Cheatsheet Rocq

1 Dictionnaire : λ-calcul simplement typé $\leftrightarrow$ Rocq

Concept Théorique	**Notation Papier (Maths/Logique)**	**Syntaxe Rocq (Gallina)**
Déclaration de type	$x : A$	`x : A`
Implication / Type flèche	$A \to B$	`A -> B`
Abstraction (λ)	$\lambda x.M$	`fun x => M`
Abstraction multiple	$\lambda x.\lambda y.M$	`fun x y => M`
Application	$M\ N$	`M N`
Parenthésage	$(f\ (g\ x))$	`f (g x)`
Conjonction (ET)	$A \wedge B$	`A /\ B`
Constructeur de $\wedge$	(a, b)	`conj a b`
Disjonction (OU)	$A \vee B$	`A \/ B`
Constructeurs de $\vee$	$\iota_1\, a,\ \iota_2\, b$	`or_introl a`, `or_intror b`
Négation (NON)	$\neg A$	`~A` ou `not A`
Absurdité	$\bot$	`False`
Vérité	$\top$	`True`
Constructeur de $\top$	$\star$	`I`
Quantificateur Universel	$\forall x.M$	`forall x, M`
Quantificateur Existentiel	$\exists x.M$	`exists x, M`
Constructeur de $\exists$	(t, p) avec $p : P\, t$	`ex_intro P t p`
Égalité Propositionnelle	$x = y$	`x = y`
Constructeur de $=$	refl_x	`eq_refl x`

2 Tactiques et Déduction Naturelle

Dans le mode **Preuve Interactive**, les tactiques construisent le λ-terme à l'envers (en remontant de la conclusion vers les prémisses de l'arbre de dérivation en déduction naturelle). Dans la suite, on utilise la notation $?M$ pour indiquer une *variable existentielle*, i.e. un terme qu'il reste à fournir par l'utilisateur, correspondant à une prémisse ouverte de la dérivation en cours de construction.

— Tactiques « Backward » (Raisonnement en arrière) —

Tactique Rocq	**Effet sur le Séquent (Contexte $\vdash$ But)**	**Règle de Déduction Naturelle**

`intros x.`	Prend l'antécédant A du but ($A \to B$) et l'introduit comme nouvelle hypothèse dans le contexte sous le nom `x`. Le but devient B.	**$\to$-Introduction** (Abstraction en λ-calcul) $\dfrac{\Gamma, x : A \vdash ?M : B}{\Gamma \vdash \lambda x.?M : A \to B}\,\text{Abs}$
`exact x.`	Résout le but courant si l'hypothèse `x` correspond *exactement* au but.	**Règle "axiome"** (Typage d'une variable) $\dfrac{}{\Gamma, x : A \vdash x : A}\,\text{Var}$
`assumption.`	Résout le but courant s'il correspond *exactement* à l'une des hypothèses (recherche automatique au lieu de nommer).	**Variante "axiome"** (Idem `exact`) $\dfrac{}{\Gamma, H : A \vdash H : A}\,\text{Var}$
`apply f.`	Si le but est B et qu'on a $f : A \to B$, change le but en A (raisonnement arrière).	**$\to$-Élimination** (Modus Ponens / Application) $\dfrac{\dfrac{}{\Gamma, f : A \to B \vdash f : A \to B}\,\text{Var} \quad \Gamma, f : A \to B \vdash ?M : A}{\Gamma, f : A \to B \vdash f\ ?M : B}\,\text{App}$
`split.`	Pour un but $A \wedge B$, crée deux sous-buts A et B.	**$\wedge$-Introduction** (Formation de paire) $\dfrac{\Gamma \vdash ?M : A \quad \Gamma \vdash ?N : B}{\Gamma \vdash (?M, ?N) : A \wedge B}\,\wedge I$
`left.` / `right.`	Pour un but $A \vee B$, choisit de prouver A (gauche) ou B (droite).	**$\vee$-Introductions** (Injections ι_1, ι_2) $\dfrac{\Gamma \vdash ?M : A}{\Gamma \vdash \iota_1 ?M : A \vee B}\vee I_1 \quad \dfrac{\Gamma \vdash ?M : B}{\Gamma \vdash \iota_2 ?M : A \vee B}\vee I_2$
`exfalso.`	Change n'importe quel but en $\bot$ (`False`).	**Principe d'Explosion** (*Ex Falso Quodlibet*) $\dfrac{\Gamma \vdash ?M : \bot}{\Gamma \vdash \texttt{abort}\ ?M : A}\,\bot E$
`admit.`	Accepte le but courant sans preuve. *N.B. : La validation finale d'un lemme contenant* `admit` *requiert* `Admitted` *au lieu de* `Qed`*.*	—

`exists t.`	Pour un but $\exists x, P(x)$, fournit le témoin t. Le but devient alors $P(t)$.	**∃-Introduction** (Paire d'un témoin et d'une preuve) $\dfrac{\Gamma \vdash ?M : P[t/x]}{\Gamma \vdash (t, ?M) : \exists x.P(x)}\exists I$

`rewrite H.`
ou
`rewrite <- H.`
	Remplace le membre gauche de l'égalité `H` par le membre droit dans le but (ou l'inverse avec `<-`).	**Substitution de Leibniz**
		(Principe d'identité)
		$\dfrac{\dfrac{}{\Gamma, x = y \vdash x = y}\text{Var} \quad \Gamma, x = y \vdash P(x)}{\Gamma, x = y \vdash P(y)}\text{Subst}$

`reflexivity.`	Résout un but de la forme $t = u$ lorsque t est *définitionnellement égal* à u (i.e. $t =_{\beta\eta} u$ modulo *unfolding* de définitions faites avec la commande `Definition`).	**Réflexivité** $\dfrac{}{\Gamma \vdash t = t}\text{Refl}$
— *Tactiques « Forward » (Raisonnement en avant)* —		
`destruct H as [x y].`	Pour $H : A \wedge B$, extrait les preuves $x : A$ et $y : B$.	**∧-Éliminations** (Projections π_1, π_2) $\dfrac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash \pi_1 M : A}\wedge E_1 \quad \dfrac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash \pi_2 M : B}\wedge E_2$
`destruct H as [x | y].`	Pour $H : A \vee B$, génère deux univers : l'un avec $x : A$, l'autre avec $y : B$.	**∨-Élimination** (Raisonnement par cas : `match ... with`) $\dfrac{\Gamma \vdash M : A \vee B \quad \Gamma, x : A \vdash N : C \quad \Gamma, y : B \vdash P : C}{\Gamma \vdash \text{match } M \text{ with } \ldots : C}\vee E$
`destruct H.`	Variante des précédents : Rocq choisit les noms d'hypothèses automatiquement.	—
`apply f in H.`	Si on a $H : A$ et $f : A \to B$, modifie l'hypothèse H pour qu'elle devienne de type B.	**→-Élimination** (Modus Ponens) —
`pose proof N as x.`	Prend un terme N comme justification pour ajouter une nouvelle hypothèse $x : A$, étant donné que $\Gamma \vdash N : A$ avec Γ le contexte du but courant.	**Règle de Coupure (Cut)** $\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash N : A\end{array} \quad \Gamma, x : A \vdash ?M : B}{\Gamma \vdash \texttt{let } x := N \texttt{ in } ?M : B}\text{Cut}$

`assert (A) as x. { ... }`	Ouvre un sous-but temporaire pour prouver d'abord A. Une fois prouvé, on obtient une nouvelle hypothèse $x : A$.	**Règle de Coupure (Cut)** (Lemme local / Lemmatisation) $\dfrac{\Gamma \vdash ?N : A \quad \Gamma, x : A \vdash ?M : B}{\Gamma \vdash \texttt{let}\ x := ?N\ \texttt{in}\ ?M : B}\ \text{Cut}$

3 Termes de preuve et commandes d'inspection

Syntaxe Rocq	Effet / Description
`match H with | conj a b => ... end`	Éliminateur à la main de H : $A \wedge B$: extrait les composantes $a : A$ et $b : B$. C'est le terme que `destruct H as [a b]` génère automatiquement.
`match H with| or_introl a => ...| or_intror b => ...end`	Éliminateur à la main de $H : A \vee B$: analyse par cas. C'est le terme que `destruct H as [a | b]` génère automatiquement.
`eq_ind`	L'éliminateur de `eq` dérivé automatiquement par Rocq et invoqué par la tactique `rewrite`. Son type formalise le **principe de Leibniz** : toute propriété de x est propriété de y dès que $x = y$.
`Compute e.`	Évalue l'expression `e` jusqu'à sa forme normale (β-réduit, déplie les définitions). Illustre l'élimination des détours intro/élim (cut elimination).
`Print nom.`	Affiche la définition complète de `nom` (type inductif, lemme...). Utile pour voir le terme de preuve généré par des tactiques.
`Check terme.`	Affiche le type de `terme` sans l'évaluer.

4 Logique Classique vs Intuitionniste

Par défaut, Rocq utilise une logique **intuitionniste** (constructiviste) : on ne peut prouver $A \vee \neg A$ que si l'on possède un algorithme pour décider lequel est vrai.

— **Axiome LEM** : On peut introduire la logique classique en rajoutant l'axiome :

```
Axiom LEM : forall X, X \/ not X.
```

— **Épistémologie** : Le Tiers-Exclu est rejeté en informatique pure car il est lié à l'**indécidabilité**

de Gödel et Turing [1] (on ne peut pas décider la vérité de n'importe quelle proposition de manière générique).

— **Double Négation** : En logique classique, $\neg\neg A \to A$ est un théorème. En logique intuitionniste, on n'a que $A \to \neg\neg A$.

1. https://fr.wikipedia.org/wiki/D%C3%A9cidabilit%C3%A9