Aide-mémoire Lean

nom: nom déjà connu de Lean — *nv_nom*: nouveau nom fourni par l'utilisateur
expr: expression — *args*: argument ou liste d'arguments
(…) : partie optionnelle

Symbole logique	**Démonstration**	**Utilisation**
∀ (pour tout)	`Soit` *nv_nom*	`On applique` *expr* (`à` *args*)
∃ (il existe)	`Montrons que` *expr* `convient`	`Par` *expr* (`appliqué à` *args*) `on obtient` *nv_nom* `tel que` *nv_nom'*
→ (implique)	`Supposons` *nv_nom*	`Par` *expr* (`appliqué à` *args*) `il suffit de montrer que` *expr'*
↔ (équivaut)	`Montrons que` *impl_directe*	`On réécrit via (←)` *expr*
∧ (et)	`Montrons que` *prop_gauche*	`Par` *expr* (`appliqué à` *args*) `on obtient` *nv_nom nv_nom'*
∨ (ou)	`Montrons que` *prop_gauche*/*prop_droite*	`On discute en utilisant` *expr*
¬ (non)	`Supposons` *nv_nom*	`Par` *expr* (`appliqué à` *args*) `il suffit de montrer que` *expr*'

Commande	**Effet**
`On conclut par` *expr* (`appliqué à` *args*)	affirme qu'on peut conclure le but courant par *expr* (appliqué à *args*)
`Fait` *nv_nom*: *énoncé*	introduit un énoncé *nv_nom* affirmant *énoncé* à démontrer
`On déplie` *nom* (`dans` *hyp*)	déplie la définition de *nom* dans le but (ou dans l'hypothèse *hyp*)
`On reformule` *hyp* `en` *expr*	transforme l'hypothèse *hyp* en l'expression *expr* qui lui est équivalente par définition
`On réécrit via` (←) *expr* (`dans` *hyp*)	dans le but (ou dans l'hypothèse *hyp*), remplace le membre de gauche (ou de droite si ← est présent) de l'égalité ou équivalence *expr* par l'autre membre.
`On calcule` (`dans` *hyp*)	déduit le but (ou simplifie l'hypothèse *hyp*) en utilisant les propriétés de l'addition et la multiplication
`On combine` [*nom*, …, *nom*]	déduit le but par combinaison linéaire des hypothèses listées
`Montrons une contradiction`	applique la règle *ex falso quod libet*
`Supposons par l'absurde` *nv_nom*	initie une démonstration par l'absurde, en appelant *nv_nom* l'hypothèse de négation du but
`On discute selon que` *expr*	scinde la démonstration en deux cas selon que *expr* est vraie ou fausse, en appelant *nv_nom* cette hypothèse
`On contrapose`	transforme l'implication but en sa contraposée
`On pousse la négation` (`dans` *hyp*)	pousse les négations dans le but (ou dans l'hypothèse *hyp*)
`Montrons par récurrence` *nv_nom* `:` *expr*	Démarre une démonstration par récurrence
`Par` *hyp* (`appliqué à` *args*) `on choisit` *nv_nom* `tel que` *nv_nom'*	utilise l'axiome du choix pour produire à partir de *hyp*: $\forall x, \exists y, P(x,y)$ une fonction $x \mapsto y(x)$ vérifiant $\forall x, P(x, y(x))$
`aide` (*nom*)	demande de l'aide basée sur la structure du but ou de l'hypothèse dont le nom est donné
`parentheses`	force l'affichage de parenthèses autour des opérations