QualN	qn	::=	$l.n$	qualified name
QualC	qc	::=	$l.c$	qualified constructor
Tm	M	::=	x	variable
			qn	declared name
			(let M x M)	local declaration
			(lam x M)	λ abstraction
			[M M]	function application
			(con qc M^*)	constructed data
			(case M C^*)	case
			(success M)	success
			(failure)	failure
			(txhash)	transaction hash
			(blocknum)	block number
			(blocktime)	block time
			(bind M x M)	computation bind
			i	primitive integer
			f	primitive float
			b	primitive bytestring
			(builtin n M^*)	built-in function
Cl	C	::=	(cl qc (x^*) M)	case clause
Prg	G	::=	(program L^*)	program
Mod	L	::=	(module l id ed ld D^*)	module
ImpD	id	::=	(imported l^*)	import decls
ExpD	ed	::=	(exported ((c a)*) (n^*))	export decls
LocD	ld	::=	(local ((c a)*) (n^*))	local decls
Def	D	::=	(define n V)	name definition
Val	V	::=	(lam x M)	λ abstraction
			(con qc V^*)	constructed data
			(success V)	success
			(failure)	failure
			(txhash)	transaction hash
			(blocknum)	block number
			(blocktime)	block time
			(bind V x M)	computation bind
			i	primitive integer
			f	primitive float
			b	primitive bytestring

Fig. 1. Grammar of Plutus Core