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

let convert_grammar gram1 = match gram1 with
(start, rules)-> let rec production nt prules = (match prules with
                            |(st1, rule)::t->if st1 = nt
                                             then rule::(production nt t)
                                             else production nt t
                           |[]->[]
) in
(start, (fun nt -> production nt rules))


let rec match_rule productionf rule acceptor derv fram =match rule with
|[]->acceptor derv fram
|hd::tl->match fram with
            |[]->None
            |hsymbol::tsymbol->match rule with
                               |(T tterm)::tail->if tterm = hsymbol
                                                 then match_rule productionf tail acceptor derv tsymbol
                                                 else None
                              |(N nterm)::tail->(match_table productionf (productionf nterm) nterm (match_rule productionf tail acceptor) derv fram)

and
match_table productionf rules nonterm acceptor derv fram = match rules with
|[]->None
|hd::tl->match (match_rule productionf hd acceptor (derv@[nonterm, hd]) fram) with
         |None->match_table productionf tl nonterm acceptor derv fram
         |x-> x



let parse_prefix gram acceptor frag = match gram with
|(nonterm, productionf)->match_table productionf (productionf nonterm) nonterm acceptor [] frag
```