=	attempt an equality (or equivalence) substitution	PP	prettyprint the current term
ACL2-WRAP	same as (lisp x)	PR	print the rules for a given name
ADD-ABBREVIATION	add an abbreviation	PRINT	print the result of evaluating the given form
AL	same as apply-linear	PRINT-ALL-CONCS	print all the conclusions of (as yet unproved) goals
APPLY-LINEAR	apply a linear rule	PRINT-ALL-GOALS	print all the (as yet unproved) goals
BASH	call the ACL2 theorem prover's simplifier	PRINT-MAIN	print the original goal
BDD	prove the current goal using bdds	PRO	repeatedly apply promote
BK	move backward one argument in the enclosing term	PROMOTE	move antecedents of conclusion's implies term to top-level hyps
BOOKMARK	insert matching ``bookends'' comments	PROTECT	run the given instructions, reverting to existing state upon failure
CASESPLIT	split into two cases	PROVE	call the ACL2 theorem prover to prove the current goal
CG	change to another goal.	PSO	print the most recent proof attempt from inside the proof-checker
CHANGE-GOAL	change to another goal.	PSO!	print the most recent proof attempt from inside the proof-checker
CL-PROC	same as clause-processor	PSOG	print the most recent proof attempt from inside the proof-checker
CLAIM	add a new hypothesis	PUT	substitute for a ``free variable''
CLAUSE-PROCESSOR	use a clause-processor	QUIET	run instructions without output
COMM	display instructions from the current interactive session	R	same as rewrite
COMMANDS	display instructions from the current interactive session	REDUCE	call the ACL2 theorem prover's simplifier
COMMENT	insert a comment	REDUCE-BY-INDUCTION	call the ACL2 prover without induction, after going into induction
CONTRADICT	same as contrapose	REMOVE-ABBREVIATIONS	remove one or more abbreviations
CONTRAPOSE	switch a hypothesis with the conclusion, negating both	REPEAT	repeat the given instruction until it ``fails''
DEMOTE	move top-level hypotheses to the conclusion	REPEAT-REC	auxiliary to repeat
DIVE	move to the indicated subterm	REPLAY	replay one or more instructions
DO-ALL	run the given instructions	RESTORE	remove the effect of an UNDO command
DO-ALL-NO-PROMPT	run the given instructions, halting once there is a ``failure''	RETAIN	drop all but the indicated top-level hypotheses
DO-STRICT	run the given instructions, halting once there is a ``failure''	RETRIEVE	re-enter the proof-checker
DROP	drop top-level hypotheses	REWRITE	apply a rewrite rule
DV	move to the indicated subterm	RUN-INSTR-ON-GOAL	auxiliary to THEN
ELIM	call the ACL2 theorem prover's elimination process	RUN-INSTR-ON-NEW-GOALS	auxiliary to then
EQUIV	attempt an equality (or congruence-based) substitution	RUNES	print the runes (definitions, lemmas, ...) used
EX	exit after possibly saving the state	S	simplify the current subterm
EXIT	exit the interactive proof-checker	S-PROP	simplify propositionally
EXPAND	expand the current function call without simplification	SAVE	save the proof-checker state (state-stack)
FAIL	cause a failure	SEQUENCE	run the given list of instructions according to a multitude of options
FINISH	require completion of instructions; save error if inside :hints	SHOW-ABBREVIATIONS	display the current abbreviations
FORWARDCHAIN	forward chain from an implication in the hyps	SHOW-LINEARS	display the applicable linear rules
FREE	create a ``free variable''	SHOW-REWRITES	display the applicable rewrite rules
GENEQV	show the generated equivalence relation maintained at the current subterm	SHOW-TYPE-PRESCRIPTIONS	display the applicable type-prescription rules
GENERALIZE	perform a generalization	SKIP	``succeed'' without doing anything
GOALS	list the names of goals on the stack	SL	simplify with lemmas
HELP	proof-checker help facility	SLS	same as SHOW-LINEARS
HELP!	proof-checker help facility	SPLIT	split the current goal into cases
HELP-LONG	same as help!	SR	same as SHOW-REWRITES
HYPS	print the hypotheses	ST	same as SHOW-TYPE-PRESCRIPTIONS
ILLEGAL	illegal instruction	SUCCEED	run the given instructions, and ``succeed''
IN-THEORY	set the current proof-checker theory	TH	print the top-level hypotheses and the current subterm
INDUCT	generate subgoals using induction	THEN	apply one instruction to current goal and another to new subgoals
LEMMAS-USED	print the runes (definitions, lemmas, ...) used	TOP	move to the top of the goal
LISP	evaluate the given form in Lisp	TYPE-ALIST	display the type-alist from the current context
MORE	proof-checker help facility	UNDO	undo some instructions
MORE!	proof-checker help facility	UNSAVE	remove a proof-checker state
NEGATE	run the given instructions, and ``succeed'' if and only if they ``fail''	UP	move to the parent (or some ancestor) of the current subterm
NIL	used for interpreting control-d	USE	use a lemma instance
NOISE	run instructions with output	WRAP	execute the indicated instructions and combine all the new goals
NX	move forward one argument in the enclosing term	WRAP-INDUCT	same as induct, but create a single goal
ORELSE	run the first instruction; if (and only if) it ``fails'', run the second	WRAP1	combine goals into a single goal
P	prettyprint the current term	X	expand and (maybe) simplify function call at the current subterm
P-TOP	prettyprint the conclusion, highlighting the current term	X-DUMB	expand function call at the current subterm, without simplifying
PL	print the rules for a given name		