Travail de Recherche Encadré :
Preuve du théorème de complétude de Gödel

Benjamin CATINAUD

Année 2020-2021

Abstract

Introduction

Ce papier a pour but d'écrire une preuve du théorème de complétude de Gödel qui établit un rapport entre les formules valides et prouvables. Plusieurs preuves existent déjà, en particulier en utilisant la sémantique de Kripke, en utilisant plutôt la partie "programme" de l'équivalence entre les preuves et les programmes.

Dans ce papier, nous allons montrer ce théorème en utilisant la sémantique de Tarski et en écrivant une preuve avec l'assistant aux preuves Coq. La sémantique de Tarski se distingue de la sémantique de Kripke au sens où cette dernière possède une "mémoire" ce qui aide à prouver le théorème.

Nous allons donc utiliser un procédé, le forcing, qui permet d'étendre les modèles de Tarski en des modèles étendus équivalents aux modèles de Kripke.

Ce travail a été fait avec l'aide du docteur Hugo Herbelin.

I. Cadre de travail

Tout d'abord, on se place dans un langage des formules de la logique. On considère d'abord les atomes, que l'on va numéroter par un entier naturel, ainsi que les constructeurs "et" ($\wedge$) et "implique" ($\implies$) :

```
Inductive form :=
    | Atome : nat -> form
    | And : form -> form -> form
    | Implies : form -> form -> form.
```

Sur cet ensemble de formules, on peut définir la notion de prouvabilité afin de déterminer la valeur de vérité d'une formule. On définit d'abord la relation $\vdash$ sur les listes de formules notée

$$\Gamma \vdash \Delta.$$

La liste Γ est appelée la liste *subséquente* et la liste Δ la liste *conséquente*. On a alors que si les prédicats de Γ sont vrais alors l'une des conséquences de Δ est vraie.

Lorsque la liste Γ est vide, on note plus simplement $\vdash \Delta$.

En pratique dans ce qui suit, on va considérer que Δ est réduite à une seule formule. De plus, on définit Γ en Coq comme étant une liste de formule appelée **context** :

```
Definition context := list form.
```

Définition 1 (Prouvabilité d'une formule)
On définit la prouvabilité d'une formule comme étant un séquent dérivable à partir d'un certain nombre de règles d'inférence et d'axiomes. Dans notre cadre les règles sont les règles de la logique intuitionniste définies comme suit :

- ***(Axiome)*** $\dfrac{}{\Gamma \vdash A}$ *lorsque* $A \in \Gamma$.
- ***($\wedge$-intro)*** $\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$
- ***($\wedge$-élim$_{1,2}$)*** $\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$ *et* $\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$.

On définit alors cette notion en Coq comme suit :

```
Inductive provable : context -> form -> Prop :=
    | Ax { ctx A } : In A ctx -> provable ctx A
    | AndI { ctx A B } : provable ctx A -> provable ctx B
                                          -> provable ctx (And A B)
    | AndE1 { ctx A B } : provable ctx (And A B) -> provable ctx A
    | AndE2 { ctx A B } : provable ctx (And A B) -> provable ctx B
```

```
| ImpliesI { ctx A B } : provable (cons A ctx) B -> provable ctx (Implies A B)
| ImpliesE { ctx A B } : provable ctx (Implies A B) -> provable ctx A
                                                    -> provable ctx B.
```

Pour pouvoir interpréter ces formules, on a alors besoin d'une sémantique. Dans ce travail, on va en définir 2 différentes, l'une est la sémantique de Kripke et l'autre la sémantique de Tarski. À chaque fois, la sémantique se base sur un modèle que l'on va d'abord définir.

Définition 2 (Modèle de Kripke)
On définit un modèle de Kripke comme un triplet $\mathcal{K} = (\mathcal{W}, \leqslant, h)$ *où*

- $\mathcal{W}$ *est l'ensemble des mondes.*
- $\leqslant$ *est une relation d'ordre sur les mondes de* $\mathcal{W}$*.*
- $h_{\mathcal{K}}$ *est une relation qui indique pour chaque atome* X *l'ensemble des mondes* $\mathcal{W}$ *où* X *est vraie. Traduit en théorie des types comme une fonction vers* ***Prop****.*

$$h_{\mathcal{K}} : \mathcal{W} \times \mathbb{N} \to \textbf{\textit{Prop}}.$$

On définit alors le modèle de Kripke standard, le modèle qui fait coïncider sémantique et prouvabilité sur les atomes :

$$\mathcal{K}_0 = \Big(\textbf{context}, \textbf{extend}, h_{\mathcal{K}_0}(\Gamma, X) \triangleq (\Gamma \vdash X)\Big)$$

Définition 3 (Sémantique de Kripke)
On définit la sémantique de Kripke à partir du modèle standard $\mathcal{K}_0$ *comme suit :*

- $\omega \Vdash_{\mathcal{K}_0} X \triangleq h_{\mathcal{K}_0}(\omega, X)$.
- $\omega \Vdash_{\mathcal{K}_0} \big(P \mathbin{\dot\wedge} Q\big) \triangleq \big(\omega \Vdash_{\mathcal{K}_0} P\big) \wedge \big(\omega \Vdash_{\mathcal{K}_0} Q\big)$.
- $\omega \Vdash_{\mathcal{K}_0} \big(P \implies Q\big) \triangleq \forall\, \omega' \geqslant \omega, (\omega' \Vdash_{\mathcal{K}_0} P) \implies (\omega' \Vdash_{\mathcal{K}_0} Q)$.

On peut déjà définir cet sémantique en Coq comme suivant :

```
Fixpoint semK (K : context -> nat -> Prop) (ctx : context) (A : form) :=
    match A with
        | Atome n   => K ctx n
        | And A1 A2 => ((semK K ctx A1) * (semK K ctx A2))%type
        | Implies A1 A2 => forall ctx', extend ctx' ctx -> (semK K ctx' A1)
                                                        -> (semK K ctx' A2)
    end.
```

Définition 4 (Modèle de Tarski)
On définit un modèle de Tarski comme étant une fonction

$$\mathcal{M} : \mathbb{N} \to \textbf{\textit{Prop}}.$$

On remarque alors qu'un modèle de Tarski n'a pas de "mémoire" contrairement à un modèle de Kripke. C'est ce qui fait la difficulté de la preuve du théorème de complétude et qui nécessite donc l'appel au plugin *Forcing*.

Définition 5 (Sémantique de Tarski)
On définit la sémantique de Tarski à partir du modèle $\mathcal{M}$ *comme suit :*

- $[\![X]\!]_{\mathcal{M}} \triangleq \mathcal{M}(X)$

- $[\![P \wedge Q]\!]_{\mathcal{M}} \triangleq [\![P]\!]_{\mathcal{M}} \wedge [\![Q]\!]_{\mathcal{M}}$.
- $[\![P \implies Q]\!]_{\mathcal{M}} \triangleq [\![P]\!]_{\mathcal{M}} \implies [\![Q]\!]_{\mathcal{M}}$.

Théorème 1 (Théorème de Complétude de Gödel)
Toute formule φ valide, au sens de Tarski, est prouvable à partir du contexte vide : $\vdash \varphi$.

II. Présentation du plugin *Forcing*

Le principe du *forcing* est d'étendre un univers pour le rendre plus expressif. Dans notre cas, le *forcing* de la sémantique de Tarski va permettre de le rendre équivalent à la sémantique de Kripke, pour laquelle on sait déjà faire la démonstration du théorème de complétude de Gödel, via les fonctions mutuellements récursives **reify** et **reflect**.

Le plugin *coq-forcing* est composé de 2 fonctions principales : **Forcing Translate** et **Forcing Definition**.

La fonction **Forcing Translate** s'appuit sur la définition suivante :

Définition 6 (Forcing Translate)
La traduction par forcing est définie par induction structurelle sur les termes comme suit :

$$
\begin{array}{rcc}
[*]_\sigma & := & \lambda(q\, f : \sigma).\Pi(r\, g : \sigma \cdot (q, f)).* \\
[\square_i]_\sigma & := & \lambda(q\, f : \sigma).\Pi(r\, g : \sigma \cdot (q, f)).\square_i \\
[x]_\sigma & := & x\, \sigma_e\, \sigma(x) \\
[\lambda x : A.M]_\sigma & := & \lambda x : [\![A]\!]^!_\sigma.[M]_{\sigma \cdot x} \\
[M\, N]_\sigma & := & [M]_\sigma [N]^!_\sigma \\
[\Pi x : A.B]_\sigma & := & \lambda(q\, f : \sigma).\Pi x : [\![A]\!]^!_{\sigma\cdot(q,f)}.[\![B]\!]_{\sigma\cdot(q,f)\cdot x} \\
[\![A]\!]_\sigma & := & [A]_\sigma\, \sigma_e\, \boldsymbol{id}_{\sigma_e} \\
[M]^!_\sigma & := & \lambda(q\, f : \sigma).[M]_{\sigma\cdot(q,f)} \\
[\![A]\!]^!_\sigma & := & \Pi(q\, f : \sigma).[\![A]\!]_{\sigma\cdot(q,f)}
\end{array}
$$

La syntaxe pour **Forcing Translate** dans le plugin est la suivante :

```
Forcing Translate foo {as id} using Obj Hom.
```

Lorsqu'aucune règle de traduction n'est applicable, on peut tout de même utiliser la fonction **Forcing Definition** pour effectuer la preuve de la traduction "à la main".

La syntaxe pour **Forcing Definition** dans le plugin est la suivante :

```
Forcing Definition foo : type {as id} using Hom Obj.
```

III. Preuve du théorème de complétude

Pour prouver le théorème, on va avoir besoin d'effectuer plusieurs étapes. Tout d'abord, on traduit via le *forcing* **valid** et **provable**. Ensuite, en spécialisant le modèle au modèle de Kripke standard, on se retrouve à montrer que $\mathbf{sem}^f$ implique $\mathbf{provable}^f$. Comme le *forcing* permet d'avoir une équivalence entre $\mathbf{sem}^f$ et **semK** ainsi qu'entre $\mathbf{provable}^f$ et **provable**, on se retrouve alors à montrer que :

$$\forall\, \varphi, \Vdash_{\mathcal{K}_0} \varphi \implies \vdash \varphi. \tag{1}$$

1) Forcing du monde de la sémantique de Tarski

2) Équivalence entre le *forcing* du monde de la sémantique de Tarski et celui de la sémantique de Kripke

Partie admise pour l'instant.

3) Fonctions reify et reflect

Bibliography

[1] Olivier Danvy, Morten Rhiger, Kristoffer H. Rose. *Normalization by Evaluation with Typed Abstract Syntax* BRICS Report Series, RS-01-16, May 2001.

[2] Catarina Coquand. *A Formalised Proof of the Soundness and Completeness of a Simply Typed Lambda-Calculus with Explicit Substitutions* Department of Computing Science, Chalmers University of Technology and University of Göteborg, 412 96 Göteborg, Sweden.

[3] Guilhem Jaber, Gabriel Lewertowski, Pierre-Marie Pédrot, Matthieu Sozeau, Nicolas Tabareau. *The Definitional Side of the Forcing* Logics in Computer Science, May 2016, New York, United States.

[4] CoqHott, SkySkimmer. *Plugin Coq pour le Forcing* https://github.com/CoqHott/coq-forcing.