Tableaux, connexions et couplages

Jean Goubault-Larrecq

February 22, 1999

Une utilisation plus directe du système **LK** de Gentzen est, comme dans le cas propositionnel, la méthode des tableaux. Bien qu'elle n'ait pas bénéficié d'autant de raffinements que la résolution, elle s'appuie sur un formalisme plus propre, à savoir **LK** sans la règle de coupure. De plus, comme les preuves sans coupure sont déjà (presque) canoniques sans que l'on ait besoin de raffiner la méthode, on peut espérer que l'espace de recherche des preuves sera considérablement plus petit qu'avec la résolution. Comme nous le verrons, ce n'est pas tout à fait vrai.

Nous présentons la méthode de base en section 1. Nous la raffinons en la méthodes des *tableaux à variables libres* en section 2, et nous décrivons en section 3 la méthode très proche des connexions, aussi connue sous le nom de méthode des couplages ("matings" en anglais). Nous comparons ces méthodes à certaines variantes de la résolution en section 3.2, ce qui montrera que ces méthodes ne sont pas si éloignées les unes des autres.

1 Tableaux au premier ordre

Grâce au théorème d'élimination des coupures pour la logique du premier ordre, nous pouvons restreindre l'espace de recherche des preuves d'un séquent donné $\Gamma \longrightarrow \Delta$ à celui des preuves sans coupure. C'est exactement ce que nous avons fait dans le cas des tableaux propositionnels.

Dans cette section, nous examinons les façons de chercher une preuve d'une formule existentielle Φ. Pour prouver une formule arbitraire du premier ordre, nous pouvons toujours commencer par l'herbrandiser pour en faire une formule existentielle, ce qui signifie que nous ne perdons pas en généralité en faisant cela. Nous traiterons des formules générales du premier ordre directement en section 2.

On peut rechercher une preuve sans coupure dans **LK** de $\Phi = \exists x_1 \cdot \ldots \exists x_n \cdot \Psi$, où Ψ est sans quantificateur, essentiellement de la même façon qu'avec les tableaux propositionnels. La seule différence est que nous devons traiter de la règle supplémentaire $\exists$R. Ceci pose un nouveau problème : pour prouver $\Gamma \longrightarrow \Delta, \exists x \cdot \Phi$ en utilisant $\exists x \cdot \Phi$ comme formule principale, nous devons deviner une instance $\Phi[t/x]$ de Φ, et produire $\Gamma \longrightarrow \Delta, \exists x \cdot \Phi, \Phi[t/x]$, ou bien $\Gamma \longrightarrow \Delta, \Phi[t/x]$.

Comme l'affaiblissement à droite des séquents est admissible dans le système classique **LK**, si $\Gamma \longrightarrow \Delta, \Phi[t/x]$ est prouvable, alors $\Gamma \longrightarrow \Delta, \exists x \cdot \Phi, \Phi[t/x]$ aussi. Donc il suffit de chercher des preuves de cette dernière. Mais, comme $\exists x \cdot \Phi$ reste présent sur le côté droit, le processus peut se répéter à l'infini : la recherche de preuve peut ne pas terminer. Mais nous devons conserver de telles duplications pour rester complet. Ce que nous avons fait là, logiquement, c'est d'appliquer $\exists$R couplé avec une instance de la règle de contraction. La logique linéaire résout le problème. La logique intuitionniste le résout aussi, en interdisant purement et simplement les affaiblissements à droite. Mais dans ces logiques, la recherche de preuve est en fait considérablement plus difficile qu'en logique classique; notamment, on ne peut déjà pas convertir les formules en forme prénexe.

(α-règles)

α	α_1	α_2
$+(X \wedge Y)$	$+X$	$+Y$
$-(X \vee Y)$	$-X$	$-Y$
$-(X \Rightarrow Y)$	$+X$	$-Y$
$+(\neg X)$	$-X$	
$-(\neg X)$	$+X$	

(β-règles)

β	β_1	β_2
$-(X \wedge Y)$	$-X$	$-Y$
$+(X \vee Y)$	$+X$	$+Y$
$+(X \Rightarrow Y)$	$-X$	$+Y$

Figure 1: Règles de tableaux

1.1 Utilisation du théorème de Herbrand

La solution la plus simple est d'utiliser le théorème de Herbrand : Φ est prouvable si et seulement s'il existe un entier k, et k substitutions $\sigma_1, \ldots, \sigma_k$ telles que $\longrightarrow \Psi\sigma_1, \ldots, \Psi\sigma_k$ soit dérivable dans **LK**. Comme on peut éliminer les coupures des preuves de **LK**, et comme les preuves sans coupure ont la propriété de sous-formule, cette dernière condition est équivalente à : $\longrightarrow \Psi\sigma_1, \ldots, \Psi\sigma_k$ est dérivable en $\mathbf{LK}_0$ sans coupure.

Nous pouvons donc étendre la méthode des tableaux propositionnelle au premier ordre comme suit :

- Deviner un entier k;
- Deviner k substitutions $\sigma_1, \ldots, \sigma_k$;
- Appliquer la méthode des tableaux propositionnelle pour déterminer s'il existe une preuve sans coupure de $\longrightarrow \Psi\sigma_1, \ldots, \Psi\sigma_k$.

Cette procédure naïve demande un travail énorme de divination. Mais on peut remplacer la tâche consistant à deviner les substitutions σ_i par le calcul d'unificateurs, comme c'était déjà le cas pour la résolution.

En premier, au lieu de deviner k substitutions, on peut se ramener à n'en deviner qu'une. Soient $\Psi_1, \Psi_2, \ldots$, des versions renommées de la formule Ψ, et d'ensembles de variables libres deux à deux disjoints. Nous devons trouver k substitutions telles que $\longrightarrow \Psi_1\sigma_1, \ldots, \Psi_k\sigma_k$ soit prouvable, ou de façon équivalente juste une substitution σ telle que $(\longrightarrow \Psi_1, \ldots, \Psi_k)\sigma$ soit prouvable : σ égale $\sigma_1 \cup \ldots \cup \sigma_k$, et réciproquement, σ_i est σ restreinte à $\mathrm{fv}(\Psi_i)$, $1 \leq i \leq k$.

Ainsi, il suffit de trouver une substitution σ qui instancie le séquent donné $\Gamma \longrightarrow \Delta$ en un que nous pouvons démontrer en $\mathbf{LK}_0$ sans coupure. Au lieu de deviner σ, nous le calculons en nous laissant guider par l'expansion du tableau. Remarquons que les α-règles et les β-règles (voir figure 1) sont inchangées par notre connaissance ou notre ignorance de la substitution à appliquer aux formules sur le chemin courant. Le seul changement est dans la façon de clore des chemins, c'est-à-dire dans la façon d'utiliser la règle Ax : au lieu de clore un chemin $\Gamma \longrightarrow \Delta$ en trouvant une formule qui soit à la fois dans Γ et Δ, nous devons trouver une formule Φ_1 dans Γ et une formule Φ_2 dans Δ telles que $\Phi_1\sigma = \Phi_2\sigma$ pour une certaine substitution σ. Lorsque Φ_1 et Φ_2 sont des formules atomiques, c'est simplement le problème de l'unification. Nous devons ensuite appliquer la substitution σ que nous avons trouvée au reste de la preuve en cours de construction avant de continuer à clore d'autres chemins. Expliquons cela plus en détail.

Pour suivre la tradition d'économie de mémoire des tableaux, voici la méthode en *profondeur d'abord* d'expansion des tableaux, définie par une procédure non déterministe `prove`. `prove` prend deux arguments, une liste l de formules signées (le chemin courant) et la substitution courante σ. `prove`(l,σ) retourne soit une nouvelle substitution σ' qui clôt tous les chemins obtenus par expansion à partir de $l\sigma$, ou un symbol spécial échec, qui signifie qu'il n'existe aucune substitution vérifiant cette propriété:

- Choisir de façon non déterministe entre clore la branche courante (trouver une formule atomique positive $+\Phi$ et une formule atomique négative $-\Phi'$ dans l telles que $\Phi\sigma$ et $\Phi'\sigma$ soient unifiables de mgu σ', et retourner $\sigma\sigma'$), ou ce qui suit :
- si l ne contient que des atomes signés, alors retourner échec;
- sinon, appliquer la stratégie d'expansion pour choisir une formule signée non atomique Φ dans l, et soit l' égale l moins cette occurrence de Φ.
  - Si Φ est une formule de type α (de formules filles $\Phi_1, \ldots, \Phi_n$), alors : si `prove`$(\Phi_1 :: l', \sigma)$ retourne échec, retourner échec; sinon, soit σ_2 la substitution retournée, alors si `prove`$(\Phi_2 :: l', \sigma_2)$ retourne échec, retourner échec; sinon, ..., soit σ_n égale `prove`$(\Phi_{n-1} :: l', \sigma_{n-1})$, et retourner `prove`$(\Phi_n :: l', \sigma_n)$.
  - si Φ est une formule de type β (de formules filles $\Phi_1, \ldots, \Phi_n$), alors retourner `prove`$(\Phi_1 :: \ldots :: \Phi_n :: l', \sigma)$.

Ceci est quasiment la réalisation que nous avons donnée pour les tableaux propositionnels, sauf pour ce qui est du choix non déterministe de la première étape. Illustrons ce qu'il signifie sur un exemple. Supposons que nous voulions prouver :

$$\begin{array}{l}\exists x \cdot \exists y \cdot \exists z \cdot \\ \quad (P(f(y), z) \wedge P(y, f(z))) \vee P(f(a), f(a)) \\ \qquad \Rightarrow P(x, f(y)) \vee P(f(x), y)\end{array}$$

Par le théorème de Herbrand, et en devinant le nombre 1 comme nombre d'instances nécessaires k, nous devons expanser et clore le tableau à partir du chemin initial :

$$\begin{array}{l}+(P(f(y), z) \wedge P(y, f(z))) \vee P(f(a), f(a)) \\ \quad \Rightarrow P(x, f(y)) \vee P(f(x), y)\end{array}$$

Puisqu'il n'y a pas de paire complémentaire de formules atomiques, nous devons expanser la formule ci-dessus. Après quelques étapes d'expansion, nous obtenons :

$$\begin{array}{r|l} \multicolumn{2}{c}{+P(x, f(y))} \\ \multicolumn{2}{c}{+P(f(x), y)} \\ -P(f(y), z) & -P(f(a), f(a)) \\ -P(y, f(z)) & \end{array}$$

Maintenant, nous pouvons décider de clore le chemin de gauche en premier par exemple, mais il y a quatre paires complémentaires possibles, correspondant à trois mgu différents :: $[z/x, z/y]$, $[f(y)/x, f(y)/z]$ et $[z/x, f(z)/y]$. Si nous choisissons $[z/x, z/y]$, alors la substitution courante σ passe de $[]$ à $[z/x, z/y]$, et le chemin de droite s'instancie par cette dernière substitution en $\{+P(z, f(z)), +P(f(z), z), -P(f(a), f(a))\}$, qu'on ne peut pas clore. Il faut donc rebrousser chemin et choisir un autre mgu, par exemple $[f(y)/x, f(y)/z]$. La substitution courante σ est maintenant $[f(y)/x, f(y)/z]$, et le chemin de droite s'instancie en $\{+P(f(y), f(y)), +P(f(f(y)), y), -P(f(a), f(a))\}$, que l'on peut clore en unifiant $P(f(y), f(y))$ avec $P(f(a), f(a))$, ce qui donne la substitution finale $[f(a)/x, f(a)/z]$.

Parce qu'on a besoin de rebrousser chemin ("backtrack" en anglais), Prolog est considéré comme un langage idéal pour réaliser des prouveurs de théorèmes à base de tableaux. Un prouveur à base de tableaux qui est à la fois petit et efficace (pour de petits problèmes au moins) est décrit dans [BP94]. Il s'agit cependant d'un prouveur basé sur la variante dite des *tableaux à variables libres*: voir la section 2.

Théorème 1 (Correction, complétude) *Soit* Φ *une formule existentielle* $\exists x_1 \cdot \ldots \exists x_n \cdot \Psi$.

Φ *est prouvable en* **LK** *si et seulement s'il existe un entier* k*, une substitution* σ*, et une expansion de tableau à partir du chemin* $\{\Psi_1, \ldots, \Psi_k\}$ *qui est close par* σ*.*

Plus précisément, étant donnée une stratégie d'expansion f*, si* Φ *est prouvable, alors la procédure de tableaux ci-dessus termine et retourne une telle substitution* σ *comme résultat, pour* k *assez grand.*

Preuve : Si σ clôt une expansion de tableau partant de $\{\Psi_1, \ldots, \Psi_k\}$, alors $\longrightarrow \Psi_1\sigma, \ldots, \Psi_k\sigma$ est dérivable en **LK**, donc Φ est prouvable.

Réciproquement, supposons que Φ soit prouvable en **LK**. Par le théorème de Herbrand, il existe une substitution σ telle que $\longrightarrow \Psi_1\sigma, \ldots, \Psi_k\sigma$ soit dérivable en $\mathbf{LK}_0$ sans la coupure (Cut). Donc le tableau (propositionnel) complètement expansé T obtenu à partir du chemin $\{+\Psi_1\sigma, \ldots, +\Psi_k\sigma\}$ voit tous ses chemins clos. Nous relevons ceci au premier ordre en considérant la tableau complètement expansé T' partant du chemin $\{+\Psi_1, \ldots, +\Psi_k\}$: soit C un chemin arbitraire dans ce tableau complètement expansé, alors $C\sigma$ doit aussi est un chemin complètement expansé de T', et donc il doit être clos par σ. Maintenant, soient C_1, ..., C_m les chemins dans T' : pour tout i, $1 \leq i \leq m$, σ unifie une paire de formules atomiques A_i et B_i de signes opposés dans C_i. Donc il existe un mgu simultané de toutes les paires A_i, B_i, $1 \leq i \leq m$, que l'on trouvera nécessairement en choisissant de façon non déterministe les paires A_i, B_i et en calculant le mgu simultané paire par paire. $\square$

La preuve montre qu'il n'est en fait pas nécessaire d'effectuer un choix non déterministe entre le fait de clore le chemin courant et de l'expanser : la procédure reste complète si nous forçons l'expansion de tous les chemins qui contiennent au moins une formule non atomique, et que nous ne tentions de clore que les chemins atomiques. Ce faisant, nous réduisons le non-déterminisme, mais cela nous force à attendre que tous les chemins aient été totalement expansés avant de les clore, c'est-à-dire essentiellement à attendre d'avoir un nombre exponentiel de chemins à clore. Ainsi, le fait d'expanser avant de clore n'est pas réaliste en général, et nous préférons essayer de clore chaque chemin aussi tôt que possible, bien que cela puisse augmenter le non-déterminisme.

Nous procédons comme suit. Nous essayons d'abord d'unifier des formules atomiques de signes opposés sur le chemin courant, et de retourner l'unificateur correspondant. Si le reste de la recherche échoue, alors le prouveur rebrousse chemin et essaie le choix suivant de paire complémentaire à unifier. S'il n'en reste aucune, alors le prouveur essaie d'expanser une formule de type α ou β. Si ce n'est pas possible, le prouveur rebrousse chemin au dernier point de choix, ou échoue s'il n'y en a pas. On peut faire un peu mieux en s'arrangeant pour que le prouveur évite d'unifier des paires complémentaires dont on sait déjà qu'elles ne mènent pas à une preuve : on conserve le chemin courant sous forme d'une liste v de formules atomiques signées anciennes, n de formules atomiques signées nouvelles, et exp de formules à expanser. Lorsque nous expansons une formule de exp, nous commençons par ajouter toutes les formules de n à v, et nous réinitialisons n à la liste vide; ensuite, nous expansons la formuleet mettons les sous-formules engendrées dans exp pour celles qui ne sont pas atomiques, ou dans n sinon. Nous pouvons ensuite nous restreindre à n'essayer de fermer le chemin courant qu'en unifiant une formule de n avec une formule de signe opposé dans v.

1.2 Étapes d'extension

Étant donné k, nous sommes maintenant capables de décider si $\longrightarrow \Psi_1, \ldots, \Psi_k$ a une instance prouvable en $\mathbf{LK}_0$. Comment devinons-nous k ?

La réponse est que nous ne le pouvons pas. En tous cas, nous ne pouvons pas calculer le k le plus petit dont nous aurions besoin, ou même un majorant quelconque de ce k. La raison est que la logique du premier ordre est indécidable. En effet, supposons que nous puissions calculer une valeur de k à partir de la formule existentielle Φ, de sorte que si Φ est prouvable, alors $\longrightarrow \Psi_1, \ldots, \Psi_k$ a une instance prouvable. Alors nous pourrions décider si Φ est prouvable en calculant un tel k, puis en appliquant l'algorithme (qui termine) décrit dans la section précédente. Ceci nous fournirait un moyen pour décider de la prouvabilité de Φ par machine, ce qui est impossible.

Il nous faut donc réellement deviner k. Une façon de faire est d'utiliser l'*approfondissement itératif* ("iterative deepening"). Initialiser k à 1, et appliquer la procédure de tableaux pour savoir si $\longrightarrow \Psi_1$ a une instance prouvable. Si oui, alors Φ est prouvée. Sinon, incrémenter k et réappliquer la procédure pour savoir si $\longrightarrow \Psi_1, \Psi_2$ a une instance prouvable. Si oui, alors Φ est prouvée. Sinon, incrémenter k, et ainsi de suite.

L'approfondissement itératif combiné avec la recherche en profondeur d'abord comme dans la section précédente, est une façon commune de coder les tableaux. Comme nous l'avons déjà dit dans le cadre de la résolution, la recherche en profondeur d'abord avec approfondissement itératif ("iterative deepening depth-first search" en anglais) est une solution raisonnable comparée à la recherche en largeur, en ce qu'elle économise quantité de mémoire. Malheureusement, le temps nécessaire à prouver des formules croît en général exponentiellement en fonction de k et de la taille de la formule, de sorte que seuls de petits théorèmes faciles peuvent être prouvés à l'aide de cette stratégie.

Pour représenter l'incrémentation de k, nous ajoutons un *étape d'extension* à la procédure d'expansion de tableau : à tout moment dans la preuve, nous avons le choix d'ajouter une nouvelle instance $\Psi\rho$ de Ψ au chemin courant, où ρ renomme toutes les variables libres de Ψ en de nouvelles variables. Nous pouvons alors faire appel à des heuristiques pour favoriser certaines étapes d'extension, dans l'espoir que nous pourrons ainsi clore le chemin plus vite qu'en se contentant les formules présentes dans le chemin courant. Le seul point à surveiller est l'*équité* du processus : nous ne devons pas appliquer les étapes d'extension en priorité indéfiniment longtemps, sinon la procédure deviendrait incomplète. Trouver les bonnes heuristiques est une tâche très compliquée, et c'est pourquoi nous ne nous y lancerons pas.

2 Tableaux à variables libres

2.1 Présentation

Une amélioration de la méthode des tableaux de base que nous avons présentée en section 1 est la méthode des *tableaux à variables libres*, comme Melvin Fitting l'appelle [Fit83].

Dans les tableaux à variables libres, les étapes d'extension sont intégrées dans le processus de recherche de preuve lui-même, en utilisant des versions paresseuses des

γ-règles

γ	γ_1
$+\exists x \cdot \Phi$	$+\Phi[x'/x]$
$-\forall x \cdot \Phi$	$-\Phi[x'/x]$

(où x' est une nouvelle variable)

Figure 2: γ-règles en tableaux à variables libres

règles $\exists$R et $\forall$L de **LK**, de la même façon que l'on pouvait expliquer la résolution en termes d'une règle de coupure paresseuse. Ces règles sont :

$$\frac{\Gamma \longrightarrow \Delta, \exists x \cdot \Phi, \Phi[x'/x]}{\Gamma \longrightarrow \Delta, \exists x \cdot \Phi}\,\exists\mathrm{R}' \qquad \frac{\Gamma, \forall x \cdot \Phi, \Phi[x'/x] \longrightarrow \Delta}{\Gamma, \forall x \cdot \Phi \longrightarrow \Delta}\,\forall\mathrm{L}'$$

où x' est une nouvelle variable, qui représente le terme t dans la règle correspondante de **LK**. Nous avons vu que, au lieu de deviner t, la méthode des tableaux le calculait en liant x' à t lors du calcul d'un unificateur servant à clore un chemin (de façon équivalente, pour faire du séquent une instance de Ax). Ceci se traduit en ce qu'on appelle les γ-règles de la méthode des tableaux à variables libres : voir la figure 2.

Nous pouvons aussi adapter les règles $\forall$R et $\exists$L, de sorte qu'il ne sera pas nécessaire d'herbrandiser la formule à prouver dans une étape de pré-traitement. Une façon de la faire est d'introduire de nouvelles versions de $\forall$R et $\exists$L, du style des règles ci-dessus, mais soumises à la condition que la variable nouvellement introduite x' *ne doit pas être liée à aucun terme pendant l'unification.* Nous pouvons faire respecter cette condition en créant une nouvelle constante plutôt qu'une nouvelle variable. Ceci mène aux règles suivantes :

$$\frac{\Gamma \longrightarrow \Delta, \Phi[c/x]}{\Gamma \longrightarrow \Delta, \forall x \cdot \Phi}\,\forall\mathrm{R}' \qquad \frac{\Gamma, \Phi[c/x] \longrightarrow \Delta}{\Gamma, \exists x \cdot \Phi \longrightarrow \Delta}\,\exists\mathrm{L}'$$

où c est une nouvelle constante.

Ceci ne suffit pas encore, cependant, parce que nous devons être sûrs non seulement que c n'apparaît pas dans Γ ni dans Δ, mais ausi qu'elle n'apparaitra dans aucune instance de Γ et Δ que nous produirons jamais par unification. Considérons en effet le séquent :

$$\longrightarrow \exists x \cdot \forall y \cdot P(x, y) \Rightarrow P(y, x)$$

qui n'est pas valide : prendre pour P la relation $\leq$ sur les entiers, par exemple. Si l'on applique $\exists$R$'$, $\Rightarrow$R et $\forall$R$'$ sans plus de ménagement, on obtient le séquent :

$$P(x', c) \longrightarrow P(c, x')$$

que l'on peut clore en liant x' à c.

Pour empêcher cela, nous maintenons une base de données de *dépendances*. Chaque fois que nous créons une nouvelle constante c par les règles $\forall$R$'$ ou $\exists$L$'$, nous enregistrons les *contraintes de dépendances* $c < y$ pour toute variable libre y dans le séquent courant : ceci formalise la condition de bord sur la variable universelle c, comme quoi c ne doit pas dépendre de y.

δ-règles

δ	δ_1
$+\forall x \cdot \Phi$	$+\Phi[c/x]$
$-\exists x \cdot \Phi$	$-\Phi[c/x]$

(où c est une nouvelle constante, et où nous ajoutons la contrainte $c < y$ pour chaque variable libre y dans le chemin courant.)

Figure 3: δ-règles en tableaux à variables libres

Il nous faut ensuite étendre le test d'occurrence de l'unification pour qu'il gère ces contraintes de dépendances supplémentaires : disons qu'un terme t dépend de x si et seulement si $t = x$, ou t est une constante c telle que la contrainte $c < x$ soit dans la base de données, ou $t = f(t_1, \ldots, t_n)$ et un t_i, $1 \leq i \leq n$, dépend de x. Nous exigeons ensuite que les unificateurs les plus généraux soient trouvés comme des substitutions σ telles que pour tout $x \in \text{dom } \sigma$, $x\sigma$ ne dépend d'aucune variable de dom σ.

En jargon des tableaux, les règles $\forall \mathrm{R}'$ et $\exists \mathrm{L}'$ donnent lieu aux δ-règles de la figure 3.

Nous pouvons désormais étendre la notion de chemins des tableaux à variables libres comme suit : si C est un chemin contenant une formule Φ de type γ, alors $C \cup \{\gamma_1\}$ est aussi un chemin, où γ_1 est défini comme dans la figure 2; et si C est un chemin contenant une formule Φ de type δ, alors $C \setminus \{\Phi\} \cup \{\delta_1\}$ est encore un chemin, où δ_1 est défini comme en figure 3. Une *stratégie d'expansion équitable* est alors une stratégie qui n'effectue pas une infinité d'étapes consécutives de γ-expansions (autrement dit, d'étapes d'extension), et, comme en section 1, toute stratégie d'expansion équitable définit une procédure de recherche de preuve correcte et complète.

2.2 Herbrandisation au vol

On peut coder la base de données des dépendances des tableaux à variables libres de façon plus directe en produisant non pas des constantes c associées à des contraintes $c < y_1$, $\ldots c < y_n$, mais un terme unique $f(y_1, \ldots, y_n)$, où la dépendance est codée directement à l'intérieur du terme $f(y_1, \ldots, y_n)$. Ce n'est qu'une autre façon d'effectuer une herbrandisation, mais au vol.

> L'ajout de nouvelles constantes ou de nouveaux symboles de fonction ne pose aucun problème en logique du premier ordre. Mais ce n'est pas le cas en logique d'ordre supérieur, parce que l'instantiation de variables d'ordres supérieurs peut propager l'introduction de constantes ou de variables dans d'autres formules. En fait, nous ne pouvons pas herbrandiser : dans ce cas, nous devons nous en tenir aux tableaux à variables libres, et même ne pas considérer c comme des constantes, mais comme des variables que l'on ne peut pas remplacer par d'autres termes [Koh95].

Il y a une autre raison pour laquelle nous souhaitons éviter de fabriquer de nouvelles constantes comme c, et c'est que cette méthode peut être très inefficace. La raison est que deux constantes c et c' créées à des instants différents lors du processus d'expansion ne s'unifient jamais. Ceci peut empêcher la procédure de tableaux d'arriver à clore des chemins rapidement, alors qu'il aurait été possible de le faire en instanciant la formule Φ de façon plus intelligente.

Ce que les δ-règles réalisent est juste l'herbrandisation ou la skolémisation; regardons donc la preuve du théorème de Herbrand-Skolem syntaxique plus en détail : pour expanser une formule de type δ, disons $+\forall x \cdot \Phi$, nous devons remplacer x dans Φ par un terme t, tel que des instances différentes de $\forall x \cdot \Phi$ doivent produire

δ^{++}-règles

δ	δ_1
$+\forall x \cdot \Phi$	$+\Phi[f(x_1, \ldots, x_n)/x]$
$-\exists x \cdot \Phi$	$-\Phi[f(x_1, \ldots, x_n)/x]$

$(\mathrm{fv}(\Phi) \setminus \{x\} = \{x_1, \ldots, x_n\},\ f = \lambda x_1 \cdot \ldots, \lambda x_n \cdot \epsilon x \cdot \Phi)$

Figure 4: δ^{++}-règles en tableaux à variables libres

des instances différentes de t. Dans les règles ci-dessus, nous avons produit un nouveau terme t (sous la forme d'une constante nouvelle) à chaque fois que nous avons instancié Φ, de sorte que la condition est vérifiée. Mais nous pouvons faire mieux, et en particulier utiliser la δ^{++}-règle de la figure 4, due à Peter Schmitt, Reiner Hähnle et Bernhard Beckert [BHS93]. Cette règle correspond aux règles de séquents suivantes :

$$\frac{\Gamma \longrightarrow \Delta, \Phi[f(x_1, \ldots, x_n)/x]}{\Gamma \longrightarrow \Delta, \forall x \cdot \Phi} \forall\mathrm{R}^+ \qquad \frac{\Gamma, \Phi[f(x_1, \ldots, x_n)/x] \longrightarrow \Delta}{\Gamma, \exists x \cdot \Phi \longrightarrow \Delta} \exists\mathrm{L}^+$$

où $\mathrm{fv}(\Phi) \setminus \{x\} = \{x_1, \ldots, x_n\}$, et f est le symbole $\lambda x_1 \cdot \ldots, \lambda x_n \cdot \epsilon x \cdot \Phi$, qui remplacent $\forall$R et $\exists$L. (ϵ est traditionnellement appelé le symbole de choix de Hilbert.)

Pour construire le symbole de fonction de Skolem f, d'abord, nous renommons toutes les variables de Φ en des variables de noms canoniques; ensuite, nous fabriquons l'expression $\lambda x_1 \cdot \ldots, \lambda x_n \cdot \epsilon x \cdot \Phi$, où $x_1, \ldots, x_n$ sont les noms canoniques des variables, et Φ est supposée renommée. Une telle expression, où λ est un opérateur lieur (comme $\forall$ ou $\exists$), se représente sous forme arborescente comme les formules. Considérons maintenant cette expression, qui n'est ni un terme ni une formule de la logique du premier ordre, comme un nouveau symbole : il s'agit du symbole de Skolem souhaité.

Il peut sembler surprenant que nous décidions qu'une expression tout entière, et en fait même une λ-expression, soit considérée comme un symbole, car nous sommes habitués à penser aux symboles comme à de simples chaînes de caractères; mais les symboles sont totalement arbitraires, et la seule contrainte sur les symboles est qu'il doit y en avoir une infinité.

On peut aussi se demander pourquoi nous produisons des symboles aussi compliqués d'apparence, alors qu'il suffisait de produire une constante nouvelle. Comme nous l'avons dit plus haut, fabriquer de nouvelles constantes peut augmenter de façon considérable la taille de la preuve à trouver. La δ^{++}-règle est une des meilleures en termes de longueur de preuve. De plus, la production de nouveaux symboles de fonction de cette façon est très simple à réaliser, car nous avons déjà tout ce qu'il faut pour construire et comparer des formules.

Cette technique est important en particulier quand deux sous-formules quantifiées d'une formule à prouver sont identiques, ou quand elle contient des équivalences logiques. Par exemple, supposons que nous souhaitions prouver la formule :

$$\Big(P(a) \Leftrightarrow \big((\exists x \cdot Q(x)) \Leftrightarrow P(a)\big)\Big) \Rightarrow \exists x \cdot Q(x)$$

Si nous expansons le tableau correspondant, nous obtenons :

$$
\begin{array}{c}
(1)\ \cancel{-P(a) \Leftrightarrow ((\exists x \cdot Q(x)) \Leftrightarrow P(a))} \\
(2)\ + \exists x \cdot Q(x) \\
(3)\ + Q(x_1) \quad (\gamma \text{ de } (2)) \\
\begin{array}{c|c}
\begin{array}{c}
(6)\ + P(a) \\
(8)\ \cancel{+(\exists x \cdot Q(x)) \Leftrightarrow P(a)} \\
\begin{array}{c|c}
\begin{array}{c}
(10)\ \cancel{-\exists x \cdot Q(x)} \\
(12)\ + P(a) \\
(14)\ - Q(c) \\
(\delta^{++} \text{ de } (10))
\end{array}
&
\begin{array}{c}
(11)\ + \exists x \cdot Q(x) \\
(13)\ - P(a) \\
\\
\\
\end{array}
\end{array}
\end{array}
&
\begin{array}{c}
(7)\ - P(a) \\
(9)\ \cancel{-(\exists x \cdot Q(x)) \Leftrightarrow P(a)} \\
\begin{array}{c|c}
\begin{array}{c}
(15)\ \cancel{-\exists x \cdot Q(x)} \\
(17)\ - P(a) \\
(19)\ - Q(c) \\
(\delta^{++} \text{ de } (15))
\end{array}
&
\begin{array}{c}
(16)\ + \exists x \cdot Q(x) \\
(18)\ + P(a) \\
\\
\\
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
$$

où (6), (7), (8), (9) ont été obtenues par des règles α et β à partir de (1); (10), (11), (12), (13) ont été obtenues par des règles α et β à partir de (8), et (15), (16), (17), (18) ont été obtenues par des règles α et β à partir de (9). Remarquons que la quantification $-\exists x \cdot Q(x)$ en (10) et en (15) est associée au même symbole de Herbrand-Skolem c en (14) et en(19) : ce tableau peut ensuite être clos par la substitution $[c/x_1]$. Mais si nous avions créé des constantes nouvelles pour herbrandiser (10) et (15), (14) serait de la forme $-Q(c)$ et (19) serait de la forme $-Q(c')$, où c et c' seraient deux constantes distinctes, et nous ne pourrions pas clore le tableau sans réappliquer d'abord une étape d'extension (une γ-règle) sur (2) pour obtenir suffisamment de variables pouvant être liées à toutes les constantes que nous avons créées.

Théorème 2 (Correction, complétude) *Soit Φ une formule close. Φ est prouvable en* **LK** *si et seulement s'il existe une substitution σ qui clôt une expansion du tableau initial $+\Phi$.*

Preuve : Soit $\mathbf{LK}^+$ le système de séquents formé par l'ajout des règles $\forall\mathrm{R}^+$, $\exists\mathrm{L}^+$, $\forall\mathrm{L}^+$ et $\exists\mathrm{R}^+$ à $\mathbf{LK}_0$ sans coupure. L'expansion de tableau revient à développer des preuves en $\mathbf{LK}^+$ à partir du but (du bas vers le haut).

Nous devons donc montrer que nous pouvons transformer toute preuve π de **LK** sans coupure en un couple (π', σ), où π' est une dérivation dans $\mathbf{LK}^+$ et σ est une substitution qui instancie les séquents sans prémisses de π' en des instances de Ax. L'argument est par récurrence sur la profondeur de π. Les cas intéressants sont :

- quand nous traduisons une dérivation se terminant en une application de $\exists$R (resp. $\forall$L) : alors nous avons dérivé $\Gamma \longrightarrow \Delta, \exists x \cdot \Phi$ de $\Gamma \longrightarrow \Delta, \Phi[t/x]$, qui est dérivé par une preuve strictement moins profonde. Affaiblissons ce dernier pour obtenir $\Gamma \longrightarrow \Delta, \exists x \cdot \Phi, \Phi[t/x]$, que nous traduisons par hypothèse de récurrence en une dérivation dans $\mathbf{LK}^+$ de $\Gamma \longrightarrow \Delta, \exists x \cdot \Phi, \Phi[t/x]$ et une substitution σ'; il s'agit aussi d'une dérivation dans $\mathbf{LK}^+$ de $\Gamma \longrightarrow \Delta, \exists x \cdot \Phi, \Phi[x'/x]$, où x' est une nouvelle variable, pour la substitution $\sigma'[t/x']$; appliquons ensuite $\exists\mathrm{R}'$ pour en déduire la dérivation désirée dans $\mathbf{LK}^+$ et la substitution $\sigma = \sigma'[t/x']$. (Idem pour $\forall$L.)

- quand nous traduisons une dérivation se terminant par une application de $\forall$R (resp. $\exists$L) : alors nous avons dérivé $\Gamma \longrightarrow \Delta, \forall x \cdot \Phi$ de $\Gamma \longrightarrow \Delta, \Phi[x'/x]$, avec x' une nouvelle variable. Ce dernier séquent est dérivé par une preuve strictement moins profonde. Remplaçons x' dans toute la preuve par le terme de Herbrand-Skolem $f(x_1, \ldots, x_n)$, où $x_1, \ldots, x_n$ sont les variables libres de Φ moins x, alors nous en obtenons une preuve de même profondeur de $\Gamma \longrightarrow \Delta, \Phi[f(x_1, \ldots, x_n)/x]$, que nous traduisons par hypothèse de récurrence en une dérivation dans $\mathbf{LK}^+$ du même séquent, avec comme substitution σ. Appliquons ensuite $\forall\mathrm{R}^+$ pour en déduire la dérivation désirée dans $\mathbf{LK}^+$ et la substitution σ. (De même pour $\exists$L.)

Réciproquement, nous devons montrer que nous pouvons transformer tout couple (π', σ), où π' est une dérivation dans $\mathbf{LK}^+$ et σ instancie tous les séquents sans prémisses de π' en des instances de Ax, en une preuve de **LK**. Ceci se montre comme avant, sauf que dans les cas de $\forall$R et de $\exists$L, nous réutilisons les arguments de la preuve du théorème de Herbrand-Skolem syntaxique pour remplacer les termes de Herbrand-Skolem par des variables et réobtenir une preuve dans **LK**. $\square$

On peut renforcer ce théorème en disant que, si une telle substitution σ existe, alors nous pouvons en trouver une en cherchant un unificateur simultané de toutes les paires de formules complémentaires sur les chemins du tableaux expansé de $+\Phi$, comme en section 1. Ceci termine notre justification des tableaux à variables libres.

2.3 Discussion

Le premier avantage qu'ont les tableaux à variables libres sur les tableaux de la section 1 est qu'il n'est plus nécessaire d'effectuer un quelconque pré-traitement sur la formule que nous souhaitons prouver. Ce n'est pas tellement un avantage en logique classique, car nous referons le travail d'herbrandisation et de skolémisation pendant la preuve plutôt qu'avant. Mais c'est crucial si l'on désire adapter la méthode aux logiques non classiques comme la logique intuitionniste. Observons en particulier que les tableaux à variables libres opèrent essentiellement de la même façon en logique intuitionniste : nous devons juste vérifier qu'aucun chemin n'a jamais plus d'une formule positive. Rappelons aussi que la conversion en forme prénexe ou la skolémisation à l'avance ne préservent pas la prouvabilité en logique intuitionniste, de sorte que les tableaux à variables libres sont essentiellement la seule méthode raisonnable. L'argument serait tout aussi vrai pour les logiques linéaire, de la pertinence ("relevant logic") ou les logiques modales.

Le deuxième avantage des tableaux à variables libres est que la recherche d'une preuve est confinée à une recherche de preuves sans coupures, ce qui forme un espace beaucoup plus étroit que l'espace des preuves générales. Ce n'est pas encore aussi réduit que l'espace exploré par les raffinements les plus efficaces de la résolution en logique classique, mais nous verrons en section 3.4 qu'il n'est pas besoin de beaucoup contraindre les tableaux pour obtenir un des raffinements les plus puissants de la résolution, l'*élimination de modèles*.

Le troisième avantage des tableaux à variables libres, qu'ils ont en commun avec la première méthode de tableaux que nous avons présentée dans ce chapitre, est que même si nous insistons pour skolémiser à l'avance, il n'est pas nécessaire de convertir la formule à prouver en forme clausale. C'est un avantage sur la résolution, car le calcul d'une forme clausale peut prendre un temps exponentiel en la taille de la formule d'entrée — bien que ça n'arrive que rarement en pratique.

Les tableaux ont cependant de graves inconvénients. Ils essaient de trouver des preuves sans coupure en développant des preuves de **LK** ou de systèmes proches de **LK** sous forme d'arbres, et aucun partage entre les chemins n'est exploité, à part dans les préfixes de ces chemins. En particulier, une grande partie du travail doit être réeffectuée d'un chemin à l'autre. Ceci est un défaut que les raffinements de la résolution n'ont pas en général, parce que toutes les tentatives de trouver une réfutation sont enregistrées dans un ensemble unique de clauses, et restent donc toujours disponibles pour savoir ce qu'on peut éviter de refaire (typiquement, par le test de subsomption). Pour optimiser les tableaux de cette façon, il nous faudrait conserver des informations sur ce qui a été fait et ce qui a échoué d'un chemin sur l'autre. Le seul point positif dû à cette absence de partage entre les calculs effectués sur différents chemins est que cela facilite la parallélisation de la méthode de tableaux. Cependant, la parallélisation ne suffit pas pour faire des tableaux une méthode plus efficace que des algorithmes séquentiels, et les réalisations parallèles les plus élaborées des tableaux utilisent déjà des techniques supplémentaires pour éviter les redondances entre les chemins.

Un autre défaut des tableaux est que, bien que nous faisions appel à l'unification pour guider la recherche d'une preuve, il est possible que nous ayons à expanser les chemins complètement ou presque complètement avant d'atteindre le point où nous pouvons enfin unifier des formules atomiques. Au moment où nous avons suffisamment expansé les chemins pour calculer des unificateurs non triviaux, il peut nous rester sur les bras un nombre trop grand de chemins pour qu'on puisse tous les clore en temps raisonnable. Il peut en effet y en avoir un nombre exponentiel en la longueur du chemin.

Un dernier défaut des tableaux est qu'il reste une certaine redondance dans l'espace de toutes les preuves sans coupure, due aux permutabilités. En logique classique, ce n'est pas grave : la plupart des règles permutent, et donc nous pouvons utiliser n'importe quelle stratégie d'expansion, au moins pour ce qui est des connecteurs propositionnels. Les seules impermutabilités proviennent des règles sur les quantificateurs, et nous les avons codées par des termes de Skolem (ou des contraintes de dépendances) : l'essence du théorème 2 est en effet que nous avons remplacé **LK**, un système avec trois impermutabilités, par $\mathbf{LK}^+$, un système sans impermutabilités mais avec une condition d'unifiabilité plus complexe. Cette idée de coder les impermutabilitées sous forme de termes de Skolem s'étend aux systèmes de séquents pour les logiques non classiques [Sha92].

3 Connexions, couplages et élimination de modèles

3.1 Connexions et couplages

La *méthode des connexions* a été inventée fin 1979 par Wolfgang Bibel, et à peu près en même temps par Peter Andrews sous le nom de *méthode des couplages* ("method of matings"). Ce sont essentiellement des réalisations légèrement différentes de la méthode de tableaux de base de la section 1.

Le changement principal en est un de vocabulaire et de présentation. En ce qui concerne la présentation, on en choisit une bi-dimensionnelle. Convenons que notre but est de prouver des formules existentielles $\exists x_1 \cdot \ldots \exists x_n \cdot \Psi$, où Ψ est sans quantificateur, et que Ψ soit en forme normale pour la négation, ou nnf. Autrement dit, Ψ est construite à l'aide des seuls connecteurs $\wedge$, $\vee$, $\neg$, et les négations ne s'appliquent qu'à des sous-formules atomiques.

Nous représentons les sous-formules de Ψ qui sont des disjonctions sous forme de colonnes, et les sous-formules qui sont des conjonctions sous forme de lignes. Ceci produit ce qu'on appelle la *représentation matricielle* de Ψ. Par exemple, $A \vee B$ est :

$$\boxed{\begin{array}{c} A \\ B \end{array}}$$

$\neg A \wedge C$ est :

$$\boxed{\begin{array}{cc} \neg A & C \end{array}}$$

et $(\neg A \wedge C) \vee A \vee (B \wedge (\neg C \vee D))$ est représentée par :

$$\boxed{\begin{array}{c} \boxed{\begin{array}{cc} \neg A & C \end{array}} \\ A \\ \begin{array}{cc} B & \boxed{\begin{array}{c} \neg C \\ D \end{array}} \end{array} \end{array}}$$

Cette représentation peut sembler sans importance au premier abord, mais elle a plusieurs vertus. En premier, elle fusionne implicitement deux colonnes qui sont l'une au-dessus de l'autre, et deux lignes qui sont côte à côte : ceci prend en compte l'associativité de $\wedge$ et de $\vee$. En second, elle permet de rendre plus évidente la dualité entre $\wedge$ et $\vee$, comme le montre la définition suivante :

Définition 3 (Chemins) *Soit Ψ une formule sans quantificateur en nnf, représentée sous forme matricielle.*

L'ensemble des chemins verticaux *de Ψ est le plus petit ensemble tel que :*

- *$\{\Psi\}$ est un chemin vertical;*
- *si C est un chemin vertical contenant une formule Ψ' de la forme :*

$$\boxed{\begin{array}{c}\Psi_1\\ \vdots\\ \Psi_m\end{array}}$$

  alors $C \setminus \{\Psi'\} \cup \{\Psi_1, \ldots, \Psi_m\}$ est un chemin vertical;
- *si C est un chemin vertical contenant une formule Ψ' de la forme :*

$$\boxed{\begin{array}{ccc}\Psi_1 & \ldots & \Psi_m\end{array}}$$

  alors $C \setminus \{\Psi'\} \cup \{\Psi_i\}$, $1 \leq i \leq m$, sont des chemins verticaux.

Dans l'exemple de la formule $(\neg A \wedge C) \vee A \vee (B \wedge (\neg C \vee D))$ ci-dessus, les chemins verticaux complètement expansés (ceux qui ne contiennent que des littéraux) sont $\{\neg A, A, B\}$, $\{\neg A, A, \neg C, D\}$, $\{C, A, B\}$, $\{C, A, \neg C, D\}$.

La notion de chemin vertical est tout à fait similaire à celle des chemins dans les tableaux. En fait, on peut transformer tout chemin atomique de tableau en un chemin vertical complètement expansé à condition de convertir les formules $+A$ en A et $-A$ en $\neg A$. Et réciproquement, tout chemin vertical complètement expansé peut être transformé en en chemin atomique de tableau en traduisant A en $+A$ et $\neg A$ en $-A$.

La méthode des connexions procède ensuite comme la méthode des tableaux de la section 1 : nous expansons les chemins verticaux et essayons de les clore tous ensemble par unification; et pour représenter les étapes d'extension, nous ajoutons des versions renommées génériques de Ψ en bas de la matrice courante tout entière, ce qui a pour effet d'étendre tous les chemins verticaux automatiquement.

Une paire complémentaire de littéraux dans un chemin vertical est appelée une *connexion* par Bibel; l'analogue des tableaux, vus comme des ensembles de chemins, est appelé un *pré-couplage* ("pre-mating" en anglais) par Andrews, et un *ensemble couvrant de chemins* ("spanning set of paths" en anglais) par Bibel. Un *ensemble couvrant de connexions* est un ensemble de connexions tel que tout chemin d'un pré-couplage a une connexion dans l'ensemble. Dans la méthode des tableaux, nous cherchions à clore tous les chemins; ici nous cherchons un ensemble couvrant unifiable de connexions. À part le vocabulaire, peu de choses changent comparées à la méthode des tableaux, c'est pourquoi nous ne décrirons pas la méthode plus avant. Son principal avantage est informatique, car nous pouvons représenter les connexions comme des liens dans ce qu'on appelle un *graphe de connexions*, sur lequel diverses opérations combinatoires peuvent être effectués efficacement. Voir [Bib87] pour une description technique complète.

3.2 Comparaison avec la résolution

Nous allons voir qu'il existe des parallèles entre les tableaux et la résolution. La première qu'il nous faut remarquer, cependant, est que la résolution fonctionne uniquement en logique classique (au moins sans quelques modifications), de sorte que la comparaison que nous allons effectuer sera limitée à la logique classique, et en fait à la logique classique du premier ordre.

La second difficulté à laquelle nous faisons face est que la résolution et les tableaux ne résolvent pas le même problème, de la façon dont nous les avons définis. Parce que nous avons tenté de présenter les deux méthodes en les dérivant de systèmes de séquents à la Gentzen, nous avons décrit la résolution comme une méthode de recherche de réfutations (sémantiquement, qui essaie de montrer qu'une formule est insatisfiable), et les tableaux comme une méthode de recherche de preuves (sémantiquement, qui essaient de montrer qu'une formule est valide). Pour les comparer, nous devons donc modifier soit la résolution pour qu'elle devienne une procédure de preuve, soit la méthode des tableaux pour en faire une procédure de réfutation.

Chacune de ces modifications est également facile, parce que Φ est prouvable (resp. valide) si et seulement si $\neg\Phi$ est incohérente (resp. insatisfiable). Mais le processus demande une certaine agilité mentale. Pour des raisons historiques, la résolution et les tableaux sont usuellement présentés comme des procédures de réfutation. Nous pensons que ce n'est pas naturel, au moins dans le cas des tableaux : les tableaux en style de recherche de réfutation nous forcent en général à raisonner en termes de modèles, autrement dit sémantiquement, alors que les tableaux peuvent aussi être utilisés pour des logiques dont la sémantique est encombrante, comme les logiques linéaire ou intuitionniste.

Pour cette raison, et pour montrer qu'en aucun cas les procédures de réfutation ne sauraient se montrer supérieures aux procédures de preuve directe (une erreur courante), nous redonnons la définition de la résolution, mais cette fois-ci en forme de recherche de validité :

Définition 4 (Résolution positive) *Supposons que nous voulions prouver une formule existentielle $\Phi = \exists x_1 \cdot \ldots \exists x_n \cdot \Psi$, où Ψ est en forme normale disjonctive (dnf), c'est-à-dire que Ψ est une disjonction de clauses conjonctives.*

La règle de résolution binaire positive *est :*

$$\frac{C \wedge A \qquad \neg A' \wedge C'}{C\sigma \wedge C'\sigma}\ \sigma = mgu(A, A')$$

où nous supposons que les deux clauses $C \wedge A$ et $\neg A' \wedge C'$ ont été renommées pour n'avoir aucune variable libre en commun. La conclusion de la règle est appelée le résolvant binaire *des deux clauses, les prémisses sont appelées les* clauses parentes*, et A (resp. $\neg A'$) est le littéral sur lequel on résout dans la première clause (resp. la seconde clause).*

Nous disons que les littéraux A et $\neg A'$ sont complementaires *si A s'unifie avec A'.*

La règle de factorisation positive *est :*

$$\frac{C \wedge A \wedge B}{C\sigma \wedge A\sigma}\ \sigma = mgu(A, B) \qquad \frac{\neg A \wedge \neg B \wedge C}{\neg A\sigma \wedge C\sigma}\ \sigma = mgu(A, B)$$

La conclusion de la règle est appelée un facteur binaire *de la clause prémisse.*

Un facteur *d'une clause est soit la clause elle-même, soit un facteur binaire d'un de ses facteurs. Un* facteur propre *d'une clause est un facteur d'un de ses facteurs binaires.*

Un résolvant *de deux clauses est un résolvant binaire de facteurs de chaque clause, non nécessairement propres.*

Une dérivation par résolution positive *est un arbre inversé dont les nœuds sont étiquetés par des clauses, les feuilles sont étiquetées par des clauses de l'ensemble initial de clauses, et les nœuds internes ont comme prédécesseurs leurs clauses parentes.*

Une preuve par résolution positive *est une dérivation se terminant sur la clause vide, autrement dit une dérivation dont la racine est étiquetée par la conjonction vide* **T** *(valant le vrai).*

Tout le chapitre sur la résolution se traduit au cas de la résolution positive. Ici, ce ne sont pas les résolvants qui sont impliqués par leurs clauses parentes, mais le contraire : les résolvants impliquent la disjonction de leur clauses parentes (conjonctives); si nous dérivons **T**, la disjonction des clauses ancêtres de **T** qui sont dans l'ensemble de clauses de départ est valide, ce qui prouve l'ensemble de clauses de départ. Aussi, pour prouver un énoncé Φ modulo un ensemble d'axiomes $\mathcal{A}$, le fait de mettre $\mathcal{A} \Rightarrow \Phi$ en dnf revient à mettre $\neg\mathcal{A}$ et Φ en dnf; c'est ce qui correspond à la conversion de $\mathcal{A}$ et $\neg\Phi$ en cnf, en résolution usuelle (qui est une méthode de réfutation). Finalement, tous les raffinements de la résolution se transposent quasiment sans modification.

Ce qui correspond aux clauses conjonctives de la résolution positive dans les tableaux (ou la méthode des connexions) est la notion de chemins orthogonale à celle que nous avons utilisée en section 3.1 :

Définition 5 (Chemins) *Soit Ψ une formule sans quantificateur en nnf, représentée sous forme matricielle.*

L'ensemble des chemins horizontaux *dans Ψ est le plus petit ensemble tel que :*

- *$\{\Psi\}$ est un chemin horizontal;*
- *si C est un chemin horizontal contenant une formule Ψ' de la forme :*

$$\boxed{\Psi_1 \quad \ldots \quad \Psi_m}$$

alors $C \setminus \{\Psi'\} \cup \{\Psi_1, \ldots, \Psi_m\}$ est un chemin horizontal;

- *si C est un chemin horizontal contenant une formule Ψ' de la forme :*

$$\boxed{\begin{array}{c}\Psi_1\\ \vdots\\ \Psi_m\end{array}}$$

alors $C \setminus \{\Psi'\} \cup \{\Psi_i\}$, $1 \leq i \leq m$, sont des chemins horizontaux.

Par exemple, les chemins horizontaux complètement expansés de notre exemple $(\neg A \wedge C) \vee A \vee (B \wedge (\neg C \vee D))$ sont $\{\neg A, C\}$, $\{A\}$, $\{B, \neg C\}$, $\{B, D\}$. Autrement dit, une dnf de la formule est l'ensemble de clauses conjonctives suivant :

$$\begin{array}{c}\neg A \wedge C\\ A\\ B \wedge \neg C\\ B \wedge D\end{array}$$

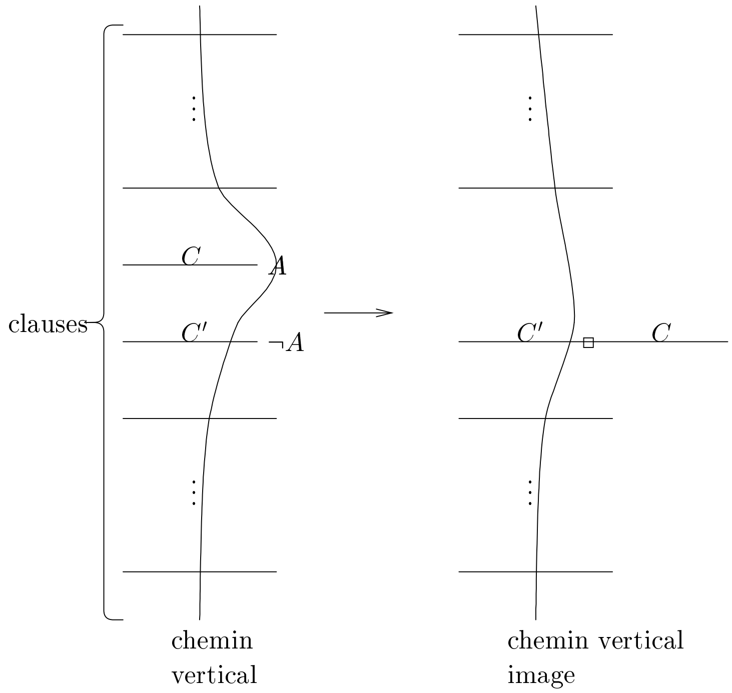

Figure 5: Effet du calcul des résolvants sur les chemins verticaux

En bref, alors que les tableaux essaient de clore des chemins verticaux, la résolution (positive) combine des paires de chemins horizontaux. De plus, la résolution intègre automatiquement les étapes d'extension des tableaux en ajoutant systématiquement les résolvants comme de nouvelles clauses renommées à l'ensemble courant des clauses. D'un autre côté, la résolution fonctionne en représentant explicitement toutes les clauses, alors qu'elles restent implicites dans les tableaux. (But on peut facilement adapter la résolution pour qu'elle fonctionne sur l'ensemble des chemins horizontaux d'une formule sans la convertir explicitement en clauses : c'est la *résolution non clausale*.)

Dans les deux sections qui suivent, nous verrons deux façons d'établir un lien entre la résolution et les tableaux. La première adapte la résolution pour fonctionner comme les tableaux, en dissociant les étapes d'extension des étapes de clôture des chemins. La second adapte les techniques de tableaux à la forme de clauses conjonctives, et fournit ce qui s'appelle la technique d'*élimination des modèles*.

3.3 La V-résolution de Chang

Quand nous créons de nouveaux résolvants par la règle de résolution, nous renommons toutes ses variables libres, de sorte que deux clauses n'aient jamais de variables libres en commun. Implicitement, les clauses C sont quantifiées existentiellement en résolution positive (elles abrègent $\exists x_1 \cdot \ldots \exists x_n \cdot C$), de sorte que la génération d'une instance de la clause avec de nouvelles variables revient exactement au même que l'utilisation d'une γ-règle.

L'idée de Chin-Lian Chang [CL73] est alors de limiter cet effect en ne renommant pas les variables libres des nouvelles clauses. Pour montrer qu'un ensemble S de clauses conjonctives est valide, nous produisons à l'avance un nombre fixé $k(C)$

de versions renommées $C\rho_1, \ldots, C\rho_{k(C)}$ de chaque clause C de sorte qu'aucune paire de clauses n'ait de variable libre en commun, ce qui fournit un nouvel ensemble S' de clauses conjonctives. Nous cherchons ensuite une substitution σ telle que $S'\sigma$ soit propositionnellement valide, en appliquant la règle de résolution propositionnelle, relevée au cas sans quantificateur comme suit. Associons à chaque clause une substitution, initialement vide. Nous appliquons ensuite les règles suivantes de *V-résolution*:

- Factorisation : étant donnée une clause $C \wedge A \wedge B$ associée à la substitution σ, où A et B sont des atomes, θ est le mgu de A et B et σ' est le mgu de θ et σ, remplacer $C \wedge A \wedge B$ par $C\theta \wedge A\theta$; (et de même pour les clauses $C \wedge \neg A \wedge \neg B$;)
- Résolution binaire : étant données deux clauses $C \wedge A$ (de substitution associée σ) et $\neg A' \wedge C'$ (de substitution associée σ'), où θ est le mgu de A et A', θ' est le mgu of σ et σ', et σ'' est le mgu de θ et θ', produire la clause conjonctive $C\theta \wedge C'\theta$, de substitution associée σ''.

Cette procédure est correcte et complète. La complétude en particulier est une conséquence directe du théorème de Herbrand et de la complétude de la résolution propositionnelle.

La factorisation et la résolution binaire doivent être appliquées de façon non déterministe, exactement comme dans le cas des règles d'expansion des tableaux. Pour comparer les deux méthodes, regardons l'effet de la V-résolution sur les chemins verticaux.

Soit S_0 un ensemble de clauses conjonctives. Choisissons un atome A_1 apparaissant dans S_0, et calculons l'ensemble S_1 de tous les résolvants de clauses dans S_0 sur A_1 et $\neg A_1$. S_1 est alors logiquement équivalent à $\forall A_1 \cdot S_0$. (De façon opposée, dans le cas de la réfutation, nous avons vu que le calcul de tous les résolvants sur un atome revenait à calculer $\exists A_1 \cdot S_0$, et qu'il s'agissait de la règle de séparation de Davis-Putnam.) Regardons maintenant de quelle façon les chemins verticaux complètement expansés dans S_0 se retrouvent transformés dans S_1. Consulter la figure 5 pour une illustration de ce qui arrive, au cas où le chemin vertical considéré passe par A mais pas par $\neg A$. En général, étant donné un chemin vertical P dans S_0 qui ne contient pas à la fois A et $\neg A$, nous trouverons un ou plusieurs chemins dans S_0 dans lesquels $P \setminus \{A, \neg A\}$ est inclus. D'un autre côté, si P contient à la fois A et $\neg A$, alors il ne correspondra à aucun chemin vertical dans S_1. (Sur la figure, un tel chemin à droite devrait rencontrer la clause résolvante $C' \wedge C$ à un endroit — un littéral — ou il rencontrait déjà C ou C' sur le côté gauche de la figure, mais un tel chemin n'existe pas.) Autrement dit, le calcul de l'ensemble de tous les résolvants sur un atome A clôt tous les chemins verticaux que l'on peut clore sur A et $\neg A$, tout en augmentant en général le nombre de chemins restant à clore. (Mais n'oublions pas que, comme les chemins restent implicites, ils sont automatiquement partagés : nous n'avons jamais deux copies du même chemin.)

La V-résolution n'est pas particulièrement intéressante en pratique. En effet, elle combine les pires aspects des tableaux (le non-déterminisme de la recherche, où des chemins différents peuvent être redondants mais seront cependant explorés de façon totalement indépendante) et les pires aspects de la résolution (l'utilisation de la résolution propositionnelle comme processus de base de recherche de preuve propositionnelle). Elle est cependant intéressante comme exemple des connexions qui peuvent exister entre les tableaux et la résolution.

3.4 Élimination de modèles

Une interaction beaucoup plus intéressante entre les tableaux et la résolution est la procédure d'élimination de modèles de Donald Loveland, aussi connue sous le nom d'OL-resolution, une forme de résolution linéaire sur des clauses ordonnées. D'autre part, nous pouvons la voir comme une variante de la méthode des tableaux de base, qui travaille sur des formules en nnf.

Supposons que nous voulions prouver un ensemble de clauses conjonctives S, contenant une clause but distinguée $L_1 \wedge \ldots \wedge L_k$. Soit S' l'ensemble S moins cette clause but. Soit T un tableau, autrement dit un ensemble de chemins. Initialement, T consiste uniquement en les chemins unitaires $\{L_1\}$, ..., $\{L_k\}$. Soit aussi σ une substitution, initialement vide. L'élimination de modèles fonctionne par transformation du tableau et de la substitution en utilisant des clauses de S' pour étendre les chemins du tableau T, par les deux règles suivantes :

- Extension : prendre un chemin P de T, et un littéral A (resp. $\neg A$) de P; choisir de façon non déterministe une clause conjonctive $\neg A' \wedge L_1 \wedge \ldots L_n$ (resp. $A' \wedge L_1 \wedge \ldots \wedge L_n$) dans S' (dont les variables libres ont été renommées en de nouvelles variables) telle que $A\sigma$ et $A'\sigma$ soient unifiables. Ensuite, soit σ' le mgu obtenu, remplacer σ par $\sigma\sigma'$, et remplacer P par les chemins $P \cup \{L_i\}$, $1 \leq i \leq n$ dans T. En bref :

$$\frac{\overbrace{P_1, \ldots, P_m, P}^{T} / \sigma \qquad \overbrace{L \wedge L_1 \wedge \ldots L_n}^{\text{dans } S'}}{P_1, \ldots, P_m, P \cup \{L_1\}, \ldots, P \cup \{L_n\} / \sigma\sigma'} \ \sigma' = mgu(\neg L'\sigma, L\sigma), L' \in P$$

- Réduction : prendre un chemin P de T, et un littéral A (resp. $\neg A$) de P; choisir de façon non déterministe un littéral $\neg A'$ (resp. A') dans P tel que $A\sigma$ et $A'\sigma$ soient unifiables. Ensuite, soit σ' le mgu obtenu, remplacer σ par $\sigma\sigma'$, et effacer P de T. En bref :

$$\frac{\overbrace{P_1, \ldots, P_m, P}^{T} / \sigma}{P_1, \ldots, P_m / \sigma\sigma'} \ \sigma' = mgu(\neg A'\sigma, A\sigma), A \in P, \neg A' \in P$$

Il est entendu que, lorsque nous disons "prendre" ci-dessus, il s'agit de prendre une objet via une stratégie arbitraire. L'ordre dans lequel nous étendons ou réduisons les chemins n'a aucune importance. Par contre, quand nous disons "choisir de façon non déterministe", l'ordre peut être important, mais il est en général inconnu : c'est là que la procédure est réellement non déterministe, et peut avoir besoin de rebrousser chemin.

L'élimination de modèles est une procédure de preuve correcte et complète, mais nous ne le montrerons pas. À la place, nous allons montrer les relations entre l'élimination de modèles, les tableaux et la résolution linéaire. Nous invitons le lecteur à adapter les preuves de correction et de complétude des tableaux ou de la résolution linéaire à l'élimination de modèles après avoir lu les sous-sections qui suivent.

3.4.1 L'élimination de modèles, une méthode de tableaux

Nous pouvons interpréter ces règles comme des règles de tableaux. L'extension consiste à effectuer une γ-expansion sur une clause de S', suivie d'une α-expansion sur le connecteur n-aire conjonctif $\wedge$. La β-expansion est traitée automatiquement par le fait de prendre la clause à l'intérieur de l'ensemble S' : bien que les autres clauses n'apparaissent pas sur le chemin P, elles n'en ont pas besoin, car nous ne nous intéressons qu'aux chemins consistant en des littéraux. Plus exactement, comme nous fermons les chemins en n'unifiant que des formules signées atomiques, la seule partie qui nous intéresse dans les chemins est l'ensemble des formules atomiques signées qui s'y trouvent.

D'autre part, les étapes de réduction sont l'analogue de la règle de clôture des tableaux. Mais tous les chemins ne sont pas clos par des étapes de réduction, puisque l'extension d'un chemin par une clause unitaire ($m = 0$) a aussi pour effet de le clore.

Les contraintes que l'élimination de modèles impose au processus d'expansion des tableaux sont les suivantes. D'abord, nous devons deviner une clause but; ceci veut dire que nous devons initialiser l'expansion du tableau en β-expansant l'ensemble S des clauses, et en choisissant une des clauses à α-expanser. Ensuite, toutes les autres expansions sont fortement contraintes : nous exigeons que, chaque fois que nous expansons une clause, au moins un des cheminus obtenus est immédiatement clos. (Il s'agit de la condition d'unification de la règle d'extension.) Finalement, comme en résolution, toute α-expansion d'une clause est précédée par une γ-expansion systématique, autrement dit par une renommage de ses variables libres.

Pour illustrer le processus, considérer l'ensemble de clauses :

$$\begin{array}{c} A \wedge B \\ A \wedge \neg B \\ \neg A \wedge B \\ \neg A \wedge \neg B \end{array}$$

Choisissons par exemple $A \wedge B$ comme clause but. Le tableau initial est alors :

$$\begin{array}{c|c} A & B \end{array}$$

Étendons le chemin de gauche en utilisant par exemple la clause $\neg A \wedge B$, ce qui donne :

$$\begin{array}{c|c} \begin{array}{c} A \\ \begin{array}{c|c} \neg A & B \\ (*) & \end{array} \end{array} & B \end{array}$$

où l'extension sur A vient de clore le chemin le plus à gauche. Nous représentons cela par le signe $*$. Nous continuons par exemple en étendant le chemin non clos le plus à gauche par $\neg A \wedge \neg B$, ce qui donne :

$$\begin{array}{c|c} \begin{array}{c} A \\ \begin{array}{c|c} \begin{array}{c} \neg A \\ (*) \end{array} & \begin{array}{c} B \\ \begin{array}{c|c} \neg A & \neg B \\ & (*) \end{array} \end{array} \end{array} \end{array} & B \end{array}$$

Mais maintenant, nous pouvons clore le chemin non clos le plus à gauche $\{A, B, \neg A\}$ par la règle de réduction. Si nous continuons, nous aboutirons au tableau qui suit, où tous les chemins sont clos, soit par extension (ce qui est signalé par une astérisque $*$) soit par réduction (ce qui est signalé par un petit cercle $\circ$) :

$$\begin{array}{c|c} \begin{array}{c} A \\ \begin{array}{c|c} \begin{array}{c} \neg A \\ (*) \end{array} & \begin{array}{c} B \\ \begin{array}{c|c} \neg A & \neg B \\ (\circ) & (*) \end{array} \end{array} \end{array} \end{array} & \begin{array}{c} B \\ \begin{array}{c|c} \begin{array}{c} A \\ \begin{array}{c|c} \neg A & \neg B \\ (*) & (\circ) \end{array} \end{array} & \begin{array}{c} \neg B \\ (*) \end{array} \end{array} \end{array} \end{array}$$

Bien que l'exemple soit propositionnel, le cas du premier ordre n'est compliqué que par le besoin de maintenir une substitution qui s'applique au tableau tout entier, et qui est calculée par unification.

3.4.2 L'élimination de modèles, une résolution linéaire

Nous pouvons aussi interpréter ces règles comme des règles de résolution linéaire. En effet, un tableau T peut être vu comme une représentation spéciale de la clause courante au sommet : il suffit de lire les littéraux qui restent en bas des chemins non clos du tableau. Par exemple, le premier tableau ci-dessus représente la clause $A \wedge B$ (le tableau initial, de toute façon, doit représenter la clause but), le deuxième représente la clause $B \wedge B$ (avant contraction), et le dernier représente la clause vide.

Alors l'application de la règle d'extension revient à calculer un résolvant binaire entre la clause courante au sommet en tant que clause centrale et une clause d'entrée (une clause de S') en tant que clause latérale : les extensions sont des étapes d'input-résolution.

En fait, la structure de tableau est plus riche que la simple description de la clause courante au sommet. Nous pouvons en effet extraire les tableaux qui avaient été produits avant le tableau courant, sous forme de sous-tableaux du tableau courant. (Un tableau étant un arbre, nous considérons comme sous-tableau n'importe quel sous-arbre du tableau courant.) Alors, au moins dans le cas propositionnel, la règle de réduction peut être vue comme une étape de résolution entre la clause centrale et une clause ancêtre, c'est-à-dire une clause qui n'est pas une clause d'entrée, qui est restée codée sous forme d'un sous-tableau du tableau courant.

Faisons-en la démonstration sur l'exemple ci-dessus. Le premier tableau correspond à la clause but $A \wedge B$. Ensuite, le deuxième tableau correspond à la clause $B \wedge B$, mais le B de gauche peut être remplacé par son littéral parent A, et ainsi nous retrouvons la clause de départ $A \wedge B$. Le troisième tableau représente $\neg A \wedge B$, mais aussi les clauses précédentes $B \wedge B$ et $A \wedge B$. Ensuite, si nous appliquons la règle de réduction sur A et $\neg A$, nous résolvons la clause au sommet $\neg A \wedge B$ avec la clause ancêtre $A \wedge B$. Ceci produit B, qui est exactement le résultat que nous lisons en bas du troisième tableau, une fois que nous avons clos le chemin de gauche par une règle de réduction.

Au premier ordre, la règle de réduction fait quelque chose de légèrement différent en réalité que la résolution binaire avec une clause ancêtre, parce que l'ensemble des variables libres dans le tableau courant est laissé intact. Autrement dit la règle de réduction calcule un résolvant binaire *sans renommage* de la clause centrale ou de la clause ancêtre. De façon étonnante, non seulement ceci ne met pas en danger la complétude de la procédure, mais en fait cela nous permet de nous passer complètement de la règle de factorisation !

Considérons l'exemple traditionnel où nous avions besoin de la factorisation :

$$\begin{gathered} P(x,a) \wedge P(a,x) \\ \neg P(y,a) \wedge \neg P(a,y) \end{gathered}$$

La résolution était incomplète sans la factorisation sur cet exemple, parce que le seul résolvant binaire que nous pouvions produire était $P(a,a) \wedge \neg P(a,a)$. En élimination de modèles, le fait de choisir la première clause comme but par exemple nous mène au tableau initial suivant :

$$P(x,a) \mid P(a,x)$$

Ensuite nous étendons le chemin de gauche par une nouvelle version renommée de la seconde clause, ce qui donne :

$$\begin{array}{c|c|c}
\multicolumn{2}{c|}{P(x,a)} & P(a,x) \\
\neg P(y_1,a) & \neg P(a,y_1) & \\
 & (*) &
\end{array}$$

avec comme substitution $[a/x, a/y_1]$. Appliquons la règle de réduction sur le chemin non clos de gauche, puis étendons le chemin de droite par une version renommée de la deuxième clause de nouveau, et réduisons finalement le chemin de droite. Nous obtenons :

$$\begin{array}{c|c|c|c}
\multicolumn{2}{c|}{P(x,a)} & \multicolumn{2}{c}{P(a,x)} \\
\neg P(y_1,a) & \neg P(a,y_1) & \neg P(y_2,a) & \neg P(a,y_2) \\
(\circ) & (*) & (*) & (\circ)
\end{array}$$

avec comme substitution $[a/x, a/y_1, a/y_2]$. Nous n'avons pas eu besoin de factoriser aucune clause, parce que la variable x que nous lions dans la moitié gauche du tableau et la variable x que nous lions dans la moitié droite sont les mêmes, ce qui a donc pour effet d'instancier la première clause en $P(a,a)$, c'est-à-dire en la clause que nous ne pouvions obtenir que par factorisation de la première clause en résolution classique.

Historiquement, l'élimination de modèles a été inventée dans les années 1970 par Donald Loveland comme un raffinement particulier de la résolution, et diverses formes de résolution linéaire on été développées par Robert Kowalski. Cette procédure a été appelée "élimination de modèles" après que l'on a découvert que ce que la méthode faisait, essentiellement, était d'énumérer des (ensembles d') interprétations sous forme de chemins dans le tableau, et d'ensuite montrer qu'elles n'étaient pas des modèles de la formule but en fermant le chemin. En fait, c'est une façon générale de comprendre les tableaux en tant que méthodes de réfutation. Par dualité, les chemins dans les tableaux réfutationnels sont à voir comme des conjonctions, de sorte que les chemins complètement expansés sont des conjonctions de littéraux. Les instances de tels chemins qui ne contiennent pas à la fois un atome et sa négation sont des interprétations de Herbrand partielles. Pour réfuter le but, nous devons trouver des isntances telles qu'aucun chemin ne peut être vu comme une interprétation partielle de Herbrand, autrement dit nous fermons tous les chemins.

References

[BHS93] Bernhard Beckert, Reiner Hähnle, and Peter H. Schmidt. The even more liberalized δ-rule in free variable semantic tableaux. In Alexander Leitsch, editor, *Third Kurt Gödel Colloquium on Computational Logic and Proof Theory*, Lecture Notes in Computer Science. Springer Verlag, 1993.

[Bib87] Wolfgang Bibel. *Automated Theorem Proving*. Vieweg, second, revised edition, 1987.

[BP94] Bernhard Beckert and Joachim Posegga. leanT^AP : Lean tableau-based theorem proving. In Alan Bundy, editor, *12th International Conference on Automated Deduction*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 793–795, Nancy, France, June 1994. Springer Verlag.

[CL73] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Computer Science Classics. Academic Press, 1973.

[Fit83] Melvin C. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. Reidel Publishing Company, Dordrecht, 1983.

[Koh95] Michael Kohlhase. Higher-order tableaux. In *Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, 1995.

[Sha92] Natarajan Shankar. Proof search in the intuitionistic sequent calculus. In Deepak Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 522–536, Saratoga Springs, New York, USA, June 1992. Springer Verlag.