Sémantique pour un langage polyadique

Marianna Girlando

Résumé

Notes de cours pour la première partie du cours Logique 2 S4 (2019 - 2020). Attention : ces notes ne suffisent pas à la préparation de l'examen, et sont à intégrer avec la lecture du livre [1] (parties des chapitres 9, 10, 11 ; chapitre 12) et les exercices vus pendant les séances de Travaux Dirigés.

1 Le langage polyadique $\mathcal{L}_p$

Le langage polyadique $\mathcal{L}_p$ est un langage pour la logique du premier ordre. À différence de $\mathcal{L}_m$, langage monadique pour la logique du premier ordre, $\mathcal{L}_p$ contient des prédicats à plusieurs places, pour exprimer des *relations*.

L'*alphabet* de $\mathcal{L}_p$ se compose de :

- Signes logiques :
  1. Connecteurs propositionnels : $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
  2. Quantificateurs : $\forall$, $\exists$
  3. Un ensemble infini dénombrable, $\mathsf{Var}(\mathcal{L}_p)$, de variables d'individu : $x, y, z, \ldots$
- Singes auxiliaires : parenthèses), (
- Signes non logiques :
  1. un ensemble, éventuellement vide, de constantes d'individu : $a, b, c, \ldots$
  2. Symboles de prédicat à n places, pour $n > 0$: P^1, Q^2, R^2, $\ldots$
     L'index indique le nombre de places, ou l'*arité*, du symbole de prédicat : P^n est un prédicat à n place, ou n-aire [1]. Il doit y avoir au moins un symbole de prédicat.

Une *expression* de $\mathcal{L}_p$ est une suite finie quelconque de symboles de l'alphabet de $\mathcal{L}_p$. On peut maintenant définir l'ensemble des termes et formules de $\mathcal{L}_p$.

Définition 1.1. L'ensemble $\mathsf{Ter}(\mathcal{L}_p)$ des *termes* de $\mathcal{L}_p$ est défini par induction.

- Une variable est un terme.
- Une constante d'individu est un terme.
- Rien d'autre est un terme.

On dénote les termes de $\mathcal{L}_p$ par $t_1, t_2, t_3, \ldots$.

1. Pour la suite, on va omettre l'index pour l'arité du prédicat quand cette information n'est pas relevante.

Définition 1.2. L'ensemble $\mathsf{For}(\mathcal{L}_p)$ de *formules* de $\mathcal{L}_p$ est défini inductivement à partir des formules atomiques.

— Soit P^n un symbole de prédicat à n places, et $t_1, \ldots, t_n$ termes de $\mathcal{L}_p$. Alors

$$P^n(t_1, \ldots, t_n)$$

est une formule atomique de $\mathcal{L}_p$.

— Soient ϕ, ψ formules de $\mathcal{L}_p$. Alors $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \to \psi$, $\phi \leftrightarrow \psi$ sont des formules.

— Soit ϕ une formule, et x une variable. Alors $\forall x\phi$ et $\exists x\phi$ sont des formules.

— Rien d'autre est une formule.

On dénote les formules de $\mathcal{L}_p$ par lettres minuscules de l'alphabet grec : ϕ, ψ, χ, ...

L'intérêt de donner une définition par induction pour un ensemble d'objets est de pouvoir raisonner par induction sur cet ensemble. C'est à dire, on peut utiliser la stratégie suivante pour démontrer qu'une certaine propriété P est satisfaite par toutes les formules (un principe similaire vaut pour les termes).

Principe d'induction sur l'ensemble de formules de $\mathcal{L}_p$. Soit P une propriété des formules. Si on vérifie que :

1. P est satisfaite par les formules atomiques (*base de l'induction*) ;
2. Pour ϕ et ψ formules de $\mathcal{L}_p$, si P est satisfaite par ϕ et ψ (*hypothèse d'induction*), alors P est satisfaite par $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \to \psi$ et $\phi \leftrightarrow \psi$.

Alors P vaut pour l'ensemble des formules de $\mathcal{L}_p$.

1.1 Variables libres, variables liées et substitution

On reprend les notions de variables libres et liées vues pour le langage monadique $\mathcal{L}_m$.

Une occurrence d'une variable x dans une formule ϕ est :

— *liée* par un quantificateur si x figure dans une sous-formule de ϕ de la forme $\exists x\psi$ ou $\forall x\psi$;

— *libre* dans ϕ si elle n'est pas liée.

Une variable est *libre dans* ϕ si elle a au moins une occurrence libre dans ϕ. Une variable est *liée dans* ϕ si elle a au moins une occurrence liée. Notez que la même variable peut être à la fois libre et liée dans une formule : par exemple, dans $\forall(P(x)) \wedge F(x)$ la première occurrence de x est liée, la deuxième est libre.

Si ϕ ne contient aucune variable libre, on dit que ϕ est un *énoncé*, ou une *formule close*.

On définit maintenant la notion de substitution, qui nous servira pour la suite. Dans un énoncé, on peut toujours remplacer d'une façon uniforme toutes les occurrences (liées) d'une variable avec une autre variable, en prenant soin que, après le remplacement, le même nombre de variables figure dans l'énoncé. Qu'est-ce qu'il se passe quand on veut définir la même opération pour une formule ? Plus précisément, qu'est-ce qu'il se passe si on veut substituer les occurrences *libres* d'une variables dans une formule ? On définit l'opération suivante :

Définition 1.3. Soit ϕ une formule avec une variable libre, x, et t un terme. On dénote par

$$\phi[t/x]$$

la formule obtenue à partir de ϕ lorsque in remplace toutes les occurrences libres de x dans ϕ par le terme s.

Exemple 1.1. Soit ϕ la formule suivante :

$$\forall x \exists y(M(x,y)) \rightarrow \exists y(M(x,y))$$

où le prédicat à deux places $M(x,y)$ signifie "y est strictement supérieur à x" et les variables rangent sur l'ensemble des nombres naturels $\mathbb{N}$. Donc ϕ affirme que "si pour tous les nombres x, il existe un nombre y tel que y est supérieur de x, alors il existe toujours un nombre y qui est supérieur d'un autre nombre".

La substitution suivante, avec la constante s, signifiant le nombre 7, ne cause pas problèmes : la conclusion qu'on tire est qu'il existe un nombre strictement supérieur à 7.

$$\phi[s/x] \; = \; \forall x \exists y(M(x,y)) \rightarrow \exists y(M(s,y)). \tag{1}$$

La substitution des occurrences libres de x avec la variable y change la signification de ϕ : on devrait conclure qu'il existe un nombre strictement supérieur à soi-même !

$$\phi[y/x] \; = \; \forall x \exists y(M(x,y)) \rightarrow \exists y(M(y,y)) \tag{2}$$

Pour éviter tout problème il vaut mieux, quand possible, de choisir pour la substitution $\phi[z/x]$ une variable z qui ne figure pas dans ϕ. Si ce n'est pas possible, on peut procéder avec un renommage des variables liées dans ϕ avec des variables différentes de z et qui ne figurent pas dans ϕ, et *après* procéder avec la substitution $\phi[z/x]$.

Facultatif. Formellement, on dit que une variable y est substituable à x dans ϕ s'il n'y a pas d'occurrence libre de x dans sous-formules de ϕ de la forme $\forall y\psi$ ou $\exists y\psi$. Dans la formule (2) de l'exemple précédent, y n'est pas substituable à x dans ϕ, vu qu'il y a une occurrence libre de x dans la sous-formule $\exists y(M(x,y))$.

2 Sémantique pour $\mathcal{L}_p$

Quand on parle de *structure*, on entend un ensemble non-vide d'objets sur lequel on définit une relation. Pour exemple, $\langle \mathbb{N}, \leq \rangle$ est la relation formé par l'ensemble des naturels et la relation à deux places $\leq$ entre les nombres. Un autre exemple de structure est l'ensemble de personnages des *Misérables* de Victor Hugo, avec la relation d'amitié ou de parenté entre les personnages.

La sémantique de Tarski pour la logique du premier ordre est définie par apport à un langage et une structure, dit *structure d'interprétation*. Dans notre cas, le langage qu'on va considérer est un langage polyadique pour la logique du premier ordre, qu'on va dénoter par $\mathcal{L}$. La sémantique assigne à chaque élément du langage $\mathcal{L}$ un valeur

dans la structure. L'objectif est la définition de la notion suivante, pour $\mathcal{I}$ structure d'interprétation et ϕ énoncé :

$$\mathcal{I} \models \phi$$

qu'on lit "ϕ est valide dans $\mathcal{I}$", ou "$\mathcal{I}$ est modèle de ϕ". La notion de validité d'un énoncé peut être interprété comme exprimant la "vérité" de l'énoncé ; mais la "vérité" dont on parle ici est une notion formelle, définie par apport à une structure.

Pour définir la validité d'un énoncé, il faut d'abord assigner une valeur à toutes les formules du langage, y compris les formules avec variables libres. Pour ça, on définit la relation de satisfaction.

2.1 Structures d'interprétation et satisfaction d'une formule

Définition 2.1. Une *structure d'interprétation* $\mathcal{I}$ pour $\mathcal{L}$, appelée aussi $\mathcal{L}$*-structure*, est composé par :

1. Un ensemble non-vide d'éléments, le *domaine*, dénoté par D.
2. Pour chaque prédicat n-aire de $\mathcal{L}$, une relation n-aire $P^{\mathcal{I}} \subseteq D^n$.
3. Pour chaque constante d'individu c, un élément $c^{\mathcal{I}} \in D$.

Exemple 2.1. À chaque prédicat unaire P du langage est associé une partie $P^{\mathcal{I}} \subseteq D$, c'est-à-dire, la partie des éléments de D qui ont la propriété décrite par P.
À chaque prédicat binaire R du langage est associé une relation binaire $P^{\mathcal{I}} \subseteq D^2$. Une relation binaire est un ensemble de couples ordonnées, et donc une partie de l'ensemble $D \times D$, qu'on dénote par D^2 (cet ensemble est le *produit cartésien* de D avec soi même).
À chaque prédicat binaire R est associé l'ensemble de couples ordonnées d'éléments de D qui sont dans la relation décrite par R.

Pour assigner une valeur aux formules du langages du premier ordre, on a besoin de savoir quelle valeur assigner aux termes du langage, en particulier aux variables. Pour ça, on définit les notions d'*assignation* et d'*interprétation des termes selon une assignation.*

Définition 2.2. Soit $\mathcal{I}$ une $\mathcal{L}$-structure. Une *assignation* σ est une fonction qui associe à chaque variable v de $\mathcal{L}_p$ un élément $\sigma(v)$ du domaine D :

$$\sigma : \mathsf{Var}(\mathcal{L}_p) \to D.$$

Définition 2.3. Soit $\mathcal{I}$ une $\mathcal{L}$-structure et σ une assignation. Une *interprétation des termes selon* σ est une fonction i qui assigne à chaque terme t de $\mathcal{L}$ un élément $i(t)$ du domaine D. Si t est une constante, $i(t)$ est l'élément $t^{\mathcal{I}}$ de D. Si t est une variable, $i(t)$ est l'élément du domaine $\sigma(t)$ indiqué par σ.

$$i : \mathsf{Ter}(\mathcal{L}_p) \to D \qquad\qquad i(t) = \begin{cases} t^{\mathcal{I}} & \text{pour } t \text{ constante} \\ \sigma(t) & \text{pour } t \text{ variable} \end{cases}$$

Pour assigner une valeur aux formules avec quantificateurs, on a besoin de définir la notion de *variante* d'une assignation.

Définition 2.4. Soit $\mathcal{I}$ une $\mathcal{L}$-structure, σ une assignation, x une variable et d un élément du domaine. Une *variante* de σ en x, dénoté par

$$\sigma\,\frac{x}{d}$$

est la *seule* assignation qui prend la valeur d sur x et assigne à toutes les autres variables la même valeur que σ.

On a enfin tous les éléments pour définir la relation de satisfaction.

Définition 2.5. Soit $\mathcal{I}$ une $\mathcal{L}$-structure, σ une assignation et ϕ une formule. On définit par induction la relation de satisfaction de ϕ en $\mathcal{I}$ selon σ, en symboles :

$$\mathcal{I} \models \phi \quad [\sigma]$$

(lire : "σ satisfait ϕ dans $\mathcal{I}$").

1. $\mathcal{I} \models P(t_1, \ldots, t_n) \quad [\sigma]$ ssi $(i(t_1), \ldots, i(t_n)) \in P^{\mathcal{I}}$,
   c'est-à-dire, ssi la relation $P^{\mathcal{I}}$ subsiste entre les éléments $i(t_1), \ldots, i(t_n)$;
2. $\mathcal{I} \models \neg\phi \quad [\sigma]$ ssi $\mathcal{I} \not\models \phi \quad [\sigma]$;
3. $\mathcal{I} \models \phi \wedge \psi \quad [\sigma]$ ssi $\mathcal{I} \models \phi \quad [\sigma]$ et $\mathcal{I} \models \psi \quad [\sigma]$;
4. $\mathcal{I} \models \phi \vee \psi \quad [\sigma]$ ssi $\mathcal{I} \models \phi \quad [\sigma]$ ou $\mathcal{I} \models \psi \quad [\sigma]$;
5. $\mathcal{I} \models \phi \to \psi \quad [\sigma]$ ssi $\mathcal{I} \not\models \phi \quad [\sigma]$ ou $\mathcal{I} \models \psi \quad [\sigma]$;
6. $\mathcal{I} \models \forall x\phi \quad [\sigma]$ ssi pour toute variante σ' de σ en x, $\mathcal{I} \models \phi \quad [\sigma']$;
7. $\mathcal{I} \models \exists x\phi \quad [\sigma]$ ssi il existe une variante σ' de σ en x t. q. $\mathcal{I} \models \phi \quad [\sigma']$.

Les conditions pour les quantificateurs peuvent être définies d'une façon équivalente, en explicitant la variante :

6'. $\mathcal{I} \models \forall x\phi \quad [\sigma]$ ssi pour tout $d \in D$, $\mathcal{I} \models \phi \quad \left[\sigma\,\frac{x}{d}\right]$;
7'. $\mathcal{I} \models \exists x\phi \quad [\sigma]$ ssi il existe un $d \in D$ tel que $\mathcal{I} \models \phi \quad \left[\sigma\,\frac{x}{d}\right]$.

2.2 Vérité d'un énoncé dans une structure

On vient de définir la relation de *satisfaction* pour ϕ formule quiconque du langage. La relation de satisfaction est définie par apport à une $\mathcal{L}$-structure $\mathcal{I}$ et une assignation σ. On s'occupe maintenant de définir la notion de vérité d'un *énoncé* dans une $\mathcal{L}$-structure. Pour ça, nous montrons que la satisfaction d'un énoncé ne dépend d'aucune assignation particulière à ses variables.

Théorème 2.1 (Coïncidence)**.** Soit ϕ une formule de $\mathcal{L}$ et $\mathcal{I}$ une structure d'interprétation. Soient σ_1 et σ_2 deux assignations qui assignent les mêmes valeurs à (coïncident sur) toutes les *variables libres* dans ϕ. Alors on a que :

$$\mathcal{I} \models \phi \quad [\sigma_1] \quad \text{ssi} \quad \mathcal{I} \models \phi \quad [\sigma_2].$$

Démonstration (Facultative). La démonstration se fait par induction sur l'ensemble de formules du langage. On définit deux fonctions d'interprétation des termes, une selon σ_1 et une selon σ_2.

$$i_1(t) = \begin{cases} t^{\mathcal{I}} & \text{pour } t \text{ constante} \\ \sigma_1(t) & \text{pour } t \text{ variable} \end{cases} \qquad i_2(t) = \begin{cases} t^{\mathcal{I}} & \text{pour } t \text{ constante} \\ \sigma_2(t) & \text{pour } t \text{ variable} \end{cases}$$

L'élément du domaine associé à chaque constante du langage est le même dans i_1 et i_2. Pour le cas de base de l'induction, on doit montrer que

$$\mathcal{I} \models P(t_1, \ldots, t_n) \quad [\sigma_1] \quad \text{ssi} \quad \mathcal{I} \models P(t_1, \ldots, t_n) \quad [\sigma_2].$$

On suppose que $\mathcal{I} \models P(t_1, \ldots, t_n) \quad [\sigma_1]$. Par définition, on a que $(i_1(t_1), \ldots, i_1(t_n)) \in P^{\mathcal{I}}$. Toutes les termes qui figurent dans $t_1, \ldots, t_n$ sont des constantes ou des variables libres. Dans les deux cas, on a que i_1 et i_2 assignent la même valeurs aux termes, c'est à dire, $i_1(t_1) = i_2(t_1), \ldots, i_1(t_n) = i_2(t_n)$. Donc on a que $(i_2(t_1), \ldots, i_2(t_n)) \in P^{\mathcal{I}}$, et que $\mathcal{I} \models P(t_1, \ldots, t_n) \quad [\sigma_2]$. La preuve de l'autre direction est similaire.
Il faut maintenant vérifier que, pour ϕ formule quiconque,

$$(*) \qquad \mathcal{I} \models \phi \quad [\sigma_1] \quad \text{ssi} \quad \mathcal{I} \models \phi \quad [\sigma_2].$$

On montre que, en supposant que la propriété $(*)$ soit satisfaite par ψ, χ formules (hypothèse d'induction), alors la propriété est satisfaite par $\phi = \neg\psi$, $\phi = \psi \wedge \chi$, $\phi = \psi \vee \chi$, $\phi = \psi \to \chi$, $\phi = \exists x\psi$, et $\phi = \forall x\psi$. On montre seulement le cas où $\phi = \forall x\psi$ (la démonstration pour les autres cas se trouve à page 257 de [1]). On veut montrer que :

$$\mathcal{I} \models \forall x\psi \quad [\sigma_1] \quad \text{ssi} \quad \mathcal{I} \models \forall x\psi \quad [\sigma_2].$$

L'hypothèse d'induction nous dit que :

$$\mathcal{I} \models \psi \quad [\sigma_1] \quad \text{ssi} \quad \mathcal{I} \models \psi \quad [\sigma_2].$$

On suppose que $\mathcal{I} \models \forall x\psi \; [\sigma_1]$. Donc, pour tout $d \in D$, on a que $\mathcal{I} \Vdash \psi \left[\sigma_1 \, \frac{x}{d}\right]$. Par hypothèse d'induction, on a que $\mathcal{I} \Vdash \psi \left[\sigma_2 \, \frac{x}{d}\right]$ (parce que : par hypothèse d'induction σ_1 et σ_2 satisfont ψ dans $\mathcal{I}$, en assignant la même valeur aux variables libres dans ψ. Les variantes de σ_1 et σ_2 assignent la même valeur (d) à la variable liée x. Donc $\mathcal{I} \Vdash \psi \left[\sigma_1 \, \frac{x}{d}\right]$ ssi $\mathcal{I} \Vdash \psi \left[\sigma_2 \, \frac{x}{d}\right]$). On peut conclure que $\mathcal{I} \models \forall x\psi \; [\sigma_2]$. La preuve de l'autre direction est similaire. □

Le théorème de coïncidence nous dit que la relation de satisfaction d'une formule à une $\mathcal{L}$-structure dépend seulement des valeurs qu'on assigne aux variables *libres* de la formule. En conséquence, la satisfaction d'un énoncé (qui n'a pas de variables libres) ne dépend pas d'une assignation particulière aux variables, mais du domaine des individus. Autrement dit, les assignation sont *indifférentes* pour un énoncé : si ϕ est un énoncé, ϕ est satisfait par une assignation ssi ϕ est satisfait par toute assignation.

Définition 2.6. Un énoncé ϕ est *vrai* (ou *valide*) dans une $\mathcal{L}$-structure $\mathcal{I}$, en symboles

$$\mathcal{I} \models \phi$$

s'il existe une assignation σ qui satisfait ϕ dans $\mathcal{I}$ (en symboles $\mathcal{I} \models \phi \quad [\sigma]$) ou, de manière équivalente, si pour toute assignation σ, $\mathcal{I} \models \phi \quad [\sigma]$. On dit aussi que $\mathcal{I}$ est *modèle* de ϕ.
Une $\mathcal{L}$-structure $\mathcal{I}$ est *modèle* d'un ensemble d'énoncés Γ, notation $\mathcal{I} \models \Gamma$, si, pour tout $\phi \in \Gamma$, $\mathcal{I}$ est modèle de ϕ.

2.3 Satisfaisabilité, validité et conséquence logique

Grace à la définition de la relation de satisfaction, on peut donner les définitions des autres notions logiques fondamentales. Notez que ces définition sont données d'une façon générale, et s'appliquent soit aux formules (ouvertes) qu'aux énoncés du langage.

Définition 2.7 (Satisfaisabilité)**.**
Une formule est *satisfaisable* si, et seulement si, il existe une $\mathcal{L}$-structure $\mathcal{I}$ qui satisfait ϕ dans $\mathcal{I}$, en symboles : $\mathcal{I} \models \phi \quad [\sigma]$.
Un énoncé est *satisfaisable* si, et seulement si, il existe une $\mathcal{L}$-structure $\mathcal{I}$ qui est modèle de ϕ, en symboles : $\mathcal{I} \models \phi$.
Un ensemble d'énoncés Γ est *satisfaisable* si, et seulement si,il existe une $\mathcal{L}$-structure $\mathcal{I}$ qui est modèle de Γ, en symboles : $\mathcal{I} \models \Gamma$.

Définition 2.8 (Contradiction)**.** Un énoncé ϕ est *invalide*, ou *contradictoire* si, et seulement si, il n'est pas satisfaisable (aucune structure n'est modèle de ϕ).
Un ensemble d'énoncés Γ est *contradictoire* si, et seulement si, il n'est pas satisfaisable (aucune structure n'est modèle de Γ).

Définition 2.9 (Validité)**.**
Une formule est *valide* si, et seulement si, dans toute $\mathcal{L}$-structure, toute assignation satisfait ϕ.
Un énoncé est *valide* si, et seulement si, toute $\mathcal{L}$-structure est modèle de ϕ.
On note la validité d'une formule ou énoncé ϕ comme :

$$\models \phi.$$

Attention : on écrit $\not\models \phi$ pour signifier que ϕ n'est pas valide, pas que ϕ est invalide.

Définition 2.10 (Conséquence logique)**.**
Un énoncé ϕ est *conséquence logique* d'un ensemble Γ d'énoncés si, et seulement si, tout modèle de Γ est modèle de ϕ, ou, de manière équivalente, dans tout modèle de Γ, ϕ est valide. En symboles :

$$\Gamma \models \phi.$$

On écrit $\not\models \phi$ pour signifier que ϕ n'est pas conséquence valide de Γ.

Un énoncé ϕ est conséquence logique de l'ensemble vide, $\emptyset \models \phi$, si et seulement si ϕ est valide, $\models \phi$. On suppose que $\emptyset \models \phi$. Par définition, pour toute $\mathcal{L}$-structure $\mathcal{I}$ qui est modèle de tous les énoncés dans $\emptyset$, $\mathcal{I}$ est modèle de ϕ. Vu que toute $\mathcal{L}$-structure est modèle d'un ensemble vide d'énoncés, toute $\mathcal{L}$-structure est modèle de ϕ ; donc, ϕ est valide. Pour démontrer l'autre direction, on suppose que ϕ est valide. Alors ϕ est conséquence logique d'un ensemble vide d'énoncés.

Facultatif. On note les trois propriétés suivantes de la relation de conséquence logique :

1. La relation de conséquence logique est *réflexive* : pour toute formule ϕ, $\phi \models \phi$.
2. La relation de conséquence logique est *transitive* : pour toutes formules ϕ, ψ, χ, si $\phi \models \psi$ et $\psi \models \chi$, alors $\phi \models \chi$.

3. La relation de conséquence logique est *monotone*. Soit Γ un ensemble d'énoncés, et soient ϕ, ψ énoncés. Si $\Gamma \models \phi$, alors $\Gamma \cup \{\psi\} \models \phi$.

La démonstration de 1 est immédiate. Pour prouver 2, il suffit d'observer que, par définition, tout modèle de ϕ est modèle de ψ, et tout modèle de ψ est modèle de χ. Donc, tout modèle de ϕ est modèle de χ, et par définition on a que $\phi \models \chi$. Démonstration de 3 : on assume que $\Gamma \models \phi$. Par définition, on a que $(*)$ tout modèle de Γ est modèle de ϕ. Vu que $\Gamma \subseteq \Gamma \cup \{\psi\}$, on a que $\Gamma \cup \{\psi\}$ est modèle de Γ (réflexivité). Donc, pour $(*)$, on a que $\Gamma \cup \{\psi\}$ est modèle de ϕ, et on peut conclure que $\Gamma \cup \{\psi\} \models \phi$.

2.4 Ensembles définissables et isomorphisme de structures

Les formules à une variable libre peuvent être utilisés pour dénoter *ensembles* dans une structure. Plus précisément, une formule à une variable libre définit l'ensemble des éléments du domaine qui satisfont la formule, c'est à dire, qui ont la propriété spécifié par la formule.

Soit $\mathcal{I}$ une $\mathcal{L}$-structure, D son domaine et σ une assignation. On dit qu'un ensemble E est *définissable* dans $\mathcal{I}$ s'il existe une formule ϕ à une variable libre, y, telle que :

$$E = \left\{ d \in D \mid \mathcal{I} \models \phi \quad \left[\sigma \frac{y}{d}\right] \right\}.$$

On définit la notion d'*isomorphisme* entre deux $\mathcal{L}$-structures $\mathcal{I}_1$ et $\mathcal{I}_2$. Le mot *isomorphisme* vient de la langue grecque, et signifie "ayant la même forme". On s'attend donc que deux structures isomorphes aient la même *forme*, c'est à dire, le même nombre d'éléments, les mêmes propriétés et les mêmes relations entre éléments. Cette définition nous permet d'établir un lien entre la *forme* des structures et les énoncés valides dans les structures.

Définition 2.11. Soient $\mathcal{I}_1$ et $\mathcal{I}_2$ deux $\mathcal{L}$-structures. Une fonction $h : D_1 \to D_2$ est un *isomorphisme* de $\mathcal{I}_1$ vers $\mathcal{I}_2$ si :

— h est une bijection de D_1 vers D_2 [2] ;
— Pour toute constante d'individu a, $h(a^{\mathcal{I}_1}) = a^{\mathcal{I}_2}$;
— Pour tout symbole de prédicat P à n places, et pour tout $(d_1, \ldots, d_n) \in D_1^n$,

$$(d_1, \ldots, d_n) \in P^{\mathcal{I}_1} \text{ ssi } (h(d_1), \ldots, h(d_n)) \in P^{\mathcal{I}_2}$$

Deux structures $\mathcal{I}_1$ et $\mathcal{I}_2$ sont *isomorphes* s'il existe un isomorphisme de l'une vers l'autre, en symboles :

$$\mathcal{I}_1 \cong \mathcal{I}_2.$$

Définition 2.12. Soient $\mathcal{I}_1$ et $\mathcal{I}_2$ deux $\mathcal{L}$-structures. On dit que les deux structures sont *élémentairement équivalentes* si tout énoncé vrai dans l'une est vrai dans l'autre. Notation :

$$\mathcal{I}_1 \equiv \mathcal{I}_2.$$

2. Une *bijection* est une fonction *injective* et *surjective*. S'il existe une bijection entre deux ensembles, les deux ensembles ont le même nombre d'éléments.

Théorème 2.2. Si deux structures sont isomorphes, elles sont élémentairement équivalentes.

Pour démontrer que deux structures ne sont pas isomorphes, il suffit de trouver un énoncé qui est valide dans une mais pas dans l'autre.

3 Tester la validité d'un énoncé

En général, on dit *vérité logique* une formule dont la vérité ne dépend pas des faits ou de l'état des choses dans le monde. On appelle *tautologie* une vérité logique pour un langage propositionnel. On peut définir une tautologie comme une formule qui prend la valeur "vrai" sur toutes les lignes de sa table de vérité. On appelle *énoncé valide* une vérité logique pour un langage du premier ordre. Un *énoncé* est une formule qui n'a pas de variables libres ; un énoncé est dit *valide* si toute structure d'interprétation est modèle de l'énoncé. Dans ce sens, la vérité d'un énoncé valide ne dépend pas de l'interpretation (ou de l'assignation) choisie.

Tester la *satisfaisabilité* d'un énoncé revient à trouver un exemple : une structure d'interpretation $\mathcal{I}$ qui est modèle de l'énoncé. Similairement, pour démontrer qu'un énoncé n'est pas valide, il suffit de trouver un *contre-exemple*, c'est à dire, une $\mathcal{L}$-structure qui n'est pas modèle de l'énoncé (voir [1], pages 188-189, pour autres exemples).

Exemple 3.1. On cherche un contre-modèle pour l'énoncé :

$$\phi = (\exists x(P(x)) \wedge \exists y(Q(y))) \rightarrow \exists z(P(z) \wedge Q(z)).$$

Il faut chercher un modèle qui satisfait l'antécédent et ne satisfait pas le conséquent de l'implication. Donc, on définit une structure d'interpretation $\mathcal{I}$ comme suit. Le domaine est constitué de deux éléments, $D = \{1, 2\}$. On assigne au prédicat unaire P l'ensemble $P^{\mathcal{I}} = \{1\}$, et au prédicat unaire Q l'ensemble $Q^{\mathcal{I}} = \{2\}$. On vérifie que $\mathcal{I} \models \exists x(P(x))$, et $\mathcal{I} \models \exists y(Q(y))$. Donc $\mathcal{I} \models \exists x(P(x)) \wedge \exists y(Q(y))$. Mais $\mathcal{I} \not\models \exists z(P(z) \wedge (Q(z))$. Donc $\mathcal{I} \not\models \phi$.

On s'intéressé à tester si un énoncé du langage du premier ordre est *valide*. Une stratégie consiste à raisonner sur une $\mathcal{L}$-structure quelconque, et montrer que l'énoncé est vrai dans la structure. Si ϕ est vrai dans une structure quelconque, on peut conclure que toute $\mathcal{L}$-structure est modèle de ϕ [3].

Exemple 3.2. On teste la validité de l'énoncé :

$$\exists x(P(x)) \rightarrow (\forall y(R(y)) \rightarrow \exists z(P(z))).$$

Soit $\mathcal{I}$ une structure d'interpretation quelconque. On suppose qu'il existe un élément $e \in D$ tel que $e \in P^{\mathcal{I}}$. On suppose aussi que pour tous les éléments d du domaine, on a que $d \in P^{\mathcal{I}}$. On conclut immédiatement qu'il existe un élément $f \in D$ tel que $f \in P^{\mathcal{I}}$: il suffit de prendre $f = e$.

3. Si on veut tester la validité d'une formule avec variables libres, il faut aussi considérer une assignation. Voir [1], page 189, pour un exemple.

Observation 3.1 (*Facultative*)**.** La formule dont on a testé la validité est une *instance de tautologie*. Une instance de tautologie est une formule ayant la même "forme" d'une tautologie, c'est à dire, qui est obtenue en remplaçant chaque formule propositionnelle figurant dans la tautologie par une formule du langage di premier ordre. Dans l'exemple, la formule est une instance de la tautologie $p \to (q \to p)$ (affaiblissement). On peut démontrer que toute formule qui est instance d'une tautologie est valide dans la logique du premier ordre (voir [1], page 192-193, et page 259 pour la démonstration).

Une autre méthode peut tester la validité d'un énoncé consiste à utiliser les arbres de vérité. On va traiter ensemble les arbres pour le langage monadique et polyadique.

3.1 Les arbres de vérité pour le premier ordre

Comme pour le cas propositionnel, la méthode des arbres consiste à chercher un contre-exemple pour la formule ou l'énoncé ϕ dont on veut tester la validité. Donc, on suppose que $\neg\phi$ soit satisfaisable, et on développe l'arbre selon les règles. Si on trouve un arbre fermée, on conclut que $\neg\phi$ n'est pas satisfaisable ; et donc, on obtient que A est un énoncé valide. Si on trouve un arbre achevée mais ouvert, on conclut qu'il existe un modèle pour $\neg\phi$, qui est un contre-modèle pour ϕ. On utilise l'équivalence :

$$\phi \text{ est valide} \quad \text{ssi} \quad \neg\phi \text{ n'est pas satisfaisable .}$$

Pour définir la méthode des arbres pour le langage du premier ordre (cas monadique et polyadique) on ajoute aux règles propositionnelles (voir le Chapitre 6 de [1]) les règles pour les quantificateurs :

$\exists x\phi$ $\phi[a/x]$	$\neg\exists x\phi$ $\forall x\neg\phi$
$\neg\forall x\phi$ $\exists x\neg\phi$	$\forall x\phi$ $\phi[a/x]$

Dans les règles pour $\exists x\phi$ et $\forall x\phi$, la formule $\phi[a/x]$ est obtenue en remplaçant toutes les occurrences *libres* de x avec la constante a (voir Définition 1.3). Dans le cas de $\exists x\phi$, la constante doit être *nouvelle*, c'est à dire, elle ne doit pas avoir d'occurrences dans l'arbre au moment où on applique la règle. On peut donc supposer que le langage soit enrichi par une constante pour chaque règle existentielle qu'on applique. Au contraire, dans le cas de $\forall x\phi$, la constante a doit figurer dans la branche de l'arbre où $\forall x\phi$ figure. Il y a une exception : s'il n'y a pas de constantes dans la branche, et si la branche n'est pas fermée après application de toutes les autres règles, on peut appliquer la règle pour l'universel $\forall x\phi$ en introduisant une nouvelle constante [4]. S'il y a plusieurs constantes

4. Cette opération est justifié par le fait que le domaine de n'importe quelle structure est non-vide. Pour exemple, essayez de construire l'arbre de vérité de la formule $\exists x(P(x) \to \forall y P(y)$.

dans la branche, la règle pour $\forall x\phi$ doit être appliquée à toutes, même à ceux introduites après application de la règle.

Comme dans le cas propositionnel, on définit *fermée* une branche de l'arbre qui contient une formule ϕ et sa négation $\neg\phi$, et on définit *ouverte* une branche qui n'est pas fermée. Après application des règles aux formules $\exists x\phi$, $\neg\exists x\phi$ et $\neg\forall x\phi$, la formule à laquelle la règle a été appliquée peut être rayée (alternativement, on utilise le symbole ✓ à coté de la formule). Vu que la règle pour la formule $\forall x\phi$ doit être appliquée à *toutes* les constantes qui figurent dans la branche, on ne peut pas rayer la formule après application de la règle. Par apport au cas propositionnel, il faut donc modifier la notion d'arbre *achevé* : on dit que un arbre est dit achevé si

1. Toutes les formules non-élémentaires [5], sauf $\forall x\phi$, qui se trouvent sur une branche ouverte ont été rayées (ou le signe ✓ figure à coté d'elles) ;
2. Toutes les formules de forme $\forall x\phi$ qui se trouvent sur une branche ouverte ont été appliquées à *toutes* les constantes qui se trouvent sur la même branche que $\forall x\phi$.

Si, au but d'un certain nombre de pas, on obtient un arbre *fermé* (un arbre où toutes les branches sont fermées), la formule de départ n'est pas satisfaisable. Donc, si on avait essayé de dresser l'arbre pour une formule $\neg\phi$, on peut conclure que ϕ est *valide*. Si, au contraire, on obtient un arbre achevé mais *ouvert* (un arbre où on a appliqué toutes les règles, mais qui a des branches ouvertes), on conclut que la formule de départ est satisfaisable. Si notre formule de départ était $\neg\phi$, on conclut qu'il y a une structure qui satisfait $\neg\phi$, et donc que ϕ n'est pas valide.

À noter : la méthode des arbres peut être utilisée pour tester si une formule ou un énoncé ϕ est satisfaisable. Dans ce cas, on construit le tableaux pour ϕ et, si on obtient un arbre fermé, on conclut que ϕ n'est pas satisfaisable ; si l'arbre qu'on obtient est achevé et ouvert, ϕ est satisfaisable.

(Facultatif). Similairement à ce qu'on faisait pour la logique propositionnelle (voir page 103 de [1]), si on a un arbre achevée et ouvert pour une formule ψ, on peut construire un modèle pour ψ à partir des informations contenues dans une des branches ouvertes de l'arbre. Dans le cas $\psi = \neg\phi$, le modèle qu'on définit est un modèle pour $\neg\phi$, et donc un contre-modèle pour ϕ. Intuitivement, la stratégie pour construire un modèle à partir d'une branche ouverte est la suivante :

— Définir une structure (et éventuellement une assignation) qui rend vraies toutes les formules élémentaires dans la branche ;

— Définir une structure *minimale* : le domaine contient le moins d'éléments possible.

On donne une définition informelle de la méthode [6], sans démonstrations.

Définition 3.1 (*Facultative*)**.** Soit $\mathcal{T}$ un arbre achevée et ouvert pour un énoncé ψ, et soit $\mathcal{B}$ une branche ouverte (quelconque) de $\mathcal{T}$. Soient $e_1, \ldots, e_n$ les constantes qui figurent dans les formules dans $\mathcal{B}$. L'énoncé ϕ est satisfait dans la structure $\mathcal{I}$ définie de la façon suivante :

5. Une formule élémentaire est une formule atomique ou une négation d'une formule atomique.

6. On indique que une des stratégies possibles, et sans détailler le cas des variables libres. Voir [2], Chapitre 29 (en anglais), pour plus de détails.

— Le *domaine* de $\mathcal{I}$ est $D = \{e_1, \ldots, e_n\}$;
— Pour P prédicat unaire, on définit $P^{\mathcal{I}}$ comme l'ensemble d'éléments $d \in D$ tels que la formule atomique $P(d)$ figure dans $\mathcal{B}$;
— Pour R prédicat binaire, on définit $R^{\mathcal{I}}$ comme l'ensemble de couples (d_1, d_2) pour $d_1, d_2 \in D$ telles que la formule atomique $R(d_1, d_2)$ figure dans $\mathcal{B}$;
— Similairement pour les prédicats à plusieurs places.

Exemple 3.3 (Facultatif)**.** On veut tester la validité de la formule de l'Exemple 3.1 :

$$\phi \;=\; (\exists x(P(x)) \wedge \exists y(Q(y))) \rightarrow \exists z(P(z) \wedge Q(z)).$$

On construit l'arbre de vérité de $\neg\phi$.

$$\begin{array}{c}
\neg((\exists x(P(x)) \wedge \exists y(Q(y))) \rightarrow \exists z(P(z) \wedge Q(z))) \quad \checkmark \\
(\exists x(P(x)) \wedge \exists y(Q(y))) \quad \checkmark \\
\neg\exists z(P(z) \wedge Q(z)) \quad \checkmark \\
\exists x(P(x)) \quad \checkmark \\
\exists y(Q(y)) \quad \checkmark \\
P(a) \\
Q(b) \\
\neg(P(a) \wedge Q(a))\checkmark \\
\neg(P(b) \wedge Q(b))\checkmark \\
\diagup \quad \diagdown \\
\neg P(a) \qquad \neg Q(a) \\
\times \qquad\quad \diagup \quad \diagdown \\
\neg P(b) \quad \neg Q(b) \\
\circ \qquad\quad \times
\end{array}$$

La seule branche ouverte est celle marquée par $\circ$. Les constantes qui figurent dans la branche sont deux : a et b. On construit le modèle $\mathcal{I}$ pour $\neg\phi$ de la façon suivante. Le domaine est constitué par deux éléments : $D = \{a, b\}$. On assigne une valeur aux constantes : $a^{\mathcal{I}} = a$ et $b^{\mathcal{I}} = b$. On assigne un valeur aux prédicats : $P^{\mathcal{I}} = \{a\}$ et $Q^{\mathcal{I}} = \{b\}$. Le modèle obtenu est un modèle pour ϕ, et un contre-modèle pour $\neg\phi$.

4 Tester la validité d'une inférence

Une *inférence* est une relation entre une ou plusieurs prémisses et une conclusion. Un raisonnement est une succession d'inférences. Pour formaliser une inférence, on transcrit les prémisses (Γ) et la conclusion (ϕ) dans le langage (dans notre cas : langage pour la logique du premier ordre). Une inférence est *valide* si, et seulement si, la conclusion est *conséquence logique* des prémisses, c'est à dire si, et seulement si, toutes les structures d'interprétations qui rendent vraies les formules de Γ rendent vraie ϕ.

On s'intéressé à tester si un énoncé est conséquence logique d'un ensemble d'énoncés. Pour prouver qu'un énoncé ϕ n'est pas conséquence logique d'un ensemble Γ, il suffit de trouver un contre-exemple : une $\mathcal{L}$-structure qui est modèle de Γ mais n'est pas modèle de ϕ (voir [1], page 189, pour un exemple).

Pour démontrer qu'un énoncé ϕ est conséquence logique d'un ensemble d'énoncés Γ il faut montrer que pour tout modèle $\mathcal{I}$, si $\mathcal{I}$ est modèle de Γ, alors $\mathcal{I}$ est modèle de ϕ. Pour ça, on peut raisonner sur en considérant une structure $\mathcal{I}$ quelconque, faire l'hypothèse qu'elle satisfait Γ et montrer qu'elle satisfait ϕ.

Autrement, similairement au cas propositionnel, on peut tester la validité d'une inférence avec la méthode des arbres. On utilise le théorème suivant :

Théorème 4.1. Soit Γ un ensemble d'énoncés, et ϕ un énoncé. On a que $\Gamma \vDash \phi$ si et seulement si l'ensemble $\Gamma \cup \{\neg\phi\}$ est contradictoire.

Démonstration. On suppose que $\Gamma \vDash \phi$. Par définition, toute structure qui est modèle de Γ est modèle de ϕ ; donc, elle n'est pas modèle de ϕ, et aucune structure est modèle de $\Gamma \cup \{\neg\phi\}$. Pour l'autre direction, on suppose que $\Gamma \cup \{\neg\phi\}$ soit contradictoire. Par définition, si une $\mathcal{L}$-structure est modèle de Γ, elle n'est pas modèle de $\neg\phi$. Donc, elle est modèle de ϕ, et on a que toute structure qui est modèle de Γ est modèle de ϕ. □

Pour tester la validité d'une inférence

$$\frac{\psi_1, \ldots, \psi_n}{\phi}$$

on transcrit dans le langage polyadique les permises $\psi_1, \ldots, \psi_n$ et la conclusion ϕ, obtenant les formules $\psi_1, \ldots, \psi_n$ et ϕ. On teste si l'ensemble $\{\psi_1, \ldots, \psi_n, \neg\phi\}$ est contradictoire avec la méthode des arbres : on démarre la construction de l'arbre et, si on obtient un arbre fermée, on conclut que l'ensemble de formule *n'est pas satisfaisable* (donc, il est contradictoire). Si on obtient un arbre achevé mais ouvert, on conclut que l'ensemble de formules est *satisfaisable*, et donc qu'il existe un modèle pour l'ensemble [7]. Pour le Théorème 4.1, dans le premier cas on a que $\{\psi_1, \ldots, \psi_n\} \vDash \phi$; dans le dernier, $\{\psi_1, \ldots, \psi_n\} \nvDash \phi$.

5 Une question de décidabilité

On a vu comment la méthode des arbres peut être utilisée pour déterminer la validité (ou l'invalidité) d'un énoncé (Section 3) et d'une inférence (Section 4.1). La méthode des arbres constitue une *procédure effective*, au sens que chaque étape du procès est déterminée, et *générale*, au sens qu'elle peut être appliquée à énoncés et inférences quelconques.

On peut se demander si la méthode des arbres est une *procédure de décision* pour le problème de tester la validité d'un énoncé et d'une inférence dans le cas du langage du premier ordre, c'est à dire, si la procédure de construction de l'arbre se termine toujours dans un nombre fini d'étapes. Notez que s'il existe une procédure de décision pour tester la validité d'une formule/énoncé, alors on a une procédure de décision pour tester la validité d'une inférence [8]. Si on a une procédure de décision pour tester la validité (et

7. Le modèle peut être extrait à partir de la branche ouverte, suivant la même idée que pour les énoncés.

8. Pour tester la validité d'une inférence $\frac{\psi_1, \ldots, \psi_n}{\phi}$, il suffit d'écrire à la racine de l'arbre l'ensemble d'énoncés $\{\psi_1, \ldots, \psi_n, \neg\phi\}$. Si la construction de l'arbre se termine pour un énoncé, elle va se terminer pour un ensable d'énoncés.

l'invalidité) d'une formule/énoncé, on dit que la logique est *décidable*.

On sait que la logique propositionnelle est décidable, et que la méthode des arbres constitue une procédure de décision pour tester si une une formule du langage propositionnel est une tautologie (ou une antilogie ; voir [1], Chapitre 6). Pour n'importe quelle formule à la racine, la construction de l'arbre de vérité se termine *après un nombre fini d'étapes*, et on obtient :

1. Soit un arbre *fermé* (où toutes les branches sont fermés),
2. Soit un arbre *achevé* et *ouvert* (un arbre qui contient au moins une branche ouverte, à laquelle on a appliqué toutes les règles).

Pour le langage du premier ordre, au contraire, la construction d'un arbre de vérité pour un énoncé (ou un ensemble d'énoncés) peut donner, autre que les cas 1 et 2, le cas suivant :

3. La construction de l'arbre continue à l'infini, sans que un arbre fermé ou achevé soit atteint.

Donc, la construction de l'arbre peu continuer indéfiniment, sans que le cas 1 ou 2 soit atteint. Dans le cas général du langage polyadique $\mathcal{L}_p$, la méthode des arbres ne constitue pas une procédure de décision pour le problème de déterminer la validité d'un énoncé ou inférence au premier ordre. Par contre, on peut montrer que le fragment du langage monadique $\mathcal{L}_m$ est *décidable*, c'est à dire, il existe une procédure effective pour déterminer dans un nombre fini d'étapes si un énoncé est valide ou pas. On analyse dans la suite les cas de $\mathcal{L}_m$ et $\mathcal{L}_p$.

5.1 Décidabilité pour $\mathcal{L}_m$

Le langage monadique est un langage du premier ordre, où tous les prédicats ont arité 1. On teste la validité de la formule $\phi = \exists x(\neg P(x) \vee \forall y(Q(y)))$ avec la méthode des arbres, et on obtient l'arbre suivant.

$$
\begin{array}{c}
\neg\exists x(\neg P(x) \vee \forall y(Q(y))) \quad \checkmark \\
\forall x\neg(\neg P(x) \vee \forall y(Q(y))) \quad (*) \\
\neg(\neg P(a) \vee \forall y(Q(a))) \quad \checkmark \\
\neg\neg P(a) \quad \checkmark \\
\neg\forall y(Q(y)) \quad \checkmark \\
P(a) \\
\exists y\neg(Q(y)) \quad \checkmark \\
\neg Q(b) \\
\neg(\neg P(b) \vee \forall y(Q(b))) \quad \checkmark \\
\neg\neg P(b) \quad \checkmark \\
\neg\forall y(Q(y)) \quad \checkmark \\
P(b) \\
\exists y\neg(Q(y)) \quad \checkmark \\
\neg Q(c) \\
\neg(\neg P(b) \vee \forall y(Q(b))) \\
\vdots
\end{array}
$$

L'arbre a une branche infinie : la règle pour le quantificateur universel doit être appliquée à la formule $(*)$ avec toutes les constantes qui apparaissent sur la branche, et la règle pour le quantificateur existentiel introduit toujours des constantes nouvelles. Donc, on ne trouve jamais un arbre achevé ou un arbre fermé. Tel qu'on l'a défini, la méthode des arbres ne constitue donc une procédure de décision pour la logique monadique.

Par contre, la logique monadique est *décidable* : c'est à dire, on peut trouver une méthode effective et générale qui nous dit, dans un temps fini, si un certain énoncé est valide ou pas. Il y a différents stratégies pour démontrer la décidabilité. Une consiste à montrer que la logique a la propriété du modèle fini : c'est à dire, si un énoncé de $\mathcal{L}_m$ est valide, il a un modèle dont le domaine est composé d'un nombre *fini* d'éléments. En plus, pour chaque énoncé il existe un nombre *fini* de modèles à tester. On peut donc essayer de construire toutes les modèles finies pour un énoncé, pour voir s'il est valide [9].

Autrement, on peut observer que la branche infinie de l'arbre se génère parce que les quantificateurs dans la formule à la racine sont *imbriqués*. Intuitivement, dans le cas monadique on peut toujours trouver une formule équivalente à une formule imbriquée telle qu'il n'y a pas de quantificateur qui occurrent à l'intérieur d'un quantificateur (donc, une formule sans quantificateurs imbriquées). Dans notre cas, la construction de l'arbre pour la formule $(\exists x \neg P(x)) \vee (\forall y Q(y))$, équivalente à ϕ mais qui n'a pas de quantificateurs imbriquées, se termine.

5.2 Décidabilité pour $\mathcal{L}_p$

Dès qu'on introduit un prédicat à deux places dans le langage, on a plus moyen de réécrire les formules avec quantificateurs imbriquées. On peut donc obtenir des arbres avec branches infinies, et on ne peut pas décider sur la validité ou invalidité de la formule à la racine de l'arbre. On essaye de construire l'arbre pour l'énoncé $\exists x \forall y(R(x,y))$.

$$\begin{array}{rl} \neg\exists x\forall y(R(x,y)) & \checkmark \\ \forall x\neg\forall y(R(x,y)) & (*) \\ \neg\forall y(R(a,y)) & \checkmark \\ \exists y\neg(R(a,y)) & \checkmark \\ \neg R(a,b) & \\ \neg\forall y(R(b,y)) & \checkmark \\ \vdots & \end{array}$$

La construction ne se termine pas : la formule $(*)$ génère une nouvelle formule pour toutes les constantes dans la branche, et la règle du quantificateur existentiel introduit toujours des nouvelles constantes.

Donc, la méthode des arbres ne constitue pas une méthode de décision pour la logique polyadique. En plus, Church et Turing ont démontré (indépendamment, en 1936) que la logique polyadique n'est pas décidable : c'est à dire, on ne peut pas trouver une procédure effective qui décide, dans un nombre fini d'étapes, si une formule est valide ou pas.

9. Plus de détails se trouvent dans [3], page 208 (en anglais).

6 Langage polyadique avec égalité

Dans cette section, on reprend la définition du langage polyadique $\mathcal{L}_e$ et on lui ajoute le symbole d'égalité, "=". On appelle $\mathcal{L}_e$ le langage polyadique avec égalité. L'alphabet de $\mathcal{L}_e$ est défini comme celui de $\mathcal{L}_e$, avec la différence que le symbole "=" figure entre les signes logiques. L'égalité est traité comme un symbole de prédicat à deux places : on n'a donc plus besoin de spécifier qu'il doit y avoir au moins un symbole de prédicat dans l'alphabet. La définition des termes de $\mathcal{L}_e$ est la même que pour $\mathcal{L}_p$. La définition des formules est la suivante.

Définition 6.1. L'ensemble $\mathsf{For}(\mathcal{L}_e)$ de *formules* de $\mathcal{L}_e$ est défini inductivement à partir des formules atomiques.

— Soit P^n un symbole de prédicat à n places, et $t_1, \ldots, t_n$ termes de $\mathcal{L}_e$. Alors

$$P^n(t_1, \ldots, t_n)$$

est une formule atomique de $\mathcal{L}_e$.

— Soient t_1, t_2 termes de $\mathcal{L}_e$. Alors $t_1 = t_2$ est une formule atomique de $\mathcal{L}_e$.

— Soient ϕ, ψ formules de $\mathcal{L}_e$. Alors $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \to \psi$, $\phi \leftrightarrow \psi$ sont des formules.

— Soit ϕ une formule, et x une variable. Alors $\forall x\phi$ et $\exists x\phi$ sont des formules.

— Rien d'autre est une formule.

La formule atomique $t_1 = t_2$ signifie que t_1 et t_2 désignent le même individu. La formule $\neg(t_1 = t_2)$ (qu'on peut aussi écrire comme $t_1 \neq t_2$) signifie que t_1 et t_2 désignent deux individus différents. En conséquence, le symbole d'égalité est interprété de la façon suivante, pour $\mathcal{I}$ structure d'interpretation et σ assignation (voir Définition 2.5) :

$$\mathcal{I} \models t_1 = t_2 \quad [\sigma] \;\text{ ssi }\; i(t_1) = i(t_2)$$

Donc, deux termes sont égales s'ils dénotent le même élément du domaine.

Avec l'égalité on a un langage beaucoup plus riche, qui permet de compter et distinguer les individus (en particulier : compter et distinguer les individus de l'interpretation du langage). Comparez les exemples suivants.

1. *Il y a au moins un chien.* $\exists x(C(x))$.
2. *Il y a au plus un chien.* $\forall x\forall y((C(x) \wedge C(y)) \to x = y)$.
3. *Il y a exactement un chien.* $\exists x(C(x) \wedge \forall y(C(y) \to x = y))$.
4. *Il y a un seul chat rouge.* $\exists x(C(x) \wedge R(x) \wedge \forall y((C(y) \wedge R(y)) \to x = y))$.
5. *Il existent exactement deux nombres premiers inférieurs à 5.*

$$\begin{aligned}\exists x\exists y \quad & (N(x) \wedge N(y) \wedge P(x) \wedge P(y) \wedge x < 5 \;\wedge y < 5 \;\wedge \\ & \wedge \forall z(N(z) \wedge P(z) \wedge z < 5 \to z = x \vee z = y)).\end{aligned}$$

Facultatif. Avec l'égalité, on peut formaliser les *descriptions définies*, en suivant la méthode exposé dans Russell dans l'essai *On Denoting* (1906). Une description définie est une locution nominale (un nom) qui a la fonction de décrire d'une façon non ambiguë un

individu ou un objet particulier, similairement à ce que un nom propre fait. Différemment d'un nom propre, une description définie ne nomme pas un individu, mais le décrit à travers une ou plusieurs propriétés qui caractérisent univoquement cet individu / objet. L'individu décrit par une description définie est *le seul* individu qui possède les caractéristiques données dans la description. Les deux locutions nominales suivantes sont des descriptions définies :

— Le président des États Unis ;
— L'actuel roi de France.

L'énoncé

Le président des États Unis est blond

se formalise en disant qu'il existe un x qui est le président des États Unis, $P(x)$; que, pour tout y ayant la même propriété, alors $x = y$; et que x est blond, $B(x)$.

$$\exists x(P(x) \wedge \forall y(P(x) \rightarrow x = y) \wedge B(x)).$$

L'énoncé est vrai si, dans le domaine, il existe exactement un élément qui a la propriété $P(x)$ et $B(x)$.

L'énoncé

L'actuel roi de France est blond

se formalise d'une façon similaire, pour $R(x)$ "être roi de France" :

$$\exists x(R(x) \wedge \forall y(R(x) \rightarrow x = y) \wedge B(x)).$$

L'énoncé est faux s'il n'y a pas un individu qui est roi de France, parce que la condition existentielle est vide. Avec cette méthode il est possible d'attribuer un valeur de vérité aussi aux énoncés qui contiennent des description définies qui n'ont pas de dénotation dans le domaine.

Références

[1] Pierre Wagner. *Logique et philosophie.* Ellipses, 2014.

[2] Peter Smith. *An introduction to formal logic.* Cambridge University Press, 2003.

[3] Geoffrey Hunter. *Metalogic : An introduction to the metatheory of standard first order logic.* Macmillan International Higher Education, 1971.