PVS & Emacs interaction

- to typecheck a file: `CTRL-c CTRL-t`
  or PVS → Parsing and Typechecking → Typecheck
- to prove a lemma: `CTRL-c CTRL-p x`
  or PVS → Prover Invocation → x prove
- `CTRL-c CTRL-c` interrupts PVS; continue with the command `restore)`
- `ALT-P` cycles through previous commands
- `CTRL-]` kills request for input on bottom line
- `CTRL-x 1` unsplits window, ie. kill all windows except the one where cursor is
- `CTRL-x 2` splits window in two
- `*pvs*` in the "Buffers" menu is the PVS prover

Most common tactics and their abbreviation

fnum: a number of antecedent of consequent
expr: a arbitrary expression
name: a name of a definition or lemma
[ ] denotes an optional parameter

control

`(undo)`	`TAB u`
`(undo undo)` ie. redo	
`(postpone)`	`TAB P`
`(help` *command*`)` eg `(help replace)`	
`(quit)`	

propositional logic

`(flatten` *[fnum]*`)`	`TAB f`
`(split` *[fnum]*`)`	`TAB s`
`(case "`*expr*`")` eg `(case "n!1 > 0")`	`TAB c`
`(lift-if` *[fnum]*`)`	`TAB l`
`(prop)`	`TAB p`

General tip: always try to `(flatten)`.

predicate logic

(skolem *fnum* "*name*" ..."*name*")	
Eg (skolem -1 "x")	
(skolem! *[fnum]*)	TAB !
(skosimp *[fnum]*)	TAB S
(skosimp*)	TAB *
(inst fnum "*expr*" ..."*expr*")	
Eg (inst -1 "2+j!1")	
(inst? *[fnum]*)	TAB ?

General tip: (nearly) always try to (skosimp*), unless you're faced with a universal quantification that you want to prove by induction.

equality

(expand "*name*" *[fnum]* *[occurrence]*)	TAB e
TAB e with cursor on formula to expand	
(replace *fnum* *[fnum]* RL)	TAB r
(replace *fnum* *[fnum]* LR)	TAB r
(rewrite)	TAB R
(assert)	TAB a

induction

(induct "*n*")	TAB I
(induct-and-simplify "*n*")	TAB CTRL-s

Goal should be of the form FORALL(..., n:nat, ...) ...

automated proving

(prop)	TAB p
(assert)	TAB a
(grind)	TAB G

misc

(lemma "*name*")	TAB l
(apply-extensionality *[fnum]*)	
(decompose-equality *[fnum]*)	TAB =