# Chapter 9. Primitivity

This chapter is concerned with partitioning the set $\Omega$ of points so that the group $G$ acts on the partition. That is, can we partition $\Omega$ into disjoint subsets $B_1, B_2, \ldots, B_r$ so that for all $i$ and for all $g \in G$, $B_i^g$ is some other subset $B_j$ (where $j$ depends on $i$ and $g$, of course)?

This is important because the group $G$ can then be studied as a permutation group on $\{1,2,\ldots,r\}$ rather than as a permutation group on $\Omega$. The smaller degree of the permutations should simplify the computations.

## Definitions

A partition

$$\Omega = \{ B_1 \mid B_2 \mid \cdots \mid B_r \}$$

of $\Omega$ into disjoint subsets is *invariant* under $G$ if the image of each subset $B_i$ of the partition under any element of the group $G$ is also a subset of the partition. That is, for all $i$ and for all $g \in G$, there is a $j$ such that $B_i^g = B_j$. Or, alternatively, each element of $G$ permutes the subsets amongst themselves.

As an example, the group $G$ acting on $\{1,2,\ldots,20\}$ and generated by

$a$=(1,2,3,4,5)(6,10,13,15,9)(7,11,14,8,12)(16,17,18,19,20) and
$b$=(1,2)(3,4)(7,11)(8,10)(9,12)(14,15)(16,17)(18,19)

has the following invariant partitions.

(1) $B_1=\{1\}, B_2=\{2\}, \ldots, B_{20}=\{20\}$

(2) $B_1=\{1,2,\ldots,20\}$

(3) $B_1=\{1,2,\ldots,5\}$ $B_2=\{6,7,\ldots,15\}$ $B_3=\{16,17,\ldots,20\}$

(4) $B_1=\{1,2,\ldots,15\}$ $B_2=\{16,17,\ldots,20\}$

(5) $B_1=\{1,16\}$ $B_2=\{2,17\}$ $B_3=\{3,18\}$ $B_4=\{4,19\}$ $B_5=\{5,20\}$ $B_6=\{6,7,\ldots,15\}$

A *trivial* invariant partition of a set $\Omega$ is either

the discrete partition where $B_i = \{i\}$, for all $i \in \Omega$, or

the complete partition where $B_1 = \Omega$, or

the partitions formed by the orbits of $G$, where each $B_i$ is a (union of ) orbits of $G$.

The only non-trivial partition in the above example is (5). The first is the discrete partition. The second is the complete partition. The third and fourth are formed by the orbits of $G$.

A group $G$ acting on a set $\Omega$ is *primitive* if there is no non-trivial partition of $\Omega$ that is invariant under $G$. A group that has a non-trivial invariant partition is called *imprimitive*.

The above group is imprimitive.

Generally, the concept of primitivity is restricted to *transitive* groups. They are groups with only one orbit, the whole of $\Omega$. In this case, all the subsets in an invariant partition are the same size.

## Finest Invariant Partition

The algorithm we will present finds the finest invariant partition of $\Omega$ where one of the subsets contains the set $\{\omega_1, \omega_2\}$, for the given points $\omega_1 < \omega_2$. The algorithm requires a generating set of the group $G$. Using this algorithm and running through all such pair of points, we can determine whether there is a non-trivial invariant partition and, hence, whether $G$ is primitive or imprimitive.

Let

$$partition(\omega_1, \omega_2) = \{ B_1 \mid B_2 \mid \cdots \mid B_r \}$$

be the finest invariant partition of $\Omega$ where the points $\omega_1$ and $\omega_2$ are in the same subset of the partition. For any point $\alpha$, let $B(\alpha)$ denote the subset of $partition(\omega_1, \omega_2)$ which contains $\alpha$. This partition may also be represented as a function

$$f(\omega_1, \omega_2) : \Omega \rightarrow \Omega$$

where each point $\alpha$ is mapped to the smallest point in the subset $B(\alpha)$. Then, if $\alpha$ is the smallest point in a subset, $B(\alpha) = f(\omega_1, \omega_2)^{-1}(\alpha)$.

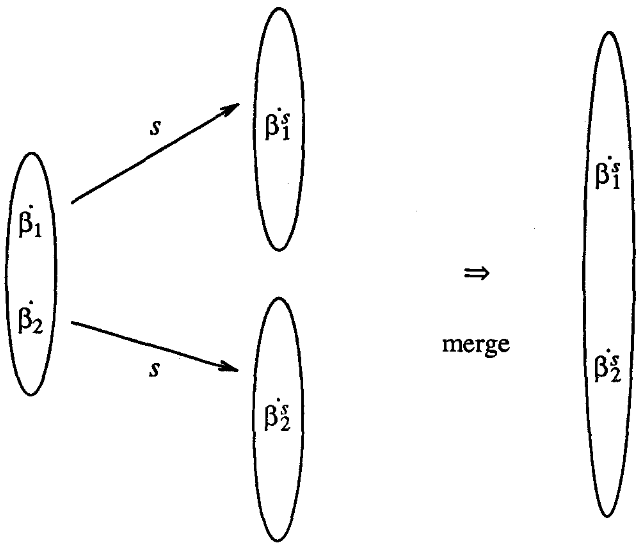
The algorithm will have a partition

$$f : \Omega \rightarrow \Omega$$

which is an approximation to $f(\omega_1, \omega_2)$. At all times $f$ will be a refinement of $f(\omega_1, \omega_2)$, so that $f^{-1}(f(\alpha)) \subseteq B(\alpha)$, for all points $\alpha$. Initially, we force $\omega_1$ and $\omega_2$ to be in the same subset. The approximation will converge to an invariant partition by merging subsets. It must converge to $f(\omega_1, \omega_2)$, since $f(\omega_1, \omega_2)$ is the finest such partition.

The only requirement we have for merging subsets of $f$ is that the final result must be invariant under $G$. So we check whether the partition $f$ is invariant and correct the situation whenever we discover it is not yet invariant. The partition is invariant if, for every pair of points $\beta_1$ and $\beta_2$ where $f(\beta_1) = f(\beta_2)$ and for every generator $s$ of $G$, $f(\beta_1^s) = f(\beta_2^s)$. When we discover a pair where $f(\beta_1) = f(\beta_2)$ but not $f(\beta_1^s) = f(\beta_2^s)$ then we merge the subsets containing $f(\beta_1^s)$ and $f(\beta_2^s)$.

**Figure 1 : Merging Subsets of the Partition**



The procedure can be simplified greatly by noting that only pairs where $\beta_1$ is the smallest point in the subset have to be used. Let $\alpha$ be the smallest point in the subset. If we verify that the generator $s$ always maps such pairs to the same subset then for a general pair we have

$$f(\beta_1^s) = f(\alpha^s) = f(\beta_2^s)$$

because we have checked the action of $s$ on the pairs $\alpha$ and $\beta_i$, for $i=1,2$. This gives us Algorithm 1.

**Algorithm 1 : Minimal partition given a pair of points**

Input : a group $G$ generated by $S = \{s_1, s_2, \ldots, s_m\}$ acting on a set $\Omega = \{1,2,\ldots,n\}$;
two points $\omega_1 < \omega_2$ of $\Omega$;

Output : the finest invariant partition $f$ of $\Omega$ such the the points $\omega_1$ and $\omega_2$
are in the same subset;

```
begin

  for each point $\alpha$ in $\Omega$ do $f(\alpha) := \alpha$; end for;
  $f(\omega_2) := \omega_1$;
  repeat

    (*check whether the partition is invariant*)
    no_change_to_f := true;

    for each point $\beta$ in $\Omega$ do
      if $f(\beta) \neq \beta$ then
        $\alpha := f(\beta)$;
        for each generator $s$ of $G$ do
          if $f(\alpha^s) \neq f(\beta^s)$ then
            no_change_to_f := false;
            let $\delta$ be the smaller and let $\gamma$ be the larger of $f(\alpha^s)$ and $f(\beta^s)$;
            for each point $\tau$ in $f^{-1}(\gamma)$ do $f(\tau) := \delta$; end for;
          end if;
        end for;

      end if;
    end for;

  until no_change_to_f;

end;
```

Let us consider the group $G$ of degree 20 that we have been using as an example. Let $\omega_1 = 3$ and $\omega_2 = 18$. The first iteration of the repeat loop in Algorithm 1 considers the pairs (18,3), (19,4), and (20,5) as $(\beta,\alpha)$ and consequently merges {19} with {4}, {20} with {5}, and {16} with {1}. The second iteration considers the pairs (16,1), (17,2), (18,3), (19,4), and (20,5) thus merging {17} with {2}. The third iteration considers the same pairs but has no merges. Therefore the loop terminates, giving the partition {1,16} {2,17} {3,18} {4,19} {5,20} {6} {7} {8} ... {14} {15}.

## Eliminating Redundant Checking

The example of Algorithm 1 clearly demonstrates that each iteration of the **repeat** loop may check the same pair $(\beta,\alpha)$. In fact, the second time we check a pair $(\beta,\alpha)$ we know that each generator $s$ maps $\alpha$ and $\beta$ to the same subset. Since each pair is actually $(\beta, f(\beta))$, the only way for a new pair to be considered during the next iteration is if $f(\beta)$ is redefined. That is, if the subset containing $f(\beta)$ is merged with some other subset.

Suppose that $f(\beta) = \alpha$ is redefined to $\alpha'$. So now $f(\alpha) = \alpha'$. Suppose we check that every generator maps $\alpha$ and $\alpha'$ to the same subset. We have previously checked that every generator maps $\beta$ and $\alpha$ to the same subset. Hence, we can conclude that every generator maps $\beta$ and $\alpha'$, the new $f(\beta)$, to the same subset.

The consequence of the above discussion is that we need only check the pairs $(\alpha, \alpha')$ where $\alpha$ was the representative of a subset at the time the subset was merged with an earlier subset. (That is, $\alpha$ once played the role of $\gamma$ (or $\omega_2$) in Algorithm 1.) Once a point $\beta$ is checked it has ceased to be a subset representative and will never be checked again.

To implement this improvement, we keep a set $C$ of the points that have been subset representatives of merged subsets and have not yet been checked since losing their status as subset representative. Initially, $\omega_2$ is the only point in the set. The improved algorithm is Algorithm 2.

**Algorithm 2 : Minimal partition given a pair of points**

```
Input : a group $G$ generated by $S = \{s_1, s_2, \ldots, s_m\}$ acting on a set $\Omega = \{1,2,\ldots,n\}$;
      two points $\omega_1 < \omega_2$ of $\Omega$;
Output : the finest invariant partition $f$ of $\Omega$ such the the points $\omega_1$ and $\omega_2$
      are in the same subset;
begin
  for each point $\alpha$ in $\Omega$ do $f(\alpha) := \alpha$; end for;
  $f(\omega_2) := \omega_1$;  $C := \{\omega_2\}$;

  repeat
    choose $\beta \in C$;  $C := C - \{\beta\}$;  $\alpha := f(\beta)$;
    for each generator $s$ of $G$ do
      if $f(\alpha^s) \neq f(\beta^s)$ then
        let $\delta$ be the smaller and let $\gamma$ be the larger of $f(\alpha^s)$ and $f(\beta^s)$;
        for each point $\tau$ in $f^{-1}(\gamma)$ do $f(\tau) := \delta$; end for;
        $C := C \cup \{\gamma\}$;
      end if;
    end for;
  until $C$ is empty;
end;
```

Repeating the previous example, we begin with $C = \{18\}$. The first value of $\beta$ is 18 which merges $\{19\}$ with $\{4\}$ so that $C = \{19\}$. The second value of $\beta$ is 19 which merges $\{20\}$ with $\{5\}$ so that $C = \{20\}$. The third value of $\beta$ is 20 which merges $\{16\}$ with $\{1\}$ so that $C = \{16\}$. The fourth value of $\beta$ is 16 which merges $\{17\}$ with $\{2\}$ so that $C = \{17\}$. The last value of $\beta$ is 17 which causes no merging.

A point $\gamma$ is added to $C$ when a merging of subsets occurs and $\gamma$ becomes an ex-representative of a subset, or when $\gamma = \omega_2$. Hence, each point (except $\omega_1$) is added to $C$ at most once. The number of points added to $C$ is at most $|\Omega| - 1$. Therefore, there are at most $|\Omega| - 1$ iterations of the **repeat** loop, and at most $|\Omega| - 2$ occasions when $f(\alpha^s) \neq f(\beta^s)$. The merging of two subsets requires running through $f^{-1}(\gamma)$. One way to do this is to run through all the points and check their function value. This requires $|\Omega|$ operation to find the points $\tau$, and at most $|\Omega| - 1$ operations to redefine all the $f(\tau)$, because the maximum size of a subset being merged is $|\Omega| - 1$. Once we add in the few miscellaneous costs the total worst case cost for Algorithm 2 becomes

$$2 \times |\Omega|^2 + 2 \times |S| \times \left[ |\Omega| - 1 \right] - 3 \times |\Omega| + 4 \text{ operations.}$$

## Testing Primitivity

This section restricts attention to transitive groups. By doing this, we know that any non-trivial invariant partition has a pair $\{1, \omega\}$ of points contained in $B(1)$.

One way to test primitivity is to determine the set of all minimal partitions of $\Omega$. If this set is empty, then the group $G$ is primitive. An invariant partition is *minimal* if it has no refinement that is non-trivial and invariant.

Our assumption of the transitivity of $G$ implies that each $\mathit{partition}(\omega_1, \omega_2)$ is $\mathit{partition}(1, \omega)$, for some $\omega \neq 1$. Consider a fixed $\omega \neq 1$ and the corresponding $\mathit{partition}(1, \omega)$. For each $\omega' \in B(1)$, $\mathit{partition}(1, \omega')$ is a refinement of $\mathit{partition}(1, \omega)$. (They may be equal.) Hence, the set of minimal partitions is contained in

$$\{\ \mathit{partition}(1, \omega) \ \mid\ \begin{array}{l} \omega \neq 1, \\ \mathit{partition}(1, \omega) \neq \Omega, \\ B(1) \text{ does not contain } \omega',\ 1 < \omega' < \omega \ \}. \end{array}$$

So an algorithm to test primitivity may use Algorithm 2 to construct $\mathit{partition}(1, \omega)$, for all $\omega \neq 1$. Those partitions that are complete are disregarded. The subset $B(1)$ may be checked to verify the last condition.

The last condition can also be checked within (a modified) Algorithm 2. There exists such an $\omega'$ if and only if Algorithm 2 merges $B(\delta)$ and $B(\gamma)$ where $\delta = 1 = \omega_1$ and $\gamma < \omega_2 = \omega$.

If we are only interested in a yes/no answer to the primitivity of the group then there is no need to construct the set of minimal partitions. The answer is no as soon as a $\mathit{partition}(1, \omega) \neq \Omega$ is constructed.

## Summary

This chapter has presented an algorithm for finding partitions of the points $\Omega$ that are invariant under the action of the group. The generators of the group provided enough information for this algorithm, which finds the finest invariant partition having two given points in the same subset. The algorithm presented is $O(|\Omega|^2)$. The primitivity of a group can be determined using this algorithm in time $O(|\Omega|^3)$.

## Exercises

(1/Easy) Determine whether the group of degree 8 generated by $a$ = (1,2,3,4)(5,6,7,8) and $b$ = (1,5)(2,8)(3,7)(4,6) is primitive or imprimitive.

(2/Easy) Determine whether the group of degree 4 generated by $a$ = (1,2,3,4) and $b$ = (1,4)(2,3) is primitive or imprimitive.

(3/Easy) Write out in detail the algorithm for testing primitivity. Assume the output is only a yes/no answer.

## Bibliographical Remarks

This chapter has presented the work of M. D. Atkinson, "*An algorithm for finding the blocks of a permutation group*", Mathematics of Computation, **29**, 131 (1975) 911-913. The paper also considers representing the set $C$ of Algorithm 2 as a circular list. However, the order of the worst case cost is not improved.

The problem is similar to the union-find problem. If one represents each subset of the partition as a list and merges the smaller subset with the larger subset then Algorithm 2 is $O(|\Omega| \times log(|\Omega|))$, and if one represents the subsets as trees and performs path compression then Algorithm 2 is essentially linear in $|\Omega|$ and $|S|$. However, in practice, a simple list is adequate. See G. Butler, "*An analysis of Atkinson's algorithm*", Basser Department of Computer Science Technical Report **259**, May 1985, for these analyses, and see M.D. Atkinson, R.A. Hassan, and M.P. Thorne, "*Group theory on a micro-computer*", in **Computational Group Theory**, M.D. Atkinson (ed.), Academic Press, New York, 1984, for an experimental comparison of these representations.