Systèmes de déduction

Richard Lassaigne

Institut de Mathématiques de Jussieu
Equipe de Logique mathématique
CNRS-Université Paris Diderot

Historiquement, les premiers systèmes de déduction pour la logique classique, dus à G.Frege et D. Hilbert, étaient basés sur des schémas d'**axiomes** et sur des **règles** de déduction. G.Gentzen eut l'idée de les remplacer par un système ne comportant que des règles et correspondant à une forme de déduction naturelle : dans une démonstration mathématique, la déduction s'effectue à partir d'hypothèses admises en cours de raisonnement, plutôt qu'à partir d'hypothèses générales, fixées une fois pour toutes. La première section est consacrée à la présentation de la **déduction naturelle**. La **méthode des tableaux** permet de déterminer si une formule du calcul propositionnel est une **tautologie**, ou si une formule est **conséquence** d'un ensemble de formules donné. Elle est basée sur un processus dit de **réfutation** : en supposant que la formule peut être fausse, on cherche à montrer que cette situation est impossible. Elle consiste en la construction d'un **arbre** dont les noeuds sont des ensembles de formules. Pour cette construction, elle utilise des **règles** prenant en compte la forme des formules à traiter. Ces règles sont les duales de règles permettant de construire des preuves dans un système de déduction comme la **déduction naturelle** ou le **calcul des séquents**, dû également à G.Gentzen.
Un certain nombre d'exemples sont présentés dans la deuxième section. La troisième est consacrée à la présentation des tableaux qui sont des arbres d'ensembles de formules, construits à l'aide de règles syntaxiques. Ces tableaux sont utilisés pour définir les notions de réfutation et de preuve d'une formule. Le **théorème de complétude**, principal résultat de ce chapitre, est démontré dans la quatrième section. Ce résultat justifie l'utilisation de cette méthode en montrant qu'elle est correcte et complète, c'est-à-dire qu'elle permet de répondre de manière générale à la question : une formule F est-elle conséquence d'un ensemble de formules Σ ? Enfin, elle fournit un **algorithme** de résolution de ce problème, qui constitue la base de méthodes de démonstration automatique.

1 La déduction naturelle

Dans la suite, Γ et Δ désignent des ensembles finis de formules et Γ, Δ leur réunion : si Δ est réduit à une seule formule A, on le note Γ, A. Les formules du calcul propositionnel considéré ci-dessous sont construites à l'aide des connecteurs $\wedge$, $\vee$, $\rightarrow$ et $\bot$, qui est un connecteur d'arité 0 représentant le faux, ou l'absurde.

Définition 1 *L'expression $\Gamma \vdash A$ exprime que la formule A peut être déduite à l'aide des formules de Γ en appliquant un nombre fini de fois les règles suivantes :*

- *axiome*

$$\frac{}{\Gamma, A \vdash A}$$

- *introduction de $\rightarrow$*

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash (A \rightarrow B)}$$

- *élimination de $\rightarrow$*

$$\frac{\Gamma \vdash A \qquad \Delta \vdash (A \rightarrow B)}{\Gamma, \Delta \vdash B}$$

- *introduction de $\wedge$*

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash (A \wedge B)}$$

- *élimination de $\wedge_1$*

$$\frac{\Gamma \vdash (A \wedge B)}{\Gamma \vdash A}$$

- *élimination de $\wedge_2$*

$$\frac{\Gamma \vdash (A \wedge B)}{\Gamma \vdash B}$$

- *introduction de $\vee_1$*

$$\frac{\Gamma \vdash A}{\Gamma \vdash (A \vee B)}$$

- *introduction de $\vee_2$*

$$\frac{\Gamma \vdash B}{\Gamma \vdash (A \vee B)}$$

- *élimination de $\vee$*

$$\frac{\Gamma \vdash (A \vee B) \qquad \Delta, A \vdash C \qquad \Delta, B \vdash C}{\Gamma, \Delta \vdash C}$$

- *élimination de* $\bot$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}$$

Dans le cas où Γ *est vide, on note* $\vdash A$.

Ces dix règles constituent un système de déduction pour la logique intuitionniste. La négation d'une formule peut être introduite comme une abréviation de la formule $(A \to \bot)$. La logique classique étudie les procédés du raisonnement mathématique : certains sont non constructifs, comme le principe du **tiers exclu**, le **raisonnement par l'absurde** ou par **contraposition**.

La logique **intuitionniste** est un cadre de formalisation du raisonnement qui n'autorise que des procédés constructifs : c'est-à-dire qui ne considère un objet que s'il est obtenu à l'aide d'une construction effective. C'est ce qui explique qu'elle est à la base de l'étude des problèmes de déduction en informatique, comme par exemple le typage dans les langages fonctionnels ou le processus d'inférence de la Programmation logique. Le système de règles pour la logique **classique** comporte une règle de plus que le système intuitionniste. Cette règle correspond à l'utilisation classique de la **double négation** :

$$\frac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}$$

Les quatre premiers exemples de preuve suivants sont construits dans le système intuitionniste, tandis que le dernier correspond au raisonnement par contraposition et utilise les règles de la logique classique.

Exemple 1 *1.* $\vdash (A \to (B \to A))$ *:*

$$\dfrac{\dfrac{B, A \vdash A}{A \vdash (B \to A)}}{\vdash (A \to (B \to A))}$$

2. $\vdash ((A \to B) \to ((B \to C) \to (A \to C)))$ *:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(A \to B) \vdash (A \to B) \qquad A \vdash A}{(A \to B), A \vdash B} \qquad (B \to C) \vdash (B \to C)}{(A \to B), (B \to C), A \vdash C}}{(A \to B), (B \to C) \vdash (A \to C)}}{(A \to B) \vdash ((B \to C) \to (A \to C))}}{\vdash ((A \to B) \to ((B \to C) \to (A \to C)))}$$

3. En particulier : $\vdash ((A \to B) \to (\neg B \to \neg A))$

4. $\vdash A \to \neg\neg A$ *:*

$$\dfrac{\dfrac{\dfrac{A \vdash A \qquad (A \to \bot) \vdash (A \to \bot)}{A, (A \to \bot) \vdash \bot}}{A \vdash ((A \to \bot) \to \bot)}}{\vdash A \to \neg\neg A}$$

5. $\vdash ((\neg B \to \neg A) \to (A \to B))$ *:*

$$\dfrac{\dfrac{\dfrac{\dfrac{(\neg B \to \neg A) \vdash (\neg\neg A \to \neg\neg B) \qquad A \vdash \neg\neg A}{(\neg B \to \neg A), A \vdash \neg\neg B}}{(\neg B \to \neg A), A \vdash B}}{(\neg B \to \neg A) \vdash (A \to B)}}{\vdash ((\neg B \to \neg A) \to (A \to B))}$$

Les propriétés usuelles des connecteurs vis-à-vis de la déduction peuvent être prouvées dans le cadre de la déduction naturelle et sont laissées en exercice. On donne le détail des démonstrations pour les deux propriétés suivantes :

- $\vdash \neg(A \wedge B)$ si et seulement si $\vdash (\neg A \vee \neg B)$
- $\vdash \neg(A \vee B)$ si et seulement si $\vdash (\neg A \wedge \neg B)$

1. Si $\vdash (\neg A \vee \neg B)$, alors $\vdash \neg(A \wedge B)$:

$$\dfrac{\vdash (\neg A \vee \neg B) \qquad \dfrac{\dfrac{(A \wedge B) \vdash (A \wedge B)}{(A \wedge B) \vdash A} \qquad \neg A \vdash \neg A}{(A \wedge B), \neg A \vdash \bot} \qquad \dfrac{\dfrac{(A \wedge B) \vdash (A \wedge B)}{(A \wedge B) \vdash B} \qquad \neg B \vdash \neg B}{(A \wedge B), \neg B \vdash \bot}}{\dfrac{(A \wedge B) \vdash \bot}{\vdash \neg(A \wedge B)}}$$

2. Si $\vdash \neg(A \vee B)$, alors $\vdash (\neg A \wedge \neg B)$:

$$\dfrac{\dfrac{\dfrac{\vdash \neg(A \vee B) \qquad \dfrac{A \vdash A}{A \vdash (A \vee B)}}{\dfrac{A \vdash \bot}{\vdash \neg A}} \qquad \dfrac{\vdash \neg(A \vee B) \qquad \dfrac{B \vdash B}{B \vdash (A \vee B)}}{\dfrac{B \vdash \bot}{\vdash \neg B}}}{\vdash (\neg A \wedge \neg B)}}$$

3. Si $\vdash \neg(A \wedge B)$, alors $\vdash (\neg A \vee \neg B)$:
   En utilisant la contraposition, il suffit de montrer que si $\vdash \neg(\neg A \vee \neg B)$, alors $\vdash (A \wedge B)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg A \vdash \neg A}{\neg A \vdash (\neg A \vee \neg B)} \qquad \vdash ((\neg A \vee \neg B) \to \bot)}{\neg A \vdash \bot}}{\vdash (\neg A \to \bot)}}{\vdash A}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg B \vdash \neg B}{\neg B \vdash (\neg A \vee \neg B)} \qquad \vdash ((\neg A \vee \neg B) \to \bot)}{\neg B \vdash \bot}}{\vdash (\neg B \to \bot)}}{\vdash B}$$

$$\dfrac{\vdash A \qquad \vdash B}{\vdash (A \wedge B)}$$

4. $\vdash (\neg A \wedge \neg B)$ implique $\vdash \neg(A \vee B)$:
   En utilisant la contraposition, il suffit de montrer que si $\vdash (A \vee B)$, alors $\vdash \neg(\neg A \wedge \neg B)$.

$$\dfrac{\dfrac{\dfrac{(\neg A \wedge \neg B) \vdash (\neg A \wedge \neg B)}{(\neg A \wedge \neg B) \vdash \neg A} \qquad A \vdash A}{(\neg A \wedge \neg B), A \vdash \bot}}$$

$$\dfrac{\dfrac{\dfrac{(\neg A \wedge \neg B) \vdash (\neg A \wedge \neg B)}{(\neg A \wedge \neg B) \vdash \neg B} \qquad B \vdash B}{(\neg A \wedge \neg B), B \vdash \bot}}$$

$$\dfrac{\dfrac{\vdash (A \vee B) \qquad (\neg A \wedge \neg B), A \vdash \bot \qquad (\neg A \wedge \neg B), B \vdash \bot}{(\neg A \wedge \neg B) \vdash \bot}}{\vdash \neg(\neg A \wedge \neg B)}$$

2 Exemples de tableaux

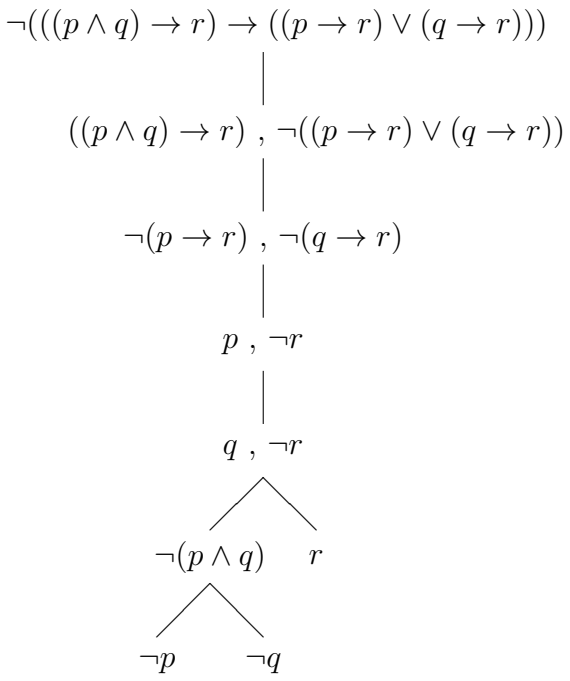
Pour déterminer si une formule est une tautologie, la définition conduit à utiliser un procédé sémantique qui consiste à examiner si toutes les valuations possibles donne la valeur vrai à la formule considérée. D'une part, dans la pratique, cette méthode est inutilisable : si n variables propositionnelles apparaissent dans la formule, il est nécessaire d'examiner 2^n valuations. D'autre part, il semble naturel de se demander si les connecteurs ont un *sens logique* suivant lequel, par exemple : l'énoncé $(p \wedge q)$ permet de *déduire* à la fois les énoncés p et q, et l'énoncé $(p \to q)$ l'alternative $\neg p$ ou q. Ce sens logique ne doit résider que dans la forme des formules, et non dans leur interprétation dans un ensemble de valeurs. La méthode des tableaux a pour objet d'élaborer deux types de règles qui prennent en compte la forme des formules et qui permettent :

- de déterminer si une formule est une tautologie
- ou de déduire une formule d'un ensemble de formules donné,

Dans le second cas, l'utilisation de ces règles peut être représentée par la construction d'un **arbre** à partir de la formule considérée. Par exemple, si la formule de départ est la formule F suivante : $(((p \wedge q) \to r) \to ((p \to r) \vee (q \to r)))$, on considère la formule $\neg F$, ce qui correspond à envisager les situations dans lesquelles F serait éventuellement fausse. Il est facile d'en tirer les conséquences pour les sous-formules immédiates de cette formule et ensuite de recommencer le processus pour ces formules :

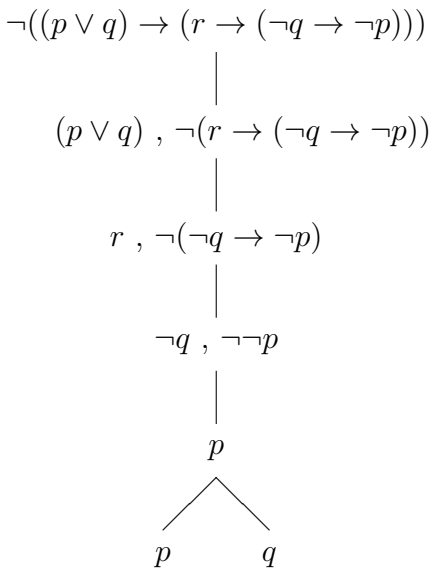
- on considère la formule $((p \wedge q) \to r)$ et la formule $\neg(((p \to r) \vee (q \to r)))$;
- à l'étape suivante, on considère les formules $\neg(p \to r)$ et $\neg(q \to r)$;
- on considère ensuite les formules p et $\neg r$;
- la même chose pour q et $\neg r$;
- d'autre part, on doit tenir compte de $((p \wedge q) \to r)$; il suffit de s'intéresser à l'une des deux formules $\neg(p \wedge q)$ ou r ;
- ce dernier cas (r) ayant été exclu précédemment, $\neg(p \wedge q)$ se ramène à l'un des deux cas $\neg p$ ou $\neg q$, qui ont déjà été exclus tous les deux.

Il n'existe donc aucune situation dans laquelle la formule F soit fausse : cette formule est bien une tautologie. Pour pouvoir développer systématiquement et vérifier commodément un raisonnement de ce genre, on le représente par un arbre de la manière suivante :



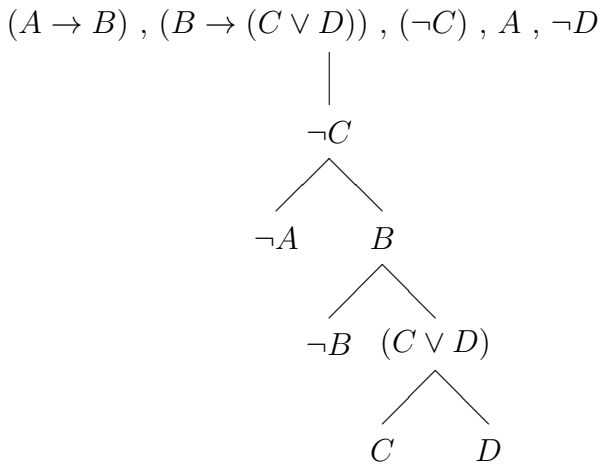
Chaque *branche* de l'arbre représente un scénario possible : plus on descend le long d'une branche, plus se précisent les contraintes. Lorsque l'on rencontre sur une branche une formule A telle que $\neg A$ figure plus haut sur la même branche, ou l'inverse, on cesse de la développer et on dit qu'elle est close, pour indiquer qu'elle est contradictoire. Si toutes les branches sont closes, tous les scénarios ont été exclus : il n'existe aucune situation dans laquelle la formule F soit fausse.

Considérons maintenant la formule G suivante : $((p \vee q) \to (r \to (\neg q \to \neg p)))$ et développons selon les mêmes principes un arbre issu de $\neg G$:

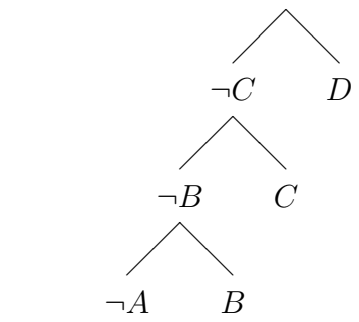


Ainsi développé, l'arbre a deux branches, dont l'une, celle de droite est close. L'autre ne l'est pas et les variables propositionnelles apparaissant sur cette branche sont : $r, \neg q, p$. Si v est une valuation telle que $v(r) = 1, v(q) = 0, v(p) = 1$, elle donne la valeur 1 à $\neg G$ et 0 à G. La branche non close fournit donc un scénario dans lequel $\neg G$ est vraie et G n'est donc pas une tautologie.

Supposons maintenant que l'on veuille montrer qu'une formule D est conséquence des formules $(A \to B), (B \to (C \vee D)), \neg C, A$. D'après le paragraphe du chapitre précédent sur la sémantique, il est possible de raisonner directement sur les valeurs que donne une valuation aux formules A, B, C, D : il suffit de vérifier que si elle donne la valeur 1 aux formules de l'ensemble considéré, alors elle donne également la valeur 1 à la formule D. La méthode précédente peut donc être appliquée pour rechercher un scénario permettant de réfuter l'affirmation en question, c'est-à-dire un scénario pour l'ensemble de formules $(A \to B), (B \to (C \vee D)), (\neg C), A, \neg D$:



Toutes les branches sont closes : la réfutation a échoué et D est bien une conséquence de l'ensemble $\{(A \to B), (B \to (C \vee D)), \neg C, A\}$. En revanche, il est possible de réfuter l'affirmation : D est conséquence de l'ensemble $\{(A \to B), (B \to C), (C \to D)\}$.



Toute valuation v satisfaisant $v(A) = v(B) = v(C) = v(D) = 0$ permet de réfuter l'affirmation considérée.

3 Méthode des tableaux

3.1 Compléments sur les arbres

Un **chemin** dans un arbre T est une suite, finie ou infinie, (x_i) d'éléments de T telle que pour tout $i \in \mathbb{N}$, x_i soit le prédécesseur de x_{i+1}. Une **branche** est un chemin que l'on ne peut pas prolonger : x_0 est la racine et si la branche est finie, $(x_0, x_1, ..., x_n)$, x_n n'a pas de successeur.

Définition 2 *La relation $\leq$ est définie sur T de la manière suivante :*
pour tout couple $(x, y) \in T^2$, $x \leq y$ s'il existe un chemin fini allant de x à y.

Un arbre peut un être ordonné à l'aide de cette relation. La démonstration de la proposition suivante est laissée à titre d'exercice.

Proposition 1 *La relation $\leq$ définit sur T un ordre partiel tel que pour tout élément x de T, l'ensemble des minorants de x, $\{y \in T : y \leq x\}$ est isomorphe à l'ensemble $\{0, 1, ..., h(x)\}$ muni de l'ordre usuel sur les entiers.*

En particulier, l'ensemble des minorants d'un élément est totalement ordonné. Dans un arbre, les éléments sans successeur s'appellent les **noeuds terminaux** ou les **feuilles**.
Tous les arbres considérés dans ce cours ont une propriété supplémentaire : ils sont **finitaires**, ou **à branchement fini**, c'est-à-dire que chaque élément a au plus un nombre fini de successeurs. Dans la plupart des cas, ce nombre sera même ≤ 2 et les arbres en question seront dits **binaires**. En général, rien n'empêche un arbre finitaire d'être **infini**, c'est-à-dire d'avoir une infinité d'éléments. Cependant, un tel arbre comporte alors une branche infinie.

Lemme 1 *(König)*
Tout arbre finitaire infini contient au moins une branche infinie.

Preuve : Un élément $x \in T$ est dit **fécond** si sa descendance, c'est-à-dire $\{y \in T : x < y\}$, est infinie. La descendance d'un élément se compose de l'ensemble de ses fils et de la descendance de chacun de ses fils :
$\{y : x < y\} = \{y : P(x, y)\} \cup \{z :$ il existe y tel que $P(x, y)$ et $y < z\}$.
Si un élément est fécond, alors l'un au moins de ses fils l'est aussi. Dire qu'un arbre est infini, c'est dire que sa racine est un élément fécond. Il est alors possible de définir par récurrence sur n, le nième élément de la branche infinie : x_0 est la racine de T et si x_n est fécond, alors x_{n+1} est défini comme l'un de ses fils féconds.

3.2 Construction des tableaux

Un **tableau** est un arbre binaire dont les éléments sont des ensembles de formules du calcul propositionnel et qui est construit à partir d'une racine quelconque par l'application répétée, un nombre fini de fois, de l'une ou l'autre des deux règles ci-dessous : la α-règle et la β-règle. Dans la suite, le calcul propositionnel est restreint aux connecteurs $\neg, \wedge, \vee, \rightarrow$. Les formules sont réparties en deux classes et à chaque formule, on associe canoniquement deux autres formules de la manière suivante :

Type α			Type β		
Formule	Formules associées		Formule	Formules associées	
$\alpha = (A \wedge B)$	$\alpha_1 = A$	$\alpha_2 = B$	$\beta = \neg(A \wedge B)$	$\beta_1 = \neg A$	$\beta_2 = \neg B$
$\alpha = \neg(A \vee B)$	$\alpha_1 = \neg A$	$\alpha_2 = \neg B$	$\beta = (A \vee B)$	$\beta_1 = A$	$\beta_2 = B$
$\alpha = \neg(A \to B)$	$\alpha_1 = A$	$\alpha_2 = \neg B$	$\beta = (A \to B)$	$\beta_1 = \neg A$	$\beta_2 = B$
$\alpha = \neg\neg A$	$\alpha_1 = A$	$\alpha_2 = A$			

Si B est une branche d'un tableau, on note $\bigcup B$ l'ensemble des formules qui apparaissent dans B, c'est-à-dire qui appartiennent à l'un des noeuds de B. Il est maintenant possible d'énoncer les deux règles de construction des tableaux :

- la α**-règle** consiste à passer d'un arbre T à un arbre T' obtenu en prolongeant une branche finie B de T par le noeud $\{\alpha_1, \alpha_2\}$, où α est une formule de type α, qui apparaît dans B ;
- la β**-règle** consiste à passer d'un arbre T à un arbre T' obtenu en prolongeant une branche finie B de T par deux noeuds $\{\beta_1\}$ et $\{\beta_2\}$, où β est une formule de type β, qui apparaît dans B.

L'arbre T' est alors une **extension** de l'arbre T : les éléments de T sont des éléments de T', les niveaux au sens de T et au sens de T' sont égaux, et la relation prédécesseur est vérifiée dans T si et seulement si elle l'est dans T' (pour des couples d'éléments de T) ; de plus, si B est la branche de T, qui est prolongée, et si x_n, de niveau n, est son élément terminal, alors :

- si la α-règle a été appliquée, $T'-T$ comprend un seul élément $\{\alpha_1, \alpha_2\}$ tel que $\alpha \in \bigcup B$, $P(x_n, \{\alpha_1, \alpha_2\})$ et $h(\{\alpha_1, \alpha_2\}) = n + 1$;
- si la β-règle a été appliquée, $T' - T$ comprend deux éléments $\{\beta_1\}$ et $\{\beta_2\}$ tels que $\beta \in \bigcup B$, $P(x_n, \beta_1)$, $P(x_n, \beta_2)$, $h(\beta_1) = n + 1$ et $h(\beta_2) = n + 1$.

On remarque que ce n'est pas nécessairement une formule du dernier élément de B qui est *développée* : la formule développée peut se situer à n'importe quel niveau dans la branche B. En pratique, les { } sont omises et les formules d'un même noeud sont disposées les unes au-dessous des autres. Dans les deux derniers cas d'application de la α-règle, c'est par souci d'uniformité que l'on a défini les formules α_1, α_2, qui sont égales, et le nouveau noeud ne comporte alors qu'une seule formule.

Définition 3 *Un arbre T' est une* **extension immédiate** *d'un arbre T si T' est obtenu par application à T de la α-règle ou de la β-règle. Un* **tableau** *est un arbre T pour lequel il existe une suite finie d'arbres $T_0, T_1, ..., T_n$ telle que :*

- *T_0 est réduit à sa racine, un ensemble quelconque de formules affectées d'un signe,*
- *pour chaque $i = 0, 1, ..., n-1$, T_{i+1} est une extension immédiate de T_i,*
- *$T_n = T$.*

3.3 Développement et clôture

Définition 4 *Soit T un tableau. Une branche B de T est* **close** *s'il existe une formule A telle que A et $\neg A$ apparaissent dans B. Dans le cas contraire, B est dite* **ouverte**. *Une branche B est dite* **développée** *si :*

- *pour toute formule $\alpha \in \bigcup B$, $\alpha_1 \in \bigcup B$ et $\alpha_2 \in \bigcup B$,*
- *pour toute formule $\beta \in \bigcup B$, $\beta_1 \in \bigcup B$ ou $\beta_2 \in \bigcup B$.*

Un tableau est **développé** *si toutes ses branches sont soit closes, soit développées. Un tableau est* **clos** *si toutes ses branches sont closes. Un tableau est* **ouvert** *s'il possède une branche ouverte. Un tableau* **pour** *une formule A (ou pour un ensemble de formules Σ) est un tableau de racine $\{A\}$ (ou $\{A : A \in \Sigma\}$).*

Proposition 2 *Si Σ est un ensemble fini ou dénombrable de formules, alors il existe un tableau développé pour Σ.*

Preuve : Si Σ est fini, la démonstration se fait par récurrence sur le nombre n de ses éléments.

- Pour $n = 1$, il suffit de remarquer que la complexité des formules α_1 et α_2 (respectivement β_1 et β_2) est strictement inférieure à celle de α (respectivement β). Le processus d'extension des branches non closes prend donc fin après un nombre fini d'étapes. Le tableau est alors développé, sans quoi il serait possible de l'étendre encore.
- Si $\Sigma = \{A_1, A_2, ..., A_{n+1}\}$, on construit d'abord un tableau développé pour $\Sigma' = \{A_1, A_2, ..., A_n\}$, ce qui est possible d'après l'hypothèse de récurrence. Si ce tableau est clos ou si A_{n+1} est une variable propositionnelle, il constitue un tableau développé pour Σ. Sinon, on étend successivement les branches ouvertes en appliquant d'abord la règle convenable pour A_{n+1}, puis en développant les branches ainsi obtenues : le processus se termine pour la même raison que ci-dessus.

Si Σ est infini dénombrable, le raisonnement précédent est utilisé pour montrer l'existence d'un tableau développé pour Σ à l'aide d'une suite (E_i) éventuellement infinie d'étapes :

- à l'étape E_1, correspond un tableau développé pour $\{A_1\}$;
- le passage de l'étape E_n à l'étape E_{n+1} s'effectue exactement comme celui du tableau développé pour $\Sigma' = \{A_1, A_2, ..., A_n\}$ au tableau développé pour $\Sigma = \{A_1, A_2, ..., A_{n+1}\}$.

Il est clair qu'il existe bien des manières de construire un tableau ayant une racine donnée. Différentes stratégies sont possibles suivant l'objectif que l'on s'est fixé. Par exemple, pour une implémentation simple, une stratégie consiste à développer la formule (non développée) placée le plus haut possible sur la branche située le plus à gauche dans l'arbre déjà obtenu. Si l'on procède à la main, et plus généralement si l'on a des raisons de préférer des arbres hauts et étroits à des arbres larges et courts, on commencera par développer selon la α-règle tout ce qui peut l'être avant d'appliquer une première fois la β-règle, et ainsi de suite.
Par ailleurs, il est intéressant d'exploiter intégralement une formule que l'on développe, c'est-à-dire d'étendre toutes les branches ouvertes auxquelles appartient la formule : il est alors possible de l'oublier, dans la mesure où elle ne peut plus livrer d'informations nouvelles.

4 Théorème de complétude

4.1 Réfutation et preuve d'une formule

Définition 5 *Une formule A est* **prouvable** *(par tableau) s'il existe un tableau clos de racine $\{\neg A\}$.*

Pour prouver une formule A, il suffit donc d'exhiber un tableau clos de racine $\{\neg A\}$. Comme il y a de nombreuses manières de développer un tableau, ne risque-t-on pas de chercher longtemps la réfutation voulue ? Qu'est-ce qui garantit qu'un premier tableau développé étant ouvert, un second sera clos, ou au contraire restera ouvert, auquel cas il faudra en construire un troisième, et ainsi de suite, le processus pouvant peut-être même ne jamais se terminer... Il est possible de voir que cette dernière éventualité ne risque pas de se présenter : pour une formule donnée, il n'existe qu'un nombre fini de tableaux développés qui sont essentiellement différents. Dans la suite, on verra qu'en fait un tableau développé suffit : s'il est clos, tout autre tableau pour la formule considérée est clos, s'il est ouvert, tout autre tableau est ouvert. La question *telle formule est-elle prouvable ?* se décide au vu d'un seul tableau développé.
Une méthode comme celle des tableaux sera considérée comme bien adaptée à la sémantique si elle est **cohérente** et **complète**, c'est-à-dire si :

- toute formule prouvable par tableau est une tautologie
- et toute tautologie est prouvable par tableau.

4.2 Cohérence de la méthode des tableaux

Définition 6 *Une branche B d'un tableau T est* **réalisable** *s'il existe une valuation v telle que pour toute formule A apparaissant sur la branche B : $v(A) = 1$ si $A \in \bigcup B$ et $v(A) = 0$ si $\neg A \in \bigcup B$. On dit alors que v réalise B. Un tableau est réalisable s'il possède une branche réalisable.*

Lemme 2 *Soit T' une extension immédiate d'un tableau T. Si T est réalisable, alors T' est réalisable.*

Preuve : Soit B une branche réalisable de T et B' la branche de T étendue dans T'. Si $B \neq B'$, alors B est une branche réalisable de T'. Si $B = B'$, B est prolongée dans T' :

- soit en une branche B_α, par application de la α-règle ;
- soit en deux branches B'_β et B''_β, par application de la β-règle.

Dans le premier cas, soit α la formule utilisée et v une valuation réalisant B : de $v(\alpha) = 1$, on déduit $v(\alpha_1) = 1$ et $v(\alpha_2) = 1$; donc v est une valuation réalisant B_α et le tableau T' est réalisable. Dans

le second cas, soit β la formule utilisée : de $v(\beta) = 1$, on déduit qu'au moins l'une des deux valeurs $v(\beta_1)$ et $v(\beta_2)$ est égale à 1 ; donc v est une valuation réalisant l'une des branches B_{β_1}, B_{β_2} et le tableau T' est réalisable.

Proposition 3 *Toute formule prouvable par tableau est une tautologie.*

Preuve : D'après le lemme, tout tableau fini dont la racine est satisfaisable est réalisable, puisqu'il est obtenu par un nombre fini d'extensions immédiates à partir de l'arbre réduit à sa racine. Or un tableau clos est non réalisable : en effet, pour chacune de ses branches, il existe une formule A telle que A et $\neg A$ apparaissent à la fois sur cette branche et aucune de ses branches n'est réalisable. Si F est une formule prouvable et T un tableau clos de racine $\neg F$, T est non réalisable et par suite, $\neg F$ est non satisfaisable : F est donc une tautologie.

Corollaire 1 *Il n'existe pas de formule F telle que F et $\neg F$ sont prouvables.*

4.3 Complétude de la méthode des tableaux

On remarque que si B est une branche développée et ouverte d'un tableau T, alors $\bigcup B$ possède les propriétés suivantes :

- il n'existe pas de variable propositionnelle p telle que $p \in \bigcup B$ et $\neg p \in \bigcup B$;
- pour toute formule $\alpha \in \bigcup B$, $\alpha_1 \in \bigcup B$ et $\alpha_2 \in \bigcup B$;
- pour toute formule $\beta \in \bigcup B$, $\beta_1 \in \bigcup B$ ou $\beta_2 \in \bigcup B$.

Lemme 3 *Toute branche développée et ouverte d'un tableau est réalisable.*

Preuve : Soit B une branche développée et ouverte d'un tableau T. On définit une valuation v par :

- si $p \in \bigcup B$, $v(p) = 1$,
- si $\neg p \in \bigcup B$, $v(p) = 0$,
- si $p \notin \bigcup B$ et $\neg p \notin \bigcup B$, $v(p) = 1$ (par exemple).

On montre par récurrence sur la complexité des formules que : si $A \in \bigcup B$, alors $v(A) = 1$, et si $\neg A \in \bigcup B$, $v(A) = 0$.

- La propriété est évidente pour les variables propositionnelles.
- Si la formule A est une formule α, $\alpha \in \cup B$ entraîne $\alpha_1 \in \bigcup B$ et $\alpha_2 \in \bigcup B$; par hypothèse de récurrence, $v(\alpha_1) = 1$ et $v(\alpha_2) = 1$ et donc $v(\alpha) = 1$.
- Si la formule A est une formule β, $\beta \in \cup B$ entraîne $\beta_1 \in \bigcup B$ ou $\beta_2 \in \bigcup B$; par hypothèse de récurrence, $v(\beta_1) = 1$ ou $v(\beta_2) = 1$ et donc $v(\beta) = 1$.

Proposition 4 *S'il existe un tableau clos de racine $\{\neg A\}$, alors tout tableau développé de racine $\{\neg A\}$ est clos.*

Preuve : Soit T un tableau développé ouvert de racine $\{\neg A\}$ et B une branche ouverte de T : d'après le lemme précédent, B est réalisable et comme $\neg A$ appartient à B, $\neg A$ est satisfaisable. A n'est pas une tautologie et n'est donc pas prouvable par tableau : il n'existe pas de tableau clos de racine $\{\neg A\}$.

Proposition 5 *Toute tautologie est une formule prouvable par tableau.*

Preuve : On suppose que A est une formule non prouvable par tableau et que T est un tableau développé de racine $\{\neg A\}$: T n'est pas clos. Comme dans la démonstration précédente, si B est une branche ouverte de T, B est réalisable et $\neg A$ est satisfaisable : A n'est pas une tautologie.
Le théorème de complétude est ainsi démontré.

Théorème 1 *Pour toute formule A, A est prouvable par tableau si et seulement si A est une tautologie.*

5 Exercices

Exercice 1. Donner une dérivation en déduction naturelle de la formule suivante :

$$\vdash (A \to (B \to C)) \to ((A \to B) \to (A \to C))$$

Exercice 2. Donner une dérivation de la proposition exprimant le tiers-exclu :

$$\vdash (A \vee \neg A)$$

Exercice 3. Donner une dérivation de chacune des propriétés suivantes de l'implication vis-à-vis de la disjonction et de la conjonction :

$$\vdash ((A \vee B) \to C) \to ((A \to C) \wedge (B \to C))$$

$$\vdash ((A \to (B \wedge C)) \to ((A \to B) \wedge (A \to C))$$

Exercice 4*. Donner une dérivation de la formule suivante (appelée loi de Pierce) :

$$\vdash ((A \to B) \to A) \to A$$

Exercice 5. Montrer que le tiers-exclu entraîne la loi de Pierce.
Exercice 6. Donner une dérivation de chacune des propriétés de distributivité suivantes :

$$\vdash ((A \wedge B) \vee C) \to ((A \vee C) \wedge (B \vee C))$$

$$\vdash ((A \vee B) \wedge C) \rightarrow ((A \wedge C) \vee (B \wedge C))$$

Exercice 7. Déterminer si les formules suivantes sont des tautologies à l'aide de la méthode des tableaux :

$$(((p \vee q) \rightarrow r) \rightarrow ((p \rightarrow r) \wedge (q \rightarrow r)))$$

$$((p \vee q) \rightarrow (r \rightarrow (\neg q \rightarrow \neg p)))$$

Exercice 8. Déterminer si la formule $F : \neg(p \rightarrow r)$ est conséquence de l'ensemble de formules Σ à l'aide de la méthode des tableaux :

$$\Sigma = \{(p \rightarrow (q \leftrightarrow r)) \ , \ \neg(p \rightarrow q)\}$$

Exercice 9. Déterminer si l'équivalence suivante est valide :

$$(p \rightarrow (q \wedge r)) \equiv ((q \rightarrow \neg r) \rightarrow \neg p)$$

Exercice 10. Déterminer si la formule $(s \rightarrow r)$ est conséquence de l'ensemble de formules :

$$\{(p \rightarrow (q \rightarrow r)) \ , \ (\neg s \vee p) \ , \ q\}$$

Exercice 11. Déterminer si la formule p est conséquence de l'ensemble de formules :

$$\{((p \wedge q) \rightarrow r)) \ , \ (r \rightarrow s) \ , \ (q \wedge \neg s)\}$$

Exercice 12. Déterminer si la formule suivante est une tautologie :

$$((A \rightarrow (B \rightarrow (C \rightarrow D)) \rightarrow (B \rightarrow (C \wedge A) \rightarrow D)))$$