Licence 3 informatique

Rapport de TER

Smullyan et le Coq

Pierre Portal
Alexandre Ribeyre
Aurélien Ambroise
Matthieu Poirot

Tuteur : David Delahaye

Année 2019

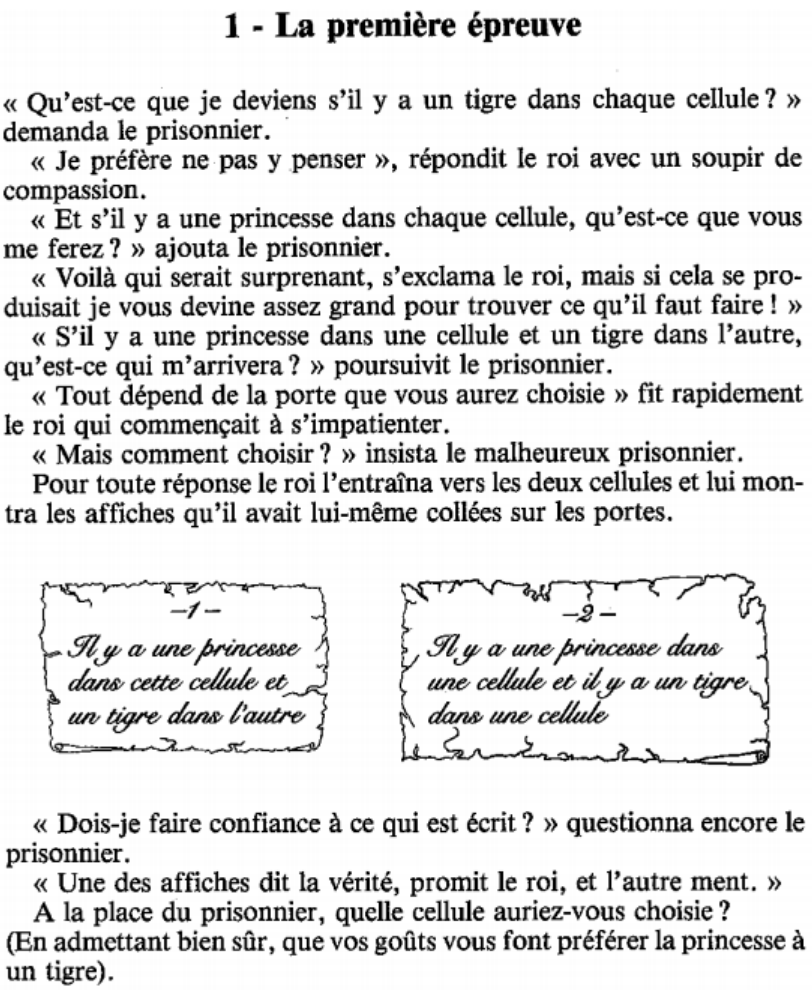
Table des matières

1 Remerciements

Nous tenons à remercier notre tuteur, Pr. David Delahaye,
pour le soutien qu'il nous a apporté tout au long de l'élaboration de ce projet.

2 Introduction

Dans le cadre de notre TER en 3ème année de Licence informatique, nous avons eu pour projet la réalisation de preuves logiques appliquées sur les 12 épreuves tirées du chapitre “Une princesse ou un tigre ?” du livre d’énigmes “Le livre qui rend fou !” écrit par Raymond Smullyan. Le principe général de chaque épreuve est le suivant : un prisonnier est confronté à plusieurs portes (cellules) sur lesquelles sont posées des inscriptions. Son but est de trouver la princesse qui est cachée derrière l’une de ces portes et, s’il se trompe, il risque de tomber sur un tigre. La véracité du contenu des inscriptions est régie par des règles énoncées préalablement par le roi.



1 - La première épreuve

« Qu’est-ce que je deviens s’il y a un tigre dans chaque cellule ? » demanda le prisonnier.

« Je préfère ne pas y penser », répondit le roi avec un soupir de compassion.

« Et s’il y a une princesse dans chaque cellule, qu’est-ce que vous me ferez ? » ajouta le prisonnier.

« Voilà qui serait surprenant, s’exclama le roi, mais si cela se produisait je vous devine assez grand pour trouver ce qu’il faut faire ! »

« S’il y a une princesse dans une cellule et un tigre dans l’autre, qu’est-ce qui m’arrivera ? » poursuivit le prisonnier.

« Tout dépend de la porte que vous aurez choisie » fit rapidement le roi qui commençait à s’impatienter.

« Mais comment choisir ? » insista le malheureux prisonnier.

Pour toute réponse le roi l’entraîna vers les deux cellules et lui montra les affiches qu’il avait lui-même collées sur les portes.

« Dois-je faire confiance à ce qui est écrit ? » questionna encore le prisonnier.

« Une des affiches dit la vérité, promit le roi, et l’autre ment. »

A la place du prisonnier, quelle cellule auriez-vous choisie ?

(En admettant bien sûr, que vos goûts vous font préférer la princesse à un tigre).

Figure 1 – L’énoncé de la première épreuve

Notre travail a d’abord été la résolution manuelle de ces épreuves comme le ferait toute personne qui lirait le livre et qui souhaiterait connaître la solution. Une fois cette étape accomplie, il a été question de modéliser l’ensemble de ces épreuves en expressions propositionnelles afin des les prouver. Puis, il a fallu résoudre ces preuves en utilisant Coq qui est un langage de résolution de preuve. Dans un premier temps, nous présenterons brièvement les domaines de l’informatique centraux pour notre projet : la logique des propositions et le langage Coq. Puis nous détaillerons les trois grandes parties de notre travail :

- la résolution manuelle des épreuves
- la modélisation des épreuves en logique propositionnelle
- la résolution des épreuves en Coq

3 Domaines de l'informatique

3.1 Logique propositionnelle

3.1.1 Structure de l'expression propositionnelle

Les épreuves sont des énigmes logiques et qui n'ont qu'une seule interprétation. Les règles qui s'appliquent sur les inscriptions sont clairement énoncées par le roi et il n'existe aucune ambiguïté dans le récit. De ce fait, il est possible de modéliser les énoncés de chaque épreuve en une expression de la logique propositionnelle et que l'on pourra résoudre. L'idée est de modéliser une implication de la forme :

$$x_1 \wedge x_2 \wedge \ldots \wedge x_i \Rightarrow S$$

Où $x_1, x_2, \ldots, x_i$ sont toutes les règles qui régissent l'épreuve (qu'elles soient explicites ou implicites) et S étant la solution de l'énigme. Il suffit ensuite d'effectuer la preuve de cette expression.

3.1.2 Terminologie

Il est nécessaire de définir une terminologie commune à toutes les épreuves afin de nommer les variables propositionnelles et donner de la clarté aux expressions que nous allons construire.

Nom	Type	Signification
pX	Variable propositionnelle	Une princesse se trouve dans la cellule X
tX	Variable propositionnelle	Un tigre se trouve dans la cellule X
affX	Formule propositionnelle	Inscription de la cellule X
hX	Formule propositionnelle	Règle axiomatique
k	Formule propositionnelle	Règle du roi

De nouvelles variables et formules s'ajouteront dans le cadre d'épreuves spécifiques et nous les expliquerons en temps voulu.

3.2 Le langage Coq

Du fait de la longueur des expressions que nous allons modéliser et pour éviter une trop grande pénibilité, nous allons utiliser le langage de preuve Coq qui va permettre de nous assister tout au long de l'élaboration de la preuve. Coq fournit un certain nombre de commandes appelées stratégies (destruct, split, intros, ...) qui permettent d'avancer dans la preuve. Ces stratégies sont analogues aux règles du calcul des séquents (LK) et la résolution de la preuve s'effectue de la même manière.

Règle LK	Tactique Coq
ax	assumption
cut	cut
$\Rightarrow_{right}$	intro
$\Rightarrow_{left}$	apply
$\Leftrightarrow_{right}$	split
$\Leftrightarrow_{left}$	elim
$\wedge_{right}$	split
$\wedge_{left}$	elim
$\vee_{right1}$	left
$\vee_{right2}$	right
$\vee_{left}$	elim
$\neg_{right}$	intro
$\neg_{left}$	elimtype False + apply
$\top_{right}$, $\bot_{left}$	auto

D'autres stratégies (tauto) permettent la résolution automatique de la preuve, nous permettant de s'assurer que les expressions ont été correctement modélisées avant d'essayer de les prouver. Nous utiliserons le logiciel CoqIde qui est un environnement de développement pour le langage Coq et qui facilite grandement son utilisation.

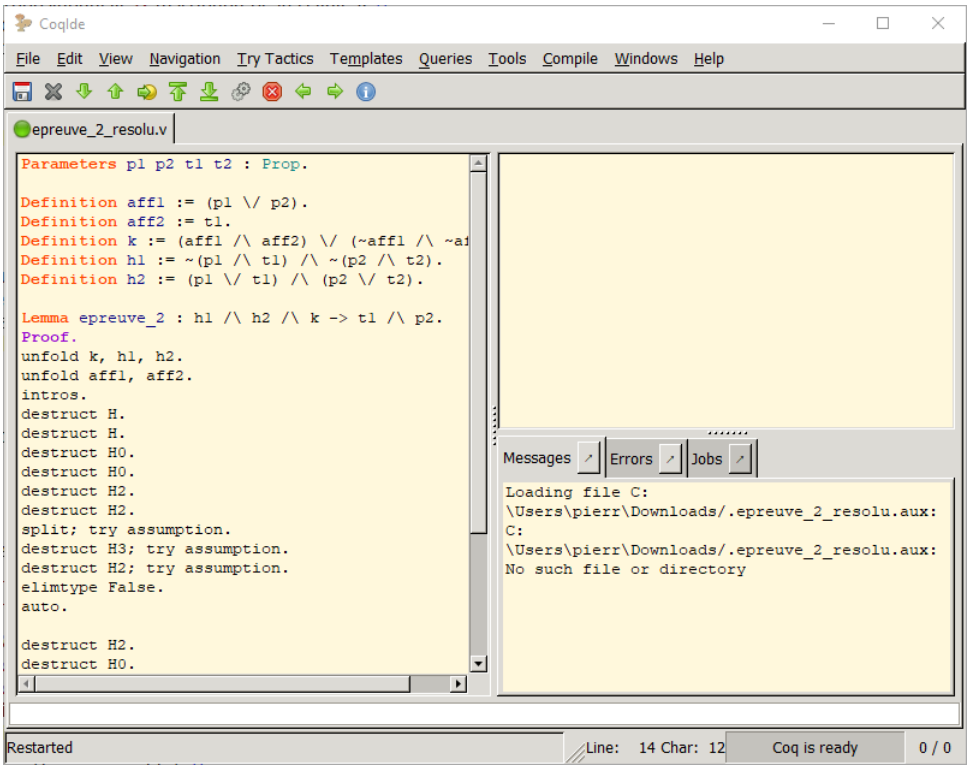


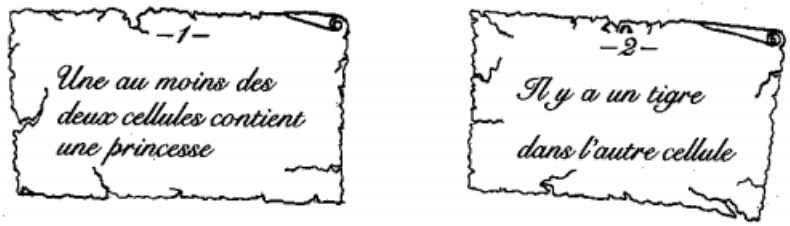
FIGURE 2 – Le logiciel CoqIde

4 Conception

4.1 Résolution manuelle

La première étape de notre projet est la résolution manuelle des épreuves. Cela nous a permis de nous familiariser avec la logique des énigmes. Pour faciliter la résolution des épreuves nous avons posé à l'écrit notre raisonnement. Nous avons remarqué qu'il fallait souvent faire des raisonnements par l'absurde pour écarter des hypothèses. Il est aussi souvent question d'incohérence, on se rend compte, par exemple, qu'une inscription dit vrai alors qu'elle doit dire faux. En procédant par élimination à l'aide de ces techniques, il ne reste alors qu'une seule solution.

Cas de l'épreuve 2 :



La règle du roi : *Soit les deux affiches sont sincères, soit elles sont fausses toutes les deux.*

Résolution manuelle : Si l'affiche 2 était fausse, alors la cellule 1 contiendrait une princesse. Il y aurait donc au moins une princesse dans une des deux cellules, ce qui impliquerait que l'affiche 1 soit vraie. Il est donc impossible que les deux affiches mentent en même temps. Ainsi, elles sont vraies toutes les deux, car la règle du roi précise qu'il est impossible qu'une cellule dise la vérité pendant que l'autre ment. Puisque la cellule 2 dit vrai, le tigre se trouve dans la cellule 1 et la princesse dans l'autre.

Cette étape permet d'obtenir les solutions des énigmes qui sont nécessaires à la modélisation des épreuves en expressions propositionnelles.

Épreuve	1	2	3	4	5	6	7	8	9	10	11	12
Solution	p1	p2	p1 et p2	p1	p2	p1	p2	p1	p1	p1	p1	p7

4.2 Modélisation des épreuves en logique propositionnelle

Une épreuve est constituée de 4 parties différentes qu'il faudra modéliser en expressions de la logique propositionnelle :

- les règles axiomatiques, qui désignent les règles de base qu'elles soient explicites dans le texte ou implicites et qui sont similaires dans toutes les épreuves.
- les inscriptions des cellules.
- la règle du roi, elle influe sur la véracité du contenu des inscriptions.
- la formule finale, c'est l'expression propositionnelle construite sous la forme d'une implication et qu'il faudra prouver.

Pour vérifier qu'une épreuve est bien modélisée, on pourra utiliser la stratégie tauto fournie par Coq et qui permet de faire la preuve automatiquement de la formule finale. Il s'avère que cette stratégie ne fonctionne pas pour l'épreuve 12 du fait de sa longueur.

4.2.1 Règles axiomatiques

La première règle axiomatique permet de préciser qu'il ne peut pas se trouver une princesse et un tigre dans la même cellule. Ainsi, pour le cas d'une énigme qui comporte deux cellules, cette règle s'écrit de la façon suivante :

$$h1 \equiv \neg(p1 \wedge t1) \wedge \neg(p2 \wedge t2)$$

La deuxième règle axiomatique précise qu'une cellule comportera obligatoirement, soit un tigre, soit une princesse. Pour le cas d'une énigme à deux cellules, l'expression propositionnelle s'écrit :

$$h2 \equiv (p1 \vee t1) \wedge (p2 \vee t2)$$

Cette règle devra être retirée quand nous commencerons à aborder le cas des épreuves où l'on considère que des cellules vides peuvent exister.

Une troisième règle axiomatique apparaît à partir de l'épreuve 9 (qui comprend 3 cellules) avec le roi qui stipule qu'une princesse se trouve dans une seule cellule et un tigre dans chacune des deux autres. Pour modéliser cette règle, on considère toutes les combinaisons de princesses et tigres possibles :

$$h3 \equiv (p1 \wedge t2 \wedge t3) \vee (t1 \wedge p2 \wedge t3) \vee (t1 \wedge t2 \wedge p3)$$

Le cas des cellules vides Ces types de cellules apparaissent à partir de l'épreuve 11. Pour définir les cellules vides, on crée une nouvelle formule propositionnelle pour chaque cellule et qui précise que si celle-ci est vide, alors elle ne contient ni une princesse, ni un tigre :

Nom	Type	Expression	Signification
vX	Formule propositionnelle	$\neg pX \wedge \neg tX$	La cellule X est vide

4.2.2 Les inscriptions des cellules

La modélisation en expression propositionnelle du contenu des inscriptions est relativement triviale. Pour cette raison, il ne nous semble pas pertinent d'aborder l'ensemble des épreuves, mais juste une sélection des épreuves que nous jugeons les plus intéressantes ou qui justifient une explication supplémentaire.

Epreuve 1, cellule 2 (2 cellules dans l'épreuve) : Cette cellule stipule : "Il y a une princesse dans une cellule et il y a un tigre dans une cellule". Pour modéliser cet énoncé en une expression logique, il suffit de préciser les deux possibilités : soit il y a un tigre dans la cellule 1 et une princesse dans la cellule 2, ou alors c'est l'inverse :

$$\mathit{aff2} \equiv (p1 \wedge t2) \vee (p2 \wedge t1)$$

Epreuve 6, cellule 1 (2 cellules dans l'épreuve) : Cette cellule stipule "Choisis n'importe quelle cellule, ça n'a pas d'importance !". Cet énoncé signifie qu'il se trouve une princesse derrière chaque cellule. Ainsi, le fait que le prisonnier choisisse l'une ou l'autre cellule n'a pas d'importance :

$$\mathit{aff1} \equiv (p1 \wedge p2)$$

La septième épreuve : La première cellule stipule "Choisis bien ta cellule, ça a de l'importance !". Cela veut dire qu'il est impossible que chaque cellule contienne une princesse car le choix du prisonnier n'aurait alors aucune importance :

$$\mathit{aff1} \equiv \neg(p1 \wedge p2)$$

La seconde cellule stipule "Tu ferais mieux de choisir l'autre cellule !", ce qui signifie simplement que la princesse se trouve dans la cellule 1 :

$$\mathit{aff2} \equiv p1$$

La huitième épreuve : Dans cette épreuve on ne connaît pas le numéro de la cellule sur laquelle chaque inscription est posée. Le roi stipule que cette information n'est pas importante pour la résolution de l'énigme. Puisqu'on connaît déjà la réponse de l'énigme, on place les inscriptions sur les cellules qui permettent de mener à la réponse.

$$\mathit{aff1} \equiv t1 \wedge t2 \; \mathit{aff2} \equiv t1$$

La douzième épreuve : Cette épreuve comporte 9 cellules et des inscriptions font référence à d'autres inscriptions. Ce tableau montre l'expression propositionnelle correspondante au contenu de l'inscription de chaque cellule :

Numéro de cellule	Expression propositionnelle
1	$p1 \vee p3 \vee p5 \vee p7 \vee p9$
2	$v2$
3	$\neg aff1$
4	$aff2 \vee aff4$
5	$\neg p1$
6	$aff5 \vee \neg aff7$
7	$\neg aff3$
8	$t8 \wedge v9$
9	$t9 \wedge \neg aff6$

4.2.3 La règle du roi

Cette règle change au fur à mesure des épreuves et elle influe sur la véracité du contenu des inscriptions.

La règle du roi de l'épreuve 1 : Dans cette épreuve, le roi stipule que "une des affiches dit la vérité, et l'autre ment". Pour modéliser cette règle on considère les deux possibilités : soit la cellule 1 dit vrai et la cellule 2 dit faux ou inversement :

$$k \equiv (aff1 \wedge \neg aff2) \vee (\neg aff1 \wedge aff2)$$

La règle du roi des épreuves 2 et 3 : Dans ces épreuves, la règle du roi est la suivante : "les affiches sont sincères toutes les deux, ou bien elles sont fausses toutes les deux". On traduit littéralement cette phrase en expression logique :

$$k \equiv (aff1 \wedge aff2) \vee (\neg aff1 \wedge \neg aff2)$$

La règle du roi des épreuves 4, 5, 6, 7, 8, 10 : Dans ces épreuves, la règle du roi est la suivante : "l'affiche de la cellule 1 dira la vérité quand il y a une princesse et mentira quand ce sera un tigre. Pour la cellule 2, ce sera exactement l'inverse". Pour modéliser cet énoncé en expression logique, on précise toutes les combinaisons possibles :

$$k \equiv ((p1 \wedge aff1) \vee (t1 \wedge \neg aff1)) \wedge ((p2 \wedge \neg aff2) \vee (t2 \wedge aff2))$$

La règle du roi de l'épreuve 9 : Dans cette épreuve, le roi signifie simplement qu'une seule cellule dit vrai et que les deux autres cellules disent fausses. Pour la modélisation, il est encore question de préciser toutes les combinaisons possibles :

$$k \equiv (aff1 \wedge \neg aff2 \wedge \neg aff3) \vee (\neg aff1 \wedge aff2 \wedge \neg aff3) \vee (\neg aff1 \wedge \neg aff2 \wedge aff3)$$

La règle du roi de l'épreuve 11 : Dans cette épreuve, le roi explique qu'il y a une princesse, un tigre et une cellule vide. L'affiche de la princesse dit vrai, celle du tigre dit faux et pour celle de la cellule vide ce n'est pas renseigné. Pour modéliser cet énoncé en expression logique, on précise toutes les informations qui sont connus par une succession d'implications :

$$k \equiv (p1 \Rightarrow \mathit{aff1}) \wedge (t1 \Rightarrow \neg \mathit{aff1}) \wedge (p2 \Rightarrow \mathit{aff2}) \wedge (t2 \Rightarrow \neg \mathit{aff2}) \wedge (p3 \Rightarrow \mathit{aff3}) \wedge (t3 \Rightarrow \neg \mathit{aff3})$$

En utilisant des implications, on n'oblige pas les cellules à être remplies, on leur donne la possibilité d'être vide et d'avoir une valeur de vérité inconnue.

La règle du roi de l'épreuve 12 : La règle du roi de cette épreuve est la même que pour l'épreuve 11 sauf qu'elle comporte 9 cellules. On fait donc exactement le même procédé pour modéliser cette règle en expression logique :

$$\begin{aligned} k \equiv (p1 \Rightarrow \mathit{aff1}) \wedge (t1 \Rightarrow \neg \mathit{aff1}) \wedge (p2 \Rightarrow \mathit{aff2}) \wedge (t2 \Rightarrow \neg \mathit{aff2}) \\ \wedge (p3 \Rightarrow \mathit{aff3}) \wedge (t3 \Rightarrow \neg \mathit{aff3}) \wedge (p4 \Rightarrow \mathit{aff4}) \wedge (t4 \Rightarrow \neg \mathit{aff4}) \\ \wedge (p5 \Rightarrow \mathit{aff5}) \wedge (t5 \Rightarrow \neg \mathit{aff5}) \wedge (p6 \Rightarrow \mathit{aff6}) \wedge (t6 \Rightarrow \neg \mathit{aff6}) \\ \wedge (p7 \Rightarrow \mathit{aff7}) \wedge (t7 \Rightarrow \neg \mathit{aff7}) \wedge (p8 \Rightarrow \mathit{aff8}) \wedge (t8 \Rightarrow \neg \mathit{aff8}) \\ \wedge (p9 \Rightarrow \mathit{aff9}) \wedge (t9 \Rightarrow \neg \mathit{aff9}) \end{aligned}$$

4.2.4 La formule finale

Cette formule est sous la forme d'une implication. Du côté gauche, on positionne toutes les règles que nous avons précédemment modélisées par une succession de conjonctions. Du côté droit, on positionne la solution de l'épreuve.

Exemple pour l'épreuve 5 :

$$\begin{aligned} &\mathit{aff1} \equiv p1 \vee p2 \\ &\mathit{aff2} \equiv p1 \\ &k \equiv ((p1 \wedge \mathit{aff1}) \vee (t1 \wedge \neg \mathit{aff1})) \wedge ((p2 \wedge \neg \mathit{aff2}) \vee (t2 \wedge \mathit{aff2})) \\ &h1 \equiv \neg(p1 \wedge t1) \wedge \neg(p2 \wedge t2) \\ &h2 \equiv (p1 \vee t1) \wedge (p2 \vee t2) \end{aligned}$$

$$\boldsymbol{\mathit{epreuve5} \equiv h1 \wedge h2 \wedge k \Rightarrow p1 \wedge t2}$$

4.3 Résolution des preuves avec Coq

4.3.1 Intégration des expressions logiques en Coq

L'étape précédente nous a permis d'obtenir l'ensemble des expressions logiques nécessaires à la construction de la preuve. Il est maintenant question de les intégrer dans l'environnement Coq. Nous prendrons comme exemple l'épreuve 2 afin de comprendre la terminologie utilisée par Coq.

On commence par définir les variables propositionnelles à l'aide de la commande Parameters et du type Prop. :

$$Parameters\ \text{p1 p2 t1 t2} : Prop.$$

Puis on définit les expressions propositionnelles avec la commande Definition :

$$\begin{aligned}
&Definition\ \textit{aff1} := (p1 \vee p2).\\
&Definition\ \textit{aff2} := t1.\\
&Definition\ k := (\textit{aff1} \wedge \textit{aff2}) \vee (\neg \textit{aff1} \wedge \neg \textit{aff2}).\\
&Definition\ h1 := \neg(p1 \wedge t1) \wedge \neg(p2 \wedge t2).\\
&Definition\ h2 := (p1 \vee t1) \wedge (p2 \vee t2).
\end{aligned}$$

Enfin, on définit l'expression à prouver avec la commande Lemma et en lui donnant un nom :

$$Lemma\ \text{epreuve2}: h1 \wedge h2 \wedge k \Rightarrow t1 \wedge p2.$$

4.3.2 Début de la preuve

Chaque début de preuve débute de la même façon. On commence en utilisant la commande unfold qu'on applique sur les formules propositionnelles afin de développer l'expression à prouver.

$$\begin{gathered}
h1 \wedge h2 \wedge k \Rightarrow t1 \wedge p2\\
\Downarrow\\
unfold\ k,\ h1,\ h2.\\
unfold\ \textit{aff1},\ \textit{aff2}.\\
\Downarrow\\
\neg(p1 \wedge t1) \wedge \neg(p2 \wedge t2)) \wedge ((p1 \vee t1) \wedge (p2 \vee t2)) \wedge ((p1 \vee p2) \wedge t1 \vee \neg(p1 \vee p2) \wedge \neg t1 \Rightarrow t1 \wedge p2
\end{gathered}$$

On utilise ensuite la commande intros pour mettre la partie gauche de l'implication en hypothèse.

$$\underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}(1/1)$$
$$\neg(p1 \wedge t1) \wedge \neg(p2 \wedge t2)) \wedge ((p1 \vee t1) \wedge (p2 \vee t2)) \wedge ((p1 \vee p2) \wedge t1 \vee \neg(p1 \vee p2) \wedge \neg t1 \Rightarrow t1 \wedge p2$$
$$\Downarrow$$
$$intros.$$
$$\Downarrow$$
$$H : (\neg(p1 \wedge t1) \wedge \neg(p2 \wedge t2)) \wedge ((p1 \vee t1) \wedge (p2 \vee t2)) \wedge ((p1 \vee p2) \wedge t1 \vee \neg(p1 \vee p2) \wedge \neg t1)$$
$$\underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}(1/1)$$
$$t1 \wedge p2$$

Avec la commande destruct on casse les conjonctions successives de l'hypothèse H pour obtenir une liste d'hypothèses.

$$H : (\neg(p1 \wedge t1) \wedge \neg(p2 \wedge t2)) \wedge ((p1 \vee t1) \wedge (p2 \vee t2)) \wedge ((p1 \vee p2) \wedge t1 \vee \neg(p1 \vee p2) \wedge \neg t1)$$
$$\underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}(1/1)$$
$$t1 \wedge p2$$
$$\Downarrow$$
$$destruct\ H.$$
$$destruct\ H.$$
$$destruct\ H0.$$
$$destruct\ H0.$$
$$\Downarrow$$
$$H : \neg(p1 \wedge t1)$$
$$H1 : \neg(p2 \wedge t2)$$
$$H0 : p1 \vee t1$$
$$H3 : p2 \vee t2$$
$$H2 : (p1 \vee p2) \wedge t1 \vee \neg(p1 \vee p2) \wedge \neg t1$$
$$\underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}(1/1)$$
$$t1 \wedge p2$$

On peut maintenant commencer à utiliser les stratégies de preuve.

4.3.3 Stratégies

Première stratégie : Cette stratégie consiste à trouver la conclusion dans une des hypothèses. En utilisant les stratégies fournies par Coq, on va essayer de développer au maximum l'hypothèse qui nous semble la plus pertinente. Si on trouve qu'une hypothèse et la conclusion sont similaires, on appele simplement la commande assumption.

$$
\begin{array}{c}
H : \neg(p1 \wedge t1) \\
H1 : \neg(p2 \wedge t2) \\
H2 : t1 \\
\hline
t1 \\
\Downarrow \\
assumption. \\
\Downarrow \\
\text{No more subgoals.}
\end{array}
\quad (1/1)
$$

Deuxième stratégie : Cette stratégie consiste à trouver deux hypothèses qui sont l'inverse l'une de l'autre. Pour cela, on va essayer de développer au maximum les hypothèses qui nous semblent pertinentes à l'aide des stratégies fournies par Coq. Quand on détecte qu'une hypothèse est l'inverse d'une autre hypothèse, on appelle la commande elimtype False qui va mettre False en conclusion puis on appelle la commande auto.

$$
\begin{array}{c}
H2 : t2 \\
H3 : \neg t2 \\
\hline
t1 \\
\Downarrow \\
elimtype\ False. \\
\Downarrow \\
H2 : t2 \\
H3 : \neg t2 \\
\hline
\text{False} \\
\Downarrow \\
auto. \\
\Downarrow \\
\text{No more subgoals.}
\end{array}
\quad (1/1)
$$

5 Conclusion

A l'issue de ce projet, nous sommes parvenus à accomplir notre objectif de prouver les 12 épreuves tirées du livre de Smullyan.

Sur une période de 3 mois, nous nous sommes réunis, en groupe, environ une fois toutes les deux semaines, pour discuter de notre travail. Une conversation en ligne et un Github ont été mis en place pour nous permettre de partager, comparer et discuter de nos travaux. Des rendez-vous réguliers ont été effectués avec notre tuteur lors desquels nous fixions un objectif pour la semaine.

Ce projet nous a permis d'apprendre un nouveau langage, Coq, ainsi qu'une mise en abyme dans le domaine des preuves mathématiques. Bien que nous connaissions la méthode de calcul de séquents, l'apprentissage de Coq s'est avéré complexe du fait de son caractère peu accessible. Il semblait parfois que l'application de règles simples s'avèrent plus compliquée que prévu dans l'environnement Coq.

Avec 386 lignes de code Coq, la preuve de l'épreuve 12 s'est avérée être un véritable challenge. Du fait de sa longueur (dès sa modélisation en expression propositionnelle), il était impossible de la résoudre automatiquement à l'aide de la règle tauto. Cela signifie que, jusqu'à la toute fin de la preuve, nous n'étions pas sûre que nous l'avions correctement modélisée.