removes entries from lists in a way similar to phase 1, but the selection of lists is more constrained. We first need some definitions.

Definition: In a table T, an ***exposed rotation*** R is an ordered subset of people E = $\{e_1, e_2, \ldots, e_r\}$, such that $s_i = h_{i+1}$, for all i from 1 to r, where i+1 is taken modulo r. Note that since the order of E is cyclic, the actual selection of which element in E is named e_1 is arbitrary, but that selection determines the rest of the ordering.

Figure 2b shows two rotations that are exposed in the phase 1 table of the running example.

We will often write "R = (E,H,S)", where H is the set of head entries of E ordered to correspond to the order of E, and S is the set of second entries of E, with corresponding order. Note that, *as sets*, S = H, and that, as *ordered sets*, S is a (backwards) cyclic rotation of H; when that point is central, we will write $S = H^r$. We will sometimes say that "e is in R" to mean that e is the E set of R; we will also say that "(e,h) is a pair in R" to mean that $e = e_i$ and $h = h_i$ for some e_i in E.

Definition: If R = (E,H,S) is an exposed rotation in table T, then the *elimination* of R from T is the following operation: for every s_i in S, remove every entry below e_i in s_i's list in T, i.e. move the bottom of s_i's list up to e_i (from e_{i+1}). Then remove s_i from k's list, for each person k who was just removed from s_i's list.

Notice that if all people are semi-engaged in a table T before a rotation elimination, then all people are semi-engaged after that elimination; hence, one affect of the elimination is to move the head of e_i's list down one place, for each e_i in R, i.e. e_i becomes semi-engaged to the s_i of table T. Figure 2c shows the table resulting from eliminating R_1 from the phase 1 table.

Phase 2 of the algorithm is simply:

```
While some person has more than one entry on his list, and no list is empty,
find and eliminate a rotation.

If every person has exactly one entry on his list, then pairing each
person with their head entry specifies a stable assignment.

If there is an empty list, then there are no stable assignments.
```

Figure 2d completes the execution of phase 2, eliminating rotations R_3 and R_4.

2.2. Correctness of algorithm I

The correctness of algorithm I is proved in [I], and will not be fully repeated here. However, we need the statements of the central lemmas that prove correctness, and we need to extend some of them; we will give proofs of the extended lemmas.

Definition: If T is a table, then roommate assignment *A* is said to be ***contained in***, or ***in***, T, if and only if every pair in *A* is in T, i.e. i is on j's list, and j is on i's list for each pair (i,j) in *A*.

The following lemmas imply the correctness of algorithm I.

Lemma 2.1 [I]: If T is a table (in phase 2) where no list is empty, and at least one person has more than one entry, then there is a rotation exposed in T.

Lemma 2.1 will be proved and extended in the next section.

Lemma 2.2 [I]: Let R = (E,H,S) be an exposed rotation in T, and let A be any stable assignment contained in T. If $e_1 \in E$ and (e_1, h_1) is a pair in A, then (e_i, h_i) must also be a pair in A, for every e_i in E.

Lemma 2.2 will be proved and extended in the next section.

Lemma 2.3 [I]: If rotation R = (E,H,S) is exposed in T, and there exists a stable assignment in T where $e_1 \in E$ pairs with h_1, then there also exists a stable assignment in T where e_1 does not pair with h_1, and by lemma 2.2, no e_i pairs with h_i, for any e_i in E.

Lemma 2.4 [I]: If the algorithm ends with a single entry on each list, then pairing each person to that entry gives a stable assignment.

2.3. Extensions of the central lemmas

We will prove lemmas 2.1 and 2.2 in order to extend them.

Proof of lemma 2.1: Let e_i be a person who has at least two entries, h_i and s_i, on its list in T. Since the head entries are a permutation of the people, and $s_i \neq h_i$, there must be a person e_j such that $s_i = h_j$. We claim that e_j must have two or more entries on its list. If not, then h_j is its only entry, and so e_j is the only entry on h_j's list; To see this, note that h_j is both the head and bottom entry on e_j's list, so e_j must also be both the head and bottom of h_j's list. But, $h_j = s_i$ which is on e_i's list, so e_i (which can't be e_j) must also be on h_j's list, and so both h_j and e_j must have at least two entries on their lists. Repeating this argument, we must eventually cycle, in which case a rotation has been found. □

Definition: The proof above gives an (implicit) algorithm for finding a rotation R, starting from any person e who has at least two entries on its list in T. Let e_1 denote the person who is visited twice by the algorithm (i.e. where the cycle is detected). Every person who is visited before the first visit to e_1 is said to be on a ***tail*** of R, and the other people are in the ***body*** of R.

Note that in a given table, an exposed rotation may have many tails, and in a different table, the same exposed rotation may have different tails. This is illustrated in the example of figure 2. We will need the following extension of lemma 2.1.

Corollary 2.1: If e is a person with two or more entries on its list in table T, then e is either in a tail or in the body of a rotation exposed in T.

rotation which is exposed in T(x), and which is the next rotation eliminated from T(x) on that execution path out of x. We use D(x) to denote the subtree of D rooted at x.

Note that D is defined only for the the phase 2 executions. In the remainder of the paper, when we talk about algorithm I, we will be referring to phase 2, unless we specifically state otherwise. Before going on, it is useful to examine the execution tree D in a running example (see figure 3). Two initial observations stand out: first, every path in the tree has the same length, and second, many of the rotations seem to come in dual pairs as defined below.

Definition: If R = (E,H,S) is a rotation in D then we define R^d to be the triple (S,E,E^r), where S and E have the same order in R^d as they have in R. Note that with this definition $(R^d)^d$ = R. Note also that R^d has the *form* of a rotation; If R^d is actually a rotation in D (i.e. is a rotation exposed in some table), then we call R and R^d a ***dual pair*** of rotations. Any rotation without a dual is called a *singleton* rotation.

In the example, rotations R_1 and R_3 are singletons, and (R_4, R_5) and (R_2, R_6) are each a dual pair of rotations. With this terminology, we can make a more precise observation about the paths in the example tree D: Each path from the root to a leaf in D contains every singleton rotation, and exactly one of each pair of dual rotations. We will prove that this is true for any execution tree D of algorithm I, and this fact will then be exploited to reveal the structure of the set of stable roommate assignments. However, we first need several technical definitions and lemmas.

3. Basic Lemmas

In this section we develop the basic (technical) tools that will be used in the rest of the paper.

Lemma 3.1: If R = (E,H,S) and R^d = (S,E,E^r) are dual rotations that are both exposed in a table T, then in T each list of E ∪ S has exactly two elements.

Proof: This follows simply from definition of duals, and the fact that person i is the head of j's list in any table if and only if person j is the bottom of i's list in that table. □

Definition: For a table T, the *active part* of T is the subtable of T consisting of those lists which contain more than one person.

Lemma 3.2: If R = (E,H,S) and R^d = (S,E,E^r) are both exposed in T, then the active part of the table resulting from eliminating R from T is the same as the active part of the table resulting from eliminating R^d from T. Further, that active part is just the active part of T minus the lists of E ∪ S.

Proof: This follows directly from the definitions dual rotations and rotation elimination, and lemma 3.1 above. □

The following lemma extends lemma 2.2.

Lemma 2.5: Let R be a rotation exposed in table T, and (e_i, h_i) a pair in R. If A is a stable assignment contained in T where e_i pairs with h_i, and if (e,h) is *any* pair in *either* the body of R or a tail of R, then (e,h) must be a pair in assignment A.

Proof: Let e_j be a person either in R or in a tail of R. If e_k is any other person such that $s_k = h_j$, then in A, e_k must pair with h_k if e_j pairs with h_j; If not, then e_k must be paired with a person below s_k on its list, since h_k is already paired with e_j, and A is in T. But s_k is the head of e_j's list in T, so e_j must be the bottom of s_k's list, and since s_k is on e_k's list, e_k is on s_k's list, and is preferred to e_j by s_k. Hence e_k and s_k would block A. It follows that if e_j is in the body of R, and if e_j pairs with h_j in A, then every e_i in R must pair with h_j in A. Now consider any tail of R (relative to T). If (e,h,s) is the last triple of the tail, then $s = h_i$ for some e_i in R, so (e,h) must be a pair in A, and the implication follows backwards along the tail. Hence in A, each person in the tail must also pair with the head person on their list in T. □

2.4. The execution tree D, and dual rotations

Algorithm I is guaranteed to produce a stable assignment if there is one. However, in this paper we are concerned with the the structure of the set of all the stable assignments for a particular instance; most of what we will deduce will be by examining the possible executions of algorithm I. Hence we need the following

Theorem 2.1: If A is any stable roommate assignment, then there is an execution of algorithm I which produces A.

Proof: Let T be any table obtained from a (partial) execution of algorithm I, where stable assignment A is in T. If in T, the head of each persons list is their partner in A, then, as in the proof of lemma 2.1, each list has only a single entry, and so T is the final table of an execution of algorithm I, and A is the resulting stable assignment. So, assume that there is a person p whose partner in A is not the head element of p's list in T. Hence p's list has at least two entries, and, by corollary 2.1, p is either in the body or in a tail of a rotation R exposed in T. We claim that no person e_i in the body of R pairs with h_i in A. This follows directly from lemma 2.5, since if (e_i, h_i) is a pair in A, then p's partner in A must also be its head entry in T, contradicting the selection of p. Hence rotation R can be eliminated from T, creating a smaller table T′ which still contains the stable assignment A. The theorem follows by repeating this argument until no rotations remain. □

Definition: We use D to refer to the resulting execution tree, when, for a given phase 1 table, phase 2 of algorithm I is executed in all possible ways. Each node x in D represents the table T(x), which is the current state of the algorithm at node x. Each edge out of x is labelled with a

Definition: Let T be a table and R = (E,H,S) be a a rotation. If for *each* e_i in R, h_i and s_i are in e_i's list in T, then we say that R is *embedded* in T.

Lemma 3.3: If R and R^d are dual rotations, then R is embedded in table T if and only if R^d is.

Proof: This follows directly from the definition of duals, and the fact that i is on j's list if and only if j is on i's list. □

Definition: Let R be a rotation exposed in table T, and let T(R) be the table resulting from eliminating R from T. If rotation $R' = (E', H', S')$ is embedded in T but not in T(R), then we say that R *removes* R′ from T. Note that for R to remove R′ from T all that is required is that h'_i or s'_i not appear on e'_i's list in T(R), for at least one e'_i in R′.

Definition: A path P in D is said to *contain* the rotations that label the edges of P.

Lemma 3.4: If P is a path from the root of D to a node x in D, and P′ is a path from the root to a node x′, and P and P′ *contain* the same rotations in different order, then table T(x) and table T(x′) are identical. Hence a table is determined by the set of rotations leading to it, not by their order.

Proof: It is clear from the way that elements are removed in phase 1 and phase 2, that at any point in phase 2, the current table T is determined by the original full table and the bottom elements of each list in T. The bottom element of person i's list is changed only if person i is in the S set of an eliminated rotation. Hence the bottom of i's list in T is given by the person p who i most prefers, such that i is the second element in p's list in some rotation on the path to T. □

We are now ready to state and prove the first non-trivial technical lemma.

Lemma 3.5: If R = (E,H,S) and R′ = (E′,H′,S′) are two distinct rotations exposed in a table T, then R removes R′ from T if and only if $R' = R^d$. Hence the only way to remove an exposed rotation is to explicitly eliminate it or its dual rotation, if it has one.

Proof: One direction is trivial. If R and R^d are dual rotations, then since one is embedded in T if and only if the other is, the elimination of R must remove R^d. To prove the other direction, suppose that R and R′ are exposed in T, and that $R \neq R'$ eliminates R′ from T. We will show that R′ must be R^d. The elimination of R moves the head of e_i to s_i, and moves the bottom of s_i to e_i, for each e_i in E. Then a person $s_i \in S$ is removed from the list of a person p if and only if p is below e_i on s_i's list in table T. Clearly, these removals of individual people from T affect the lists of people in E′ only if s_i is in H′ (hence in S′), or if s_i is in E′. To see that the first case is not possible, recall that in table T the elements in the H column are a permutation of the 2n people, and that in each *rotation* R, the set of S elements and H elements in R are the same. So

even though the S column of table T is not necessarily a permutation (i.e. a person can appear more than once in the S column), no person can appear in the S set of more than one rotation exposed in T. Hence $S \cap S' = S \cap H' = \emptyset$, and so the first case is not possible. Hence the elimination of R removes R′ from T only if some s_i is in $S \cap E'$. For ease of discussion, assume wlog that $s_1 \in S \cap E'$, and that $s_1 = e'_j$.

If $e_1 \neq h'_j$, then the change of the bottom of s_1 to e_1 (the consequence of eliminating R) will not affect R′, so we assume also that $e_1 = h'_j$. Let T(R) be the table resulting from eliminating R in T. Since the bottom of s_1's list moves up to e_1, which is the head of s_1's list in T (since $s_1 = e'_j$ and $e_1 = h'_j$), s_1's list in T(R) contains only the single element e_1. It follows that if e'_i is in E′, then e'_i's list in T(R) must also contain only a single element; if not, then, in T(R), e'_i will be on a tail that leads to no rotation, since H′ is unchanged by the elimination of R. This would be a contradiction of corollary 2.1. So in T(R) each e'_i in E′ contains only a single element in its list. Now R′ is exposed in T, so each e'_i in E′ has two or more elements on its list in T, so the affect of eliminating R in T is to move the bottom of each e'_i in R′. But this is possible only if for each $e'_i \in E'$, $e'_i = s_k$ and $h'_i = e_k$, for some e_k in R.

So we now know that if R removes R′ from T, then *as sets*, E′ = H, and H′ = E = S′. This is necessary if $R' = R^d$, but in order to actually prove that equality, we need to show that the order inside the sets is correct. We already know that the correspondence between E′ and H′ is correct, i.e. that $e'_i = s_k$ and $h'_i = e_k$, for the same k. So, assuming wlog that $s_1 = e'_1$, we must show that $s'_i = e_{i+1}$ for each i, where i+1 is taken (mod r), and r is the size of R. To do this, we first note that in T the list of every element in R contains exactly two elements; this follows from the fact shown above that in T, each e_i in R is the head of s_i's list, so s_i is the bottom of e_i's list, so each element in R has exactly two people on its list in T. But then each e_i can be only on the list of h_i or s_i in T. Further, e_i appears once in H′ and once in S′. Now e_i is the head of s_i's list in T, so e_i must be the second element in h_i's list in T. So $e'_i = s_i = h_{i+1}$, and the second element on h_{i+1}'s list is e_{i+1}, so $s'_i = e_{i+1}$, as claimed. Hence if R removes R′ from T, then $R' = R^d$. □

Later in the paper we will strengthen this theorem to show that if R is exposed in T, and R′ is *embedded* in T, but perhaps not exposed, then R removes R′ from T if and only if $R' = R^d$.

4. The structure of D

In this section we examine the structure of D, as this structure will reveal the structure of the set of stable assignments.

4.1. Covering Rotations

Definition: Let x be a node in D and D(x) the subtree of D rooted at x. The *active part* of D(x) is the tree D(x) where at each node y in D(x), T(y) is replaced by the active part of T(y). Note that the edge labels of D(x) do not change.

Definition: If R and R^d are dual rotations, and path P in D contains *either* of them, then we say that P *covers* R *and* R^d. If R is a singleton, and P contains it, then then P covers R.

Lemma 4.1: Let x be a node in D with associated table T(x). Every path from x to a leaf in D(x) covers the same set of rotations.

Proof: Let d(x) denote the maximum number of edges on any path from x to a leaf in D. The theorem will be proved by induction on d(x). For $d(x) = 1$, if there is only one edge out of x (i.e. only one rotation exposed in T(x)) then the basis is trivially true. If there are two rotations R and R′ exposed in T(x), then, by lemma 3.5, they must be duals of each other, since rotating either one of them results in a table with no rotations (i.e. each removes the other). Similarly, there cannot be more than two rotations in T(x), since the elimination of any of them removes them all. So the basis is proved.

Assuming that the theorem holds for $d(x) \leq k$, let x be a node in D where $d(x) = k+1$, and let z be a child of x such that $d(z) = k$; by the induction hypothesis, all paths from x through z to a leaf must cover the same rotations; let P be any such path, and let R be the rotation labelling the edge (x,z). If R is the only rotation exposed in T(x) then there is nothing to prove, so let y be another child of x, and let R′ be the rotation on the edge (x,y). We will show that every path from x through y to a leaf of D(y) covers the same set of rotations as P.

If $R' = R^d$, then, by lemma 3.1 (since both R and R^d are exposed in T(x)), the *active* parts of T(z) and T(y) are identical, and hence the active parts of D(z) and D(y) are identical. Further, $d(z) = k$, and $d(y) \leq k$, so each path from z covers the same rotations, and each path from y covers the same rotations, so, since the subtrees from z and y are identical, any path from z must cover the same rotations as any path from y. Then every path from x through y covers the same rotations as P.

If $R' \neq R^d$, then, by lemma 3.5, R is still exposed in table T(y), and R′ is still exposed in table T(z). Let z′ be the node associated with the table obtained by eliminating R′ from T(z), and let y′ be the node associated with the table obtained by eliminating R from T(y); and let P(z′) and P(y′) be paths from x to leafs in D that pass through z′ and y′ respectively. Now the set of rotations on the path from the root of D to z′ is exactly the same as the set of rotations on the path to y′, hence by lemma 3.4, T(z′) is identical to T(y′), and so D(z′) = D(y′). It follows, as in the case above, that P(z′) and P(y′) cover the same set of rotations. But, P(z′) covers the same set as P, and since $d(y) \leq k$, any path out of y covers the same set of rotations

as P(y′), hence covers the same set as P. Node y was an arbitrary child of x such that $y \neq z$, so the theorem is proved. □

By definition, every rotation is exposed somewhere in D, hence the major consequence of this theorem is the following

Path Theorem

Theorem 4.1: Every path from the root of D to a leaf covers all the rotations. Further, since no path can contain both a rotation and its dual, each path contains every singleton and exactly one of each dual pair of rotations.

Corollary 4.1: Every path in D from the root to a leaf has the same length.

Hence the observations in the example hold in general.

We can now strengthen lemma 3.5.

Corollary 4.2: If R is exposed in table T, and R′ is *embedded* in T, then R removes R′ if and only if $R' = R^d$.

Proof: Clearly, R removes R^d whether R^d is exposed or not. To prove the converse, let x be a node in D with associated table T(x), and let y be the child of x obtained by eliminating R from T(x). Since R′ is embedded in T(x), neither R′ nor R'^d are on the path from the root to x. If R removes R′, it removes R'^d also, so neither of these rotations is on any path from x to a leaf. Hence to avoid contradicting theorem 4.1, it follows that $R' = R^d$. □

We continue to clarify the structure of D. The following lemma augments lemma 3.2, and will be needed in the proof of the next theorem.

Lemma 4.2: If P is a path from the root of D to a node x in D, and P′ is a path from the root to a node x′, and P and P′ *cover* the same set of rotations, then the *active parts* of table T(x) and table T(x′) are identical, and hence the active parts of D(x) and D(x′) are also identical.

Proof: Note first that since no path can contain both rotations in a dual pair, the length of P and P′ are the same. Let dP and dP′ be the parts of P and P′ respectively, after the point v in D where P and P′ diverge. The proof of the lemma is by induction of the length of dP (which is, of course, also the length of dP′). For length of one, dP must contain R while dP′ contains R^d, for some dual pair of rotations. Then in T(x), both R and R^d are exposed and the basis follows from lemma 3.2. Now assuming the theorem holds for dP of length k, consider dP of length k+1, and let R and R′ be the first rotations on dP and dP′ respectively, and let v_R and $v_{R'}$ be the first nodes below v on these paths (see figure 4a). If $R' = R^d$ then the active tables are the same after eliminating either rotation, and hence there must be a path from v_R that is identical to the part of dP′ starting at $v_{R'}$. Hence the table T at the end of that path is T(x′). But, by

the induction hypothesis, the active part of T(x) is the same as the active part of T, hence the theorem follows in this case.

Now suppose that $R' \neq R^d$. There are two cases to consider: either R is on dP′, or R^d is on dP′.

Let w be the point on dP′ where R (in the first case) or R^d (in the second case) is eliminated. Since both R and R′ are exposed at v, R must be exposed at every table on dP′ down to w.

In the first case, consider the edge on dP′ into w, and let R* be the rotation eliminated there (see figure 4b). If instead of eliminating R*, R is eliminated at that point, R* will not be removed (since $R^* \neq R^d$), hence the path which is identical to dP′ except that the order of R* and R is reversed, is in fact a path from v; call that path dP*. Now dP′ and dP* contain exactly the same rotations, so the table at the end of dP* is T(x′). Repeating this argument up the length of dP′, moving R up at each step and leaving the rest of the path the same, it follows that there is a path from v through v_R which contains the same rotations as dP′, and hence ends with table T = T(x′). But, by the inductive hypothesis, as above, the active parts of T(x) are identical to the active parts of T, and hence of T(x′).

In the second case, there must be a path, P(R), from v through v_R, which is identical to dP′, except that R replaces R^d (see figure 4c). This follows from lemma 3.2, and the fact that R is exposed at w. But now dP′ and P(R) diverge below v, hence by the induction hypothesis, the active part of the table, T, at the end of P(R) is identical to the active part of T(x′). Now we can repeat the step argument of the first case, moving R up P(R) to v, and conclude that there is a path from v through v_R which contains exactly the same rotations as P(R). Then, by the inductive hypothesis, the active part of T(x) is the same as the active part of T, which is the same as the active part of T(x′). □

Theorem 4.2: Let R and R^d be dual rotations, and x a point in D. If R is exposed in T(x), then R^d is exposed in D(x).

Proof of the theorem: Let z be the closest ancestor of x such that R^d is exposed in D(z), and let y (possibly x) be the child of z on the path from z to x; Let R_1 be the rotation on the (z,y) edge. Let P be a path from z to a leaf, where P contains R^d, and let R_2 be the first rotation on P (see figure 5a). If $R_1 = R_2{}^d$, then the active tables after eliminating either rotation are the same, so the subtrees below those two points must be the same, hence D(y) must contain R^d. So, assume that $R_1 \neq R_2{}^d$. Neither R_1 nor $R_1{}^d$ is on the path from the root to z, so either R_1 or $R_1{}^d$ must be on P, say at a point w. Note that in either case, R_1 is exposed at w, as in the above proof of lemma 4.2. Hence if R^d is before w on P, then we can assume P contains R_1. If P contains $R_1{}^d$ and R^d is after w (see figure 5b), then consider the affect of eliminating R_1 at w; the resulting active table is the same as after eliminating $R_1{}^d$, so there is a path from w which

contains R^d. So we can always assume that P contains R_1 and R^d. But now, we can move R_1 up a step at a time, as the preceding proof, concluding that there is a path from z through y that contains R^d. This contradicts the selection of z, and proves the theorem. □

Corollary 4.3: If R is the only rotation exposed in T(x), then R has no dual rotation.

Corollary 4.4: If R and R^d are duals embedded in T(x), then both are exposed somewhere in D(x).

Corollary 4.5: If R and R^d are duals, then there is a point x in D where both R and R^d are exposed in the table T(x).

Corollary 4.6: If R and R^d are dual rotations, then there is a stable roommate assignment where each e_i is paired with h_i, and also one where each e_i is paired with s_i, where e_i is in R.

Proof: At point x where both R and R^d are exposed, eliminating R^d makes h_i the only element on e_i's list, and eliminating R makes s_i the only element on e_i's list. With either elimination, the algorithm is guaranteed to find a stable assignment in the resulting table. □

4.2. Finding all the rotations in $O(n^4)$ time

At this point we digress to point out that the above corollaries give an $O(n^4)$ time algorithm to identify all the rotations. We run Irving's algorithm once, following a path P in D, finding the rotations on P. If R is a rotation on P, then, using corollaries 4.1 and 4.3, we can test if R^d is a rotation by simply returning to the point on P where R is eliminated and successively choosing and eliminating any rotation other than R. By corollary 4.2, we will either expose and eliminate R^d, or we will have a table where only R is exposed; in the latter case, R can have no dual rotation, by corollary 4.3. There are at most n(n-1) rotations on P, and each running of Irvings algorithm costs $O(n^2)$ time, so although the size of D can be exponential in n, all rotations in D can be found in $O(n^4)$ time. Of course, in practice this procedure can be sped up by noting at each step which other rotations are exposed.

5. The structure of the rotations and stable assignments

5.1. Unique elimination

Definition: Let e_i, h_i, s_i be a triple in rotation R; hence when R is eliminated from any table it is exposed in, the bottom of s_i's list moves from e_{i+1} to e_i, where i+1 is taken mod r. Let A(R,i) denote the set of people on s_i's list between e_i and e_{i+1}, including e_i but excluding e_{i+1}. Similarly, let B(R,i) be the people between e_i and e_{i+1}, including e_{i+1} but excluding e_i.

Lemma 5.1: For any e_i in R, R is the only rotation whose elimination moves the bottom of s_i's list *to* a person in A(R,i), and is the only rotation whose elimination moves the bottom of s_i's list *from* a person in B(R,i).

Proof: Let R′ be a different rotation whose elimination moves the bottom of s_i's list to a person p in A(R,i), from a person q. Clearly, since e_i is above q, and p is above e_{i+1}, no path in D can contain both R and R′. Further, R^d (if it is a rotation) cannot precede R′ on any path, since R^d moves the head of s_i's list to e_{i+1}, which is below p. Similarly, R′ cannot precede R^d on any path, since it moves s_i's bottom to p, which is above e_{i+1}. But every path contains either R or R^d, so no path contains R′, contradicting the definition of a rotation. The proof for moves from B(R,i) is similar. □

Corollary 5.1: A person p is the H element a person q's list in at most one rotation, and is the S element of q's list in at most one rotation.

Corollary 5.2: If $p \neq e_i$, and $p \in A(R,i)$, then s_i can never be paired with p in any stable roommate assignment.

Proof: Consider any path P where p is paired with s_i. Since s_i prefers p to e_i, p is not the bottom of s_i's list in the phase I table. Hence, somewhere on P, p must become the bottom of s_i's list. But this contradicts lemma 5.1 above. □

5.2. The partial order Π^* of rotations

Let R be a rotation with the triple e_i, h_i, s_i in R. If $p \neq h_i$, and p is above s_i in e_i's list, then R will never be exposed until p is removed from e_i's list.

Lemma 5.2: Let p be a person who must be removed from e_i's list before R is exposed. There is only one rotation, R′, other than R^d, whose elimination removes p from e_i's list. Hence R′ must precede R on any path in D which contains R (i.e. on which R is eliminated).

Proof: From examination of algorithm I, there are only two ways in which p can be removed from e_i's list: either the bottom of e_i's list moves up above p, or the bottom of p's list moves up above e_i. The first case happens when R^d (if it is a rotation) is eliminated, and by lemma 5.1, that is the only way that it can happen. By lemma 5.1 again, the second case can happen only when a particular unique rotation, R′, is eliminated. To see that R′ must precede R on any path containing R, recall that if R^d is eliminated, then R is removed. □

Definition: If p must be removed from e_i's list before R is exposed, and if R′ is the (unique) rotation whose elimination causes the removal of p from e_i's list, then we say that R′ *explicitly precedes* R.

Definition: Let Π^* be the reflexive transitive closure of the above relation of explicit

precedence. It is clear that Π^* is a partial order on the rotations.

Figure 6 shows the Hasse diagram of Π^* for the running example.

The following lemma follows directly from the definition of Π^* and lemma 5.2.

Lemma 5.3: If (R′, R) $\in \Pi^*$, then R′ is on every path in D that contains R, and R′ precedes R on each of these paths.

The converse of the lemma is also true, but it will not be needed. However, it gives us the freedom to say R′ *precedes* R, either in the context of Π^*, or in the context of paths in D. Hence, we call Π^* the precedence relation on the rotations.

5.3. The structure of the stable assignments

Definition: In partial order Π^*, a subset C of rotations is called *closed* if and only if it is closed under the predecessor relation, i.e. if R is in C, and R′ precedes R in Π^*, then R′ is in C.

Theorem 5.1: There is a one-one correspondence between stable assignments and the set of those closed subsets of Π^* which contain all the singleton rotations, and contain exactly one of each dual pair of rotations.

Proof: One direction is trivial. Let A be a stable assignment and P be a path in D which results in assignment A. We claim that the set C of the rotations on P forms a closed subset in Π^* of the required type. We know that each path in D contains all the singleton rotations and exactly one of each dual pair of rotations, hence we only need to show that C is closed in Π^*. But, by lemma 5.3 above, any rotation that precedes R $\in$ C must be contained on P, hence C is closed. Conversely, let C be a closed set of the required type; we will show that there is a path P in D which contains C exactly. First, the maximal elements $C_0 \subseteq$ C (those with no predecessors in Π^*) must be exposed rotations in the phase I table, and since only one of any dual pair is in C, there is a subpath from the root of D consisting of the rotations C_0. After the C_0 rotations are eliminated, the elements $C_1 \subset$ C whose only predecessors are in C_0 must now be exposed, and again, each remain exposed until eliminated, and hence there is a path from the root consisting of C_0 followed by C_1. Continuing in this way, there is a path from the root to a leaf in D consisting of the rotations in C. □

5.3.1. The lattice generated from Π^*

It is well known that if P is a partial order, then the family of *all* closed subsets of P defines a distributive lattice L under the relation of set inclusion; that is, each element of L is a closed subset of P, and if C and C′ are two closed subsets in P, then C $\leq$ C′ in L if and only if C′ $\subseteq$ C in P. Further, every finite distributive lattice can be generated in this way from some partial order [B,GR]. It is also well known [K,GS84,R] that, for a fixed instance of the stable marriage

problem, the set of all stable marriages forms a distributive lattice under the relation of male dominance, and hence, as shown explicitly in [IL], there is a partial order Θ such that the stable marriages correspond one-one to the closed subsets in Θ. This partial order Θ never has more than $O(n^2)$ nodes for an n person instance of the marriage problem; it can be found quickly [ILG, G]; and once obtained, it is the key to the efficient solution of many problems concerning stable marriage [ILG, G].

Because of the beauty and the theoretical and algorithmic utility of the lattice structure for stable marriages, it is desirable to find a similar algebraic structure to tie together the set of stable roommate assignments. However, the situation for the stable roommate problem at first seems more complex; the closed subsets of Π^* that correspond to assignments are constrained to be of a special form: they must contain every singleton rotation, and must contain exactly one of every dual pair of rotations. It isn't immediately clear if there is a known and useful algebraic structure defined by these particular closed subsets. We will answer this question, showing that the set of stable assignments also forms a distributive lattice under the same relation as in the stable marriage problem.

Lemma 5.4: Let (R, R^d) and (R_1, R_1^d) be two dual pairs of rotations, and R' a singleton rotation. Then:

1) Neither R nor R^d can precede R' in Π^*, i.e. only a singleton rotation can precede a singleton.

2) R precedes R_1 in Π^* if and only if R_1^d precedes R^d.

3) R cannot precede both R_1 and R_1^d in Π^*.

Proof: Since each singleton rotation is on every path in D, any rotation which precedes a singleton rotation must be on every path in D. So if R precedes R', R is on every path, and R^d is on no paths in D, contradicting the assumption that R^d is a rotation; so the first fact is proved. For the second fact, observe first that since R precedes R_1, no path can contain both R_1 and R^d, so any path containing R^d contains R_1^d. Now if P is a path with R^d before R_1^d, consider the point x where R^d is eliminated. Both R_1 and R_1^d are embedded in T(x), so, by corollary 4.4, R_1 is exposed in D(x), contradicting the fact that no path can contain both R^d and R_1. So R_1^d precedes R^d, and fact 2 is proved. Fact 3 follows from fact 2, for if R precedes both R_1 and R_1^d, then by transitivity, R precedes itself, which is impossible. □

5.4. The dual partial orders Π and Π^d

As a consequence of lemma 5.4 above, if we remove the singleton rotations from Π^*, the dual rotations can be partitioned[1] into two sets, Π and Π^d, so that the following conditions hold for any non-singleton rotations R and R′: R is in Π if and only R^d is in Π^d; no element of Π precedes an element of Π^d, nor does an element of Π^d precede an element of Π; and R precedes R′ in Π if and only if R'^d precedes R^d in Π^d. Hence Π and Π^d are themselves each partial orders under the precedence relation, and they are inverted images of each other. Figure 7 illustrates this for the running example. With these observations, we have the following

Definition: Let $\mathfrak{R}$ be the rotations in Π, and $\mathfrak{R}^d$ the rotations in Π^d. For a subset $C \subseteq \mathfrak{R}$, let C^d be the set of duals of the rotations in C (of course, $C^d \subseteq \mathfrak{R}^d$).

It follows immediately from lemma 5.4 that

Corollary 5.3: C is a closed set of rotations in Π if and only if $(\mathfrak{R} - C)^d$ is a closed set of rotations in Π^d.

Hence there is a one-one correspondence between the closed subsets in Π, and the those closed subsets in Π^* which contain every singleton and exactly one of each dual pair of rotations: a closed subset C in Π corresponds to the subset in Π^* consisting of the singleton rotations, plus C, plus $(\mathfrak{R} - C)^d$, and this subset is closed in Π^*; conversely, a closed subset K in Π^* corresponds to the subset $(\mathfrak{R} \cap K)$ in Π, and this subset is closed in Π.

Summarizing we have

Theorem 5.2: There is a one-one correspondence between the closed subsets in Π and the set of stable assignments.

As mentioned above, the set of all closed subsets in a partial order forms a distributive lattice under the relation of set inclusion. Hence, as in the stable marriage problem, the set of stable roommate assignments can be represented by a distributive lattice which is generated from Π, which, as a function of n, is a small, efficiently obtainable partial order. A stable assignment A is less than stable assignment A' in L if and only if the closed subset in Π associated with A contains the closed subset associated with A'. In the case of stable marriage, the relation in the lattice can be similarly expressed in terms of closed subsets and set inclusion, but it can also be easily expressed as the relation of (male) dominance [K]. For the roommate problem, we do not (yet) have such a simple characterization of the relation. It will be clear from the next section, and made explicit in the last section, that the relation in L is also one of dominance.

[1]The partition is not unique if the set of all non-singleton rotations form more than two connected components of Π^*. However, any partition that satisfies the conditions will do.

6. The partition, and reduction to stable marriage

6.1. The partition

In this section we will show that there is a partition of the 2n people into two sets M and W, called men and women, such that *every* stable assignment is a mapping between M and W, i.e. there are no (M,M) pairs or (W,W) pairs in *any* of the stable roommate assignments. We will then show a reduction of the roommate preference lists to lists specifying an instance of the stable marriage problem between M and W, so that the stable assignments are exactly the stable marriages in the derived problem. We will also give a (first) algorithm to find a partition.

Lemma 6.1: If person p is in the H set of a rotation R in $\mathfrak{R}$, then p is not in the H set of any rotation in $\mathfrak{R}^d$, but, of course, p is in the E set of $R^d \in \mathfrak{R}^d$. Similarly, if person p is in the E set of a rotation R in $\mathfrak{R}$, then p is not in the E set of any rotation in $\mathfrak{R}^d$, although it is in the H set of $R^d \in \mathfrak{R}^d$.

In other words, once the singleton rotations have been deleted, the people in the remaining rotations can be *partitioned* into those who are in the E sets of rotations in $\mathfrak{R}$ (hence in the H = S sets of rotations in $\mathfrak{R}^d$), and those who are in the E sets$\mathfrak{R}$, then of rotations in $\mathfrak{R}^d$ (and in the H = S sets of rotations in $\mathfrak{R}$). This is certainly reflected in the rotations of the running example.

Proof: Suppose p is the H element of e_i in $R \in \mathfrak{R}$, and also the H element of e′ in $R' \in \mathfrak{R}^d$. In any table where R is exposed, e_i is the last element on p's list, and similarly, when R′ is exposed, e′ is the last element on p's list. Suppose wlog that p prefers person e_i to e′. Then by lemma 5.2, R′ must be eliminated before R can be exposed, hence R′ is on any path that R is on. But $R' \in \mathfrak{R}^d$, so $R'^d \in \mathfrak{R}$, and R and R'^d generate a closed subset in $\mathfrak{R}$ (simply take R and R'^d and all predecessors of them in $\mathfrak{R}$), so there must be a path in D containing both R and R'^d. But that path would then contain both R′ and its dual, which is impossible. The second part of the lemma follows from the first, for if p is in the E set of $R \in \mathfrak{R}$, then it is in the H set of $R^d \in \mathfrak{R}^d$, hence it can't be in the H set of any rotation in $\mathfrak{R}$, hence it can't be in the E set of any rotation in $\mathfrak{R}^d$. □

Definition: Let D* be the subtree of D on which all the singleton rotations are eliminated before any non-singletons are, and in which no rotation in $\mathfrak{R}^d$ is eliminated before a rotation in $\mathfrak{R}$.

By lemma 5.4 and theorem 5.2, it is clear that D* generates all the stable assignments of D. Since a table is determined only by the eliminated rotations, every path in D* has the same table after the elimination of the singletons; let T* be this table (hence T* contains all the stable assignments). Without loss of generality we can also assume that there is only one path leading to T*, and we let x* be the (assumed) unique node associated with T*.

Definition: If persons i and j are paired together in some stable assignment, then they are called a *stable pair*. If they are paired together in all assignments, then they are called a *fixed pair*.

Lemma 6.2: Person i is in a fixed pair if and only if i's list in T^* has only a single entry.

Proof: First, since D^* contains all the stable assignments, if i's list in T^* contains only one entry, j, then (i,j) is a pair in every stable assignment. If i's list in T^* is not a single entry, then i must be in the E set of a rotation R, and in the H = S sets of rotation R^d. Hence by corollary 4.6, i is in at least two stable pairs. □

Lemma 6.2, combined with corollary 4.6 gives the following

Lemma 6.3: If (e_i,j) is not a fixed pair, then it is a stable pair if and only if $j = h_i$ in some non-singleton rotation.

We can now define a partition of the 2n people into two sets M and W.

Definition: If (i,j) is a fixed pair, then arbitrarily put either i or j into set M and the other into set W. For a remaining person i, if i is in the E set of some rotation in $\mathfrak{R}$, then put i in M, else put i in W. By corollary 4.6 and lemmas 6.1 and 6.2, the sets M and W form a partition of the 2n people. We will often call the people in M "men", and the people in W "women". Such a partition for the running example is shown in figure 1.

With this definition and the above results, the following is easy to prove, after first noting that only the elimination of rotations in $\mathfrak{R}$ move the head of a man's list, and similarly, only the elimination of rotations in $\mathfrak{R}^d$ move the head of a woman's list.

Theorem 6.1: Let T be any table in D, for a solvable roommate instance. Matching each man to his head entry in T is a stable assignment, and similarly, matching each woman to her head entry in T is also a stable assignment.

Finally, we have the main theorem of the paper.

Partition Theorem

Theorem 6.2: If (i,j) is any stable pair, then exactly one of i or j is in M, and the other is in W.

Proof: This is true by definition for any fixed pair. Suppose i is not in a fixed pair, and that i is in set M; consider any path P in D^* from x^* to a leaf where i pairs with j. Since (i,j) is not a fixed pair, j cannot both be the head and the bottom of i's list in T^*. However, j is both the head and the bottom of i's list at the end of P, so, at some point on P, either the head or the bottom of i's list must change. When the head changes, i is an E element in the rotation R being

eliminated, and the new head of i's list is an S element of R; hence, by lemma 6.1, the new head must be in W. Similarly, when the bottom changes, say from k to k′, then i must be $s_k = h_{k+1}$ in the rotation being eliminated; hence, again by lemma 6.1, k′ is in W. At some point on P, j becomes either the new head or the new bottom of i's list, and so j is in W. □

Corollary 6.1: |M| = |W| = n.

6.1.1. Finding a partition

It should be clear from the results so far that we can find a partition M,W in the following way: find the set of rotations; eliminate all the singleton rotations from the table after phase I of Irvings algorithm; using the precedence relations on rotations, partition the non-singleton rotations into those in $\mathfrak{R}$ and those in $\mathfrak{R}^d$; and note the people in the E sets of $\mathfrak{R}$, putting them into M, putting the people in the H set of $\mathfrak{R}$ into W, and dividing each fixed pair arbitrarily between M and W. Note that we only need to identify the rotations in $\mathfrak{R}$; we do not need to explicitly build the partial orders Π or Π^d. In the next section we will present a simpler procedure that finds a partition of the people without the need to classify the rotations into $\mathfrak{R}$ and $\mathfrak{R}^d$.

6.2. Reducing stable assignment to stable marriage

In this section we show that, given a partition of the 2n people into sets M and W, we can construct an instance of the stable marriage problem between M and W, such that every stable marriage in the derived problem is a stable roommate assignment in the original roommate problem, and visa versa. The existence of this reduction completes the relationship of the solvable stable roommate problem to the stable marriage problem. It also permits a simple exposition of efficient solutions to several problems of selecting particular "good" stable assignments, if there are more than one, and for the efficient enumeration of the set of all stable assignments. The analogous stable marriage problems are discussed and solved in [ILG, G], and although those solutions can be modified and applied directly to the roommate problem, reducing the roommate problem to stable marriage allows a more efficient exposition.

There are many (M,W) preference lists that will work. Unfortunately, the most obvious and "attractive" reduction, that of deleting every man from each man's roommate list, and removing every woman from each woman's roommate list, does not work. In this paper we will discuss a reduction that is easiest to prove correct.

Definition: For T a subtable of T*, let C(T) denote the table where the list of each person p is obtained from T by deleting every person who is not in a stable (roommate) pair with p.

We will show that C(T*) defines an (incomplete) instance of the stable marriage problem between M and W, such that the stable assignments in T* (recall that T* contains all the stable

assignments) are stable marriages in C(T*), and conversely. The proof of this, in a following sequence of lemmas, will be somewhat indirect: we will reduce the stable *marriage* problem on C(T*) to a stable *roommate* problem on C(T*), so that we can apply the (roommate) theorems already obtained in this paper. We will then see that the set of rotations obtained from the roommate problem on C(T*) is exactly the same as the set of rotations obtained from the roommate problem on T*; it will then follow that the stable marriage of C(T*) are exactly the stable assignments of T*.

Definition: Let MP be a complete instance of the stable marriage problem (between M and W), obtained from C(T*) by adding to the end of each man (woman) i's list any woman (man) who is not on i's list in C(T*).

Lemma 6.3: All stable marriages for instance MP are contained in C(T*).

Proof: We first show that the head of each list i in T* is in a stable pair with i. To see this, not that only rotations in $\mathfrak{R}$ move the head of a man's list, and only rotations in $\mathfrak{R}^d$ move the head of woman's list. Hence the maximal assignment in L (the empty subset of $\mathfrak{R}$) must assign each man to the head of his list in T*. Similarly, the minimal element in L (the complete set $\mathfrak{R}$) gives assigns each woman to the head element in her list in T*.

So the head entries of T* and C(T*) and, of course, MP are the same. It also follows that the bottom entries of T* and C(T*) are the same, for j is the head of i's list in T*, if and only if j is the bottom of i's list in T*. But (i,j) is a stable pair, hence i is on j's list in C(T*). C(T*) is a subtable of T*, so the bottoms of C(T*) and T* are the same.

Now T* is a table obtained from running algorithm I, so the head entries of T* form a permutation of the 2n people. Hence, if we run the classical Gale-Shapley stable marriage algorithm on MP with the men proposing, then each man gets his first choice in table MP, hence in table C(T*). This marriage is called the man-optimal marriage, so in the man optimal marriage, each man gets his first choice in C(T*), and each woman gets her last choice in C(T*). Symmetrically, if the woman make the proposals, yielding the woman-optimal marriage, then each woman gets her first choice in MP and in C(T*), and each man gets his last choice in C(T*). It is well known [K] that on the preference list for each person i (for the stable marriage problem), any person outside the interval specified by i's mates in the man-optimal and the woman-optimal marriages can be deleted without changing the set of stable marriages. Hence the stable marriages of MP are exactly those of C(T*). □

We now reduce the problem of finding stable *marriages* in MP (hence in C(T*)) to the problem of finding stable *assignments* in a larger table RMP.

Definition: Let RMP be an instance of the stable roommate problem on M ∪ W, obtained

from MP by adding to the end of each man (woman) p's list all the men (women) other than p.

Note that C(T*) is an initial sub-table of MP, which is an initial sub-table of RMP. Clearly, if *A* is a stable assignment in RMP that is contained in C(T*), then *A* specifies a stable marriage in MP. The converse is proved true in the following

Lemma 6.4: Every stable roommate assignment in RMP is in C(T*), hence is a stable *marriage* in C(T*).

Proof: If man m is paired with a man in an assignment in RMP, then there is a woman w who is paired with a woman in that assignment. But w prefers m to any woman, and m prefers w to any man, hence the assignment is not stable. So all stable ***assignments*** in RMP are contained in MP, and so are stable marriages in MP. But all stable marriages in MP are contained in C(T*). □

So the set and the structure of the stable ***marriages*** in C(T*) is exactly that of the set and structure of the stable ***roommate assignments*** in RMP. Since we have now reduced the marriage problem on C(T*) to a roommate problem, we can apply what we know about the roommate problem.

Lemma 6.5: The phase 1 table of algorithm I applied to RMP is C(T*).

Proof: The proof here is exactly as in lemma 6.3, i.e. that the head elements of each list in RMP are a permutation, and if j is the head of i's list in RMP, then i is in C(T*) on j's list, and is the last entry on i's list in C(T*). □

Hence the question of what phase 2 of algorithm I does on C(T*) is meaningful. Let C(D*) be the execution tree resulting from running algorithm I, in all possible ways, starting with the phase 1 table C(T*).

Lemma 6.6: Let P be a path from x* in D* (hence starting with table T*) containing rotations $\{R_1, \ldots, R_k\}$. Then there is a path in the C(D*) containing the same set of rotations, and the resulting stable assignment is the same.

Proof: By corollary 4.6, if (e_i,h_i,s_i) is a triple in non-singleton rotation R, then both (e_i, h_i) and (e_i, s_i) are stable pairs in D*, hence all the elements of any rotation on P are in C(T*), and their relative order is unchanged in each list. Hence all the exposed rotations of T* are exposed rotations in C(T*). At the start of P (at x*) the bottom entries of T* are the same as the bottoms of C(T*), and it follows from lemma 5.1 that each time a person p is the S element of a rotation being eliminated on P, the bottom of p's list moves up to the person who is the next person on p's list *in C(T*)*. Hence by induction on the order of the rotations on P, every initial subpath of P is also a subpath from C(T*) in C(D*), and the lemma follows. □

Hence every rotation in D^* is in $C(D^*)$. Now we will show the converse.

Lemma 6.7: All rotations in $C(D^*)$ are in D^*.

Proof: Consider a person e_i, and let q be the bottom of e_i's list in $C(T^*)$. We first claim that every person on e_i's list in $C(T^*)$, other than q, is h_i in some rotation in D^* containing e_i. To see this, note that since (e_i,q) is a stable roommate pair in T^*, there is a path P in D^* where q is made the head of e_i's list. Let $j \neq q$ be another person such that (e_i,j) is a stable pair in T^*, i.e. j is on e_i's list in $C(T^*)$. Now on P, j must be removed from e_i's list before q can become the head. This happens either when $j = h_i$ (and e_i is the bottom of j's list) in some rotation R, and R is then eliminated, or when the bottom of j's list is below e_i and is then moved strictly above e_i by the elimination of some rotation. But by corollary 5.2, the second case would imply that (e_i,j) is not a stable pair in T^*. Hence each person on i's list in $C(T^*)$, other than q, is h_i for some rotation in D^*, hence for some rotation in $C(D^*)$.

We can now prove the lemma. By corollary 5.1, each person j can be the H element for person e_i in at most one rotation in $C(D^*)$ (we can apply corollary 5.1 here because $C(D^*)$ is an execution tree of a *roommate* problem). But each person other than q is already the H element for e_i in a rotation in D^*, hence in $C(D^*)$. Further, q can't be h_i for any rotation in $C(D^*)$, so $C(D^*)$ contains only those rotations that are in D^*. □

Hence D^* and $C(D^*)$ contain exactly the same set of rotations, and hence produce exactly the same stable assignments. It now follows that

Theorem 6.3: The stable marriages in $C(T^*)$ are exactly the stable assignments in the original stable roommate problem, and the lattice L of stable assignments obtained from Π is identical to the lattice of stable marriages obtained from $C(T^*)$.

6.2.1. A simpler algorithm for finding a partition

The correctness of the above reduction of the stable roommates problem to the stable marriage problem gives a simple way to find a partition of the 2n people into sets M and W. Given the stable pairs, which, by lemma 6.3, are known once the rotations have all been found and the singletons have been identified, a partition can easily be obtained by using the fact that i is on j's list in $C(T^*)$ if and only if i and j have different gender. Hence, once the stable pairs are known, the partition and reduction can be found in $O(n^2)$ time. The easiest way to see this is to consider the stable pairs as a graph, where each person is a node, and there is an edge between two nodes if and only if the associated people are in a stable pair. Then the partition theorem says that this graph is bipartite, with an equal number of nodes on each side. Any two-coloring of the nodes yields a proper partition of the nodes, and hence, of the people (note that any proper partition can be obtained this way). It is well known how to obtain such a coloring in time proportional to the number of edges, here the number of stable pairs. Hence the total time for

the partition and reduction is $O(n^4)$, the time needed to find all the rotations, and identify the singletons. We believe that the time can be reduced; this the focus of current research.

7. Extensions and Specializations

Given a partition of the 2n people into men and women as above, we have the following:

Definition: Stable assignment A dominates (from a male perspective) A' if and only if no man prefers his partner in A' to his partner in A.

It is known that in the case of stable marriage, the relation in lattice of all stable marriages is (male) dominance. An immediate consequence [K] of this is that if M and M' are two stable marriages, then the mapping of each man to his most preferred mate in those two marriages, is also a marriage, and it is stable. Given theorem 6.3, this must also hold for stable assignments, and the relation in L must also be that of (male) dominance. Many other theorems which hold for stable marriages can similarly be defined and proved for the roommates case, via the imposition of gender and the reduction to stable marriage. For example, theorems about polygomy, misrepresentation of preferences, and the case of people refusing to pair with certain other people, are easily extended to the roommate case.

As shown in the proof of lemma 6.4, the stable marriage problem is a specialization of the stable roommate problem, hence it is interesting to see what the general results in this paper imply for the marriage problem. In particular, it is interesting to compare the results here to those of [IL], where a structure of the set of stable marriages was first obtained. The results in this paper specialize to those in [IL] for the marriage problem. However, the structure of stable marriages is somewhat simpler than the general roommate structure, and this allows faster algorithms to construct it, and a simpler view of how to construct stable marriages from the partial order(s). Hence the structure resulting from specializing the roommate structure to stable marriage, at first appears different than that in [IL]. In the next paragraphs we sketch additional observations that connect the structures and the expositions.

By lemma 6.5, there are no singleton rotations in an instance of the marriage problem. Hence any path in D essentially yields all the rotations in the stable marriage case. The time for a single execution of algorithm I is $O(n^2)$, and hence for the stable marriage problem, all the rotations can be obtained that quickly; that bound was first obtained in [G] by a different method and argument. Further, since the partition is known in the marriage case, it is easy to follow the path $P(\mathfrak{R})$ in D which contains only rotations in $\mathfrak{R}$. While the exposition in [IL] is very different from the one given here, one can interpret the method in [IL] for finding rotations as a traversal of $P(\mathfrak{R})$. Finally, in the marriage case, any particular stable marriage can be extracted from Π more easily than in the roommate case. Before detailing this, we note the

following theorem for roommates.

Theorem: Let C be a closed subset in Π, and let *A(C)* be the corresponding stable roommate assignment. If T is the table obtained by starting with T^* and eliminating exactly the rotations in C, then in T, each man's mate in *A(C)* is his head entry in T.

This theorem holds both for the stable roommate and the stable marriage problems, but it is much more useful in the marriage problem, because the partition is known ahead of time, and T^* can be obtained in $O(n^2)$ time; in the roommate case, T^* cannot be identified until the singletons are known, and, at present, that takes $O(n^4)$ time. This theorem essentially connects the structure in [IL] to that obtained by specializing the results in this paper to the stable marriage problem. Hence in [IL], all stable marriages are obtained from the man optimal stable marriage, via elimination of rotations in $\mathfrak{R}$, and the concept of dual rotations, or singletons, is never discussed, or needed.

7.1. Comment

A final comment on the relationship of the paper [IL] to the present paper. It was known for some time that the set of stable marriages forms a distributive lattice, and although the stable marriage literature contains many calls for a more compact and useful representation of the set, the partial order representation was made explicit only recently in [IL], where that representation is proved entirely in the setting of the stable marriage problem, and the Gale-Shapely algorithm. It was subsequently noted in [GILS] that the structure results of [IL] could be obtained by making reference to classical theorems about distributive lattices, in particular that any finite distributive lattice is isomorphic to the closed subsets of a partial order, under the relation of set inclusion. However, in the case of the solvable roommate problem, it was unknown that the stable assignments form a distributive lattice, and hence that the assignments can be represented by the small partial order Π and obtained from the rotations. The proofs of these results, in this paper, follow the spirit of the proofs in [IL], i.e. the arguments are made entirely in the setting of the stable roommates problem and algorithm I. If I had learned the structure of the stable marriage problem only from the more concise lattice theoretic view, I doubt that I would have developed the needed intuition for these results, and I doubt I would have conjectured and obtained the structure of the roommate assignments. Hence I owe a large intellectual debt to the ideas and exposition in [IL], no matter if they are subsequently replaced by a more compact and algebraic exposition.

8. References

[B] G. Birkhoff, *Lattice Theory*, American Mathematical Society, Providence, R.I., 1967.

[GR] G. Gratzer, *Lattice Theory*, W.H. Freeman, 1971.

[GS] D. Gale, and L. Shapley. College Admissions and the Stability of Marriage. Am. Math. Monthly 69 (1962).

[GS84] D. Gale, and M. Sotomayor. Some Remarks on the Stable Matching Problem. Discrete Applied Math (to appear).

[G] D. Gusfield, Three Fast Algorithms for Four Problems in Stable Marriage. Yale Computer Science technical report #407, July 1985.

[GILS] D. Gusfield, R. Irving, P. Leather, M. Saks. Every Finite Distributive Lattice is a Set of Stable Matchings for a *Small* Stable Marriage Instance. Yale Computer Science technical report #434, September 1985.

[I] R. Irving. An Efficient Algorithm for the Stable Room-mates Problem. J. of Algorithms, to appear.

[IL] R. Irving and P. Leather. The complexity of counting stable marriages. SIAM J. on Computing, to appear.

[ILG] R. Irving, P. Leather and D. Gusfield. An algorithm for the optimal stable marriage problem. July 1985.

[K] D. E. Knuth. *Mariages Stables* (in French). Les Presses de L'Universite de Montreal, 1976.

[L] E. Lawler. *Combinatorial Optimization: networks and matroids*. Holt, Rhinehard and Winston, 1976.

[PTW] G. Polya, R.E. Tarjan, D.R. Woods. *Notes on Introductory Combinatorics*, Birkhauser Verlag, Boston, 1983.

[Q] M. Quinzil. Core and Competitive Equilibria with Indivisibilities. International J. of Game Theory, Vol. 13 #1, 1984.

[R] A. Roth. Common and Conflicting Interests in Two-Sided Matching Markets. Department of Economics, University of Pittsburgh, working paper #186, Nov. 1984.

9. Acknowledgements

There are many people I want to thank. First, I thank the students in the fall 1985 Yale computer science 260 class, who generated many useful execution trees at the beginning of this research; the trees of A. Zoler and E. Winters were particularly helpful, the latter demonstrating a counter-example to an early conjecture of mine. I thank David Warren who acted as a sounding-board, and who contributed the version of phase 1 of algorithm I that appears in this paper. I also thank all the people in the Yale theory group, particularly Dana Angluin, who listened to many parts of this work as it developed. Finally, I thank Carrie Shepard, my ex-roommate. Together, in the midst of deriving and writing these results, we put theory into practice, demonstrating that roommate stability does indeed lead to marriage.

1	7	2	6	8	5	3	4
2	4	6	5	3	8	1	7
3	5	2	1	7	4	6	8
4	1	7	3	6	5	8	2
5	7	1	8	4	6	2	3
6	7	3	8	4	5	1	2
7	2	8	4	3	5	6	1
8	4	2	3	5	6	7	1

1a. 8 person preference lists.

(1,6)	(1,5)	(1,4)
(2,3)	(2,3)	(2,3)
(7,4)	(7,4)	(7,5)
(8,5)	(8,6)	(8,6)

1b. Three stable roommate assignments. The "men" are {1, 2, 7, 8} and the "women" are {3, 4, 5, 6}.

1	6	5	4	
2	3			
7	4	5		
8	5	6		

3	2		
4	1	7	
5	7	1	8
6	8	1	

1c. mens preference lists womens preference lists.

Any woman missing from a man's list may be put at the end, but are irrelevant in this stable marriage instance.

Figure 1.

1	2	6	5	3	4		
2	6	5	3	8	1		
3	5	2	1	7	4	6	
4	1	7	3	6	5	8	
5	7	1	8	4	6	2	3
6	3	8	4	5	1	2	
7	8	4	3	5			
8	4	2	5	6	7		

2a. phase 1 table

E_1	H_1	S_1
1	2	6
2	6	5
3	5	2

E_2	H_2	S_2
4	1	7
5	7	1

2b. Rotation $R_1 = (E_1, H_1, S_1)$ both exposed in the phase 1 table

Rotation $R_2 = (E_2, H_2, S_2)$

In the phase 1 table 6, 7, 8 is a tail of R_1, and R_2 has no tail.

Figure 2

1	6 5 3 4
2	5 3
3	2 1 7 4 6
4	1 7 3 6 5 8
5	7 1 8 4 6 2
6	3 8 4 5 1
7	8 4 3 5
8	4 5 6 7

2c. table after eliminating R_1 from the phase 1 table.

Note that R_2 is still exposed, and now has a tail of 3.

Rotation $R_3 = (E_3, H_3, S_3)$ where $E_3 = \{2, 6, 7, 8\}$ is now also exposed.

1	6 5
2	3
3	2
4	7
5	1 8
6	8 1
7	4
8	5 6

Table after elimination of R_3.

Now R_4, R_5 are exposed, where

$E_4 = \{1, 8\}$ $E_5 = \{5, 6\}$

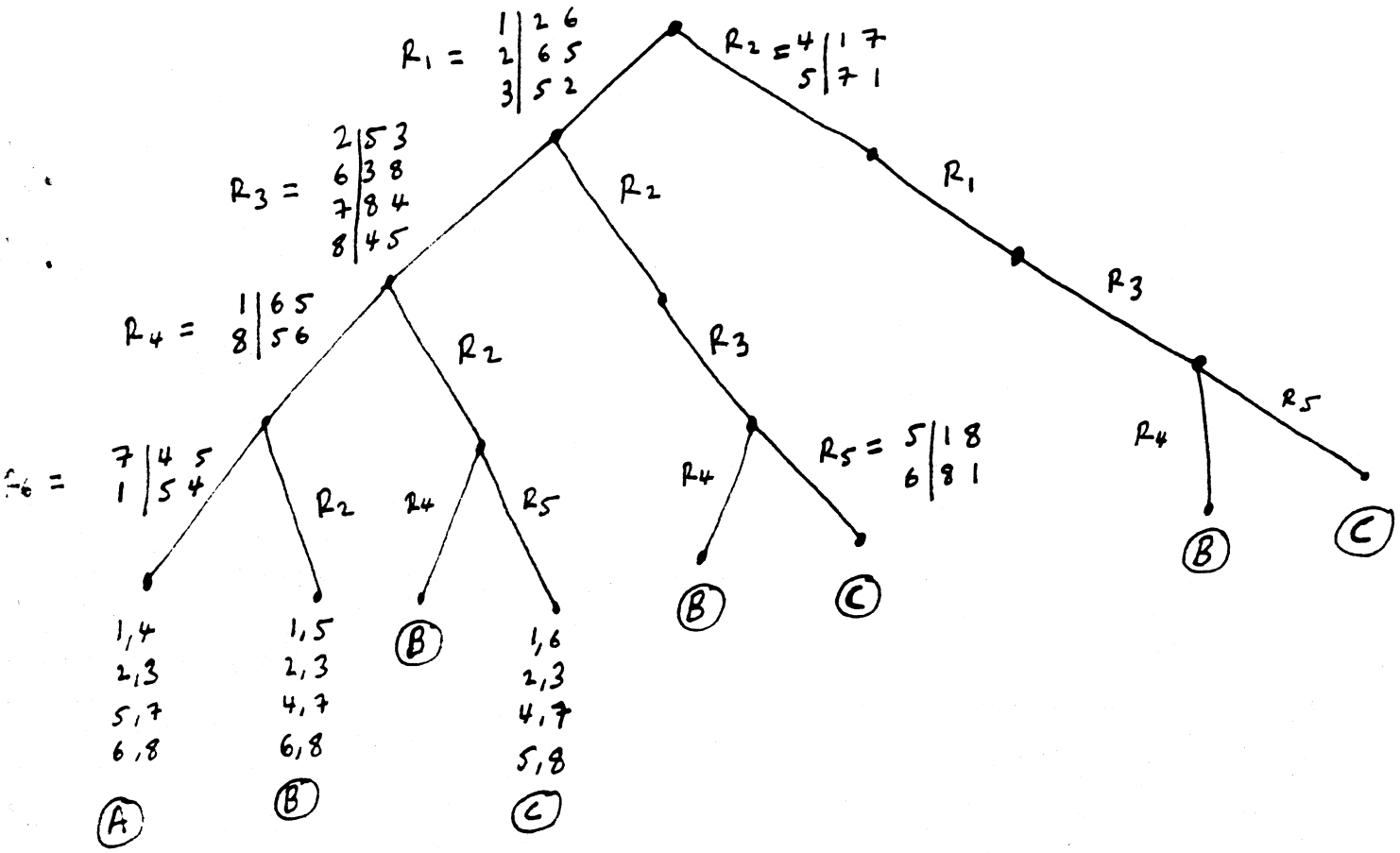
1	6 5
2	5 3
3	2 4 6
4	7 3 6 5 8
5	1 8 4 6 2
6	3 8 4 5 1
7	8 4
8	4 5 6 7

2d. Table after eliminating R_2.

R_3 is the only exposed rotation

1	5
2	3
3	2
4	7
5	1
6	8
7	4
8	6

Table after elimination of R_4.



Tree D for the example.
The tables at the nodes are not shown.
All paths have length 4. (R_4, R_5) and (R_2, R_6) are each a dual pair of rotations. R_1 and R_3 have no duals.

Figure 3.

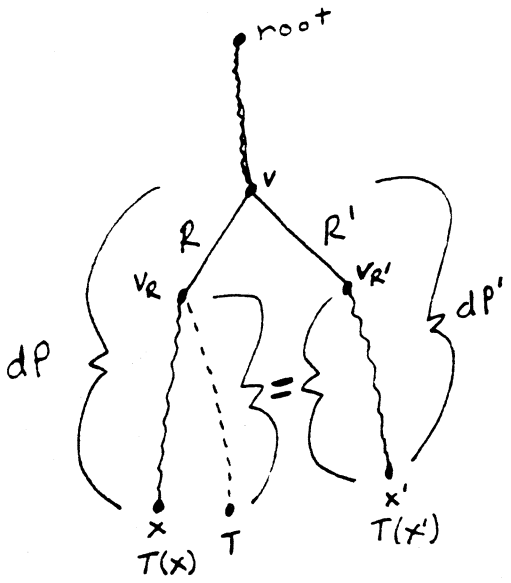
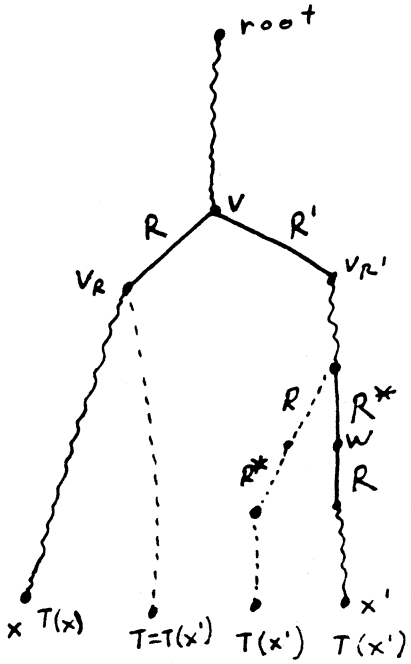
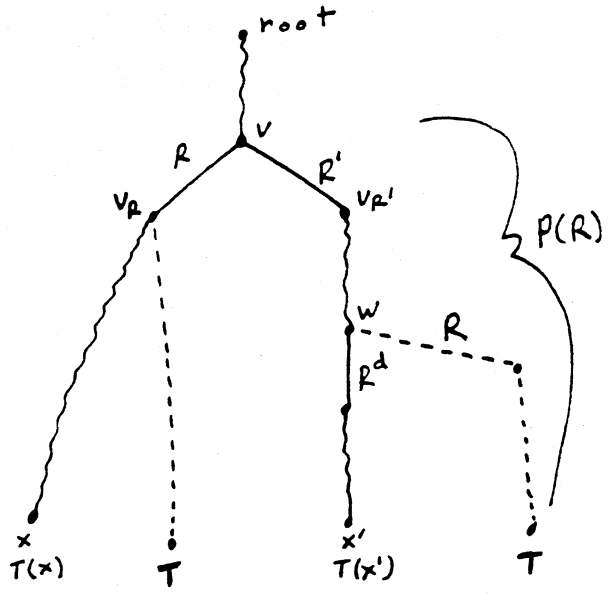


Figure 4a.

Proof of Lemma 4.2, when $R' = R^d$. Wavy lines are known paths in D, solid edges are known single edges in D; labelled dashed lines are inferred edges, and unlabelled dashed lines are inferred paths in D.



4b. $R' \neq R^d$ and R is on dP'.



4c. $R' \neq R^d$ and R^d is on dP'.

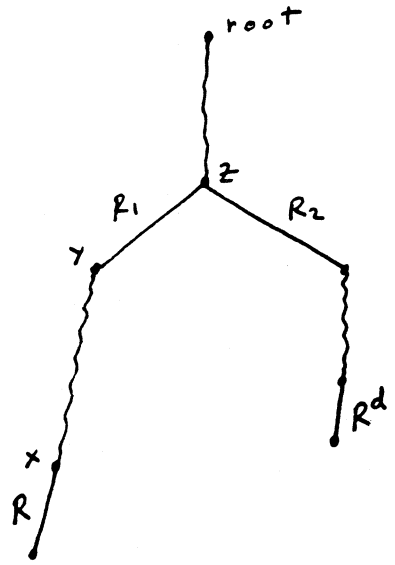
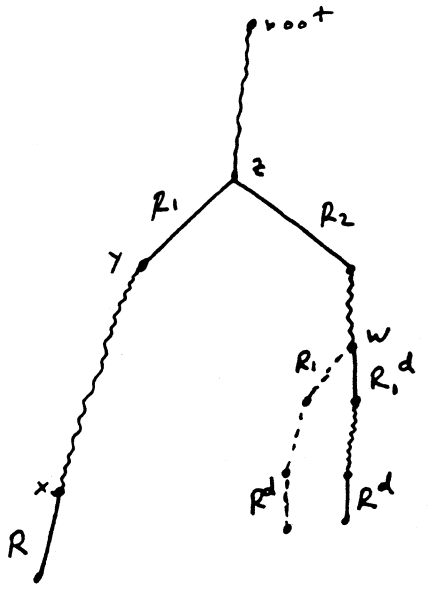


Figure 5a.



5b. R^d is on P after w

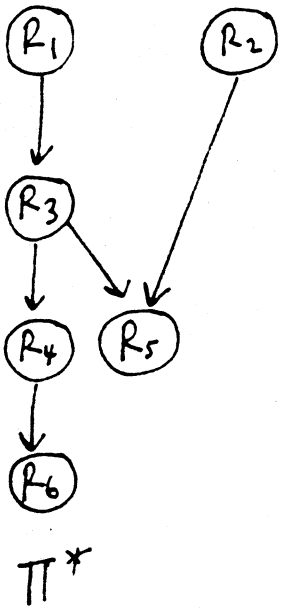


Figure 6.

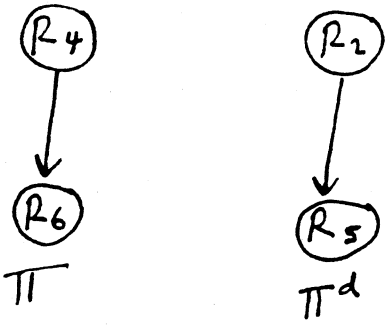


Figure 7. The dual partial orders.
Recall that $R_5 = R_4^d$ and $R_2 = R_6^d$.