Grammar

var	::=	[a-z]+
var_seq	::=	var var_seq | var
factor	::=	var | (term)
abstraction	::=	`"\"` var_seq `"."` term
factor_seq	::=	factor factor_seq | factor
application	::=	factor factor_seq
term	::=	abstraction | application | factor
bind	::=	`"let"` var `"="` term
program	::=	bind program | term

Example

x y z
program -> term -> application -> factor factor_seq -> var factor_seq ->
x factor_seq -> x factor factor_seq -> x var factor_seq ->
x y factor_seq -> x y factor -> x y var -> x y z

let T = \x y.x
T a b
program -> bind program
-> `"let"` var `"="` term program
-> `"let"` T `"="` term program
-> `"let"` T `"="` abstraction program
-> `"let"` T `"="` \var_seq.term program
-> `"let"` T `"="` \var var_seq.term program
-> `"let"` T `"="` \x var_seq.term program
-> `"let"` T `"="` \x var.term program
-> `"let"` T `"="` \x y.term program
-> `"let"` T `"="` \x y.factor program
-> `"let"` T `"="` \x y.var program
-> `"let"` T `"="` \x y.x program
-> `"let"` T `"="` \x y.x term
-> `"let"` T `"="` \x y.x application
-> `"let"` T `"="` \x y.x factor factor_seq
-> `"let"` T `"="` \x y.x var factor_seq
-> `"let"` T `"="` \x y.x T factor_seq
-> `"let"` T `"="` \x y.x T factor factor_seq
-> `"let"` T `"="` \x y.x T var factor_seq
-> `"let"` T `"="` \x y.x T a factor_seq
-> `"let"` T `"="` \x y.x T a factor
-> `"let"` T `"="` \x y.x T a var
-> `"let"` T `"="` \x y.x T a b