Aide-mémoire Lean

Dans le tableau suivant, *nom* désigne toujours un nom déjà connu de Lean tandis que *nv_nom* désigne un nouveau nom fourni par l'utilisateur. Lorsque qu'un de ces mots apparait deux fois dans la même ligne, les apparitions ne désignent pas le même nom. *expr* désigne une expression, par exemple le nom d'un objet du contexte, une expression arithmétique fonction d'objets du contexte, ou une hypothèse du contexte ou un lemme appliqué à un objet du context.

Symbole logique	Démonstration	Utilisation
$\forall$ (pour tout)	`intro` *nv_nom*	`apply` *expr* ou `specialize` *nom expr*
$\exists$ (il existe)	`use` *expr*	`cases` *expr* `with` *nv_nom nv_nom*
$\to$ (implique)	`intro` *nv_nom*	`apply` *expr* ou `specialize` *nom expr*
$\leftrightarrow$ (équivaut)	`split`	`rw` *expr* ou `rw` ← *expr*
$\wedge$ (et)	`split`	`cases` *expr* `with` *nv_nom nv_nom*
$\vee$ (ou)	`left` ou `right`	`cases` *expr* `with` *nv_nom nv_nom*
$\neg$ (non)	`intro` *nv_nom*	`apply` *expr* ou `specialize` *nom expr*

Remarque : la pratique traditionnelle sur papier utilise $\Rightarrow$ pour l'implication, utilise $\Leftrightarrow$ pour l'équivalence, et n'utilise pas de notation pour « et », « ou » et « non ».

Dans la colonne de gauche du tableau suivant, les parties entre parenthèses sont optionnelles. L'effet de ces parties est aussi entre parenthèses dans la colonne de droite. Il s'agit presque toujours de préciser qu'une manipulation, qui agit par défaut sur le but, doit être faite plutôt sur une certaine hypothèse nommée *hyp*.

Tactique	Effet
`exact` *expr*	affirme que le but est exactement *expr*
`have` *nv_nom* : *fait*	introduit un énoncé *nv_nom* affirmant *fait* à démontrer
`unfold` *nom* (`at` *hyp*)	déplie la définition de *nom* dans le but (ou dans l'hypothèse *hyp*)
`change` *expr* (`at` *hyp*)	transforme le but (ou l'hypothèse *hyp*) en l'expression *expr* qui lui est équivalente par définition
`rw` (←) *expr* (`at` *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. L'expression à remplacer doit apparaître explicitement, il faut parfois utiliser `unfold` ou `change` pour l'assurer.
`compute` (`at` *hyp*)	déduit le but (ou simplifie l'hypothèse *hyp*) en utilisant les propriétés de l'addition et la multiplication
`linarith`	déduit le but par combinaison linéaire des hypothèses
`choose` *nv_nom nv_nom* `using` *expr*	utilise l'axiome du choix pour produire à partir de *expr* : $\forall x, \exists y, P(x,y)$ une fonction $x \mapsto y(x)$ vérifiant $\forall x, P(x, y(x))$
`exfalso`	applique la règle *ex falso quod libet*
`by_contradiction` *nv_nom*	initie une démonstration par l'absurde, en appelant *nv_nom* l'hypothèse de négation du but
`by_cases` *nv_nom* : *expr*	scinde la démonstration en deux cas selon que *expr* est vraie ou fausse, en appelant *nv_nom* cette hypothèse
`contrapose`	transforme l'implication but en sa contraposée
`push_neg` (`at` *hyp*)	pousse les négations dans le but (ou dans l'hypothèse *hyp*)