RECOGNIZING CIRCLE GRAPHS IN POLYNOMIAL TIME

Csaba P. Gabor
Wen-Lian Hsu
Kenneth J. Supowit

CS-TR-059-86

July 1986

Recognizing Circle Graphs in Polynomial Time

Csaba P. Gabor *

Wen-Lian Hsu **

Kenneth J. Supowit *

July 10, 1986

Abstract. Our main result is an $O(|V| \times |E|)$ time algorithm for deciding whether a given graph is a circle graph, that is, the intersection graph of a set of chords on a circle. Our algorithm utilizes two new graph-theoretic results, regarding necessary induced subgraphs of graphs having neither articulation points nor similar pairs of vertices. Furthermore, as a substep of our algorithm, we show how to find in $O(|V| \times |E|)$ time a decomposition of a graph into prime graphs, thereby improving on a result of Cunningham.

---

* Department of Computer Science, Princeton University. The work of these authors was supported in part by the National Science Foundation under grant DMC-8451214, and by grants from Hewlett-Packard Co. and the IBM Corp.

** Department of Industrial Engineering and Management Sciences, Northwestern University. The work of this author was supported in part by the National Science Foundation under grant DMS-8504691.

1. Introduction

An undirected graph G is called a *circle graph* if there exists a set of chords C on a circle and a one-to-one correspondence between vertices of G and chords of C such that two distinct vertices are adjacent if and only if their corresponding chords intersect. Such a set C is called a *(circle-graph) model* for G. Fig. 1 shows a circle graph, along with a model for it. Fig. 2 shows a graph that is *not* a circle graph. Our main result is an algorithm that, given a graph $G = (V, E)$, decides in $O(|V| \times |E|)$ time whether G is a circle graph, and if so outputs a model for it.

Our circle-graph recognition algorithm utilizes two new graph-theoretic results, which may also be of interest in their own right. Say that a pair of vertices in a graph is *similar* if each third vertex is adjacent to both of them or neither of them. We show that a graph having no similar pairs must have an induced subgraph isomorphic to a certain graph (Theorem 1). We then use this result to show that a graph having neither articulation points nor similar pairs must have an induced subgraph isomorphic to some member of a certain family of graphs (Theorem 2).

Another interesting result that we prove and utilize here is that a decomposition of a graph into prime elements, with respect to the join decomposition [CE], can be found in $O(|V| \times |E|)$ time for undirected graphs. This improves on the $O(|V|^3)$ algorithm of [Cu].

Circle graphs, also known as overlap graphs, were introduced in [EI]. Given a model, the maximum clique and the maximum independent set problems on its corresponding graph can be solved in polynomial time ([Ga], [Hs], [RU]), but the coloring problem is NP-hard [GJ]. The recognition problem for circle graphs was posed as open in [GJ], [Go].

Polynomial-time algorithms have been found for recognizing permutation graphs ([PL], [Sp]), as well as for circular-arc graphs [Tu], which are defined as the intersection graphs of the matching diagram of a permutation, and of a set of arcs of a circle, respectively. These and other types of intersection graphs are discussed in [Go]. Note that the recognition problem for circle graphs is at least as hard as that for permutation graphs. More precisely, given a graph G we can add a new vertex adjacent to each other vertex to get a new graph G ; it is easily seen that G is a permutation graph if and only if G is a circle graph.

A preliminary version of this paper appeared in [GH]. An $O(|V|^5)$ time algorithm for recognizing circle graphs was independently discovered by Bouchet [Bo].

2. Overview of the algorithm

The idea of the algorithm is to attempt to build up a model C for the given graph $G = (V, E)$ by adding one chord at a time. Thus, after each iteration, C is a model for the subgraph induced by some $W \subseteq V$. At all times, we maintain a placement for each vertex v not yet in W, but adjacent to at least one member of W. In particular, we associate with v a pair of arcs $\{\phi_1, \phi_2\}$ with the property that the chord for v in any model for G must have one endpoint in ϕ_1 and one in ϕ_2 (more precisely, the pair $\{\phi_1, \phi_2\}$, which we refer to as ch(v), is a "necessary placement" for v relative to C, as defined in Section 3 below). Furthermore, at all times, C is essentially the only model for the subgraph induced by W that could be contained by a model for the entire graph G (more precisely, C is a "necessary model" for G/W, as defined in Section 3).

At each iteration, we add to W a vertex w chosen arbitrarily from among those not in W but adjacent to at least one member of W. We also add a chord to C corresponding to w whose endpoints lie in the two members of ch(w), respectively. We then must update ch(v) for various vertices $v \in V - W$, since our model C is now more detailed. In particular, for each vertex v such that ch(v) contained some $\phi \in$ ch(w), we now must be more specific about where w must be placed. That is, the arc ϕ was split into two subarcs by the new chord for w; we must update ch(v) by replacing ϕ by one of these two subarcs. Furthermore, we must compute ch(v) for each vertex v adjacent to w but to no other members of W, since previously such ch(v) were undefined. We must perform the updates to these various ch(v) in such a way that they remain necessary placements relative to C.

The algorithm terminates either when $W = V$ (in which case C is a model for the entire graph G) or when we find that a chord with endpoints in ch(w) -- where w is the chosen vertex -- does not intersect precisely those chords in C corresponding to vertices adjacent to w (in which case G is not a circle graph, since ch(w) is necessary).

To facilitate this process, we first do some pre-processing to remove certain degeneracies from G. In particular, we first find the connected components of G. It is easily seen that G is a circle graph if and only if each of its connected components are, and that a model for G can be obtained in $O(|V|)$ time from models for those connected components. Therefore we assume in the remainder of this paper that G is connected. Next we decompose G into components which are prime with respect to Cunningham's join decomposition [CE], [Cu] (defined in Section 3 below). Again, G is a circle graph if and only if each of its prime components are, and a model for G can be obtained quickly from models for those connected components (see Lemma 1).

After decomposing into prime elements, we assume that G is prime. The next major task is to start off the iterative process, that is, to find some initial W and a necessary model C for it and the necessary placements ch(v) relative to C. It turns out (Theorem 2) that each graph containing neither similar pairs nor articulation points must contain as an induced subgraph at least one member of a certain family **F** of graphs (see Fig. 9). This is interesting because prime graphs have neither similar pairs nor articulation points (Lemma 2) and because each member of **F** is easily seen to be prime and are "uniquely representable" circle graphs (defined in Section 3), which essentially means that each of these graphs has only one model up to rotations, reflections and order-preserving mappings of the circle. Furthermore, this induced member of **F** can be found quickly.

Section 3 contains much of the definitions and notation used in the paper. Lemmas 1 and 2 (which pertain to primeness) as well as the improved algorithm for finding a prime decomposition are contained in Section 4. Section 5 contain the algorithm for finding a W that induces a member of **F** within a graph without similar pairs or articulation points. It also presents another result (used as a substep of that algorithm) for finding a particular graph (see Fig. 7) within a graph not containing similar pairs. Section 6 contains the algorithm for computing necessary placements relative to this initial W. Section 7 contains the main, iterative part of the overall algorithm. A subroutine used in Section 6, which we call Algorithm A, is also used in Section 7. An interesting corollary (Section 8) of this work is that primeness and unique representability are equivalent notions for circle graphs.

The complexity analyses of the various steps are given in the sections in which they appear. Performing the prime decomposition takes $O(|V| \times |E|)$ time. Finding a member of **F** takes $O(|E|)$ time. Algorithm A can be performed in $O(|E|)$ time. The initial placement (Section 6) consists of $|W|$ calls to Algorithm A, and the main iteration (Section 7) consists of at most $|V| - |W|$ calls to Algorithm A. Hence the total time used by the algorithm is $O(|V| \times |E|)$.

3. Definitions and Notation

All graphs considered in this paper are undirected, with no self-loops. Let $G = (V, E)$ be a graph and $W \subseteq V$. Then G/W is the subgraph of G induced by W. If $v \in V$, then $N_W(v) = \{w \in W : (w, v) \in E\}$, i.e. it is the neighborhood of v restricted to W. Vertices $a, b \in V$ are said to be W-similar if $N_W(a) - \{b\} = N_W(b) - \{a\}$. Thus, vertices that are V-similar are what we called "similar" above.

A *path* in G is an ordered set of vertices $p_1, p_2, ..., p_s$ such that $(p_i, p_{i+1}) \in E$ for all $i, 1 \leq i < s$. The path is *primitive* if in addition $(p_i, p_j) \notin E$ for all $i, j, |j-i| \geq 2$.

We consider only sets of chords whose endpoints are all distinct; this is without loss of generality because if G is a circle graph then it has a model with all endpoints distinct. We also find it convenient to consider only sets of chords on a particular circle, say the unit circle centered at the origin (thus we will refer to "the circle").

We now give a more precise definition of unique representability. Let C be a set of chords on the circle. Then $G(C)$ denotes the graph for which C is a model, and let $\{v_1, v_2, ..., v_n\}$ be its vertices. Let $\pi(C)$ denote the permutation on the multiset $\{v_1, v_1, v_2, v_2, ..., v_n, v_n\}$, obtained by traversing (starting at some arbitrary point) the $2n$ endpoints of chords of C clockwise around the circle, and at each such endpoint reporting the vertex of $G(C)$ corresponding to the chord of C that contains it. If $\pi = (\pi_0, \pi_1, ..., \pi_k)$ is a permutation then a *shift* of π results in the permutation $(\pi_k, \pi_0, \pi_1, ..., \pi_{k-1})$, and a *reverse* of π results in the permutation $(\pi_k, \pi_{k-1}, ..., \pi_0)$. If C_1 and C_2 are circle-graph models then we write $C_1 \sim C_2$ if $\pi(C_2)$ can be obtained from $\pi(C_1)$ by a sequence of shift and reverse operations. Note that $\sim$ is an equivalence relation. It is easily seen that $C_1 \sim C_2$ implies that $G(C_1)$ is isomorphic to $G(C_2)$. A circle graph G is *uniquely representable* if each two models C_1 and C_2 for it satisfy $C_1 \sim C_2$.

A model D for G/W is *necessary* if either G is not a circle graph or each model for G has a subset D' corresponding to the vertices of W such that $D' \sim D$. The endpoints of the chords of D define $2|D|$ $(= 2|W|)$ *empty arcs* on the circle, each containing none of these endpoints. A *placement* of a vertex $v \in V - W$ relative to D is a pair $\{\phi_1, \phi_2\}$ of these empty arcs. This placement of v is *necessary* if $D \cup \{d\}$ is a necessary model for $G/(W \cup \{v\})$, where d is some chord having one endpoint in ϕ_1 and the other in ϕ_2.

Finally, we review the notion of the join composition, following [CE], [Cu]. Let $G = (V, E)$ be a graph such that V can be partitioned into V_0, V_1, V_2 and V_3 such that

$$V_1 \times V_2 \subseteq E,$$

$$E \cap ((V_0 \times V_2) \cup (V_0 \times V_3) \cup (V_1 \times V_3)) = \emptyset,$$

and

$$|V_0 \cup V_1|, |V_2 \cup V_3| \geq 2$$

(see Fig. 3). Let $G_1 = G/(V_0 \cup V_1 \cup \{m_1\})$ for some $m_1 \in V_2$. Analogously, let $G_2 = G/(\{m_2\} \cup V_2 \cup V_3)$ for some $m_2 \in V_1$. Then we say that $\{G_1, G_2\}$ is a *simple decomposition* of G, yielded by the partition $\{V_0, V_1, V_2, V_3\}$. A *(general) decomposition* of a graph G is defined inductively to be either $\{G\}$ or a set of graphs obtained from a decomposition $\mathbf{G}$ of G by replacing some $H \in \mathbf{G}$ by the members of a simple decomposition of H. A graph is *prime* if it has no simple decomposition. A decomposition is *prime* if each of its elements is prime.

4. Decomposition into prime elements

The following result enables us to restrict our attention to prime graphs.

LEMMA 1: Let $\{G_1, G_2\}$ be a simple decomposition of G. Then G is a circle graph if and only if G_1 and G_2 both are. Furthermore, given models for G_1 and G_2, respectively, a model for G can be constructed in $O(|V|)$ time.

PROOF: If G is a circle graph then G_1 and G_2 are as well (since they are induced subgraphs).

To prove the converse, assume that G_1 and G_2 are circle graphs, with models C_1 and C_2, respectively. Let

$$\pi(C_1) = (m_1, A_1, m_1, B_1)$$

and

$$\pi(C_2) = (m_2, A_2, m_2, B_2),$$

where m_1 and m_2 are as specified in the definition of decomposition above, and where, for $i = 1,2$, A_i (resp. B_i) denotes the subsequence of vertices in $\pi(C_i)$ appearing after the first (resp. second) occurrence of m_i.

Then the sequence

$$(A_1 A_2 B_1 B_2)$$

can be obtained from a traversal of a set of chords (see Fig. 4) that, as is easily verified, is a circle model for G. **QED**

Thus our circle-graph recognition algorithm first finds a prime decomposition $\mathbf{G}$ of G, then decides whether each of its elements is a circle graph. If so, then it constructs a model for G in a pairwise manner from the models for the elements of $\mathbf{G}$; otherwise it declares that G is not a circle graph. Thus we assume from now on that the input graph G is prime.

The following properties of primeness will be used extensively.

LEMMA 2: If $G = (V, E)$ is a prime graph having at least five vertices then (1) G has no articulation points, and (2) G has no similar pairs.

PROOF: (1) If G has an articulation point v, then let $\{A, B\}$ be a partition of $V - \{v\}$ sets such that $|B| > |A| \geq 1$ and the removal of v from G leaves A disconnected from B. Then the partition of V into

$$V_0 = A$$

$$V_1 = \{v\}$$

$$V_2 = N_B(v)$$

$$V_3 = B - N_B(v)$$

is a simple decomposition of G.

(2) If a, $b \in V$ are a similar pair then the partition of V into

$$V_0 = \varnothing$$

$$V_1 = \{a, b\}$$

$$V_2 = N_V(a) - \{b\}$$

$$V_3 = V - (N_V(a) \cup \{a, b\})$$

is a simple decomposition of G. **QED**

We now present an O time algorithm to find a prime decomposition of an undirected graph G, thus improving on the $O(|V|^3)$ time algorithm of [Cu]. As is shown in [Cu], this problem can be solved by making $O(|V|)$ calls to a subroutine that solves the following problem:

INPUT: a graph $G = (V, E)$ and an edge $(x, y) \in E$.

OUTPUT: a partition $\{V_0, V_1, V_2, V_3\}$ of V yielding a decomposition, such that $x \in V_1$ and $y \in V_2$, if such a partition exists; otherwise output "no".

An $O(|V|^2)$ time algorithm to solve this problem is given in [Cu]; our algorithm, which runs in $O(|E|)$ time, is shown in Fig. 5. We maintain a partition $\{S, T\}$ of V, such that $x \in S$ and $y \in T$. We try to construct these sets so that there is a partition $\{V_0, V_1, V_2, V_3\}$ of V yielding a decomposition, where $S = V_0 \cup V_1$ and $T = V_2 \cup V_3$; if such a decomposition exists then we refer to the set $\{S,\ T\}$ as a *split.* Initially S contains only x and one other (arbitrarily chosen) vertex w, and T contains all other vertices. Define a *violation* as a pair $\{s,\ t\}$ such that $s \in S$, and $t \in T$ and one of the following four cases is true (see Fig. 6)

1. $(s,\ t) \in E, \quad (s,\ y) \notin E, \quad (t,\ x) \in E.$
2. $(s,\ t) \in E, \quad (s,\ y) \in E, \quad (t,\ x) \notin E.$
3. $(s,\ t) \in E, \quad (s,\ y) \notin E, \quad (t,\ x) \notin E.$
4. $(s,\ t) \notin E, \quad (s,\ y) \in E, \quad (t,\ x) \in E.$

It is easily verified that if there is a violation then $\{S,\ T\}$ is not a split. The algorithm iteratively looks for violations, and whenever it finds one, eliminates it by moving the member of T involved in the violation into S. We use a set U to hold all vertices s that have been moved into S, but that we have not yet examined to see whether there is a $t \in T$ with which s forms a violation.

```
(1)  w ← some element of V − {x, y} ;
(2)  S ← {x, w} ;
(3)  T ← V − S ;
(4)  U ← {w} ;
(5)  WHILE U ≠ ∅ DO
(6)      BEGIN
(7)        s ← some member of U ;  U ← U − {s} ;
(8)        [[ Look for violations of the form {s, t} where t ∈ T ]]
(9)          IF (s, y) ∈ E
(10)             THEN FOR each t ∈ N_T(s) ∪ N_T(x) DO
(11)                    IF {s, t} is a violation THEN
(12)                       BEGIN
(13)                          T ← T − {t} ;
(14)                          S ← S ∪ {t} ;
(15)                          U ← U ∪ {t}
(16)                       END
(17)             ELSE FOR each t ∈ N_T(s) DO
(18)                    BEGIN
(19)                       [[ {s, t} must be a violation ]]
(20)                         T ← T − {t} ;
(21)                         S ← S ∪ {t} ;
(22)                         U ← U ∪ {t}
(23)                    END;
(24)     END;
(25)   [[ Now there are no violations ]]
(26)   IF |T| > 1
(27)      THEN BEGIN
(28)              V_1 ← {s ∈ S : (s, y) ∈ E};
(29)              V_0 ← S − V_1;
(30)              V_2 ← {t ∈ T : (t, x) ∈ E};
(31)              V_3 ← T − V_2;
(32)              output {V_0, V_1, V_2, V_3}
(33)           END
(34)      ELSE
(35)        Interchange x and y and repeat the WHILE loop with S = {y, w} and T = {x}.
(36)        Again, if |T|>1 then output a partition as in steps (28)-(32); otherwise output "no".
```

Fig. 5

The algorithm to check for a decomposition splitting a particular edge.

PROOF OF CORRECTNESS:

If there is a split $\{S,\ T\}$ with $x \in S,\ y \in T$ then either $w \in S$ or $w \in T$. Hence it suffices to check that the algorithm correctly determines whether there is a split $\{S,\ T\}$ with $\{x,\ w\} \in S,\ y \in T$. We show first that after the termination of the WHILE loop there is no violation. Assume for a contradiction that there were such a violation $\{s,\ t\}$. Since every element of S other than x was removed from U at some point, we can consider the point at which s was removed from U. If $(s,\ y) \in E$ then (since $\{s,\ t\}$ is a violation) we must have that $t \in N_T(s) \cup N_T(x)$. But this implies that the violation $\{s,\ t\}$ would have been detected in step (10), and therefore t would have been included in S, a contradiction. On the other hand, if $(s,\ y) \notin E$ then the violation $\{s,\ t\}$ would have been detected in step (17) and therefore t would have been included in S, again a contradiction.

Now if $|T| > 1$ then it is easy to verify that the partition $\{V_0,\ V_1,\ V_2,\ V_3\}$ yields a decomposition. On the other hand, if there is a split $\{S\ ,\ T\ \}$ with $x \in S\ ,\ y \in T\ $ then no element of $S\ $ could constitute a violation with an element of $T\ $.

ANALYSIS OF TIME COMPLEXITY:

For each vertex $v \in V$, we maintain a doubly-linked list of its neighbors in T (i.e. the set $N_T(v)$). When an element t is removed from T, it must be deleted from each of the deg(t) such lists. To facilitate this, we also maintain, for each $t \in T$, a list of the nodes corresponding to it in these lists; thus there is one such node (in the list for $N_T(v)$) for each v adjacent to t.

Note that no vertex is removed from U more than once. Furthermore, we claim that whenever an element s is removed from U, the amount of time spent in checking for violations involving s (which is proportional to the number of executions of the FOR loops in steps (10) and (17)) is

$$O(\ \deg(s) + \#\text{ of violations found involving } s)$$

To see this, note that step (10) examines some $t \in T$ either when $t \in N_T(s)$ (which happens for at most deg(s) values of t), or when $t \in N_T(x) - N_T(s)$ (which implies that $\{s, t\}$ is a violation, of type 4). To enumerate the elements in the set $N_T(x) - N_T(s)$ efficiently, we first traverse the list $N_T(s)$, marking each element; we then traverse the list $N_T(x)$ reporting each unmarked element. These two traversals take O(deg(s) + # of violations found involving s) time. Finally, step (17) executes $|N_T(s)|$ ($\leq$ deg(s)) times.

Thus, the running time of the algorithm is at most

$$\sum_{s \in V - \{x,y\}} O(\deg(s) + \text{\# of violations found involving } s)$$

$$= O(|V| + |E|) \;=\; O(|E|),$$

since the number of violations found is at most $|V| - 2$ (since each such violation causes an element to be removed from T).

In summary, the running time of algorithm of [Cu] to find a prime decomposition is dominated by O($|V|$) calls to this algorithm; thus its total time is O($|V| \times |E|$).

5. Finding a member of F

We assume throughout the remainder of the paper that G is prime and has at least five vertices. Recall (Lemma 2) that this implies that G has neither similar pairs nor articulation points. We begin by showing (Theorem 1) that G contains, as an induced subgraph, the graph pictured in Fig. 7 (defined below). This will help us in showing (Theorem 2) that G must contain, as an induced subgraph, one of the graphs pictured in Fig. 9 (also defined below). These theorems are constructive, and indeed the required induced subgraph can be found in $O(|E|)$ time.

DEFINITIONS: a graph is called a P4 if it is isomorphic to the graph with vertices $\{a, b, c, d\}$ and edges $\{(a, b), (b, c), (c, d)\}$ (see Fig. 7).

THEOREM 1: Let $G = (V, E)$, $|V| \geq 2$, be a graph having no similar pairs. Then some induced subgraph of G is a P4. Furthermore, such a subgraph can be found in $O(|E|)$ time. (Note that the absence of similar pairs implies $|E| = \Omega(|V|)$.)

PROOF: For all $k \geq 0$, let H_k denote the graph with vertices $\{a_i : 0 \leq i \leq 2k+1\}$, and edges $\{(a_i, a_j) : i \neq j$ and j is even and (i is even or $i > j$)$\}$ (see Fig. 8). Our algorithm iteratively finds an induced subgraph of G isomorphic to H_k for $k = 0, 1, \cdots$ until eventually finding an induced P4.

More precisely, we initially chose some edge $(w_0, w_1) \in E$ (E cannot be empty since otherwise each vertex of G would be isolated and hence, since $|V| \geq 2$, G would have a similar pair). Thus $G/\{w_0, w_1\}$ is isomorphic to H_0.

Now assume that we have found a set $W = \{w_j : 0 \leq j \leq 2k+1\} \subseteq V$ that induces a subgraph of G isomorphic to H_k, for some $k \geq 0$, where w_j corresponds to a_j under the isomorphism. Note that since w_{2k} and w_{2k+1} are W-similar there must exist some $z_0 \in V - W$ adjacent to exactly one of them (otherwise w_{2k} and w_{2k+1} would be V-similar). Assume that z_0 is adjacent to w_{2k} but not to w_{2k+1}; this is without loss of generality by the W-similarity of w_{2k}

and w_{2k+1}. Now if z_0 is adjacent to w_{2j+1} for some j, $0 \leq j < k$, then $\{w_{2j+1}, z_0, w_{2k}, w_{2k+1}\}$ induces a P4 (and so we halt and output it). Otherwise if z_0 is not adjacent to w_{2j} for some j, $0 \leq j < k$, then $\{z_0, w_{2k}, w_{2j}, w_{2j+1}\}$ induces a P4. So assume that neither of these two cases hold.

At this point we know that z_0 and w_{2k+1} are $W \cup \{z_0\}$-similar. Therefore there must be some $z_1 \in V - (W \cup \{z_0\})$ adjacent to exactly one of $\{z_0, w_{2k+1}\}$, for otherwise z_0 and w_{2k+1} would be similar. Since z_0 and w_{2k+1} are $W \cup \{z_0\}$-similar, it is without loss of generality to assume that z_1 is adjacent to z_0 but not to w_{2k+1}. Now if z_1 is not adjacent to w_{2j} for some j, $0 \leq j \leq k$, then $\{z_1, z_0, w_{2j}, w_{2k+1}\}$ induces a P4. Otherwise if z_1 is adjacent to w_{2j+1} for some j, $0 \leq j < k$, then $\{w_{2j+1}, z_1, w_{2k}, w_{2k+1}\}$ induces a P4.

Thus if we have not yet found an induced P4 in G then the set $W \cup \{z_0, z_1\}$ induces a subgraph isomorphic to H_{k+1}.

By induction, the algorithm must find an induced P4 in G, since otherwise it finds an induced subgraph in G isomorphic to H_k for each $k \geq 0$, contradicting the finiteness of G.

We now describe how to implement the algorithm in the specified time bound. At the kth iteration we must find a vertex z_0 adjacent to w_{2k}, but not to w_{2k+1}, or vice versa. To do this, we can traverse the adjacency list for w_{2k}, placing a marker in the slots corresponding to its elements in an array representing all the members of V. Then we traverse the adjacency list for w_{2k+1}; if we ever find an unmarked vertex in it, we halt and return that as z_0 (since it is adjacent to w_{2k+1} but not to w_{2k}). Similarly, we can search for a vertex adjacent to w_{2k} but not to w_{2k+1}). We check whether z_0 is adjacent to w_{2j+1} for some j, $0 \leq j < k$, by scanning the adjacency list for z_0. We check whether z_0 is not adjacent to w_{2j} for some $0 \leq j < k$ by scanning the adjacency list for z_0 and marking off the even-indexed elements of W as they are encountered --- thus the work is O(deg(z_0)), unless this is the last iteration, in which case it is O($|V|$). Finding z_1 and checking its adjacencies is done analogously. Therefore the total time required after the sorting is $O(|E|)$.

Since we have shown that G has an induced P4, another way to find it is to use the algorithm of [CP] that finds an induced P4 (whenever it exists) in a graph (V, E) in $O(|V| + |E|)$ time.

QED Theorem 1

DEFINITIONS: A graph is a *house* if it is isomorphic to the graph with vertices $\{a, b, c, d, e\}$ and edges $\{(a, b), (a, c), (b, d), (c, d), (c, e), (d, e)\}$. A graph is a *tepee* if it is isomorphic to the graph with vertices $\{a, b, c, d, e\}$ and edges $\{(a, b), (b, c), (c, d), (a, e), (b, e), (c, e), (d, e)\}$. A graph is a *figure-8* if it is isomorphic to the graph with vertices $\{a, b, c, d, e, f\}$ and edges $\{(a, b), (a, c), (b, d), (c, d), (c, e), (d, f), (e, f)\}$. A graph is a *primitive k-cycle* if it is isomorphic to the graph with vertices $\{a_1, a_2, \cdots, a_k\}$ and edges $\{(a_1, a_2), (a_2, a_3), \ldots, (a_{k-1}, a_k), (a_k, a_1)\}$ (see Fig. 9). Let

$$\mathbf{F} = \{G : G \text{ is a house, a tepee, a figure-8 or a primitive } k\text{-cycle for some } k \geq 5\}.$$

The key fact, as is easily verified, is that each member of **F** has at least 5 vertices, is prime, and is a uniquely representable circle graph. Furthermore, we can efficiently find a member of **F** within G because of the following result.

THEOREM 2: Let $G = (V, E)$, $|V| \geq 2$, be a graph having neither similar pairs nor articulation points. Then some induced subgraph of G is a member of **F**. Furthermore, such a subgraph can be found in $O(|E|)$ time.

PROOF: Our algorithm to find such an induced subgraph first finds a P4 (Step 1). Then in each of Steps 2 through 9, we test for some condition and if it is satisfied then we can find a member of **F** and halt; otherwise we continue, and utilize the falsity of the condition in subsequent steps. Finally (Step 10), if none of the conditions has been met, then we can find a set $U \subseteq V$ such that (1) there is a particular vertex w that is adjacent to each member of U, (2) $|U| \geq 2$, and (3) U contains no U-similar pairs. Therefore we can simply apply Theorem 1 to the subgraph G/U to obtain another P4; this P4 together with w induces a tepee.

More precisely, the algorithm is:

Step 1: Find vertices $a,\ b,\ c,\ d \in V$ that induce a P4, where $(a,\ b)$, $(b,\ c)$, $(c,\ d) \in E$, by means of the algorithm in the proof of Theorem 1.

Step 2: If there exists $v \in V$ adjacent to both a and d then output the subgraph induced by $\{a,\ b,\ c,\ d,\ v\}$ (which must be a house, tepee or primitive 5-cycle) and halt.

Step 3:

$$B \leftarrow \{b \ \in V : (a,\ b\ ) \in E \text{ and there exists } c\ \in V \text{ such that } (b\ ,c\ ),\ (c\ ,d) \in E\};$$

$$C \leftarrow \{c \ \in V : (d,\ c\ ) \in E \text{ and there exists } b\ \in V \text{ such that } (a,\ b\ ),\ (b\ ,\ c\ ) \in E\}.$$

If there exist vertices $b_1 \in B,\ c_1 \in C$ such that $(b_1, c_1) \notin E$ then we can find an induced member of **F** as follows. Since $b_1 \in B,\ c_1 \in C$, there exist vertices $b_2,\ c_2$ such that $(a, b_2), (b_2, c_1) \in E$, and $(b_1, c_2), (c_2, d) \in E$ (Fig. 10.1). Note that Step 2 ensures that $(a, c_2), (b_2, d) \notin E$. There are eight cases, depending on which subset of $\{(b_1, b_2), (b_2, c_2), (c_1, c_2)\}$ are contained in E. The reader may verify that there is an induced member of **F** in each case.

Thus

FACT 1: for all $b_1 \in B,\ c_1 \in C,\ (b_1, c_1) \in E$ (see Fig. 10.15).

Step 4: If $|B| = 1$ then let $(a = p_0, p_1, p_2, \ldots, p_r = c)$ be the shortest path from a to c not containing b (such a path must exist, otherwise b would be an articulation point). We know that $r \geq 3$, since otherwise B would have at least two elements, namely b and p_1. Consider the subgraph induced by $\{b, a, p_1, p_2, \ldots, p_{r-1}, c\}$; it must be of the form shown in Fig. 10.2. Let k be the smallest integer greater than 2 such that p_k is adjacent to b.

We consider four cases:

CASE 1: (p_1, b), $(p_2, b) \notin E$.

Then $\{a,p_1,p_2,...,p_k,b\}$ is a primitive cycle of length $k+2 \geq 5$.

CASE 2: $(p_1,b) \in E$, $(p_2,b) \notin E$.

If $k=3$ then $\{a,p_1,p_2,p_3,b\}$ induces a house. If $k \geq 4$ then $\{p_1,p_2,...,p_k,b\}$ is a primitive cycle of length $k+1 \geq 5$.

CASE 3: $(p_1,b) \notin E$, $(p_2,b) \in E$.

If $k=3$ then $\{a,p_1,p_2,p_3,b\}$ induces a house. If $k=4$ then $\{a,p_1,p_2,p_3,p_4,b\}$ induces a figure-8. If $k \geq 5$ then $\{p_2,p_3, \ . \ . \ . \ ,p_k,b\}$ is a primitive cycle of length $k \geq 5$.

CASE 4: $(p_1,b),(p_2,b) \in E$.

If $k=3$ then $\{a,p_1,p_2,p_3,b\}$ induces a tepee. If $k=4$ then $\{p_1,p_2,p_3,p_4,b\}$ induces a house. If $k \geq 5$ then $\{p_2,p_3,...,p_k,b\}$ is a primitive cycle of length $k \geq 5$.

Step 5:

$W_1 \leftarrow \{v \in V-B : v$ is adjacent to at least one element of B and at least one element of $C \cup \{a,d\}$ $\}$;

$W_2 \leftarrow \{v \in V-B : v$ is adjacent to at least one element of B but no elements of $C \cup \{a,d\}$ $\}$.

Thus $\{W_1, W_2\}$ is a partition of the vertices outside of B but adjacent to elements of B.

After Step 6, each element of W_1 will be adjacent to each element of B. After Step 7, each element of W_2 will be adjacent to at least two elements of B. Steps 8 and 9 will use these two facts to produce a set $U \subseteq B$ having at least two elements and no U-similar pairs, which then allows us to find a member of $\mathbf{F}$ in Step 10.

Step 6: If there exists $w \in W_1$ such that $N_B(w) \neq B$ then we find a member of $\mathbf{F}$ as follows.

First note that it cannot be that $N_C(w) \neq \emptyset$ and $(w,a) \in E$, otherwise we would have

$w \in B$. Furthermore, it cannot be that $N_C(w) = \emptyset$ and $(w,a) \notin E$, otherwise we would have $(w,d) \in E$ and hence (since $N_B(w) \neq \emptyset$) we would have $w \in C$. Also note that $w \neq a$, since otherwise there would be some $c_1 \in N_C(w)$ adjacent to a and also adjacent to d (by the definition of C), which cannot be true because of Step 2. Since w is adjacent to at least one, but not all elements of B, there exists $b_1, b_2 \in B$ such that w is adjacent to b_1 but not to b_2.

CASE 1: $N_C(w) \neq \emptyset$, $(w,a) \notin E$.

Let c_1 be an element of C adjacent to w (Fig. 10.3). Then $\{b_1, a, b_2, c_1, w\}$ induces a tepee if $(b_1, b_2) \in E$, and a house otherwise.

CASE 2: $N_C(w) = \emptyset$, $(w,a) \in E$ (Fig. 10.4).

Then $\{b_1, c, b_2, a, w\}$ induces a tepee if $(b_1, b_2) \in E$, and a house otherwise.

Step 7:

$$Q \leftarrow \{w \in W_2 : |N_B(w)| = 1\} .$$

If Q is non-empty then we find a member of $\mathbf{F}$ as follows. For each $q \in Q$ define b_q as the sole element of $N_B(q)$, and define path(q) as a shortest path not containing b_q from q to some $b \in B - \{b_q\}$ (such a path must exist since otherwise b_q would be an articulation point, since $|B| \geq 2$ by Step 4). Find some $q_0 \in Q$ such that path(q_0) has minimum length; let $(q_0, p_1, p_2, \ldots, p_k, b)$ be path(q_0). Note that for all $1 \leq i \leq k$, p_i is not adjacent to both a and c, since otherwise $p_i \in B$ (by the definition of B) and therefore path(q_0) is not minimum.

Since $|N_B(q_0)| = 1$ we must have $k \geq 1$; hence q_0 is not adjacent to b. Also note that $p_1 \notin \{a, c\}$, by the definition of Q.

CASE 1: $k = 1$.

If $(a, p_1) \in E$ then (as shown above) p_1 is not adjacent to c. We also know that q_0 is not adjacent to c, since $q_0 \in Q$ implies $N_C(q_0) = \emptyset$ (since $Q \subseteq W_2$). Thus (Fig. 10.5)

$\{b_{q_0}, q_0, p_1, b\ , c\}$ induces either a tepee (if b_{q_0} is adjacent to both p_1 and b), a house (if it is adjacent to exactly one of them), or a primitive 5-cycle (if it is adjacent to neither).

On the other hand, if $(a, p_1) \notin E$ then (since $(q_0, a) \notin E$, by the definition of Q) we have that $\{a, b\ , b_{q_0}, q_0, p_1\}$ induces either a tepee, house or primitive 5-cycle in each of the four cases depending on which subset of $\{b\ , p_1\}$ is adjacent to b_{q_0}. (Fig. 10.6).

The following two facts are used in cases 2 and 3:

FACT 2: for all $i, 1 \leq i \leq k-2$, p_i is adjacent to neither a nor c, since otherwise there would be a path from q_0 to b shorter than path(q_0) (b is adjacent to c by Fact 1).

FACT 3: for all $i, 1 \leq i \leq k-1$, p_i is not adjacent to b_{q_0}. To see this, assume that it is. Then p_i is not adjacent to some $b \in B - \{b_{q_0}\}$, since otherwise there would be a path from q_0 to b shorter than path(q_0). Therefore since $|B| \geq 2$ (by Step 4), we have $N_B(p_i) \neq B$. Therefore we have (as a result of Step 6) that p_i is adjacent to no element of $C \cup \{a, d\}$. Thus $p_i \in Q$. Furthermore path(p_i) is of shorter length than path(q_0), a contradiction.

CASE 2: $k = 2$.

CASE 2.1: $(b_{q_0}, b\ ) \in E$ (Fig. 10.7).

Then $\{b_{q_0}, q_0, p_1, p_2, b\ \}$ induces a house if $(b_{q_0}, p_2) \in E$ and a primitive 5-cycle otherwise.

CASE 2.2: $(b_{q_0}, b\ ) \notin E$.

CASE 2.2.1: $(p_1, a) \in E$.

Then $(p_1, c) \notin E$ (Fig. 10.8), since otherwise $p_1 \in B$, giving a path shorter than path(q_0). Then $\{b_{q_0}, q_0, p_1, a, b\ , c\}$ induces a figure-8.

CASE 2.2.2: $(p_1, a) \notin E$ (Fig. 10.9).

Then if $(p_2, a) \in E$ then $\{b_{q_0}, q_0, p_1, p_2, a\}$ induces a house or a primitive 5-cycle (depending on whether $(p_2, b_{q_0}) \in E$). On the other hand, if $(p_2, a) \notin E$ then $\{b_{q_0}, q_0, p_1, p_2, b\ \ , a\}$ induces a figure-8 or a primitive 6-cycle (depending on whether $(p_2, b_{q_0}) \in E$).

CASE 3: $k \geq 3$.

CASE 3.1: $(p_{k-1}, c) \in E$ (Fig. 10.10).

Then $\{b_{q_0}, q_0, p_1, p_2, \ldots, p_{k-1}, c\}$ is a primitive cycle of length $k+2 \geq 5$.

CASE 3.2: $(p_{k-1}, c) \notin E$.

CASE 3.2.1: $(p_k, b_{q_0}) \in E$.

Then $\{b_{q_0}, q_0, p_1, p_2, \ldots, p_k\}$ is a primitive cycle of length $k+2 \geq 5$.

CASE 3.2.2: $(p_k, b_{q_0}) \notin E$ (Fig. 10.11).

If $(p_k, c) \in E$ then $\{b_{q_0}, q_0, p_1, p_2, \ldots, p_k, c\}$ is a primitive cycle of length $k+3 \geq 6$. Otherwise $((p_k, c) \notin E)$ either $\{b_{q_0}, q_0, p_1, p2, \ldots, p_k, b\ \ \}$ is a primitive cycle of length $k+3 \geq 6$ (if $(b\ \ , b_{q_0}) \in E$) or $\{b_{q_0}, q_0, p_1, p_2, \ldots, p_k, b\ \ , c\}$ is a primitive cycle of length $k+4 \geq 7$ (if $(b\ \ , b_{q_0}) \notin E$).

Thus

FACT 4: $|N_B(w)| \geq 2$ for all $w \in W_2$.

Step 8:

$w \leftarrow$ the element of W_2 having the fewest neighbors in B, i.e. $|N_B(w)| \leq |N_B(w\ \ )|$ for all $w\ \ \in W_2$ (We know $W_2 \neq \emptyset$ since $a \in W_2$);

$U \leftarrow N_B(w)$.

If there exists $v \in B - U$ adjacent to some $b_1 \in U$ but not adjacent to some $b_2 \in U$ then

(Fig. 10.12) $\{b_1, w, b_2, c, v\}$ induces a tepee if $(b_1, b_2) \in E$ and a house otherwise.

Thus each vertex in B outside of U is adjacent to either none or all of the vertices in U.

Step 9: If there exist $w\ \in W_2$ adjacent to some $b_1 \in U$ but not adjacent to some $b_2 \in U$, then we can find a member of **F** as follows. Let b_3 be an element of $B - U$ that is adjacent to w (such a vertex must exist, since otherwise $|N_B(w)| = |U| > |U - \{b_2\}| \geq |N_B(w\ )|$ contradicting the choice of w).

CASE 1: $(w, w\ ) \in E$ (Fig. 10.13).

Then $\{b_1, a, b_2, w\ , w\}$ induces a tepee if $(b_1, b_2) \in E$ and a house otherwise.

CASE 2: $(w, w\ ) \notin E$ (Fig. 10.14).

CASE 2.1: $(b_2, b_3) \in E$.

Then $\{b_1, w, b_2, b_3, w\ \}$ induces a tepee (if b_1 is adjacent to both b_2 and b_3), a primitive 5-cycle (if b_1 is adjacent to neither b_2 nor b_3), or a house otherwise.

CASE 2.2: $(b_2, b_3) \notin E$.

If b_1 is adjacent to both b_2 and b_3 then $\{b_1, w, b_2, a, b_3\}$ induces a tepee. If b_1 is adjacent to b_2 but not b_3 then $\{b_1, w\ , b_3, a, b_2\}$ induces a house. If b_1 is adjacent to b_3 but not b_2 then $\{b_1, w, b_2, a, b_3\}$ induces a house. Finally, if b_1 is adjacent to neither b_2 nor b_3 then $\{b_1, w, b_2, a, b_3, w\ \}$ induces a figure-8.

FACT 5: U has no U-similar pairs.

PROOF: We claim that each vertex $v \in V - U$ is adjacent to all of U or none of U. Step 6 ensures this for each $v \in W_1$ (since $U \subseteq B$). Step 8 ensures this for each $v \in B - U$. Step 9 ensures this for each $v \in W_2$. Therefore if there exists $u_1, u_2 \in U$ that are U-similar, then u_1, u_2 are also V-similar, a contradiction. **QED**

Step 10: Find vertices $u_1, u_2, u_3, u_4 \in U$ that induce a P4, by again using the algorithm of the proof of Theorem 1 (Facts 4 and 5 allow us to do this). Thus $\{w, u_1, u_2, u_3, u_4\}$ induces a tepee.

It is a simple matter to implement Steps 1-6 and 8-10 in $O(|E|)$ time. For Step 7 it is only slightly more complicated: recall that there we must find a shortest path $(q_0, p_1, p_2, \ldots, p_k, b)$ subject to $q_0 \in Q$, $p_i \neq b_{q_0}$ for all $1 \leq i \leq k$, and $b \in B - \{b_{q_0}\}$. To do this, we first find a shortest path P_1 of this form but subject also to $p_k \notin Q$. Secondly, we find a shortest path $(q, p_1, p_2, \ldots, p_k)$ subject to $q, p_k \in Q$, $b_q \neq b_{p_k}$; and then let P_2 be the path $(q, p_1, p_2, \ldots, p_k, b_{p_k})$. Both P_1 and P_2 can be found in $O(|E|)$ by means of breadth-first search. We then choose the shorter of P_1 and P_2 as our desired path.

QED Theorem 2

6. Initial placement

Now assume that we have found (using the algorithm of the previous section) some $W \subseteq V$ that induces a member of **F**. We then construct a model C for this graph G/W, which is easily done in $O(|W|)$ time. The model C is (of course) necessary for G/W since each member of **F** is uniquely representable (the reader may wish to review the definition of "necessary" models and placements from Section 3). Let $\text{chord}_C(w)$ denote the chord in C corresponding to w, for each $w \in W$. In this section, our task is to assign a necessary placement relative to C to each $v \in V - W$ adjacent to at least one member of W. Having made this placement for v, we then pick an arbitrary point within each of these two empty arcs as endpoints of a chord for v. The important decision is which two principal arcs are chosen; the actual points chosen within the arcs are irrelevant.

We first define certain subsets of $V - W$ as follows: Let

$$A_0 = \{v \in V - W : N_W(v) = \emptyset\} .$$

For each $w \in W$, let

$$A_1(w) = \{v \in V - W : N_W(v) = \{w\}\}$$

and

$$M(w) = \{v \in V - W : v \text{ is } W\text{-similar to } w\} .$$

Let

$$A_1 = \bigcup_{w \in W} A_1(w) \; (= \{v \in V - W : |N_W(v)| = 1\})$$

and

$$M = \bigcup_{w \in W} M(w).$$

Finally, let

$$A_2 = \{a \in V - (W \cup M) : |N_W(a)| \geq 2\}$$

Thus, $\{A_0, A_1, A_2, M\}$ is a partition of $V - W$.

Thus our task is to give each $v \in A_1 \cup A_2 \cup M$ a necessary placement ch(v) relative to C. We assign such placements to members of A_2 using one method, and then to members of $A_1 \cup M$ using another.

6.1. Placing members of A_2

Our algorithm that assigns a necessary placement (relative to C) to each $v \in A_2$, is straightforward: for each pair $\{\alpha, \beta\}$ of empty arcs defined by C, we consider a chord $c_{\alpha,\beta}$ having one endpoint in α and the other in β. We then check which chords of C are intersected by $c_{\alpha,\beta}$. If we ever find a pair $\{\alpha, \beta\}$ such that $c_{\alpha,\beta}$ intersects precisely those chords corresponding to members of $N_W(v)$, then we return the pair $\{\alpha, \beta\}$ as the placement for v. Otherwise, we report that G is not a circle graph and halt the entire algorithm. We can do this with confidence that (in the former case) the chord returned for v is necessary or that (in the latter case) G is not a circle graph, because of the following result.

LEMMA 3: For each $v \in A_2$, if $G/(W \cup \{v\})$ is a circle graph then there exists a unique pair $\{\alpha, \beta\}$ of empty arcs defined by C such that there exists a chord c_v with one endpoint in α and one in β, and $C \cup \{c_v\}$ is a model for $G/(W \cup \{v\})$.

PROOF: Let c and c be chords having endpoints in α and β, and in α and β, respectively, such that $C \cup \{c\}$ and $C \cup \{c\}$ are both models for $G/(W \cup \{v\})$. Assume for a contradiction that $\{\alpha, \beta\} \neq \{\alpha, \beta\}$. Consider two cases.

CASE 1: α, β, α and β are *not* pairwise distinct.

Then there exists some $\gamma \in \{\alpha, \beta\} \cap \{\alpha, \beta\}$. Let ϕ_0, ϕ_1 and ϕ_2 be the three arcs separating the three distinct members of $\{\alpha, \beta, \alpha, \beta\}$, such that γ does not share a bounding point with ϕ_0 (see Fig. 11(a)). We regard ϕ_i, $i = 0,1,2$, as being closed (i.e. it includes its bounding points). For each $w \in W$, chord$_C(w)$ does not have exactly one endpoint in ϕ_0, since otherwise chord$_C(w)$ would intersect exactly one of $\{c, c\}$. Therefore the set

$$W \quad = \{w \in W : \text{chord}_C(w) \text{ has an endpoint in } \phi_0\},$$

is non-empty and is disconnected from $W - W$, which is also non-empty (since ϕ_1 has at least one bounding point, which is an endpoint of a chord of C). But then G/W is disconnected, a contradiction (since $G/W \in \mathbf{F}$).

CASE 2: α, β, α and β are pairwise distinct.

Consider the arcs ϕ_0, ϕ_1, ϕ_2 and ϕ_3, separating α, β, α and β , in clockwise order around the circle, where ϕ_i, $i = 0,1,2,3$ is regarded as being closed (i.e. it includes its two bounding points). For example, see Fig. 11(b). Note that for $i = 0,1,2,3$, for all $w \in W$, it is *not* the case that $\text{chord}_C(w)$ has one endpoint in ϕ_i and the other in $\phi_{i+1 \bmod 4}$, since otherwise $\text{chord}_C(w)$ would intersect exactly one of $\{c, c\ \}$. If both ϕ_0 and ϕ_2 contain only one point, then these two points must be the endpoints of $\text{chord}_C(w)$ for some $w \in W$. Therefore, either v and w are W-similar or $N_W(v) = \{w\}$; either case contradicts the fact that $v \in A_2$. Similarly, it cannot be that both ϕ_1 and ϕ_3 contain only one point. There is at least one vertex v_1 (resp. v_2) in W such that one endpoint of $\text{chord}_C(v_1)$ lies in ϕ_0 (resp. ϕ_1) and the other in ϕ_2 (resp. ϕ_3). Hence v must be adjacent to v_1 or v_2 (or both); without loss of generality, assume that v is adjacent to v_1. Therefore $\{V_0, V_1, V_2, V_3\}$ is a partition of W, where

$$V_0 = \{u \in W : \text{ the endpoints of chord}_C(u) \text{ are both in } \phi_0 \text{ or both in } \phi_2\}$$

$$V_1 = \{u \in W : \text{ one endpoint of chord}_C(u) \text{ is in } \phi_0 \text{ and the other is in } \phi_2\}$$

$$V_2 = \{u \in W : \text{ one endpoint of chord}_C(u) \text{ is in } \phi_1 \text{ and the other is in } \phi_3\}$$

$$V_3 = \{u \in W : \text{ the endpoints of chord}_C(u) \text{ are both in } \phi_1 \text{ or both in } \phi_3\}$$

(thus $v_1 \in V_1$ and $v_2 \in V_2$). This partition gives a simple decomposition of G/W. To see this, it is easily verified that each member of V_1 is adjacent to each member of V_2, that no member of V_0 is adjacent to a member of $V_2 \cup V_3$, and that no member of V_1 is adjacent to a member of V_3. Furthermore, $|V_0 \cup V_1| \geq 2$, because (as noted above) $\phi_0 \cup \phi_2$ contains at least three endpoints of chords in C. Similarly, $|V_2 \cup V_3| \geq 2$. But G/W is a member of $\mathbf{F}$ and is therefore prime, a contradiction.

QED LEMMA 3

6.2. Placing members of $A_1 \cup M$

We assign a necessary placement, relative to C, to the members of $A_1 \cup M$ by calling Algorithm A (described below) once for each $w \in W$. Algorithm A, given some $w \in W$, returns a necessary placement relative to C for each member of $A_1(w) \cup M(w)$.

In order to describe the operation of Algorithm A, given w, we need some notation. Let c be the chord in C corresponding to w. Let α and β denote the two empty arcs defined by the model $C - \{c\}$ that contain the endpoints of c. Let α_0 and α_1 (resp. β_0 and β_1) denote the two empty arcs of model C into which c splits α (resp. β), in such a way that α_0 and β_0 are on the same side of c, as is illustrated in Fig. 12. We refer to the arcs α_0, α_1, β_0 and β_1 as the ***principal arcs***. Let γ and δ denote the two arcs between α and β so that the sequence of arcs α_0, α_1, δ, β_1, β_0, γ are encountered during a clockwise traversal of the circle. Thus, there are vertices w_1, w_2, w_3, $w_4 \in W$ such that α is bounded by one endpoint of $\text{chord}_C(w_1)$ and one of $\text{chord}_C(w_2)$, δ is bounded by one endpoint of $\text{chord}_C(w_2)$ and one of $\text{chord}_C(w_3)$, β is bounded by one endpoint of $\text{chord}_C(w_3)$ and one of $\text{chord}_C(w_4)$, γ is bounded by one endpoint of $\text{chord}_C(w_4)$ and one of $\text{chord}_C(w_1)$.

Thus, Algorithm A finds, for each $v \in A_1(w) \cup M(w)$, a pair of principal arcs that must contain the endpoints of v's chord in all models for G. Note that, in each model for G, the placement for each member of $A_1(w)$ must have either one endpoint in α_0 and the other in α_1, or one in β_0 and the other in β_1. Similarly, there are two possibilities for the placement of each $m \in M(w)$. In particular, if $(m,w) \notin E$ then the chord for m must have either one endpoint in α_0 and the other in β_0, or one in α_1 and the other in β_1. Otherwise ($(m, w) \in E$) the chord for m must have either one endpoint in α_0 and the other in β_1, or one in α_1 and the other in β_0. Thus, for each $v \in A_1(w) \cup M(w)$, once Algorithm A has found that some endpoint of v's chord must (in all models for G) lie in a particular principal arc, then it is a trivial matter to decide which principal arc must contain the other endpoint of v's chord.

Since the chord for each member of $M(w)$ must have one endpoint in α and one in β, we refer to $M(w)$ by the symbol $F_{\alpha\beta}$. The reason for this extra notation is that Algorithm A will be called again while iteratively adding chords (Section 7). The algorithm for adding chords will also identify a set of vertices needing chords with one endpoint in α and one in β, but this set will not directly correspond to the notion of W-similarity with some vertex; this will be clarified in Section 7.

Now the information that allows Algorithm A to make these binary choices of placements for vertices in $A_1(w) \cup F_{\alpha\beta}$ is provided to it in the form of four sets of vertices: F_{α_0}, F_{α_1}, F_{β_0} and F_{β_1}.

6.2.1. Computing F_{α_0}

We first motivate and describe the computation of F_{α_0} in detail, and then briefly mention the analogous sets F_{α_1}, F_{β_0} and F_{β_1}. Intuitively, F_{α_0} is the set of vertices whose chords must have one endpoint in $\gamma \cup \delta$ and the other *might* lie in α_0 but cannot lie in any of the other three principal arcs. In other words, these vertices' chords cannot have an endpoint in $\alpha_1 \cup \beta$, but can have (at most) one endpoint in α_0. Identifying these vertices is useful to Algorithm A because if, for example, some vertex p with this property is adjacent to some $q \in A_1(w)$, then we know that the endpoints of the chord for q must lie in α_0 and α_1, rather than in β_0 and β_1.

We begin by including in F_{α_0} each $v \in A_2$ for which $\text{ch}(v)$ (computed in Section 6.1 above) contains α_0 (note that the other arc in $\text{ch}(v)$ must be contained in $\gamma \cup \delta$, since $v \in A_2$). As is shown in Section 6.1, $\text{ch}(v)$ is necessary.

We also include each $v \in A_1(w_1)$ in F_{α_0}. We claim that, in each model for G, the chord for v cannot have an endpoint in $\alpha_1 \cup \beta$. To see this, let y_1 denote the endpoint of $\text{chord}_C(w_1)$

bounding α and γ, and let y_2 denote $\text{chord}_C(w_1)$'s other endpoint. Then the endpoints of v's chord must lie either in the two empty arcs defined by C surrounding y_1 or in those surrounding y_2, because v is adjacent to no vertices of W other than w. If $(w_1, w) \notin E$ then $y_2 \in \gamma$ and hence the endpoints of v's chord must either lie both in γ or one in γ and the other in α_0 (see Fig. 13(a)). Otherwise ($(w_1, w) \in E$) they both lie in δ or one lies in γ and the other in α_0 (Fig. 13(b)). This follows easily from the following result:

LEMMA 4: w_1, w_2, w_3 and w_4 are pairwise distinct.

PROOF: If $w_1 = w_2$ (see Fig. 14(a)) then $N_W(w_1) = \{w\}$, which contradicts the fact that no graph in **F** has an articulation point. Analogously, we have $w_3 \neq w_4$. If $w_1 = w_3$ (Fig. 14(b)) or $w_1 = w_4$ then w and w_1 are W-similar, contradicting the fact that no graph in **F** has a similar pair. Analogously, we have $w_2 \neq w_4$ and $w_2 \neq w_3$. **QED**

In either case, v's chord has no endpoint in $\alpha_1 \cup \beta$.

Finally, we include each $v \in M(w_1)$ in F_{α_0}. There are four possibilities for the placement of v's chord: two if $(w, w_1) \in E$ (Fig. 15(a)) and two if $(w, w_1) \notin E$ (Fig. 15(b)). No two members of $\{w_1, w_2, w_3, w_4\}$ are W-similar (since no graph in **F** has a similar pair); hence v's chord cannot have an endpoint in $\alpha_1 \cup \beta$.

In summary, we compute:

$$F_{\alpha_0} \leftarrow \{v \in A_2 : \alpha_0 \in \text{ch}(v)\} \cup A_1(w_1) \cup M(w_1)$$

$$F_{\alpha_1} \leftarrow \{v \in A_2 : \alpha_1 \in \text{ch}(v)\} \cup A_1(w_2) \cup M(w_2)$$

$$F_{\beta_0} \leftarrow \{v \in A_2 : \beta_0 \in \text{ch}(v)\} \cup A_1(w_4) \cup M(w_4)$$

$$F_{\beta_1} \leftarrow \{v \in A_2 : \beta_1 \in \text{ch}(v)\} \cup A_1(w_3) \cup M(w_3)$$

For convenience, we define $F_1 = F_{\alpha_0} \cup F_{\alpha_1} \cup F_{\beta_0} \cup F_{\beta_1}$.

6.2.2. Algorithm A

Throughout the remainder of this section, when we refer to the placement of a vertex we mean relative to the model $C - \{c\}$, unless otherwise specified.

Algorithm A is shown in Fig. 16, and works as follows. We iteratively place a member of $A_1(w) \cup F_1 \cup F_{\alpha\beta}$ and in doing so, learn how to place other such members. We traverse this subset of $A_1(w) \cup F_1 \cup F_{\alpha\beta}$ as a breadth-first search, implemented by means of a first-in-first-out queue. Thus whenever placing a vertex, we then put it in the queue. The algorithm iteratively dequeues a vertex v, which then may cause a number of other vertices to be placed and enqueued; in this case we say that v *determines* these other vertices. We maintain a set P of all members of $A_1(w) \cup F_{\alpha\beta}$ that have ever been in the queue, i.e. the members of $A_1(w) \cup F_{\alpha\beta}$ that have been placed so far.

A few definitions are needed to explain Fig. 16. If $k \geq 1$ then a ***chain*** is a path $a_1, a_2, ..., a_k$ in G such that $a_i \in A_0$ for $i = 1,2,...,k$. Two vertices v, $v \in V$ are said to be ***connected by*** this chain if $(v,\ a_1)$, $(a_k,\ v) \in E$. We refer to the four arcs α_0, α_1, β_0, β_1 as the ***principal arcs***. Thus, the output of Algorithm A consists of a pair of principal arcs for each member of $A_1(w) \cup F_{\alpha\beta}$. We define the *x_reflect* of a principal arc as the other principal arc on the same side of chord c ; more precisely

$$\text{x\_reflect}(\alpha_0) = \beta_0, \quad \text{x\_reflect}(\beta_0) = \alpha_0, \quad \text{x\_reflect}(\alpha_1) = \beta_1, \quad \text{x\_reflect}(\beta_1) = \alpha_1 \ .$$

Analogously, define the *y_reflect* of a principal arc as the principal arc differing from it only by being on the other side of c ; more precisely

$$\text{y\_reflect}(\alpha_0) = \alpha_1, \quad \text{y\_reflect}(\alpha_1) = \alpha_0, \quad \text{y\_reflect}(\beta_0) = \beta_1, \quad \text{y\_reflect}(\beta_1) = \beta_0 \ .$$

By "placing" a vertex v, what the algorithm actually does is to pick the two principal arcs to contain the endpoints of a chord for v. We distinguish between these two principal arcs, calling them $\text{arc}_1(v)$ and $\text{arc}_2(v)$, respectively.

The algorithm begins by placing each $f \in F_{\alpha_0}$ as follows. Note that, in each model for G, no endpoint of fs chord may lie in $\alpha_1 \cup \beta$, but there may be an endpoint of fs chord in α_0. We indeed set $\mathrm{arc}_2(f)$ to α_0, for the following reason. If f will determine some $q \in A_1(w) \cup F_{\alpha\beta}$, then indeed fs chord must have an endpoint in $\alpha \cup \beta$ and hence in α_0. On the other hand, if f does not determine any vertices, then $\mathrm{arc}_2(f)$ will never be referenced and hence it does not matter what value it receives here. We let $\mathrm{arc}_1(f)$ be δ if f is adjacent to w, otherwise we set it to γ. Notice that $\mathrm{arc}_1(f)$ will not be referenced in the algorithm, but will appear in the proof of correctness (Lemma 6). We place each member of F_{α_1}, F_{β_0} and F_{β_1} analogously.

In the body of the main loop of Algorithm A, a vertex p is removed by the queue and then the vertices q (not previously determined) that are determined by p are placed and enqueued. There are conditions under which p determines q, illustrated in Fig. 17 (a)-(e):

(a) p and q are connected by a chain,

(b) $q \in A_1(w)$ and $(p, q) \in E$,

(c) $q \in F_{\alpha\beta}$, $p \in A_1(w)$ and $(p, q) \notin E$,

(d) $q \in F_{\alpha\beta}$, $p \in F_1 \cup F_{\alpha\beta}$, $(p, w) \in E$ and $(p, q) \notin E$,

(e) $q \in F_{\alpha\beta}$, $p \in F_1 \cup F_{\alpha\beta}$, $(p, w) \notin E$ and $(p, q) \in E$.

The key idea (made more formal in the proof of correctness) of the algorithm is that the first endpoint of q's chord must lie in the same principal arc x as the second endpoint of p's arc. The first and second endpoints of the arcs are distinguished by 1's and 2's in the figure. The second endpoint of q's arc is then forced to lie in the y_reflect of x if $q \in A_1(w)$, or in the x_reflect of x if $q \in F_{\alpha\beta}$ and $(q, w) \notin E$, or in the y_reflect of the x_reflect of x if $q \in F_{\alpha\beta}$ and $(q, w) \in E$.

```
queue ← ∅;
FOR each x ∈ {α0, α1, β0, β1} DO
   FOR each f ∈ Fx DO
      BEGIN
         IF ( (f, w) ∈ E ) iff ( x ∈ {α0,β0} )
            THEN arc1(f) ← δ
            ELSE arc1(f) ← γ ;

         arc2(f) ← x ;

         enqueue(f)
      END;

P ← ∅ ;
WHILE the queue is non-empty DO
     BEGIN
       dequeue(p) ;

       FOR each q ∈ A1(w) − P adjacent to p or connected by a chain to p DO
          BEGIN
             arc1(q) ← arc2(p) ;
             arc2(q) ← y_reflect(q) ;

             enqueue(q) ;   P ← P ∪ {q}
          END;

       FOR each q ∈ Fαβ − P DO
          IF    (p is connected by a chain to q)
             OR (p ∈ A1(w) AND (p, q) ∉ E)
             OR ( p ∈ F1 ∪ Fαβ
                    AND
                    (   ((p, w) ∈ E, AND (p, q) ∉ E)
                      OR ((p, w) ∉ E, AND (p, q) ∈ E) ) ) THEN
            BEGIN
               arc1(q) ← arc2(p) ;

               arc2(q) ← x_reflect( arc1(q) );
               IF q is adjacent to w THEN
                  arc2(q) ← y_reflect( arc2(q) );

               enqueue(q) ;  P ← P ∪ {q}
            END
     END;
```

Fig. 16
Algorithm A

PROOF OF CORRECTNESS OF ALGORITHM A

The proof of correctness of Algorithm A follows immediately from Lemmas 6 and 7; Lemma 5 is used in proving Lemma 6.

Let P denote the set P at the termination of the algorithm. For each $p \in F_1 \cup P$, let ch(p) denote the placement given to p by the algorithm; that is, ch(p) = {arc$_1$(p), arc$_2$(p)}.

Throughout these lemmas, we fix a model D for G; let chord$_D$(v) denote the chord in D corresponding to v, for each vertex v of G. Since C is a necessary model for G/W, we can assume without loss of generality for our purposes that for each $w \in W$, the chord corresponding to w in model C is identical to chord$_D$(w). Thus, we are able to refer to the principal arcs α_0, α_1, β_0 and β_1 with regard to model D just as we have for model C. Since the choice of D was arbitrary, Lemma 6 implies that ch(p) is necessary, for each $p \in A_1(w) \cup F_{\alpha\beta}$.

LEMMA 5: If $a_1, a_2, \cdots, a_k$ is a chain such that a_1 is adjacent to some $v \in A_1(w) \cup F_{\alpha\beta}$ there is some principal arc x such that for all i, $1 \leq i \leq k$, both endpoints of chord$_D$(a_i) are in x.

PROOF: Assume the contrary. Then since a_1 is adjacent to v and chord$_D$(v) has both endpoints in $\alpha \cup \beta$, one endpoint of chord$_D$(a_1) lies in some principal arc (since $(a_1, w) \notin E$), say α_0. By assumption, there is some vertex in the chain whose chord in D has endpoints not both in α_0; let a_j be such a vertex for the least j. Now chord$_D$(a_j) has one endpoint in α_0 (since, if $j > 1$, a_j is adjacent to a_{j-1} and chord$_D$(a_{j-1}) has both endpoints in α_0). The other endpoint of chord$_D$(a_j) cannot lie in $\alpha_1 \cup \beta_1 \cup \delta$ (since $(a_j, w) \notin E$), nor in β_0 (since a_j is not W-similar to w), nor in γ (since otherwise W would be disconnected, since a_j is adjacent to no member of W).

QED

LEMMA 6: The endpoints of chord$_D$(p) lie in arc$_1$(p) and arc$_2$(p), respectively, for each $p \in P$.

PROOF:

We begin with a few definitions: For each $p \in F_1 \cup P$, we distinguish between the two endpoints of $\text{chord}_D(p)$, calling them first(p) and second(p), respectively. In particular, we will show (as mentioned above) by induction that the endpoints of $\text{chord}_D(p)$ lie in $\text{arc}_1(p)$ and $\text{arc}_2(p)$, respectively. Define first(p) (resp. second(p)) as that endpoint of $\text{chord}_D(p)$ lying within $\text{arc}_1(p)$ (resp. $\text{arc}_2(p)$). These are well defined since $\text{arc}_1(p)$ cannot equal $\text{arc}_2(p)$.

If z is a point within arc α (resp. β) then outside(z) denotes the arc bounded by z and by one of the two bounding points of α (resp. β) such that outside(z) is entirely contained in some principal arc. In other words, outside(z) does *not* contain an endpoint of chord c (see Fig. 18). If z is a point on the circle *not* in $\alpha \cup \beta$ then define outside(z) as the empty set.

As a notational convenience, if $p \in F_1 \cup P$ then we let

$$\text{outside}_1(p) = \text{outside}(\text{ first}(p)\ ) \qquad \text{and} \qquad \text{outside}_2(p) = \text{outside}(\text{ second}(p)\ )$$

(note that outside(first(p)) $= \varnothing$ if $p \in F_1$). Also, let

$$\text{live}(p) = \text{outside}_2(p) \;\cup\; \Big(\bigcup_{q \text{ determined by } p} \text{outside}_1(q) \Big)$$

Finally, we have already defined what it means for vertices v_1 and v_2 to be connected by a chain. Also, we have seen (Lemma 5) that for each chain $a_1, a_2, \ldots, a_k$ such that a_1 is adjacent to a member of P, there is a principal arc x containing both endpoints of each of the chords $\text{chord}_D(a_1)$, $\text{chord}_D(a_2)$, ..., $\text{chord}_D(a_k)$. Now we say that points $z_1, z_2 \in x$ are connected by this chain if

$$z_1,\ z_2 \in \bigcup_{1 \le i \le k} \phi_i$$

where ϕ_i is the arc within x defined by $\text{chord}_D(a_i)$ (see Fig. 19).

Let $p_1, p_2, \ldots, p_{|F_1 \cup P|}$ be the vertices of $F_1 \cup P$ in the order of their inclusion in the queue (i.e. in the order in which they were placed by the algorithm). We will show the following facts about p_k, for $k = 1, 2, \ldots, |F_1 \cup P|$, by induction on k:

1. If p_k determines a vertex q then

(A) If p_k is connected by a chain to q then this chain connects second(p_k) to some endpoint of chord$_D(q)$,

(B) The endpoints of chord$_D(p_k)$ lie in arc$_1(p_k)$ and arc$_2(p_k)$ and the endpoints of chord$_D(q)$ lie in arc$_1(q)$ and arc$_2(q)$ (this permits us to define first(p_k), second(p_k), first(q) and second(q) as discussed above),

(C) If second(p_k) $\in$ outside$_1(q)$ then p_k and q are connected by a chain.

2. If $r \in A_1(w) \cup F_{\alpha\beta}$ and chord$_D(r)$ has an endpoint in live(p_k) then r was determined by p_j for some $j \leq k$ (that is, r was determined by p_k if it had not already been determined earlier).

Part 1.B of the induction hypothesis is what we are really interested in; parts 1.A, 1.C and 2 are carried along in order to facilitate its proof.

BASIS OF INDUCTION: As a basis for the induction, assume $1 \leq k \leq |F_1|$. Then $p_k \in F_1$, since the algorithm uses a first-in-first-out queue. In this basis step, we find it convenient to prove the four parts of the claim in the order 1.B, 1.A, 1.C, 2. In the proof of parts 1.A, 1.B and 1.C of the claim, let q be a vertex determined by p_k.

PROOF OF PART 1.B: We will first show that the endpoints of chord$_D(p_k)$ lie in arc$_1(p_k)$ and arc$_2(p_k)$. Then we will show that one endpoint of chord$_D(q)$ (which we will define as first(q)) lies in arc$_2(p_k)$. The other endpoint of chord$_D(q)$ (which we will define as second(q)) must therefore lie in the y_reflect of this arc (if $q \in A_1(w)$), or in the y_reflect of its x_reflect (if $q \in F_{\alpha\beta}$ and $(q, w) \in E$) or in simply its x_reflect (if $q \in F_{\alpha\beta}$ and $(q, w) \notin E$). But this is precisely the way the algorithm chooses arc$_2(q)$.

Consider two cases.

CASE 1: $(p_k, w) \notin E$.

Then q is either adjacent to p_k or connected by a chain to p_k. Since chord$_D(q)$ has both endpoints in $\alpha \cup \beta$ (since $q \in A_1(w) \cup F_{\alpha\beta}$), this implies that chord$_D(p_k)$ has at least one endpoint in $\alpha \cup \beta$. But since $p_k \in F_1$, chord$_D(p_k)$ also has at least one endpoint in $\gamma \cup \delta$. Therefore the endpoints of chord$_D(p_k)$ lie in arc$_1(p_k)$ and arc$_2(p_k)$. For example (see Fig. 20(a)) if $p_k \in F_{\alpha_0}$ then chord$_D(p_k)$ has one endpoint in γ and the other *not* in $\alpha_1 \cup \beta$. Therefore since this second endpoint must lie in $\alpha \cup \beta$, it must lie in α_0.

If q is adjacent to p_k then one endpoint of chord$_D(q)$ is in arc$_2(p_k)$ = arc$_1(q)$. On the other hand, if q is connected by a chain to p_k, then some endpoint of chord$_D(p_k)$ is connected to some endpoint of chord$_D(q)$. Therefore since endpoints of chord$_D(q)$ lie in $\alpha \cup \beta$ and since first(p_k) does not lie in $\alpha \cup \beta$, we have that this endpoint of chord$_D(q)$ is connected by a chain to second(p_k) and hence lies in arc$_2(p_k)$ = arc$_1(q)$.

CASE 2: $(p_k, w) \in E$.

If either q is a member of $A_1(w)$, or if q is a member of $F_{\alpha\beta}$ and connected to p_k by a chain, then we can show the claim by using the argument of CASE 1 above.

So assume that $q \in F_{\alpha\beta}$ and that q is not connected to p_k by a chain. Then q is *not* adjacent to p_k (since p_k determined q). Therefore since chord$_D(q)$ has one endpoint in α and one in β, and since $(p_k, w) \in E$, chord$_D(p_k)$ has at most one endpoint in $\gamma \cup \delta$. Therefore the endpoints of chord$_D(p_k)$ lie in arc$_1(p_k)$ and arc$_2(p_k)$. For example (see Fig. 20(b)) if $p_k \in F_{\alpha_0}$ then chord$_D(p_k)$ has one endpoint in δ and the other *not* in $\alpha_1 \cup \beta$. Since p_k is *not* adjacent to q, this second endpoint must lie in $\alpha \cup \beta$ and hence in α_0.

Therefore one endpoint (which we define as first(q)) of chord$_D(q)$ is in arc$_2(p_k)$ = arc$_1(q)$.

PROOF OF PART 1.A: Assume that p_k and q are connected by a chain. Then some endpoint of chord$_D(p_k)$ is connected to some endpoint of chord$_D(q)$. Therefore since endpoints of chord$_D(q)$ lie in $\alpha \cup \beta$ (since $q \in A_1(w) \cup F_{\alpha\beta}$), and first($p_k$) does not lie in $\alpha \cup \beta$ (since $p_k \in F_1$), we have that second(p_k) is connected by a chain to some endpoint of chord$_D(q)$.

PROOF OF PART 1.C:

Assume second(p_k) $\in$ outside$_1(q)$.

If $q \in A_1(w)$ then $(p_k,\ q) \notin E$ and hence (since p_k determined q) p_k and q are connected by a chain (see Fig. 21(a)).

Otherwise $(q \in F_{\alpha\beta})$ if $(p_k, w) \in E$ (Fig. 21(b)) then $(p_k, q) \in E$; on the other hand, if $(p_k, w) \notin E$ (Fig. 21(c)) then $(p_k, q) \notin E$. In either case, p_k and q are connected by a chain (since p_k determined q).

PROOF OF PART 2:

Let $r \in A_1(w) \cup F_{\alpha\beta}$ be such that chord$_D(r)$ has an endpoint z_1 in live(p_k). Let z_2 denote the other endpoint chord$_D(r)$.

CASE 1: second(p_k) $\in$ outside(z_1).

Then since $z_1 \in$ live(p_k), there is a vertex q determined by p_k such that $z_1 \in$ outside$_1(q)$ and second(p_k) $\in$ outside$_1(q)$ (Fig. 22(a)). Since q was determined by p_k, the endpoints of chord$_D(q)$ lie in arc$_1(q)$ and arc$_2(q)$, as is shown in part 1.B above. Therefore, by part 1.C, p_k is connected to q by a chain. Then, by part 1.A, second(p_k) is connected by a chain to first(q). Hence r is connected by a chain to p_k, and therefore was determined by p_k if not earlier.

CASE 2: second(p_k) $\notin$ outside(z_1).

Then either $(p_k,\ w) \in E$ (Fig. 22(b)) or $(p_k,\ w) \notin E$ (Fig. 22(c)). In either case, if $r \in A_1(w)$ then $(p_k,\ r) \in E$, implying that r was determined by p_k if not earlier. Otherwise $(r \in F_{\alpha\beta})$ we have that $(p_k,\ r) \notin E$ (if $(p_k,\ w) \in E$)) or $(p_k,\ r) \in E$ (if $(p_k,\ w) \notin E$)), implying in either case that r was determined by p_k if not earlier.

INDUCTIVE STEP: Consider some $k > |F_1|$, and assume the claim for all j, $1 \leq j < k$. Thus $p_k \in P$. Define p as the vertex that determined p_k, Then $p = p_j$ for some $j < k$, and hence the inductive hypothesis applies to p. Therefore the endpoints of $\text{chord}_D(p_k)$ lie in $\text{arc}_1(p_k)$ and $\text{arc}_2(p_k)$. In the proof of parts 1.A, 1.B and 1.C of the claim, let q be a vertex determined by p_k.

PROOF OF PART 1.A:

Assume that p_k and q are connected by a chain. Then some endpoint z_1 of $\text{chord}_D(q)$ lies in the same principal arc as does $\text{first}(p_k)$ or $\text{second}(p_k)$. Assume for a contradiction that z_1 and $\text{first}(p_k)$ lie in the same principal arc. Let z_2 be the other endpoint of $\text{chord}_D(q)$. We know that $z_1 \notin \text{live}(p)$, since otherwise q would have been determined by p or earlier, by part 2 of the inductive hypothesis applied to p. Hence $z_1 \notin \text{outside}_1(p_k)$ and $z_1 \notin \text{outside}_2(p)$. Consider two cases, depending on the relative order of first(p_k) and second(p').

CASE 1: $\text{first}(p_k) \in \text{outside}_2(p)$ (see Fig. 23(a)).

> Then since z_1 is connected by a chain to $\text{first}(p_k)$ and since $\text{second}(p)$ is between z_1 and $\text{first}(p_k)$, q must be connected by a chain to p. Hence q was determined by p or earlier, and therefore not by p_k, a contradiction.

CASE 2: $\text{first}(p_k) \notin \text{outside}_2(p)$ (see Fig. 23(b)).

> Then p_k is connected to p by a chain which, by part 1.B of the inductive hypothesis applied to p, connects $\text{second}(p)$ to $\text{first}(p_k)$. But then $\text{second}(p)$ is connected by a chain to z_1. Hence q was determined by p or earlier, and therefore not by p_k, a contradiction.

PROOF OF PART 1.B:

The endpoints of $\text{chord}_D(p_k)$ lie in $\text{arc}_1(p_k)$ and $\text{arc}_2(p_k)$ by part 1.B of the inductive hypothesis applied to p; thus we need only show that the endpoints of $\text{chord}_D(q)$ lie in $\text{arc}_1(q)$ and $\text{arc}_2(q)$.

If p_k and q are connected by a chain then, as shown above, $\text{second}(p_k)$ and some endpoint of $\text{chord}_D(q)$ lie in the same principal arc, namely $\text{arc}_2(p_k)$ $(= \text{arc}_1(q))$ by part 1.B of the induction hypothesis applied to p. The other endpoint of $\text{chord}_D(q)$ must therefore lie in $\text{arc}_2(q)$.

So assume that p_k and q are not connected by a chain. We will prove the claim by considering a number of cases. A key fact in many of these is no endpoint of $\text{chord}_D(q)$ may lie in $\text{outside}_1(p_k)$. To see this, note that $\text{outside}_1(p_k) \subseteq \text{live}(p)$, implying that if an endpoint of $\text{chord}_D(q)$ were within $\text{outside}_1(p_k)$ then (by part 2 of the inductive hypothesis applied to p) q would have been determined by p or earlier and hence not by p_k.

CASE 1: $q \in A_1(w)$.

CASE 1.1: $p_k \in A_1(w)$ (Fig. 24(a)).

Then $(p_k, q) \in E$. Therefore the endpoints of $\text{chord}_D(q)$ must lie in the same pair of principal arcs as do the endpoints of $\text{chord}_D(p_k)$, hence the claim is true.

CASE 1.2: $p_k \in F_{\alpha\beta}$ (Fig. 24(b)).

Then $(p_k, q) \in E$. Therefore since some endpoint of $\text{chord}_D(q)$ must lie in the same principal arc as some endpoint of $\text{chord}_D(p_k)$, the claim is true.

CASE 2: $q \in F_{\alpha\beta}$.

CASE 2.1: $p_k \in A_1(w)$ (see Fig. 24(c)).

Then $(p_k, q) \notin E$ (since p_k determined q). Therefore since some endpoint of $\text{chord}_D(q)$ must lie in the same principal arc as some endpoint of $\text{chord}_D(p_k)$, and since no endpoint of $\text{chord}_D(q)$ lies in $\text{outside}_1(p_k)$ the claim (i.e. that the endpoints of $\text{chord}_D(q)$ lie in $\text{arc}_1(q)$ and $\text{arc}_2(q)$) is true.

CASE 2.2: $p_k \in F_{\alpha\beta}$.

CASE 2.2.1: (p_k, w), $(q, w) \notin E$ (Fig. 24(d)).

Then $(p_k, q) \in E$. Therefore the endpoints of $\text{chord}_D(q)$ must lie in the same pair of principal arcs as do the endpoints of $\text{chord}_D(p_k)$, hence the claim is true.

CASE 2.2.2: $(p_k, w) \notin E$, $(q, w) \in E$ (Fig. 24(e)).

Then $(p_k, q) \in E$. Therefore, since no endpoint of $\text{chord}_D(q)$ lies in $\text{outside}_1(p_k)$, one endpoint of $\text{chord}_D(q)$ must lie in $\text{outside}_2(p_k)$. Hence the claim is true.

CASE 2.2.3: $(p_k, w) \in E$, $(q, w) \notin E$ (Fig. 24(f)).

Then $(p_k, q) \notin E$. Therefore, since no endpoint of $\text{chord}_D(q)$ lies in $\text{outside}_1(p_k)$, one endpoint of $\text{chord}_D(q)$ must lie in $\text{outside}_2(p_k)$. Hence the claim is true.

CASE 2.2.4: (p_k, w), $(q, w) \in E$ (Fig. 24(g)).

Then $(p_k, q) \notin E$. Therefore the endpoints of $\text{chord}_D(q)$ must lie in the same pair of principal arcs as do the endpoints of $\text{chord}_D(p_k)$, hence the claim is true.

PROOF OF PART 1.C:

Assume $\text{second}(p_k) \in \text{outside}_1(q)$.

CASE 1: $q \in A_1(w)$.

CASE 1.1: $p_k \in A_1(w)$ (Fig. 25(a)).

Then $\text{second}(q) \notin \text{outside}_1(p_k)$, since otherwise q would have have been determined by p or earlier (and hence not by p_k) by part 2 of the inductive hypothesis applied to p, since $\text{outside}_1(p_k) \subseteq \text{live}(p)$).

Therefore $(p_k, q) \notin E$ and hence (since p_k determined q) p_k and q are connected by a chain.

CASE 1.2: $p_k \in F_{\alpha\beta}$ (Fig. 25(b)).

Then $(p_k, q) \notin E$ and hence (since p_k determined q) p_k and q are connected by a chain.

CASE 2: $q \in F_{\alpha\beta}$.

CASE 2.1: $p_k \in A_1(w)$ (Fig. 25(c)).

Then $(p_k,\ q) \in E$ and hence (since p_k determined q) p_k and q are connected by a chain.

CASE 2.2: $p_k \in F_{\alpha\beta}$. .

CASE 2.2.1 $(p_k, w) \notin E$ and $(q, w) \notin E$. (Fig. 25(d))

Then second(q) $\notin$ outside$_1(p_k)$, since otherwise q would have have been determined by p or earlier (and hence not by p_k) by part 2 of the inductive hypothesis applied to p, since outside$_1(p_k) \subseteq$ live(p)).

Therefore $(p_k,\ q) \notin E$; hence (since p_k determined q) p_k and q are connected by a chain.

CASE 2.2.2 $(p_k, w) \notin E$ and $(q, w) \in E$. (Fig. 25(e))

Then $(p_k,\ q) \notin E$; hence (since p_k determined q) p_k and q are connected by a chain.

CASE 2.2.3 $(p_k, w) \in E$ and $(q, w) \notin E$. (Fig. 25(f))

Then $(p_k,\ q) \in E$; hence (since p_k determined q) p_k and q are connected by a chain.

CASE 2.2.4 $(p_k, w) \in E$ and $(q, w) \in E$. (Fig. 25(g))

Then second(q) $\notin$ outside$_1(p_k)$, since otherwise q would have been determined by p or earlier (and hence not by p_k) by part 2 of the inductive hypothesis applied to p, since outside$_1(p_k) \subseteq$ live(p)).

Therefore $(p_k,\ q) \in E$; hence (since p_k determined q) p_k and q are connected by a chain.

PROOF OF PART 2:

Let $r \in A_1(w) \cup F_{\alpha\beta}$ be such that chord$_D(r)$ has an endpoint z_1 in live(p_k). Let z_2 denote the other endpoint chord$_D(r)$.

CASE 1: second$(p_k) \in$ outside(z_1).

Then since $z_1 \in \text{live}(p_k)$, there is a vertex q determined by p_k such that $z_1 \in \text{outside}_1(q)$ and $\text{second}(p_k) \in \text{outside}_1(q)$ (Fig. 26(a)). Since q was determined by p_k, the endpoints of $\text{chord}_D(q)$ lie in $\text{arc}_1(q)$ and $\text{arc}_2(q)$, as is shown in part 1.B above. Therefore, by part 1.C, p_k is connected to q by a chain. Then, by part 1.A, $\text{second}(p_k)$ is connected by a chain to $\text{first}(q)$. Hence r is connected by a chain to p_k, and therefore was determined by p_k if not earlier.

CASE 2: $\text{second}(p_k) \notin \text{outside}(z_1)$ and $z_2 \in \text{outside}_1(p_k)$. (Fig. 26(b))

Then $z_2 \in \text{live}(p)$, since $\text{outside}_1(p_k) \subseteq \text{live}(p)$. Therefore, by part 2 of the inductive hypothesis applied to p, we have that r was determined by p if not earlier.

CASE 3: $\text{second}(p_k) \notin \text{outside}(z_1)$ and $z_2 \notin \text{outside}_1(p_k)$.

Consider the three cases:

Case 3.1: $p_k \in A_1(w)$ (Fig. 26(c))

Case 3.2: $p_k \in F_{\alpha\beta}$ and $(p_k, w) \in E$ (Fig. 26(d))

Case 3.3: $p_k \in F_{\alpha\beta}$ and $(p_k, w) \notin E$ (Fig. 26(e))

In each of these three cases, if $r \in A_1(w)$ then $(p_k, r) \in E$, implying that r was determined by p_k if not earlier. Otherwise ($r \in F_{\alpha\beta}$) we have that $(p_k, r) \notin E$ in Cases 3.1 and 3.2 and $(p_k, r) \in E$ in Case 3.3, implying in each case that r was determined by p_k if not earlier.

QED LEMMA 6

LEMMA 7: all members of $A_1(w) \cup F_{\alpha\beta}$ are placed by Algorithm A.

PROOF: We must show $P = A_1(w) \cup F_{\alpha\beta}$. Assume for a contradiction that $(A_1(w) \cup F_{\alpha\beta}) - P$ is non-empty. Partition V as follows:

$$\begin{aligned}
V_0 = \quad & (A_1(w) - P) \\
& \cup \{a_0 \in A_0 : \text{there is a member of } (A_1(w) \cup F_{\alpha\beta}) - P \text{ adjacent to } a_0 \\
& \qquad \text{or connected by a chain to } a_0\}, \\
V_1 = \; & (F_{\alpha\beta} - P) \cup \{w\}, \\
V_2 = \quad & N_W(w), \\
& \cup F_{\gamma\delta} \\
& \cup \{f \in F_1 : (f, w) \in E\} \\
& \cup (P \cap (A_1(w) \cup \{f \in F_{\alpha\beta} : (f, w) \in E\})) \\
V_3 = \; & V - (V_0 \cup V_1 \cup V_2),
\end{aligned}$$

where $F_{\gamma\delta}$ is defined as

$$\{v \in V - (W \cup A_0 \cup A_1 \cup F_1 \cup F_{\alpha\beta}) : (v, w) \in E\}.$$

It is easy to see that the chord (in D) for a vertex in $F_{\gamma\delta}$ has one endpoint in γ and one in δ. We will show that $\{V_0, V_1, V_2, V_3\}$ is a simple decomposition of G, contradicting the primeness of G.

First, note that $V_0 \cup V_1$ contains at least two members, namely w and the members of $(A_1(w) \cup F_{\alpha\beta}) - P$ (which, as we have assumed, is non-empty). Furthermore $V_2 \cup V_3$ contains at least two members, since

$$|V_2 \cup V_3| \geq |W - \{w\}| \geq 4.$$

To see that every vertex in V_1 is adjacent to every vertex in V_2, let $v_1 \in V_1$ and $v_2 \in V_2$. The proof that $(v_1, v_2) \in E$ is straightforward if either $v_1 = w$ or if $v_1 \in F_{\alpha\beta}$ and $v_2 \in N_W(w) \cup F_{\gamma\delta}$. On the other hand, if $v_1 \in F_{\alpha\beta}$ and

$$v_2 \in \{f \in F_1 : (f, w) \in E\} \cup (P \cap A_1(w) \cup \{f \in F_1 \cup F_{\alpha\beta} : (f, w) \in E\})$$

then if v_1 were not adjacent to v_2 then v_2 would have determined v_1 (unless it were previously placed). In either case we would have $v_1 \in P$; thus $(v_1, v_2) \in E$.

We next show that no vertex in V_2 is adjacent to a vertex in V_0. Let $v_0 \in V_0$. Since $\text{chord}_D(v_0)$ has both endpoints in α or both in β, it cannot be adjacent to a member $N_W(w) \cup F_{\gamma\delta}$. So consider the other case, i.e. v_0 is adjacent to some $v_2 \in F_1 \cup P$. If $v_0 \in A_1(w) - P$ then it cannot be adjacent to v_2, since otherwise v_2 would determine v_0 and we would have $v_0 \in P$. On the other hand, if $v_0 \in A_0$ then it is connected by a chain to some $v \in (A_1(w) \cup F_{\alpha\beta}) - P$; v_0 cannot be adjacent to v_2 since otherwise v_2 would determine v, giving $v \in P$.

Finally, we show that no vertex $v_3 \in V_3$ is adjacent to a vertex $v \in V_0 \cup V_1$, by considering four cases. These cases are exhaustive since v_3 cannot be a member of $A_1(w)$ ($\subseteq V_0 \cup V_2$) or of $F_{\gamma\delta}$ ($\subseteq V_2$).

CASE 1: $v_3 \in W$.

Then $v_3 \neq w$ (since $v_3 \notin V_1$) and v_3 is not adjacent to w (since $v_3 \notin V_2$). Therefore the endpoint of $\text{chord}_D(v_3)$ must be both in γ or both in δ, whereas those of v are both in $\alpha \cup \beta$. Hence $(v_3, v) \notin E$.

CASE 2: $v_3 \in A_0$.

Then $(v_3, w) \notin E$, by the definition of A_0. If $v \neq w$ then $(v_3, v) \notin E$ since otherwise v_3 would be connected by a chain to some member of $(A_1(w) \cup F_{\alpha\beta}) - P$, implying $v_3 \in V_0$.

CASE 3: $v_3 \in F_1 \cup F_{\alpha\beta}$.

Then $(v_3, w) \notin E$. To see this note that if $v_3 \in F_1$ then $(v_3,\ w) \in E$ would imply $v_3 \in V_2$. On the other hand, if $v_3 \in F_{\alpha\beta}$ then $v_3 \in P$ (since $F_{\alpha\beta} - P \subseteq V_1$) so that again $(v_3,\ w) \in E$ would imply $v_3 \in V_2$.

Hence if $v \neq w$ and v_3 were adjacent to v then either $v \in (A_1(w) \cup F_{\alpha\beta}) - P$ in which case v_3 would determine v, or else v would be in A_0 and adjacent to or connected by a chain to some $v \in (A_1(w) \cup F_{\alpha\beta}) - P$ which would therefore be determined by v_3. In either case we have a contradiction; hence $(v_3, v) \notin E$.

CASE 4: $v_3 \in V - (W \cup A_0 \cup A_1(w) \cup F_1 \cup F_{\alpha\beta} \cup F_{\gamma\delta})$.

Then no endpoint of $\text{chord}_D(v_3)$ lies in $\alpha \cup \beta$, but both endpoints of $\text{chord}_D(v)$ lie in $\alpha \cup \beta$ (since $v \in V_0 \cup V_1$). Furthermore, $(v_3,\ w) \notin E$. Therefore $(v_3, v) \notin E$.

QED LEMMA 7

ANALYSIS OF TIME COMPLEXITY OF ALGORITHM A

We now describe how to implement Algorithm A in $O(|E|)$ time.

In order to facilitate finding the vertices connected by a chain to the vertex just dequeued, we do some preprocessing, as follows. We first find the connected components of the graph G/A_0. Then for each such component A , we construct a linked list of the vertices in $A_1(w) \cup F_{\alpha\beta}$ adjacent to at least one vertex in A . These lists can be constructed in a total of $O(|E|)$ time, as follows:

FOR each $v \in A_1(w) \cup F_{\alpha\beta}$ DO

FOR each $a \in A_0$ adjacent to v DO

add v to the list associated with a's component ;

Having done this preprocessing, whenever we dequeue some vertex p, for each $a \in A_0$ adjacent to p, we place and enqueue each q in the list associated with a's component, and then delete this list.

Next, we place and enqueue each $q \in A_1(w) - P$ adjacent to p.

Next, if $p \in F_1 \cup F_{\alpha\beta}$ and $(p,\ w) \notin E$ then we place and enqueue each $q \in F_{\alpha\beta} - P$ adjacent to p.

These three steps described so far can all be accomplish in $O(\deg(p))$ time.

Finally, if $p \in A_1(w)$ or if ($p \in F_1 \cup F_{\alpha\beta}$ and $(p,\ w) \in E$) then we place and enqueue each member of $F_{\alpha\beta} - P$ *not* adjacent to p. This can be done efficiently by maintaining a doubly-linked list to represent the set $F_{\alpha\beta} - P$; we first traverse p's adjacency list and marking (in the list for $F_{\alpha\beta} - P$) each member adjacent to p. This requires O(deg(p)) time. We then traverse the list for $F_{\alpha\beta} - P$, placing and enqueueing (and deleting from the list, i.e. inserting into P) each unmarked element. Thus, each $q \in F_{\alpha\beta}$ will be visited during such traversals at most a total of deg(q) + 1 times (since when q is placed and enqueued it is inserted into P so as never to be visited again).

Thus the total time of Algorithm A is O($|E|$).

7. Adding chords

At this point we have already computed a necessary model C for G/W, as well as a necessary placement $\mathrm{ch}(v)$ relative to C for each $v \in V - W$ adjacent to at least one member of W (recall from the definitions that $\mathrm{ch}(v)$ is a pair of empty arcs in the model defined by C). We now describe how to iteratively add vertices to W along with corresponding chords to C, while maintaining the necessary placements relative to C for these members of $V - W$. The algorithm is as follows:

$A_0 \leftarrow \{v \in V - W : N_W(v) = \varnothing\}$;
WHILE $W \neq V$ DO
  BEGIN
   $w \leftarrow$ some member of $V - W$ such that $|N_W(w)| \geq 1$;
       [[such a vertex must exist since G is connected]]
   $\alpha, \beta \leftarrow$ the two members of $\mathrm{ch}(w)$;
   $c_w \leftarrow$ some chord with one endpoint in α and one in β ;

   IF the subset of chords in C intersected by c_w does not equal $N_W(w)$
     THEN declare that G is not a circle graph and *halt* ;

   $A_1(w) \leftarrow N_V(w) \cap A_0$;
   $A_0 \leftarrow A_0 - N_V(w)$;
   $F_{\alpha\beta} \leftarrow \{v \in V - W : \mathrm{ch}(v) = \{\alpha, \beta\}\}$;
   $T \leftarrow V - (W \cup A_1(w) \cup A_0 \cup F_{\alpha\beta})$;

   $F_{\alpha_0} \leftarrow \{v \in T : \mathrm{ch}(w) = \{\alpha, \phi\}$ for some arc ϕ such that $(\phi \subseteq \gamma)$ iff $(v, w) \notin E\}$;
   $F_{\alpha_1} \leftarrow \{v \in T : \mathrm{ch}(w) = \{\alpha, \phi\}$ for some arc ϕ such that $(\phi \subseteq \gamma)$ iff $(v, w) \in E\}$;
   $F_{\beta_0} \leftarrow \{v \in T : \mathrm{ch}(w) = \{\beta, \phi\}$ for some arc ϕ such that $(\phi \subseteq \gamma)$ iff $(v, w) \notin E\}$;
   $F_{\beta_1} \leftarrow \{v \in T : \mathrm{ch}(w) = \{\beta, \phi\}$ for some arc ϕ such that $(\phi \subseteq \gamma)$ iff $(v, w) \in E\}$;
   $F_1 \leftarrow F_{\alpha_0} \cup F_{\alpha_1} \cup F_{\beta_0} \cup F_{\beta_1}$;

   FOR each $x \in \{\alpha_0, \alpha_1, \beta_0, \beta_1\}$ DO
     FOR each $v \in F_x$ DO
       $\mathrm{ch}(v) \leftarrow (ch(v) - \{\alpha, \beta\}) \cup \{x\}$;

   Call Algorithm A with parameter w, update $\mathrm{ch}(v)$ for each $v \in A_1(w) \cup F_{\alpha\beta}$
       with its value returned by this call;

   $C \leftarrow C \cup \{c_w\}$;
  END;

[[G is a circle graph]]
output C as a model for G ;

The arcs α_0, α_1, β_0, β_1, γ and δ are defined here as they were in Section 6 (see Fig. 12).

Thus, at each iteration, the algorithm chooses some w adjacent to at least one member of W (hence at all times, G/W is connected). We then use Algorithm A to refine the placements given to the vertices; that is, we must choose among two possibilities for placing each $v \in A_1(w)$ (whose necessary placement has both endpoints in α or both in β) and each $v \in F_{\alpha\beta}$ (whose necessary placement has one endpoint in α and one in β). The set $A_1(w)$ is computed exactly as in Section 6. The set $F_{\alpha\beta}$ is computed by looking at the necessary placements ch(v), relative to W, that we have already computed; note that now there may be some vertices W-similar to w but *not* included in $F_{\alpha\beta}$, since w may be W-similar to other members of W.

In general, we cannot assume here (as we did in Section 6) that G/W is prime. So, for example, it might be that δ consists of a single point; that is, w_2 might equal w_3, (making w an articulation point of $G/(W \cup \{w\})$) so that Lemma 4 would not be true in this context. However, Lemmas 5, 6 and 7, which constitute the proof of correctness of Algorithm A, do not depend on $G/(W \cup \{w\}$ being prime; it suffices that it is connected and has at least five vertices.

The sets F_{α_0}, F_{α_1}, F_{β_0} and F_{β_1} have a simpler interpretation and are conceptually simpler to compute than they were in Section 6. Recall that in Section 6, F_{α_0} was interpreted as the union of those vertices that *might* have one endpoint in a_0 but none in any other principal arc. This uncertainty was not harmful because it turned out that if those vertices did indeed determine any members of $A_1(w) \cup F_{\alpha\beta}$ then we knew that they must have an endpoint in a_0. Here we can simply define F_{α_0} as the vertices $v \in V - W$ having necessary placements with one endpoint in α_0 and one in $\gamma \cup \delta$; we can do this because we already have ch(v) (from the previous iteration). There is only one minor detail: ch(v) $= \{\alpha, \phi\}$ for some arc ϕ is a placement of v relative to C, but we need a placement of v relative to $C \cup \{c_w\}$; that is, we need to replace α in ch(v) by either α_0 or α_1. This is accomplished by making the simple test as described in step

(14): if ϕ is contained within γ then choose α_0 if $(v, w) \notin E$ and α_1 otherwise; on the other hand, if ϕ is contained within δ then choose α_0 if $(v, w) \in E$ and α_1 otherwise. This placement of v is clearly necessary relative to $C \cup \{c\}$, so we record it in step (21). The sets F_{α_1}, F_{β_0}, F_{β_1} are computed analogously.

Having computed A_0, $A_1(w)$, and F_1 in this way, we perform Algorithm A precisely as it appears in Fig. 16. The proof of correctness also is unchanged.

8. Remarks

We can now characterize unique representability as follows:

COROLLARY: Let G be a circle graph with at least five vertices. Then G is prime if and only if it is uniquely representable.

PROOF: If G is prime, then our algorithm either finds that it is not a circle graph, or constructs a necessary model for it.

Conversely, assume that G is uniquely representable. Assume for a contraction that there is a partition $\{V_0, V_1, V_2, V_3\}$ yielding a simple decomposition $\{G/(V_0 \cup V_1 \cup m_1), G/(V_2 \cup V_3 \cup m_2)\}$, for some $m_1 \in V_2$, $m_2 \in V_1$, with models C_1 and C_2, respectively. Let

$$\pi(C_1) = (m_1, A_1, m_1, B_1)$$

and

$$\pi(C_2) = (m_2, A_2, m_2, B_2),$$

where, for $i = 1,2$, A_i (resp. B_i) denotes the subsequence of vertices in $\pi(C_i)$ appearing after the first (resp. second) occurrence of m_i.

Then the sequences

$$(A_1 A_2 B_1 B_2)$$

and

$$((A_1)^R A_2 B_1 B_2)$$

where $(A_1)^R$ is the reverse of sequence A_1, can each be obtained from a traversal of a set of chords D_1 and D_2, respectively, (again see Fig. 4). It is easily verified that D_1 and D_2 are each models for G and that it is not true that $D_1 \sim D_2$, contradicting the unique representability of G. **QED**

Unique representability of certain circle graphs is discussed in [Bu].

REFERENCES

[Bo] A. Bouchet, "Reducing Prime Graphs and Recognizing Circle Graphs," submitted for publication.

[Bu] M. A. Buckingham, "Circle Graphs", Courant Computer Science Report No. 21, Ph.D. thesis, Courant Inst. of Math. Sciences, Computer Science Dept., New York Univ.

[Cu] W. H. Cunningham, "Decomposition of Directed Graphs," *SIAM J. Alg. Disc. Meth.*, Vol. 3, No. 2 (1982), pp. 214-228.

[CE] W. H. Cunningham and J. Edmonds, "A Combinatorial Decomposition Theory," *Canadian Journal of Mathematics*, Vol. 22 (1980), 734-765.

[CP] D. G. Corneil, Y. Perl and L. K. Stewart, "A Linear Recognition Algorithm for Cographs", *SIAM J. Comput.*, vol. 14, no. 4 (Nov. 1985), pp. 926-934.

[EI] S. Even and A. Itai, "Queues, Stacks and Graphs," in *Theory of Machines and Computations,* A. Kohavi and A. Paz (ed.), Academic Press, New York, 1971, pp. 71-86.

[GJ] M. R. Garey, D. S. Johnson, G. L. Miller and C. H. Papadimitriou, "The Complexity of Coloring Circular Arcs and Chords," *SIAM J. Alg. Discr. Meth.,* Vol. 1 (1980), pp. 216-228.

[Ga] F. Gavril, "Algorithms for a Maximum Clique and a Maximum Independent Set of a Circle Graph," *Networks,* Vol. 3 (1973), pp. 261-273.

[GH] C. P. Gabor, W.-L. Hsu and K. J. Supowit, "Recognizing Circle Graphs in Polynomial Time," *Proc. 26th IEEE Symp. on Foundations of Computer Science* (1985), pp. 106-116.

[Go] M. Golumbic, *Algorithmic Graph Theory and Perfect Graphs,* Academic Press, New York, 1980.

[Hs] W.-L. Hsu, "Maximum Weight Clique Algorithms for Circular-Arc Graphs and Circle Graphs," *SIAM J. Comput.*, Vol. 14, No. 1 (Feb. 1985), pp. 224-231.

[PE] A. Pneuli, S. Even and A. Lempel, "Transitive Orientation of Graphs and Identification of Permutation Graphs," *Canadian J. of Mathematics*, Vol. 23 (1971), pp. 160-175.

[RU] D. Rotem and J. Urrutia, "Finding Maximum Cliques in Circle Graphs," *Networks,* Vol. 11 (1981), pp. 269-278.

[Sp] J. Spinrad, "On Comparability and Permutation Graphs," *SIAM J. Comput.*, Vol. 14, No. 3 (August 1985), pp. 658-670.

[Tu] A. Tucker, "An Efficient Test for Circular-arc Graphs," *SIAM J. Comput.*, Vol. 9, No. 1 (Feb. 1980), pp. 1-24.

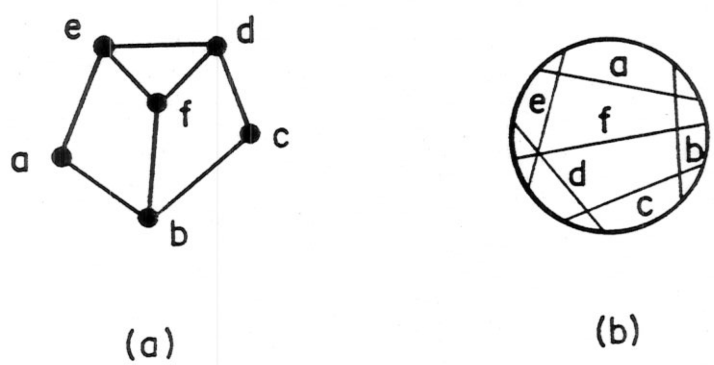

Fig. 1

A graph along with a model for it.

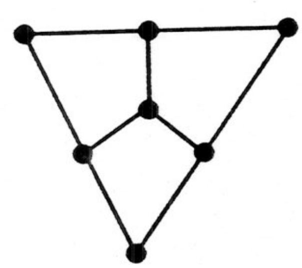

Fig. 2

A graph that is *not* a circle graph.

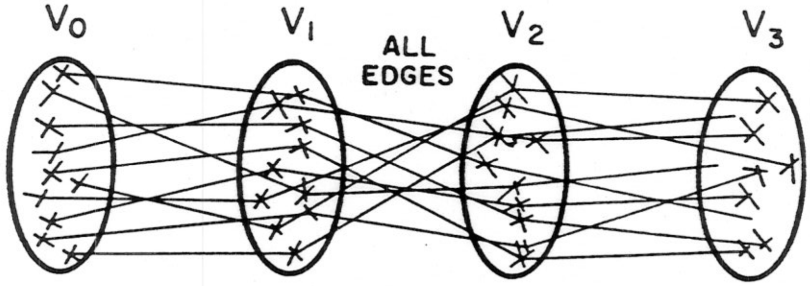

Fig. 3

A simple decomposition of a graph.

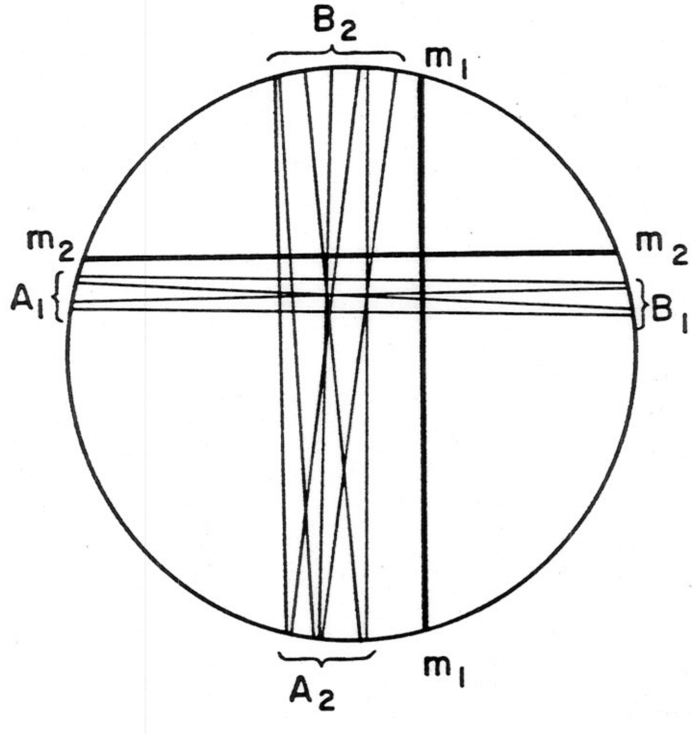

Fig. 4

Illustration for the proof of Lemma 1.

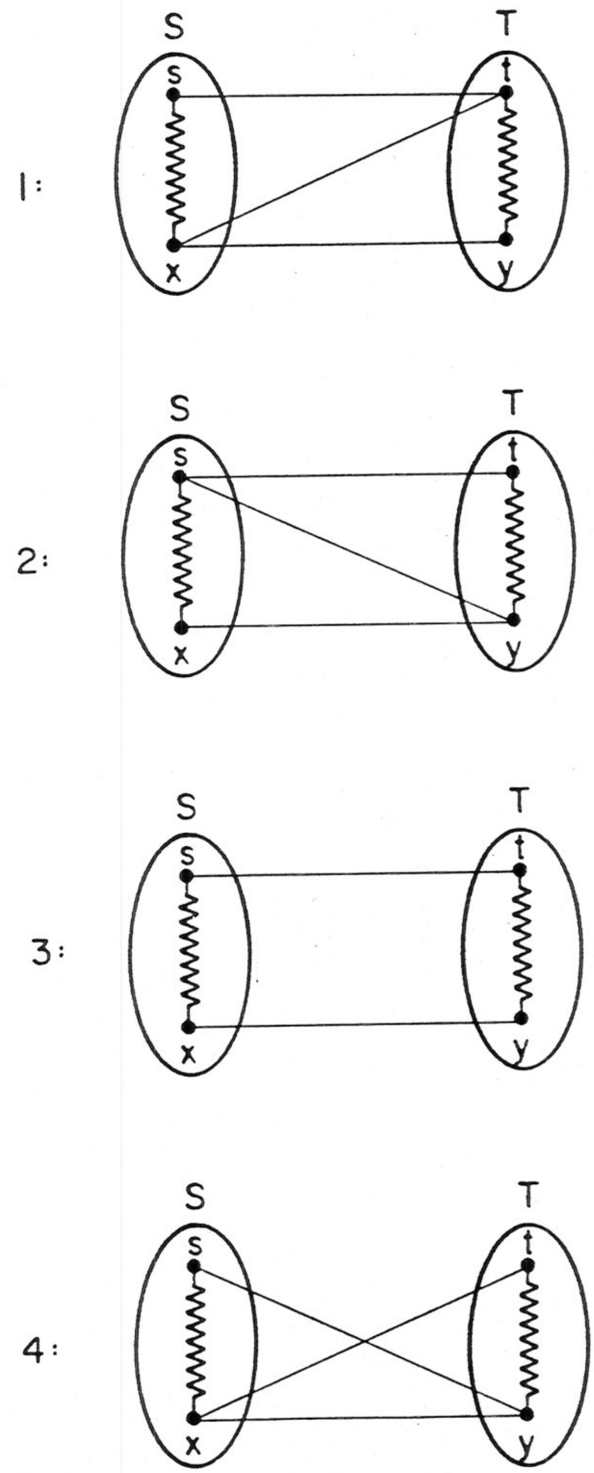

Fig. 6

The four types of violations.

A straight line indicates an edge, an absence of line indicates no edge, a wavy line indicates that an edge may or may not be present.

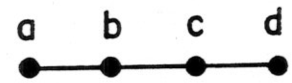

Fig. 7

A graph that is a P4.

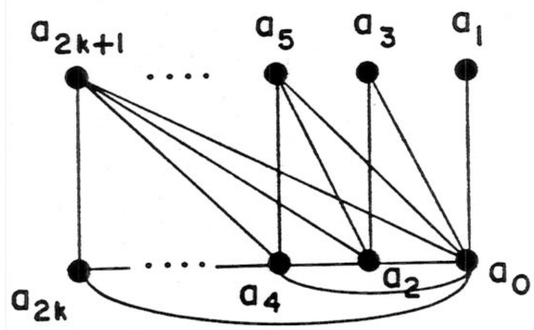

Fig. 8

The graph H_k.

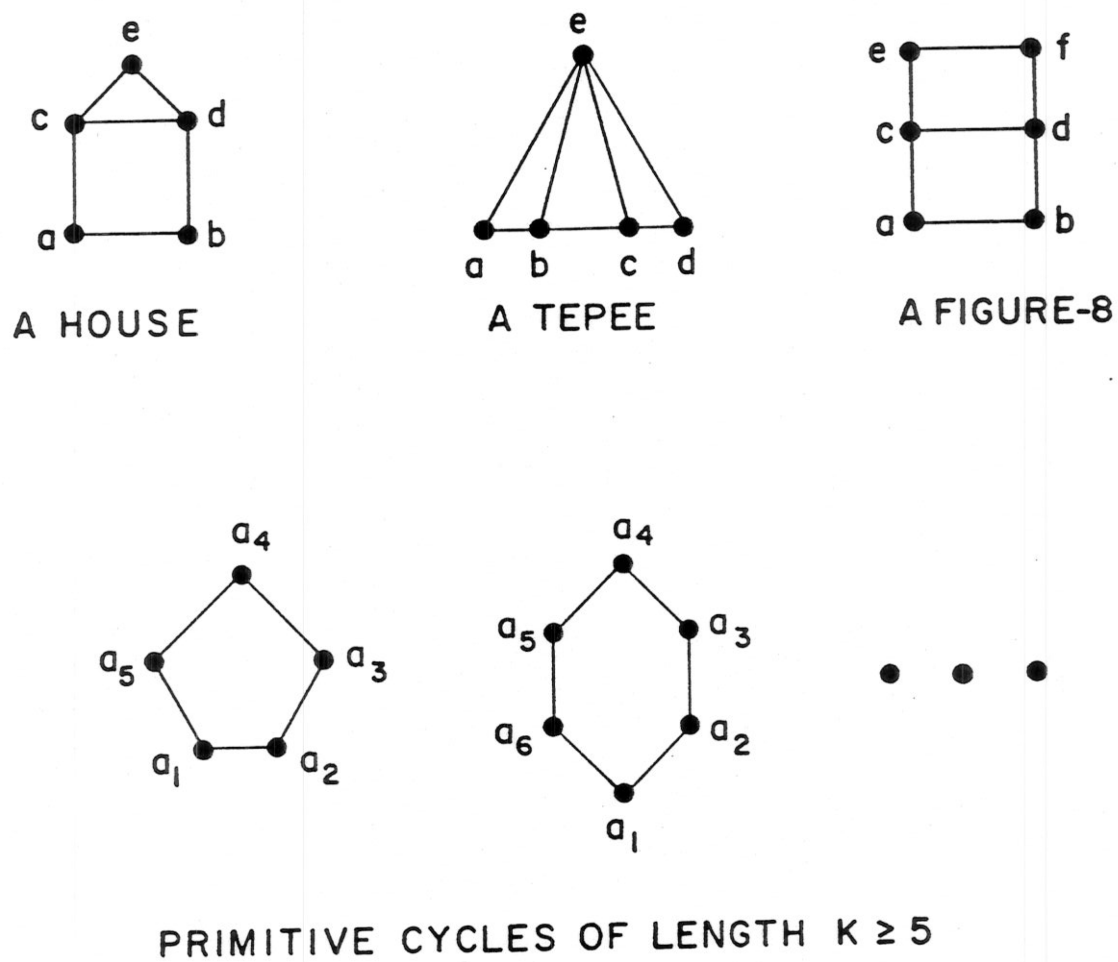

Fig. 9

Graphs in the set **F**.

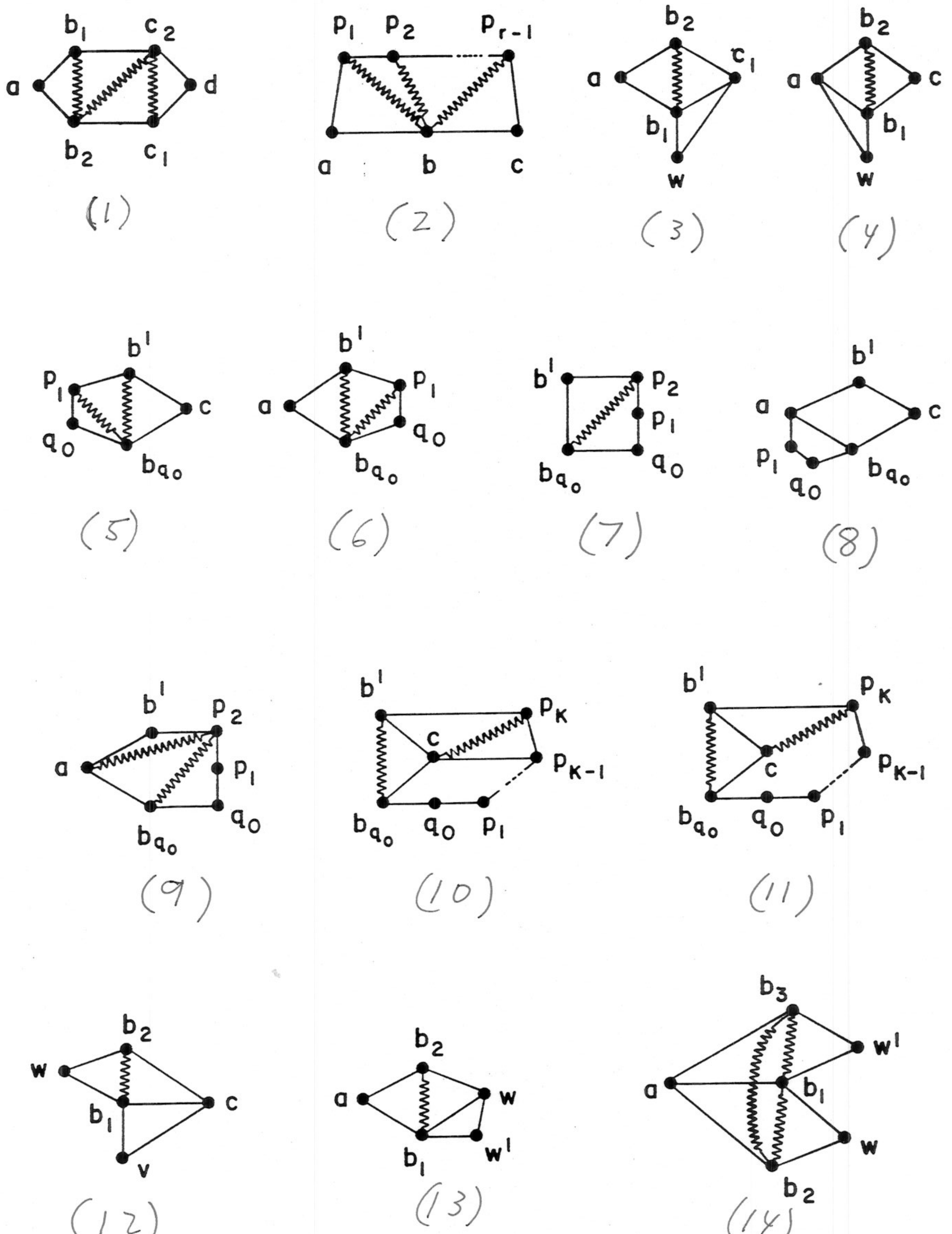

Fig. 10

Illustrations for the proof of Theorem 2.

A straight line indicates an edge, an absence of line indicates no edge, a wavy line indicates that an edge may or may not be present.

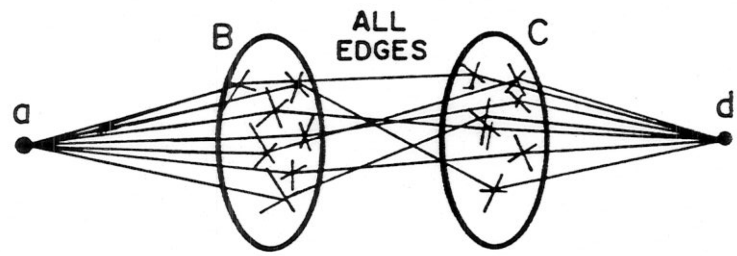

Fig. 10.15

Another illustration for the proof of Theorem 2.

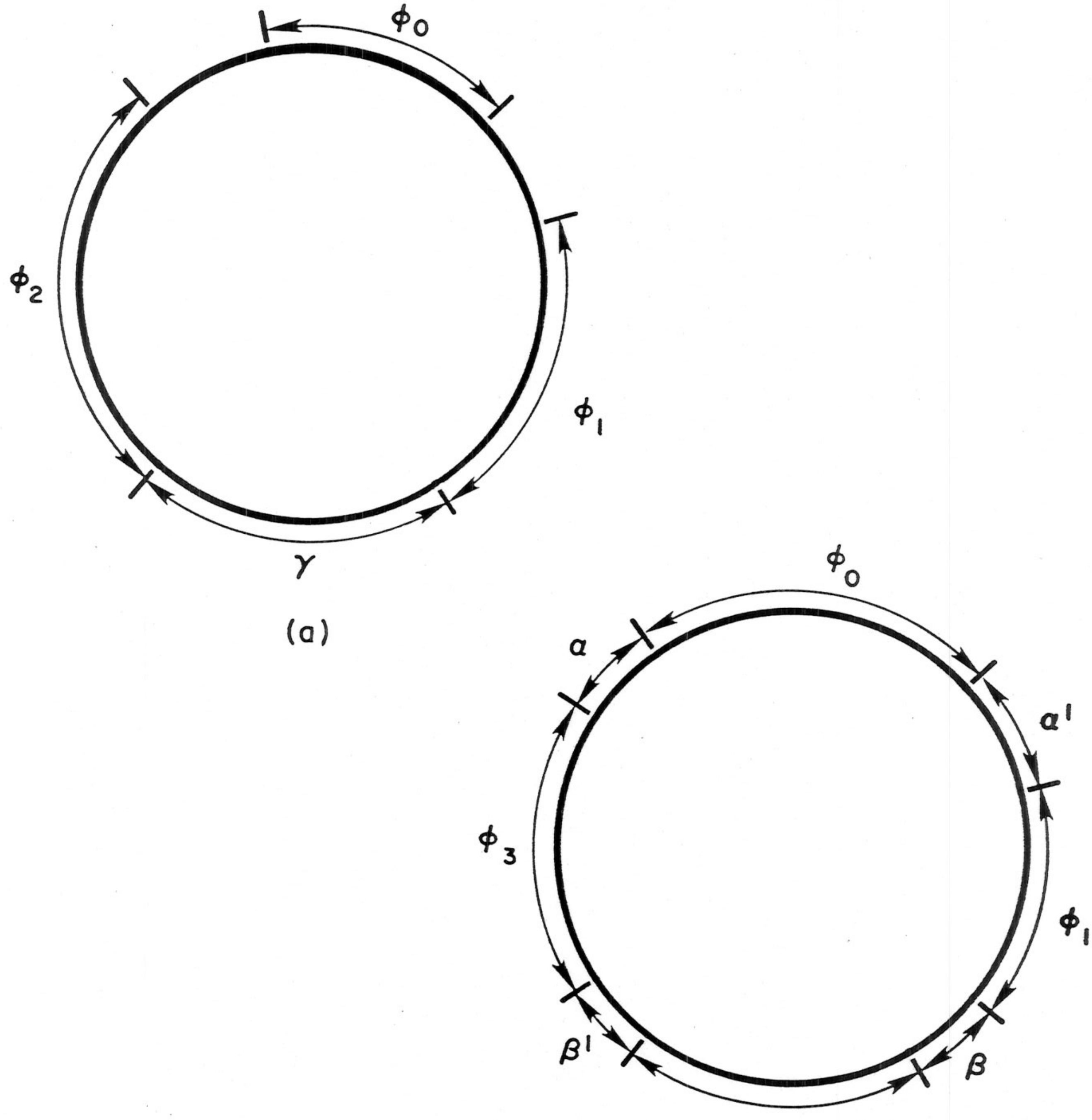

Fig. 11

Illustrations for the proof of Lemma 3.

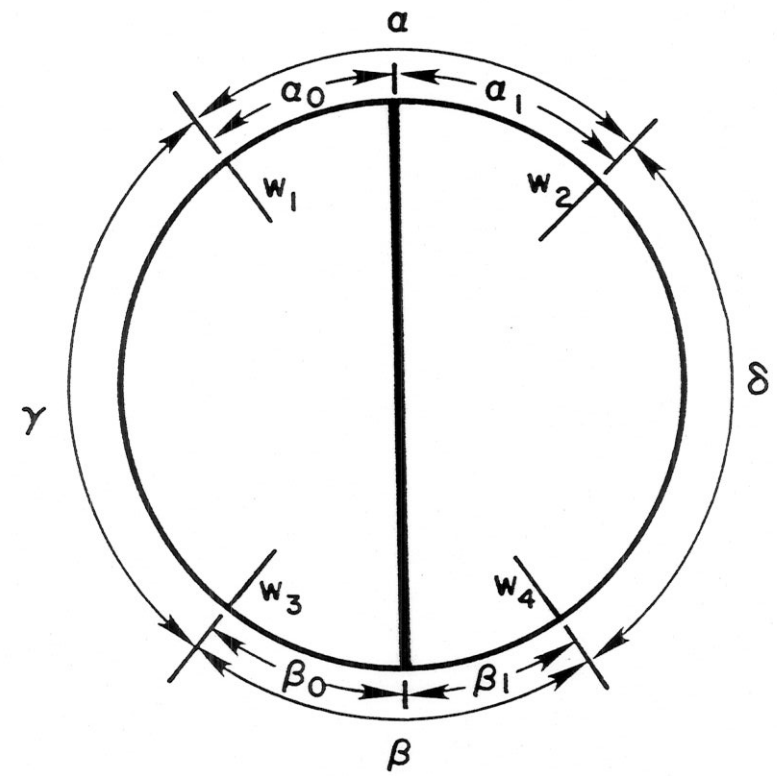

Fig. 12

The arcs referenced by Algorithm A.

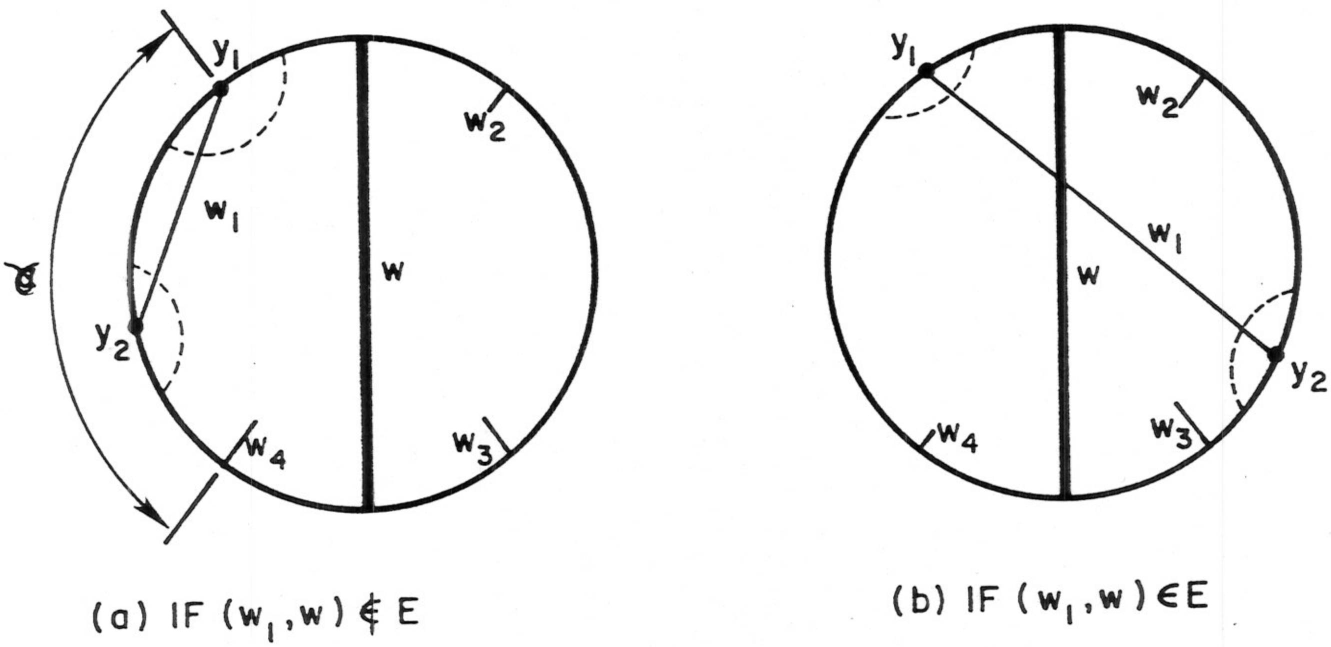

Fig. 13

The possible placements of $v \in A_1(w_1)$ are shown in broken line.

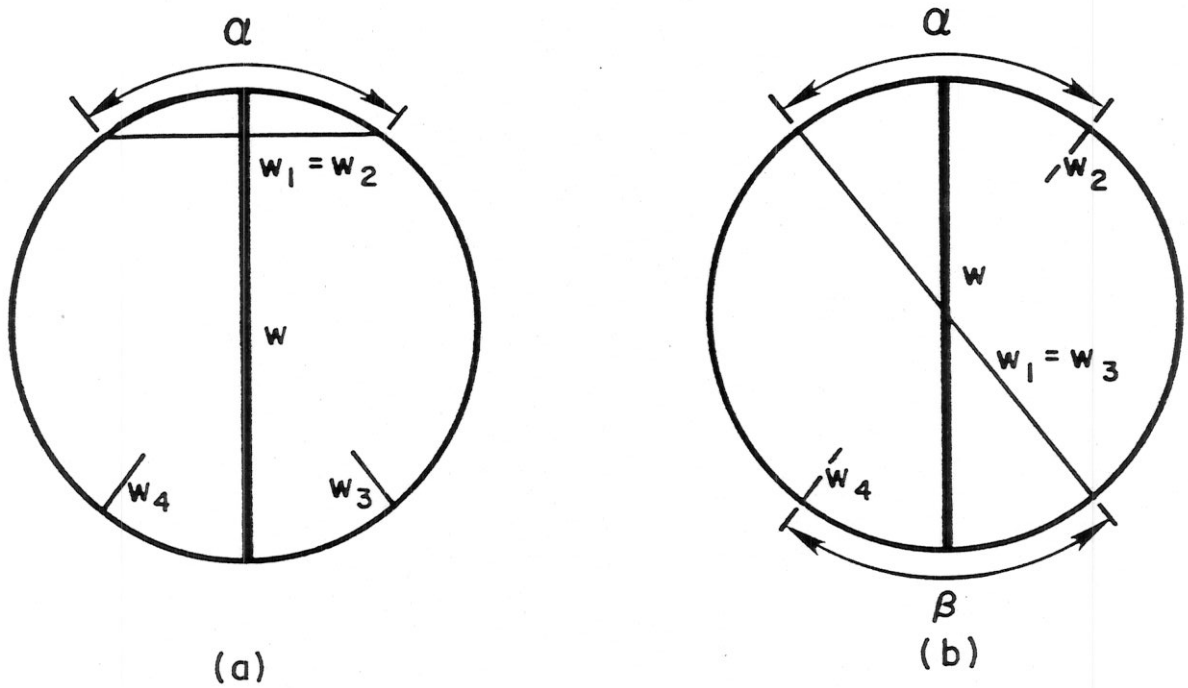

Fig. 14

Illustrations for the proof of Lemma 4.

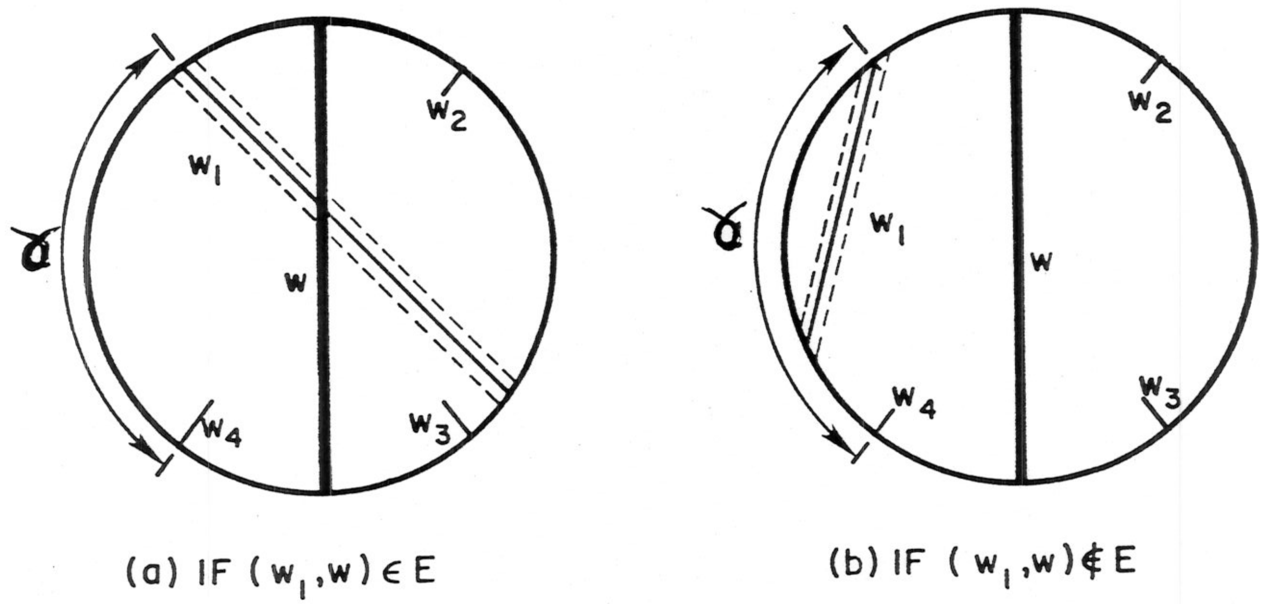

Fig. 15

The possible placements of $v \in M(w_1)$ are shown in broken line.

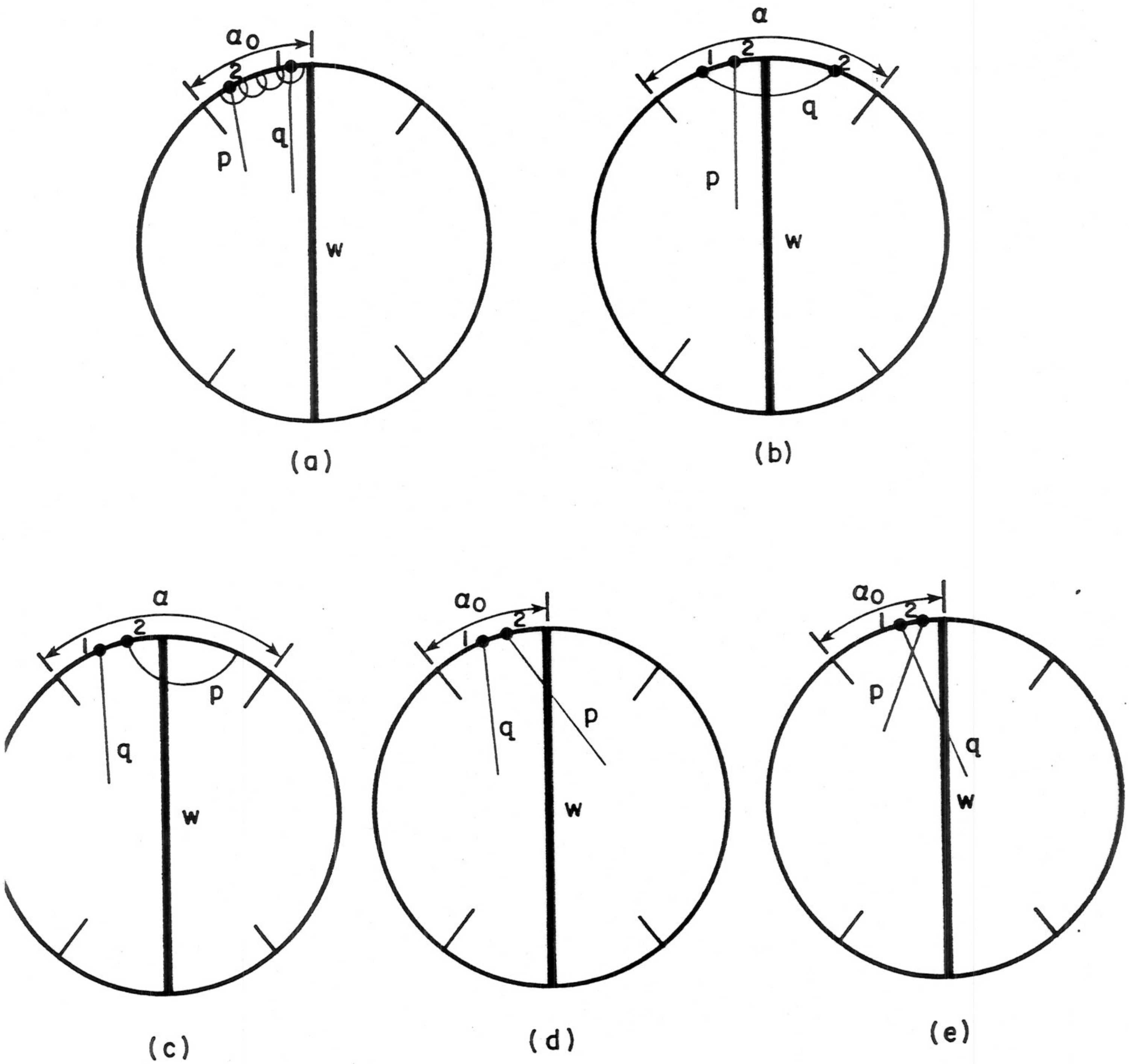

Fig. 17

The five ways for p to determine q.

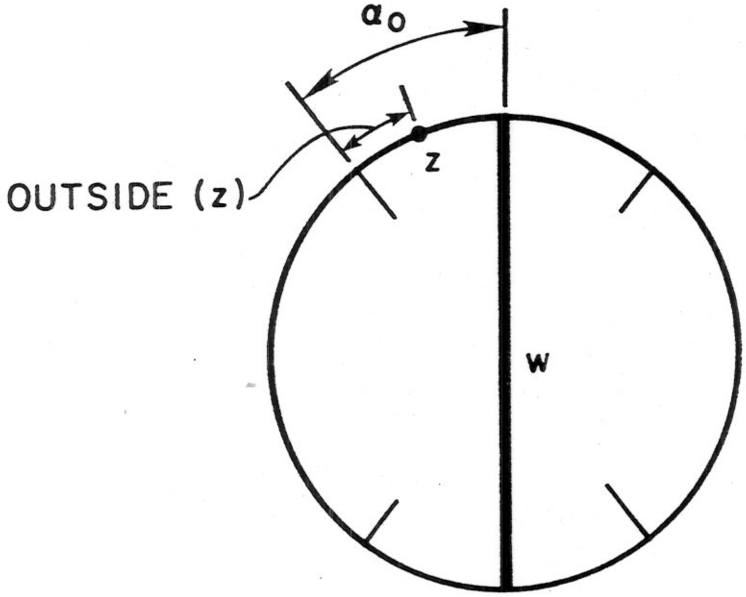

Fig. 18

Definition of outside(z).

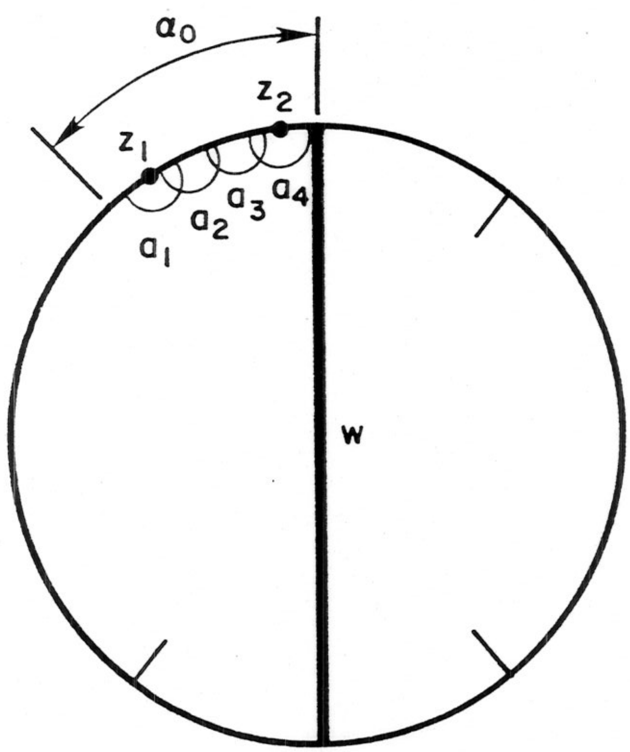

Fig. 19

The chain a_1, a_2, a_3, a_4 connects points z_1 and z_2.

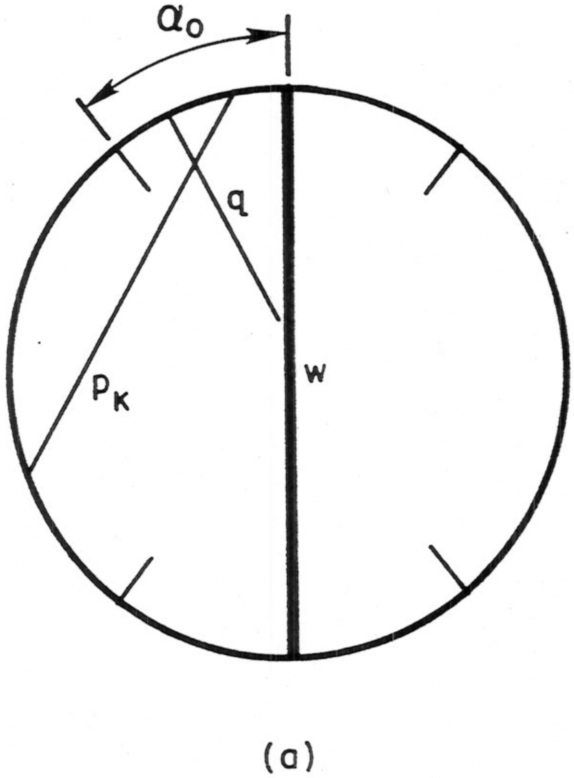

Fig. 20

Illustrations for proof of part 1.B, basis step, in Lemma 6.

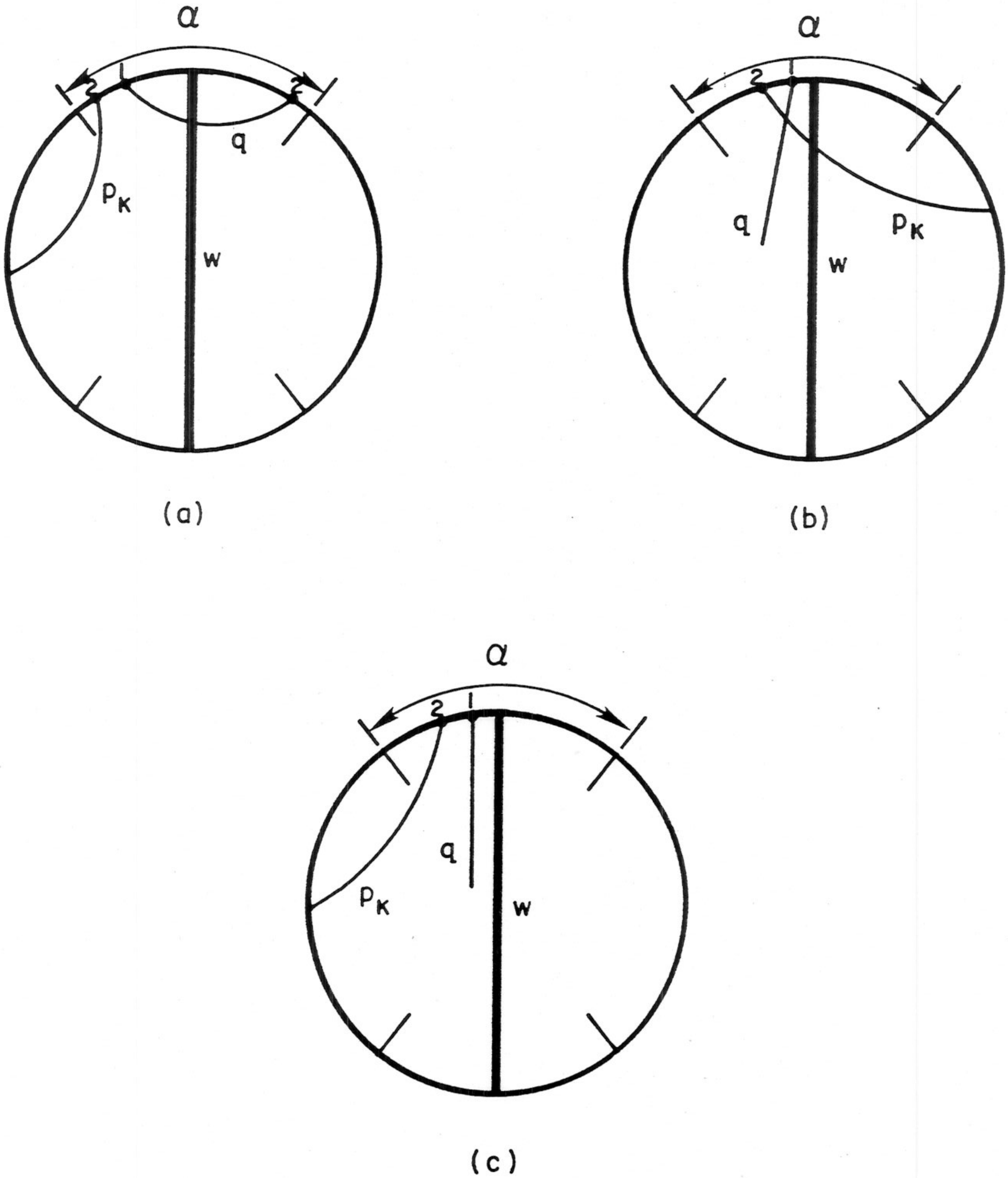

Fig. 21

Illustrations for proof of part 1.C, basis step, in Lemma 6.

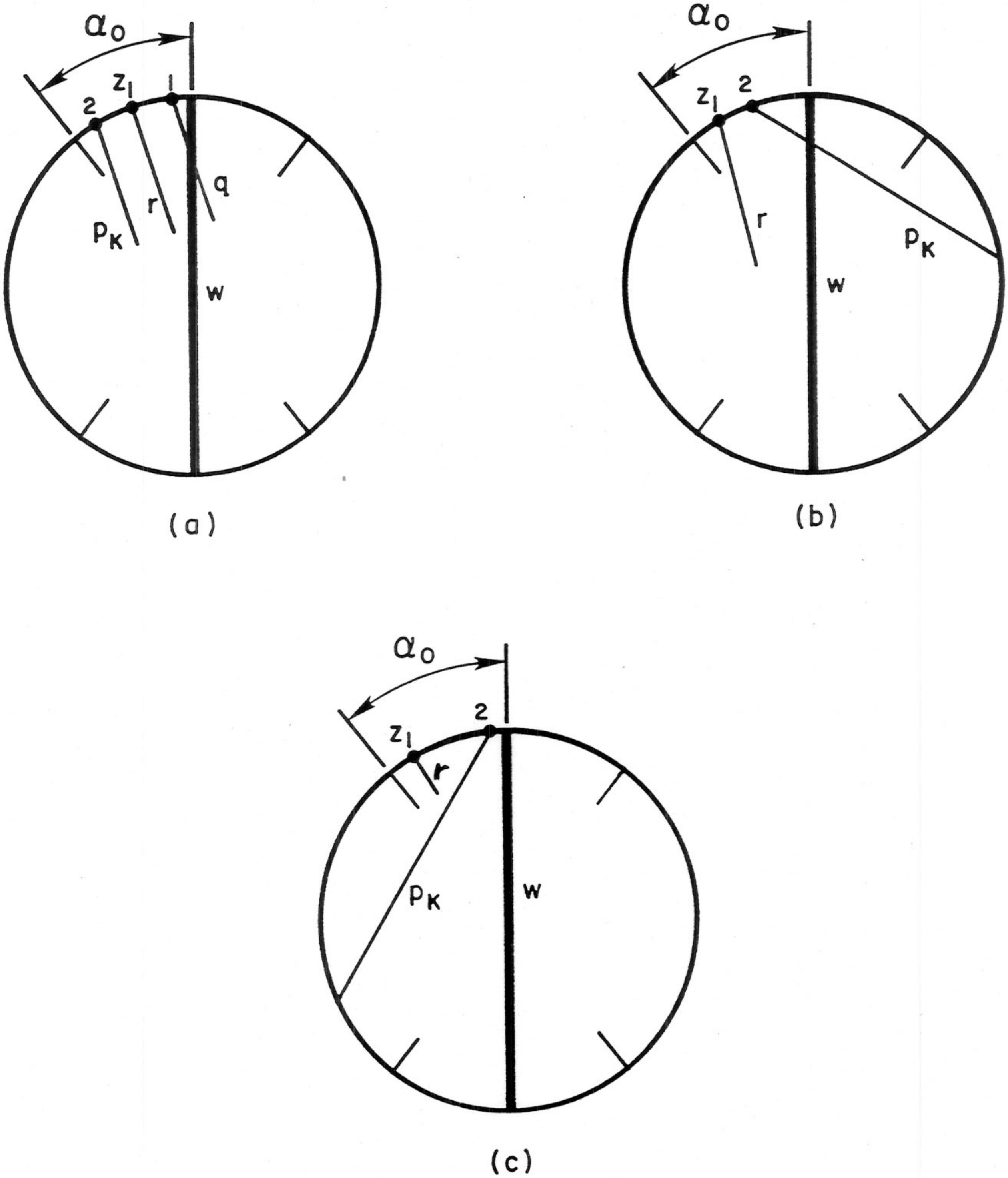

Fig. 22

Illustrations for proof of part 2, basis step, in Lemma 6.

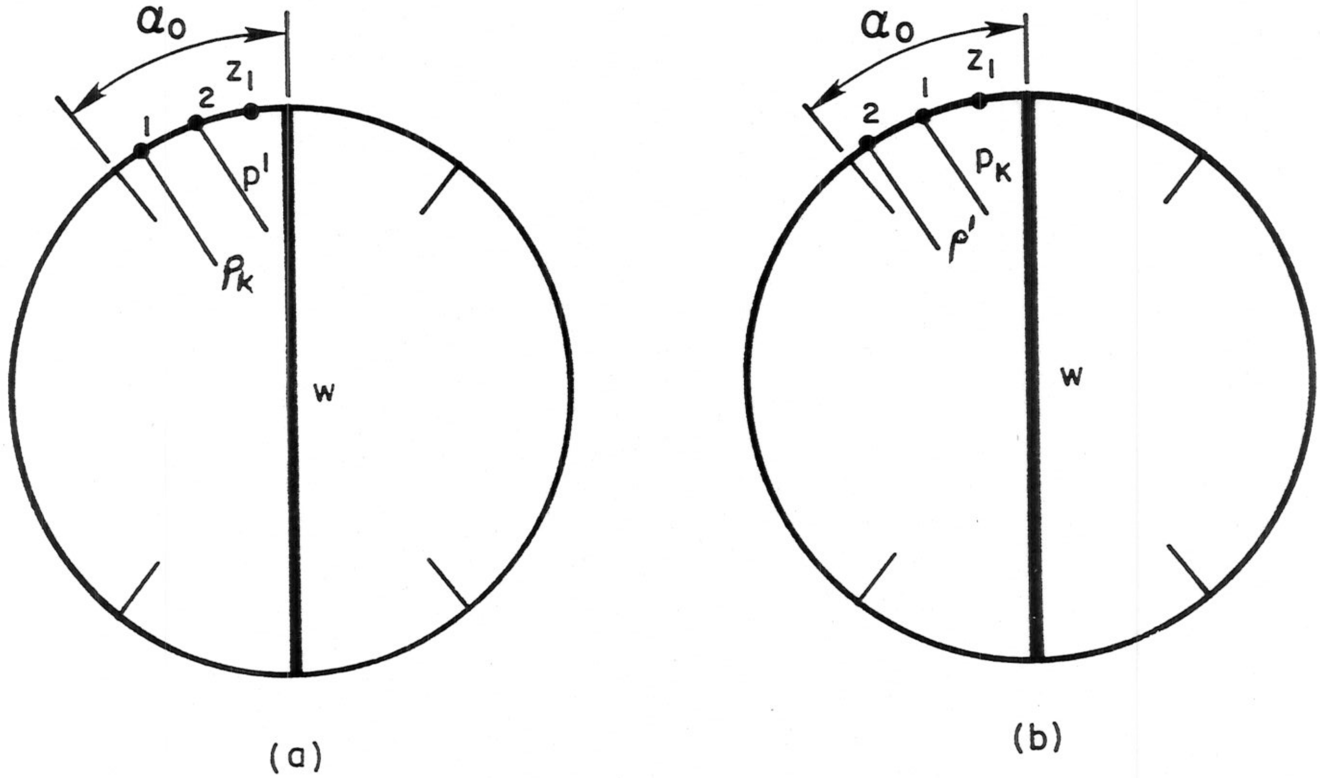

Fig. 23

Illustrations for proof of part 1.A, inductive step, in Lemma 6.

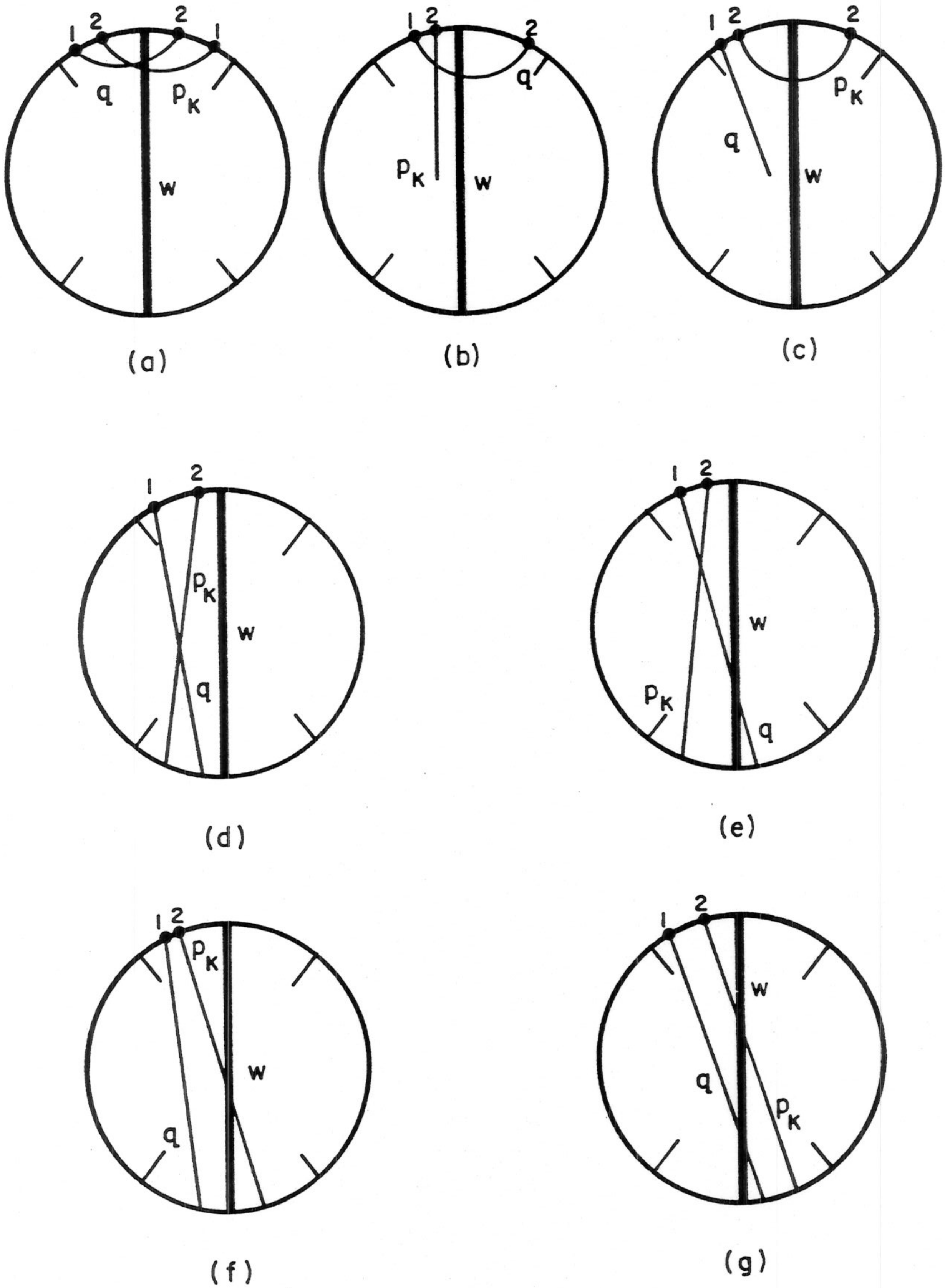

Fig. 24

Illustrations for proof of part 1.B, inductive step, in Lemma 6.

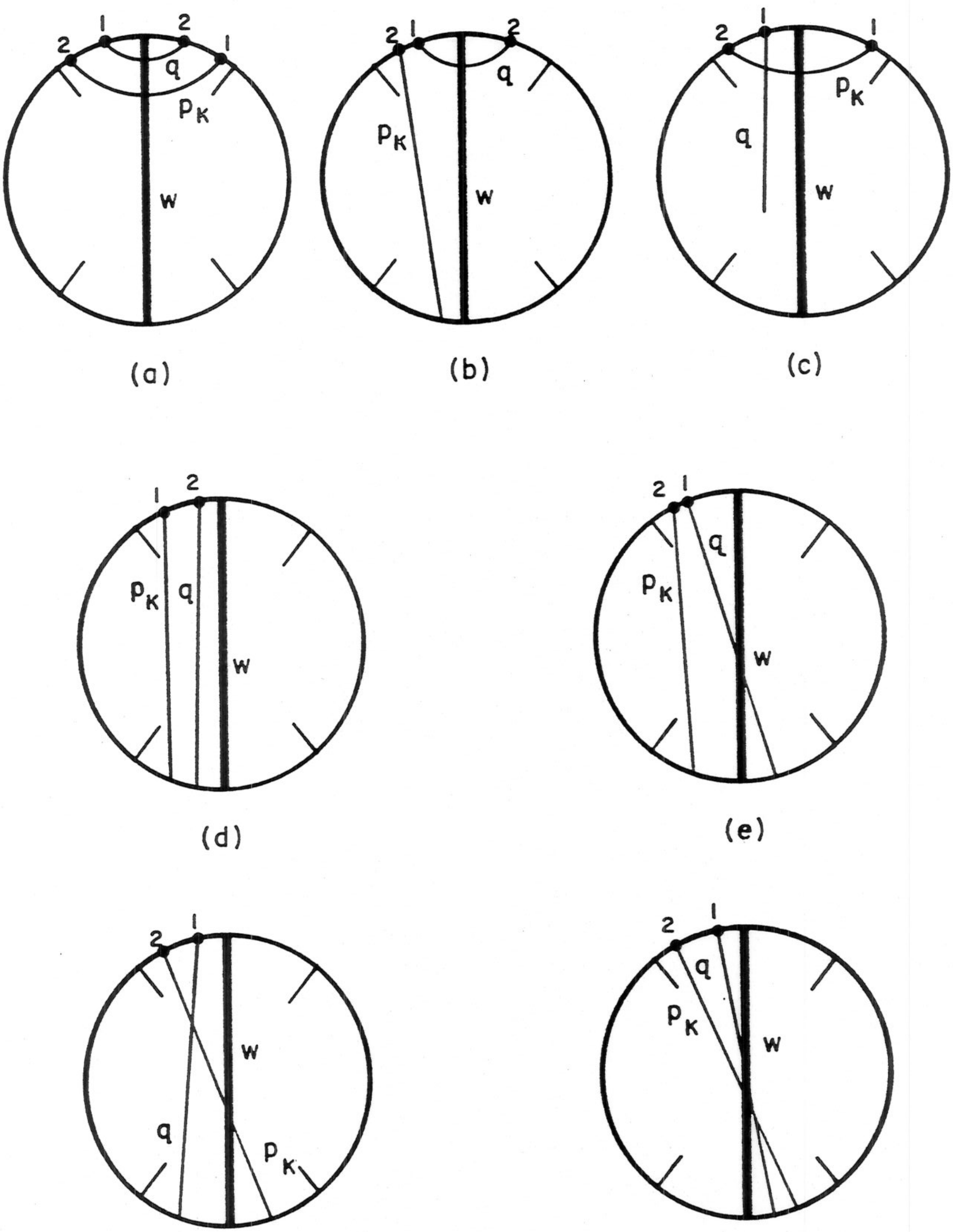

Fig. 25

Illustrations for proof of part 1.C, inductive step, in Lemma 6.

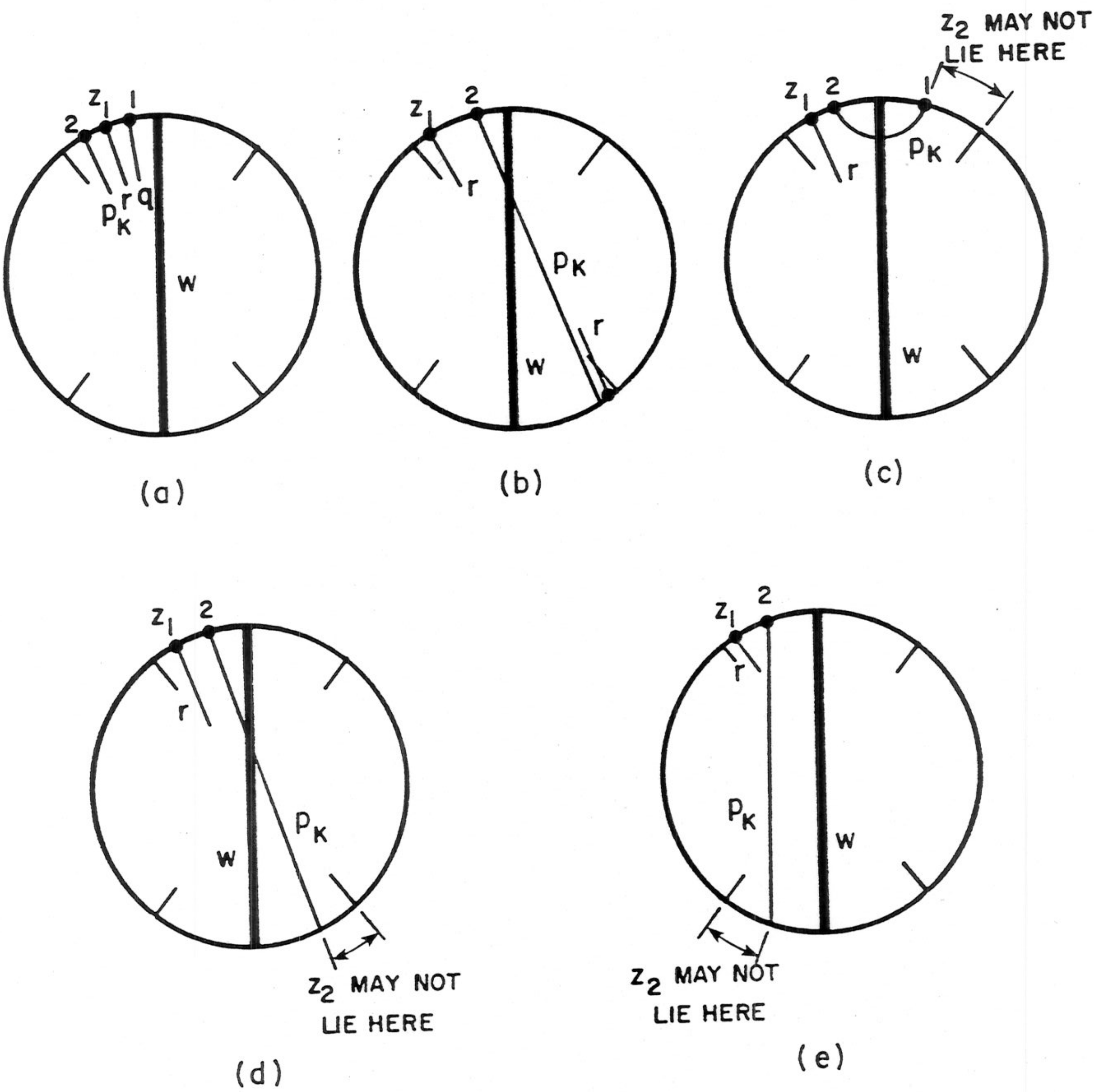

Fig. 26

Illustrations for proof of part 2, inductive step, in Lemma 6.