```
(* Function: subset a b *)

let rec subset a b = match a with
| [] -> true
| h::t ->
(match List.mem h b with
| true -> subset t b
| false -> false);;

(* Function: equal_sets a b *)

let equal_sets a b = match (subset a b, subset b a) with
| (true, true) -> true
| (_, _) -> false;;

(* Function: set_union a b *)

let set_union a b = List.sort_uniq Pervasives.compare (List.append a b);;

(* Function: set_intersection a b *)

let rec intersect a b c = match a with
| [] -> c
| h::t ->
(match List.mem h b with
| true -> intersect t b (List.cons h c)
| false -> intersect t b c);;

let set_intersection a b = List.sort_uniq Pervasives.compare (intersect a b []);;

(* Function: set_diff a b *)

let rec unintersect a b c = match a with
| [] -> c
| h::t ->
(match List.mem h b with
| true -> unintersect t b c
| false -> unintersect t b (List.cons h c));;

let set_diff a b = List.sort_uniq Pervasives.compare (unintersect a b []);;

(* Function: computed_fixed_point eq f x *)

let rec computed_fixed_point eq f x =
if eq x (f x) then x
else computed_fixed_point eq f (f x);;

(* Function: filter_reachable g *)
```

```
type ('nonterminal, 'terminal) symbol =
  | N of 'nonterminal
  | T of 'terminal;;

let rec get_reachable_set sym fixed_rules rules reachable_set =
let rec process_rhs f_r aux_rhs aux_rset = match aux_rhs with
| [] -> aux_rset
| h::t ->
        (match h with
        | N element ->
                (match List.mem element aux_rset with
                | true -> process_rhs f_r t aux_rset
                | false -> process_rhs f_r t (get_reachable_set element f_r f_r (List.cons element aux_rset)))
        | T element -> process_rhs f_r t aux_rset)
in
match rules with
| [] -> reachable_set
| (lhs, rhs)::t ->
        if lhs = sym then
        get_reachable_set sym fixed_rules t (process_rhs fixed_rules rhs reachable_set)
        else
        get_reachable_set sym fixed_rules t reachable_set;;

let filtering rules reachable_set =
List.filter (fun rule -> List.mem (Pervasives.fst rule) reachable_set = true) rules;;

let filter_reachable g = match g with
| (start_symbol, rules) -> (start_symbol, filtering rules (get_reachable_set start_symbol rules rules [start_symbol]));;
```