INFO0054 - Programmation Fonctionnelle

Projet : Tableaux Sémantiques

Christophe Debruyne
Septembre 2022

Le projet

Le but de ce projet est d'implémenter l'algorithme des tableaux sémantiques pour la logique propositionnelle. Cet algorithme est utilisé pour déterminer, entre autres :

- La **satisfiabilité des ensembles finis de formules** $F = \{\phi_1,..., \phi_n\}$ de logique propositionnelle ; i.e., est-ce qu'il existe au moins un modèle pour lequel toutes les formules *F* sont vraies ?
- Montrer qu'une **formule ϕ est valide sous les hypothèses** $F = \{\phi_1,..., \phi_n\}$.
- Montrer qu'une **formule ϕ est une tautologie.**
- Montrer qu'une **formule ϕ (ou un ensemble de formules *F*) n'a pas de modèle (contradiction)**.
- **Déterminer les modèles** d'un ensemble de formules
- **Déterminer les contre-exemples** d'une formule ϕ sous les hypothèses *F*.

Vous voyez que quelques de ces tâches font appel à cet algorithme des tableaux sémantiques.

Nous prenons comme exemple l'ensemble des formules $F = \{p \vee q,\ r,\ \neg q\}$ (donc "p OR q", "r", et "NOT q". Vous avez sans doute vu comment identifier les modèles d'un ensemble de formules en utilisant des tableaux. Chaque ligne correspond à des attributions de valeurs de vérité à toutes les variables apparaissant dans F.

	p	q	r	$p \vee q$	r	$\neg q$	F
v1	0	0	0	0			
v2	0	0	1	0			
v3	0	1	0	1	0		
v4	0	1	1	1	1	0	
v5	1	0	0	1	0		
v6	1	0	1	1	1	1	1
v7	1	1	0	1	0		
v8	1	1	1	1	1	0	

Dans le cas de v6, nous avons v6(p) = 1, v6(q) = 0, et v6(r) = 1. Si une attribution rend toutes les formules de F vraies, on nomme cette attribution un modèle.

	p	q	r	$p \vee q$	r	$\neg q$	F
v1	0	0	0	0			
v2	0	0	1	0			
v3	0	1	0	1	0		
v4	0	1	1	1	1	0	
v5	1	0	0	1	0		
v6	1	0	1	1	1	1	1
v7	1	1	0	1	0		
v8	1	1	1	1	1	0	

L'ensemble des modèles pour *F* est *{v6}*. Vu que *F* a au moins un modèle, nous pouvons conclure que *F* est *satisfiable*.

L'algorithme des tableaux sémantiques nous permet de faire le même exercice d'une manière plus élégante et efficace. Le but est de décomposer les formules, en tenant compte de leurs sémantiques, et de constater la présence d'une contradiction. Nous reprenons l'exemple ci-dessus. Nous créons un tableau que nous nommons T1.

T1: $p \vee q,\ r,\ \neg q$

Nous constatons que ce tableau ne contient pas une contradiction ; il n'y pas une formule ϕet ¬ϕ.

Les formules dans ce tableau ne sont pas toutes des formules de base (i.e., une proposition ou sa négation) ; nous avons des opérateurs que nous pouvons éliminer. La formule $p \vee q$ est vraie quand *p* est vrai (peu importe la valeur de *q*), ou quand *q* est vrai (peu importe la valeur de *p*). Dans ce cas, on doit créer deux tableaux (qui contiendront, ensemble, les trois possibilités : *p* et *q*, *p* et $\neg q$, et $\neg p$ et *q*).

T1: $p,\ r,\ \neg q$

T2: $q,\ r,\ \neg q$

T2 contient une contradiction et ce tableau est dit "fermé". Toutes les formules en T1 sont des formules de base (nous n'avons pas de formules composées à éliminer) et T1 ne contient pas de contradiction. T1 est dit "ouvert". Le tableau T1 contient les modèles de *F*.

Attention : un tableau peut contenir plusieurs modèles ! Si *F* fait référence aux propositions *r*, *s*, et *t*, et un tableau Tx ne contient que la formule $\neg r$ (par exemple), ça implique que ce tableau contient les modèles ou *r* est faux, peu importe les valeurs des propositions *s* et *t* ! Le tableau Tx contient donc 4 modèles :

- v1(r) = 0, v1(s) = 0, v1(t) = 0
- v2(r) = 0, v2(s) = 0, v2(t) = 1
- v3(r) = 0, v3(s) = 1, v3(t) = 0
- v4(r) = 0, v4(s) = 1, v4(t) = 1

Partie 1 : La fonction `semtab` (30%)

C'est à vous de créer une fonction `semtab` qui prend comme argument un ensemble de formules de logique propositionnelle et renvoie une liste de tableaux.

Vous devez creer un ADT pour representer les formules de logique propositionnelle.

- `p, q, r, s, …`
- $((p \vee q) \wedge \neg(s \wedge \neg s))$
- $(p \rightarrow (p \vee r))$
- $\neg\neg s$

C'est a vous de voir si des ADT pour les tableaux sont opp

Vous devez implémenter les règles d'élimination de base. Vous devez, pour chaque opérateur logique, analyser quand cet opérateur renvoie vrai ou faux, et déterminer les tableaux que chaque règle renvoie. Les règles d'élimination (de base) pour NOT, OR, AND, et IFTHEN ne renvoient qu'un ou deux tableaux :

- $\neg\neg\phi$ est vrai si ϕ est vrai;
- $(\phi \vee \psi)$ est vrai si ϕ OU ψ sont vrais
- $(\phi \wedge \psi)$ est vrai si ϕ ET ψ sont vrais
- $(\phi \rightarrow \psi)$ est vrai si ϕ est faux OU ψ est vrai
- $\neg(\phi \vee \psi)$ est vrai si ϕ ET ψ sont faux
- $\neg(\phi \wedge \psi)$ est vrai si ϕ est faux OU ψ est faux
- $\neg(\phi \rightarrow \psi)$ est vrai si ϕ est vrai ET ψ est faux

Assurez-vous que votre approche et votre implémentation sont élégantes et extensibles.

Partie 2 : La bibliothèque (30%)

La fonction `semtab` vous fournit la base d'autres fonctions pour déterminer si une formule est une tautologie, si une formule est valide sous des hypothèses, etc. La liste des fonctions que vous devez implémenter sont:

- `isSatisfiable`
  - Est-ce qu'il existe au moins un modèle pour lequel toutes les formules *F* sont vraies ?
- `isValid`
  - Est-ce que tous les modèles de des hypothèses F sont aussi des modèles de la formule f ? Ou, est-ce que $F \cup \{\neg f\}$ est insatisfiable ?
- `isTautology`
  - Est-ce que la formule f est toujours vraie ? (attention, ne soyez pas naif!)
- `isContradiction`
  - Est-ce que la formule f est toujours fausse ? (attention, ne soyez pas naif!)
- `models`
  - Quels sont les modèles d'un ensemble de formules F ?
- `counterexamples`
  - Si la formule f n'est pas valide sous les hypothèses F, quels sont les modèles qui constituent les contre-exemples ?

Partie 3 : Extensions (30%)

- Vous devez conceptualiser, formaliser et implémenter les règles d'élimination pour EQUIV, XOR, NAND, et XNOR.
- Vous devez également proposer une généralisation pour les opérateurs OR et AND prenant n >= 1 arguments.

NAND = $\neg(p \wedge q)$ XOR = $(p \vee q) \wedge \neg(p \wedge q)$ XNOR = $\neg((p \vee q) \wedge \neg(p \wedge q))$

Partie 4 : Parser (bonus: 10%)

Pour des points "bonus", vous pouvez créer un parser pour les formules de la logique propositionnelle en utilisant "Chapter 9. Parser combinators" de votre livre. Vous pouvez vous limiter aux opérations AND, OR, NOT, et IFTHEN. Si vous essayez cet exercice, assurez-vous que cette partie du code se trouve dans un module séparé.

Rapport (10 %)

En plus de l'implémentation, il vous est demandé de soumettre un petit (!) rapport au format PDF reprenant les réponses aux questions suivantes :

1. Proposez
   a. Des règles d'élimination pour EQUIV, XOR, NAND, et XNOR.
   b. Une extension pour les opérateurs OR et AND prenant n >= 1 arguments.
2. Décrivez la représentation que vous avez utilisée pour les formules et les tableaux.
3. Décrivez vos choix d'implémentation pour les règles d'élimination.
4. Décrivez vos choix d'implémentation pour l'extension.
5. Décrivez l'utilisation de vos modules et donnez des exemples.
6. ~~Décrivez vos choix d'implémentation pour la visualisation des tableaux.~~

Dates d'échéances et attentes

- Le projet est à réaliser par groupe de **trois** étudiants et est à rendre pour le **2 décembre 2022 à 23:59** au plus tard.
- Vous devez rendre une archive au format **zip** comprenant votre rapport, ainsi que les fichiers .scala.
- Toutes les soumissions se feront via eCampus.
- Afin d'anticiper tous problèmes liés à la formation des groupes, vous devez constituer ceux-ci sur la plateforme de soumission pour le **20 octobre 2022**. Passé cette date, il ne sera plus possible de constituer un groupe, et donc de soumettre le projet.
- N'oubliez pas de **spécifier toutes les fonctions auxiliaires** ainsi que toutes les structures que vous implémentez.
- Pour rappel, le **plagiat est sévèrement sanctionné**.

Bon travail!