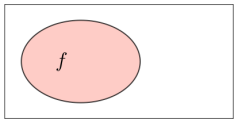
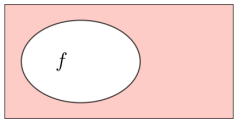
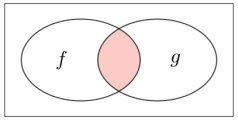
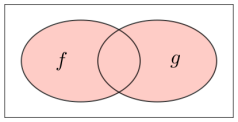
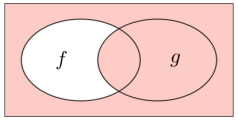
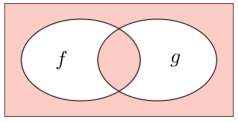
Pense-bête VIP : Modèles basés sur la logique

Afshine Amidi et Shervine Amidi

8 septembre 2019

Bases

❒ **Syntaxe de la logique propositionnelle** – En notant f et g formules et $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ opérateurs, on peut écrire les expressions logiques suivantes :

Nom	Symbole	Signification	Illustration
Affirmation	f	f	
Négation	$\neg f$	non f	
Conjonction	$f \wedge g$	f et g	
Disjonction	$f \vee g$	f ou g	
Implication	$f \rightarrow g$	si f alors g	
Biconditionnel	$f \leftrightarrow g$	f, c'est à dire g	

Remarque : n'importe quelle formule peut être construite de manière récursive à partir de ces opérateurs.

❒ **Modèle** – Un modèle w dénote une combinaison de valeurs binaires liées à des symboles propositionnels.

Exemple : l'ensemble de valeurs de vérité $w = \{A : 0, B : 1, C : 0\}$ est un modèle possible pour les symboles propositionnels A, B and C.

❒ **Interprétation** – L'interprétation $\mathcal{I}(f,w)$ outputs whether model w satisfies formula f :

$$\boxed{\mathcal{I}(f,w) \in \{0,1\}}$$

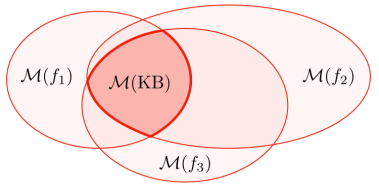
❒ **Ensemble de modèles** – $\mathcal{M}(f)$ dénote l'ensemble des modèles w qui satisfont la formule f. Sa définition mathématique est donnée par :

$$\boxed{\forall w \in \mathcal{M}(f), \quad \mathcal{I}(f,w) = 1}$$

Base de connaissance

❒ **Définition** – La base de connaissance KB est la conjonction de toutes les formules considérées jusqu'à présent. L'ensemble des modèles de la base de connaissance est l'intersection de l'ensemble des modèles satisfaisant chaque formule. En d'autres termes :

$$\boxed{\mathcal{M}(\text{KB}) = \bigcap_{f \in \text{KB}} \mathcal{M}(f)}$$



❒ **Interprétation en termes de probabilités** – La probabilité que la requête f soit évaluée à 1 peut être vue comme la proportion des modèles w de la base de connaissance KB qui satisfait f, i.e. :

$$\boxed{P(f|\text{KB}) = \frac{\displaystyle\sum_{w \in \mathcal{M}(\text{KB}) \cap \mathcal{M}(f)} P(W = w)}{\displaystyle\sum_{w \in \mathcal{M}(\text{KB})} P(W = w)}}$$

❒ **Satisfaisabilité** – La base de connaissance KB est dite satisfaisable si au moins un modèle w satisfait toutes ses contraintes. En d'autres termes :

$$\boxed{\text{KB satisfaisable} \iff \mathcal{M}(\text{KB}) \neq \varnothing}$$

Remarque : $\mathcal{M}(KB)$ dénote l'ensemble des modèles compatibles avec toutes les contraintes de la base de connaissance.

❒ **Relation entre formules et base de connaissance** – On définit les propriétés suivantes entre la base de connaissance KB et une nouvelle formule f :

Nom	Formulation mathématique	Illustration	Notes
KB déduit f	$\mathcal{M}(\text{KB}) \cap \mathcal{M}(f) = \mathcal{M}(\text{KB})$		- f n'apporte aucune nouvelle information - Aussi écrit KB $\models f$
KB contredit f	$\mathcal{M}(\text{KB}) \cap \mathcal{M}(f) = \varnothing$		- Aucun modèle ne satisfait les contraintes après l'ajout de f - Équivalent à KB $\models \neg f$
f est contingent à KB	$\mathcal{M}(\text{KB}) \cap \mathcal{M}(f) \neq \varnothing$ et $\mathcal{M}(\text{KB}) \cap \mathcal{M}(f) \neq \mathcal{M}(\text{KB})$		- f ne contredit pas KB - f ajoute une quantité d'information non triviale à KB

❒ **Vérification de modèles** – Un algorithme de vérification de modèles (*model checking* en anglais) prend comme argument une base de connaissance KB et nous renseigne si celle-ci est satisfaisable ou pas.

Remarque : DPLL et WalkSat sont des exemples populaires d'algorithmes de vérification de modèles.

❒ **Règle d'inférence** – Une règle d'inférence de prémisses $f_1,...,f_k$ et de conclusion g s'écrit :

$$\boxed{\frac{f_1,...,f_k}{g}}$$

❒ **Algorithme de chaînage avant** – Partant d'un ensemble de règles d'inférence Rules, l'algorithme de chaînage avant (en anglais *forward inference algorithm*) parcourt tous les $f_1,...,f_k$ et ajoute g à la base de connaissance KB si une règle parvient à une telle conclusion. Cette démarche est répétée jusqu'à ce qu'aucun autre ajout ne puisse être fait à KB.

❒ **Dérivation** – On dit que KB dérive f (noté KB $\vdash f$) par le biais des règles Rules soit si f est déjà dans KB ou si elle se fait ajouter pendant l'application du chaînage avant utilisant les règles Rules.

❒ **Propriétés des règles d'inférence** – Un ensemble de règles d'inférence Rules peut avoir les propriétés suivantes :

Name	Formulation mathématique	Notes
Validité	$\{f : \text{KB} \vdash f\} \subseteq \{f : \text{KB} \models f\}$	- Les formules inférées sont déduites par KB - Peut être vérifiée une règle à la fois - "*Rien que la vérité*"
Complétude	$\{f : \text{KB} \vdash f\} \supseteq \{f : \text{KB} \models f\}$	- Les formules déduites par KB sont soit déjà dans la base de connaissance, soit inférées de celle-ci - "*La vérité dans sa totalité*"

Logique propositionnelle

Dans cette section, nous allons parcourir les modèles logiques utilisant des formules logiques et des règles d'inférence. L'idée est de trouver le juste milieu entre expressivité et efficacité.

❒ **Clause de Horn** – En notant $p_1,...,p_k$ et q des symboles propositionnels, une clause de Horn s'écrit :

$$\boxed{(p_1 \wedge ... \wedge p_k) \longrightarrow q}$$

*Remarque : quand q = *false*, cette clause de Horn est "négative", autrement elle est appelée "stricte".*

❒ **Modus ponens** – Sur les symboles propositionnels $f_1,...,f_k$ et p, la règle de modus ponens est écrite :

$$\boxed{\frac{f_1,...,f_k, \quad (f_1 \wedge ... \wedge f_k) \longrightarrow p}{p}}$$

Remarque : l'application de cette règle se fait en temps linéaire, puisque chaque exécution génère une clause contenant un symbole propositionnel.

❒ **Complétude** – Modus ponens est complet lorsqu'on le munit des clauses de Horn si l'on suppose que KB contient uniquement des clauses de Horn et que p est un symbole propositionnel qui est déduit. L'application de modus ponens dérivera alors p.

❒ **Forme normale conjonctive** – La forme normale conjonctive (en anglais *conjunctive normal form* ou *CNF*) d'une formule est une conjonction de clauses, chacune d'entre elles étant une disjonction de formules atomiques.

Remarque : en d'autres termes, les CNFs sont des $\wedge$ de $\vee$.

❒ **Représentation équivalente** – Chaque formule en logique propositionnelle peut être écrite de manière équivalente sous la forme d'une formule CNF. Le tableau ci-dessous présente les propriétés principales permettant une telle conversion :

Nom de la règle		Début	Résultat
Élimine	$\leftrightarrow$	$f \leftrightarrow g$	$(f \rightarrow g) \wedge (g \rightarrow f)$
	$\rightarrow$	$f \rightarrow g$	$\neg f \vee g$
	$\neg\neg$	$\neg\neg f$	f
Distribue	$\neg$ sur $\wedge$	$\neg(f \wedge g)$	$\neg f \vee \neg g$
	$\neg$ sur $\vee$	$\neg(f \vee g)$	$\neg f \wedge \neg g$
	$\vee$ sur $\wedge$	$f \vee (g \wedge h)$	$(f \vee g) \wedge (f \vee h)$

❒ **Règle de résolution** – Pour des symboles propositionnels $f_1,...,f_n$, et $g_1,...,g_m$ ainsi que p, la règle de résolution s'écrit :

$$\boxed{\frac{f_1 \vee ... \vee f_n \vee p, \quad \neg p \vee g_1 \vee ... \vee g_m}{f_1 \vee ... \vee f_n \vee g_1 \vee ... \vee g_m}}$$

Remarque : l'application de cette règle peut prendre un temps exponentiel, vu que chaque itération génère une clause constituée d'une partie des symboles propositionnels.

❒ **Inférence basée sur la règle de résolution** – L'algorithme d'inférence basée sur la règle de résolution se déroule en plusieurs étapes :

— Étape 1 : Conversion de toutes les formules vers leur forme CNF

— Étape 2 : Application répétée de la règle de résolution

— Étape 3 : Renvoyer "non satisfaisable" si et seulement si False est dérivé

Calcul des prédicats du premier ordre

L'idée ici est d'utiliser des variables et ainsi permettre une représentation des connaissances plus compacte.

❒ **Modèle** –Un modèle w en calcul des prédicats du premier ordre lie :

— des symboles constants à des objets

— des prédicats à n-uplets d'objets

❒ **Clause de Horn** – En notant x_1,...,x_n variables et a_1,...,a_k,b formules atomiques, une clause de Horn pour le calcul des prédicats du premier ordre a la forme :

$$\boxed{\forall x_1,...,\forall x_n, \quad (a_1 \wedge ... \wedge a_k) \rightarrow b}$$

❒ **Substitution** – Une substitution θ lie les variables aux termes et Subst(θ,f) désigne le résultat de la substitution θ sur f.

❒ **Unification** – Une unification prend deux formules f et g et renvoie la substitution θ la plus générale les rendant égales :

$$\boxed{\text{Unify}[f,g] = \theta} \quad \text{t.q.} \quad \boxed{\text{Subst}[\theta,f] = \text{Subst}[\theta,g]}$$

Note : Unify[f,g] renvoie Fail si un tel θ n'existe pas.

❒ **Modus ponens** – En notant x_1,...,x_n variables, a_1,...,a_k et a'_1,...,a'_k formules atomiques et en notant $\theta = \text{Unify}(a'_1 \wedge ... \wedge a'_k, a_1 \wedge ... \wedge a_k)$, modus ponens pour le calcul des prédicats du premier ordre s'écrit :

$$\boxed{\frac{a'_1,...,a'_k \quad \forall x_1,...,\forall x_n (a_1 \wedge ... \wedge a_k) \rightarrow b}{\text{Subst}[\theta, b]}}$$

❒ **Complétude** – Modus ponens est complet pour le calcul des prédicats du premier ordre lorsqu'il agit uniquement sur les clauses de Horn.

❒ **Règle de résolution** – En notant $f_1, ..., f_n$, $g_1, ..., g_m$, p, q formules et en posant $\theta = \text{Unify}(p,q)$, le règle de résolution pour le calcul des prédicats du premier ordre s'écrit :

$$\boxed{\frac{f_1 \vee ... \vee f_n \vee p, \quad \neg q \vee g_1 \vee ... \vee g_m}{\text{Subst}[\theta, f_1 \vee ... \vee f_n \vee g_1 \vee ... \vee g_m]}}$$

❒ **Semi-décidabilité** – Le calcul des prédicats du premier ordre, même restreint aux clauses de Horn, n'est que semi-décidable.

— si KB $\models f$, l'algorithme de chaînage avant sur des règles d'inférence complètes prouvera f en temps fini

— si KB $\not\models f$, aucun algorithme ne peut le prouver en temps fini