DEA PROGRAMMATION : Sémantique, Preuves et Langages

Tautologies classiques et combinateurs de contrôle

Emmanuel BEFFARA

Stage encadré par M. Vincent DANOS
Université Paris VII, équipe PPS

septembre 2002

Résumé

On commence par introduire le $\lambda\kappa$-calcul, extension du λ-calcul avec contrôle, dont les règles de typage étendent la correspondance entre preuves et programmes à la logique propositionnelle classique. On rappelle ensuite le formalisme de la réalisabilité qui fournit des modèles de la logique propositionnelle permettant d'étudier le comportement calculatoire que spécifient les types. Dans ce cadre, on montre que certaines tautologies de la logique classique spécifient des mécanismes comparables à des structures de contrôle plus ou moins familières en programmation. De plus, on montre comment le calcul peut être enrichi par des combinateurs qui réalisent ces tautologies tout en faisant gagner de l'expressivité au calcul.

Table des matières

Introduction

Cette étude se place dans une extension du λ-calcul dans laquelle on dispose d'une notion explicite de *contrôle* global. Dans le λ-calcul classique, on définit l'évaluation des termes par la relation de β-réduction ; cette relation étant confluente, toutes les stratégies de réduction sont équivalentes. Le contrôle permet quant à lui d'agir sur le processus de réduction, en autorisant la mémorisation et la restitution de termes dans un état particulier. Plusieurs extensions du λ-calcul ont été proposées pour permettre un tel contrôle. Parmi elles, le $\lambda\mu$-calcul de Parigot [5] a été très étudié. Un autre approche est le λc-calcul, introduit par Griffin [3], qui formalise l'introduction dans le langage Scheme de l'opérateur *call-with-current-continuation*. Le calcul utilisé ici est le $\lambda\kappa$-calcul, défini en détail dans la section 1.1, qui est en fait une reformulation du λc-calcul. Ces modèles de calcul, tous fondamentalement équivalents, sont associés chacun à un système de réduction particulier, en général fondé sur une machine virtuelle spécifique.

L'intérêt d'une telle extension du calcul ne réside pas tant dans l'expressivité du calcul lui-même (on ne peut bien sûr rien calculer de plus qu'avec le λ-calcul) que dans l'enrichissement qui en découle dans le typage. En effet, l'introduction du contrôle dans le calcul permet d'étendre la correspondance de Curry-Howard au cadre de la déduction naturelle classique, alors que le λ-calcul se restreignait au cadre intuitionniste. Pour étudier les modèles de la déduction naturelle engendrés par le λ-calcul avec contrôle, on utilise ici la théorie de la réalisabilité, présentée en section 1.2. Elle se révèle être un outil efficace pour la construction systématique de preuves de spécification. On entend par *spécification* le problème général consistant à étudier le comportement calculatoire commun de toutes les preuves d'un théorème donné (voir section 1.2.3). Ce problème a été étudié dans différents systèmes de types, pour l'étude de différentes théories logiques, dont notamment l'arithmétique de Peano ou la théorie des ensembles de Zermelo-Frænkel [4].

On se pose ici le problème de la spécification, non dans une théorie logique particulière, mais dans le cas de certaines tautologies de la logique classique. On se place donc dans une théorie purement propositionnelle. Les tautologies étudiées sont explicitement classiques, c'est-à-dire fausses en logique intuitionniste. Comme le passage à la logique classique se fait en introduisant les instructions de contrôle, ces tautologies spécifient des *structures de contrôle*, que l'on cherche ici à comprendre et à relier à des constructions connues en programmation. Dans le chapitre 2, on s'intéresse aux tautologies dites disjonctives ; le cas simple du tiers-exclu s'avère spécifier une structure rappelant le traitement des exceptions dans certains langages, et le cas général en est en fait assez proche. Dans le chapitre 3, on étudie ensuite des tautologies sous forme disjonctive ; celles-ci engendrent des comportements plus complexes, qui manquent pour l'instant d'une interprétation calculatoire bien claire.

Je remercie bien entendu Vincent DANOS pour son encadrement bienveillant et enthousiaste, ainsi que ses collègues de l'équipe PPS, notamment Jean-Louis KRIVINE et Olivier LAURENT pour quelques discussions fructueuses, ainsi que Chantal BERLINE pour m'avoir prêté un coin de bureau où travailler. Je remercie également le DEA de Programmation pour m'avoir mené là et le DEA de Logique pour m'avoir écouté présenter mon sujet. Merci aussi à Emmanuel JEANDEL pour ses contributions TEXniques, ainsi qu'à tous ceux que j'aurais eu le malheur d'oublier.

Chapitre 1

Modèle de calcul

Dans ce premier chapitre, on présente le cadre calculatoire dans lequel se place l'étude qui suit. On va dans un premier temps introduire le $\lambda\kappa$-calcul, typé dans le système F enrichi pour le typage des instructions de contrôle. On présente ensuite le formalisme de la réalisabilité telle qu'elle s'applique au $\lambda\kappa$-calcul pour la construction de modèles à partir du typage.

1.1 Le $\lambda\kappa$-calcul

L'extension du λ-calcul dans laquelle on se place dans cette étude est le $\lambda\kappa$-calcul, qui est essentiellement une reformulation du λc-calcul introduit initialement par Griffin [3].

1.1.1 Termes et réductions

Pour introduire la notion de contrôle dans le λ-calcul, il faut pouvoir identifier précisément l'état d'un terme dans le processus de réduction, c'est-à-dire qu'il faut que dans tout terme réductible on puisse identifier la prochaine réduction à effectuer. Pour cela, on introduit un concept de pile, qui représente les arguments du programme. Le calcul ne consiste plus alors à réduire un terme par β-réduction, mais à réduire un terme sur une pile, par un système de règles un peu plus élaboré.

Définition 1 (*termes, piles, exécutables*)

On définit inductivement l'ensemble Λ des termes et l'ensemble Π des piles par :

$$\begin{array}{rl} \textit{termes :} & t ::= x \mid (t)t \mid \lambda x.t \mid \kappa x.t \mid k_\pi \\ \textit{piles :} & \pi ::= \varepsilon \mid t \cdot \pi \end{array}$$

où x est élément d'un ensemble dénombrable V des variables, et où ε représente la pile vide. On appelle exécutable *tout élément de $\Lambda \times \Pi$, et on notera $t * \pi$ au lieu de (t, π).*

Dans ce contexte, le calcul est défini par réduction sur les exécutables ; la réduction aura toujours lieu en tête, c'est-à-dire sur le membre gauche de l'exécutable. Le système de réductions utilisé est déterministe, il définit un calcul en appel par nom. Les règles de réduction sont les

suivantes :

$$\begin{array}{rrcl} \text{[push]} & (t)u * \pi & \succ & t * u \cdot \pi \\ \text{[pop]} & \lambda x.t * u \cdot \pi & \succ & t[u/x] * \pi \\ \text{[save]} & \kappa x.t * \pi & \succ & t[k_\pi/x] * \pi \\ \text{[restore]} & k_\pi * t \cdot \pi' & \succ & t * \pi \end{array}$$

On pourrait aussi présenter ces réductions sous la forme d'une machine abstraite à base d'environnements, comme la KAM pour le $\lambda\mu$-calcul, mais ce n'est pas le propos ici.

Les termes k_π sont des *continuations*, c'est-à-dire qu'ils servent à sauvegarder les piles. Bien que ce calcul fasse des continuations des objets de première classe, on ne s'autorisera pas à les écrire explicitement dans un terme. On dira donc qu'un terme est *pur* s'il ne contient pas de k_π.

1.1.2 Typage

Pour le typage, on se place ici dans le système F étendu au $\lambda\kappa$-calcul, c'est-à-dire que l'algèbre des types se définit inductivement par

$$\tau ::= X \mid \tau \to \tau \mid \forall X\, \tau$$

Dans le cadre logique, les types correspondent naturellement aux formules du calcul des prédicats. Par conséquent, selon le contexte, on parlera indifféremment de *type* ou de *formule* pour désigner les éléments de cette algèbre.

Les règles de typage sont celles du λ-calcul, auxquelles on ajoute la règle du κ qui correspond du point de vue logique à la loi de Peirce :

$$\text{axiome :} \quad \frac{}{\Gamma, x : A \vdash x : A}$$

$$\text{application :} \quad \frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash (t)u : B}$$

$$\text{abstraction :} \quad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}$$

$$\text{continuation :} \quad \frac{\Gamma, x : A \to B \vdash t : A}{\Gamma \vdash \kappa x.t : A}$$

$$\text{introduction du quantificateur :} \quad \frac{\Gamma \vdash t : A \qquad X \notin FV(\Gamma)}{\Gamma \vdash t : \forall X\, A}$$

$$\text{élimination du quantificateur :} \quad \frac{\Gamma \vdash t : \forall X\, A}{\Gamma \vdash t : A[B/X]}$$

Naturellement, ces règles ne permettent donc de typer que des termes purs. En fait, on n'aura pas besoin ici de toute l'expressivité du système F, mais on utilisera plutôt un sorte de système de types simples enrichi avec une constante $\bot$ qui représente l'absurde.

1.1.3 Équivalence avec les calculs similaires

Il existe d'autres extensions du λ-calcul dans lesquelles une notion de contrôle est introduite. Les formalismes varient un peu mais tous ces calculs sont essentiellement équivalents, chacun

pouvant simuler les autres.

En particulier, on montre facilement que le $\lambda\kappa$-calcul est équivalent au λc-calcul, introduit par Griffin [3] en 1990. En effet, dans le λc-calcul, la loi de Peirce est posée comme un axiome en tant que type du terme cc, et la règle de réduction est

$$cc * t \cdot \pi \succ t * k_\pi \cdot \pi$$

donc si l'on pose

$$cc \equiv \lambda f.\kappa x.fx$$

on a bien un terme du bon type et dont la réduction est identique (au nombre de pas de réduction près). Réciproquement, le passage du λc au $\lambda\kappa$ se fait en posant

$$\kappa x.t \mapsto cc(\lambda x.t)$$

On verra par la suite que cette façon d'introduire la loi de Peirce par un $\lambda\kappa$-terme est en fait la seule possible à équivalence près. Le fait d'utiliser ici le lieur κ au lieu de la constante cc est essentiellement un choix cosmétique qui met simplement en évidence la dualité entre abstraction sur les termes et abstraction sur les piles.

Le $\lambda\kappa$-calcul est aussi équivalent au $\lambda\mu$-calcul, introduit par Parigot [5] en 1992, dans sa forme stricte où seule la construction $\mu\alpha[\beta]t$ est autorisée. En effet on peut transformer un $\lambda\kappa$-terme pur en $\lambda\mu$-terme par la transformation

$$\kappa x.t \mapsto \mu\alpha[\alpha](\lambda x.t)(\lambda y.\mu\beta[\alpha]y)$$

dont on vérifie qu'elle conserve le typage ainsi que la réduction des exécutables[1]. Pour la traduction réciproque, on considère simplement les μ-variables comme des variables de termes (dont on sait qu'elles représenteront des continuations). Alors la transformation

$$\mu\alpha[\beta]t \mapsto \kappa\alpha.(\beta)t$$

suffit à obtenir le résultat, et on vérifie que le typage est conservé et que les réductions sont équivalentes.

1.2 Réalisabilité

Rappelons maintenant le formalisme de la réalisabilité, telle qu'elle est définie par Krivine, et que l'on utilisera ensuite pour étudier les tautologies.

1.2.1 Définitions

On suppose défini un ensemble $\perp\!\!\!\perp$ d'exécutables, clos par anti-réduction. On comprend $\perp\!\!\!\perp$, l'ensemble des *observables*, comme l'ensemble des *bons* exécutables, dans un sens que l'on définira selon le besoin. On pourra parfois imposer plus de conditions sur cet ensemble.

L'idée est que l'on ne peut pas, en général, étudier un programme (c'est-à-dire un terme) dans l'absolu, mais que l'on ne peut qu'étudier son évaluation pour des données d'entrée particulières (c'est-à-dire sur une pile particulière). La propriété que l'on *observe* est alors le fait que l'exécution aboutisse ou pas dans l'ensemble $\perp\!\!\!\perp$. C'est alors par le choix de $\perp\!\!\!\perp$ que l'on décide quelle propriété est à observer.

[1] L'évaluation des $\lambda\mu$-termes est définie sur une machine abstraite différente, mais en adaptant les réductions d'un formalisme vers l'autre, on vérifie bien l'équivalence voulue.

Il est alors naturel de se demander, pour un programme donné, quelles sont les données pour lesquelles le programme se comporte bien, c'est-à-dire quel est l'ensemble des piles qui, appliquées à un terme donné, font un exécutable de $\perp\!\!\!\perp$. La question duale est de savoir quels sont les programmes qui se comportent bien sur des données particulières. La formalisation de cette idée est la notion d'orthogonal suivante :

Définition 2 (*orthogonal*)

Pour un ensemble $X \subset \Lambda$ de termes, on note $X^{\perp\!\!\!\perp} = \{\pi \mid \forall t \in X, (t, \pi) \in \perp\!\!\!\perp\}$,
et pour un ensemble $Z \subset \Pi$ de piles, on note $Z^{\perp\!\!\!\perp} = \{t \mid \forall \pi \in Z, (t, \pi) \in \perp\!\!\!\perp\}$.

Cet opérateur est effectivement un orthogonal, au sens où tout ensemble est inclus dans son bi-orthogonal, i.e. $X \subseteq X^{\perp\!\!\!\perp\perp\!\!\!\perp}$ pour tout X, et tout orthogonal est égal au tri-orthogonal correspondant, i.e. $X^{\perp\!\!\!\perp} = X^{\perp\!\!\!\perp\perp\!\!\!\perp\perp\!\!\!\perp}$ pour tout X. De plus, on a des propriétés de monotonie : pour $A \subseteq B$, on a $B^{\perp\!\!\!\perp} \subseteq A^{\perp\!\!\!\perp}$, pour les termes comme pour les piles.

On construit maintenant, pour chaque $\perp\!\!\!\perp$, un modèle de la théorie logique que définit notre système de types. Les objets que l'on considère ici comme des valeurs de vérité sont les ensembles de piles :

Définition 3 (*valeur de vérité*)

Soit $e : V \to \mathcal{P}(\Pi)$ un environnement, on définit la valeur de vérité $[A]_e$ d'une formule (i.e. d'un type) A inductivement par les règles suivantes :

$$\begin{aligned} [X]_e &= e(X) \\ \left[\forall X\, A[X]\right]_e &= \bigcup_{\mathcal{Z} \subseteq \mathcal{P}(\Pi)} [A]_{e[\mathcal{Z}/X]} \\ [A \to B] &= [A]^{\perp\!\!\!\perp} \cdot [B] \end{aligned}$$

Pour les formules closes, on s'autorisera à écrire $[A]$ car la valeur de vérité d'une formule close est indépendante de l'environnement. De cette notion, on déduit celle d'interprétation, qui en est en fait l'orthogonal :

Définition 4 (*interprétation*)

On appelle interprétation *d'une formule A dans un environnement e donné et pour un $\perp\!\!\!\perp$ donné l'ensemble de termes $|A|_e = [A]_e^{\perp\!\!\!\perp}$.*

Définition 5 (*réalisation*)

On dit qu'un terme t réalise *une formule A (dans un environnement $e : V \to \mathcal{P}(\Pi)$ donné), et on note $t \Vdash_e A$, lorsqu'on a $t \in |A|_e$.*

Une fois de plus, dans le cas de formules closes, on s'autorisera à écrire l'interprétation $|A|$ et la réalisation $t \Vdash A$ car la relation est alors indépendante de l'environnement.

1.2.2 Lemme d'adéquation

On a donc défini deux relations entre termes et types : d'une part l'inférence de type $\vdash t : A$ et d'autre part la réalisation $t \Vdash_e A$. Le lemme d'adéquation prouve l'inclusion de la première relation dans la seconde :

Théorème 1

Si le jugement $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ est dérivable, alors pour toute valeur de $\perp\!\!\!\perp$, tout environnement $e : V \to \mathcal{P}(\Pi)$, tous termes $t_1 \in |A_1|_e \ldots t_n \in |A_n|_e$ et toute pile $\pi \in [B]_e$, $t[t_1/x_1, \ldots, t_n/x_n] * \pi \in \perp\!\!\!\perp$.

Démonstration : La preuve se fait par induction sur la dérivation de typage. On note Γ l'environnement de typage $x_1 : A_1, \ldots, x_n : A_n$ et on suppose choisis l'environnement $e : V \to \mathcal{P}(\Pi)$ et les termes $t_i \in |A_i|_e$. On note $\bar{t}$ pour $t[t_1/x_1, \ldots, t_n/x_n]$.

axiome le jugement est de la forme $x : A \vdash x : A$, donc si $t \in |A|_e$, par définition $t * \pi$ est élément $\perp\!\!\!\perp$ pour toute pile π de $[A]_e$;

application soient t et u tels que $\Gamma \vdash t : A \to B$ et $\Gamma \vdash u : A$ soient dérivables. Par hypothèse d'induction, on a donc $\bar{t} \in |A \to B|_e$ et $\bar{u} \in |A|_e$. Soit $\pi \in [B]_e$, par définition $[A \to B]_e = |A|_e \cdot [B]_e$ donc $\bar{u} \cdot \pi$ est élément de $[A \to B]_e$, et par conséquent $\bar{t} * \bar{u} \cdot \pi$ est élément de $\perp\!\!\!\perp$, et comme $\perp\!\!\!\perp$ est clos par anti-réduction on a bien $\overline{(t)u} * \pi \in \perp\!\!\!\perp$;

abstraction supposons $\Gamma, x : A \vdash t : B$ dérivable. Alors, par hypothèse d'induction, pour tout terme $u \in |A|_e$ et toute pile $\pi \in [B]_e$ on a $\bar{t}[u/x] * \pi \in \perp\!\!\!\perp$, d'où par anti-réduction $\overline{\lambda x.t} * u \cdot \pi \in \perp\!\!\!\perp$. Comme l'ensemble $\{u \cdot \pi \mid u \in |A|_e, \pi \in [B]_e\}$ est, par définition, égal à $[A \to B]_e$, on a bien le résultat voulu ;

continuation si $\Gamma, x : A \to B \vdash t : A$ est dérivable, pour tout $u \in |A \to B|_e$ et toute pile $\pi \in [A]_e$ on a par hypothèse d'induction $\bar{t}[u/x] * \pi \in \perp\!\!\!\perp$. Comme $\kappa x.t * \pi$ se réduit en $t[k_\pi/x] * \pi$, il suffit de prouver que k_π est élément de $|A \to B|_e$ pour obtenir le résultat. Soit donc $u \cdot \pi'$ une pile de $[A \to B]_e$, c'est-à-dire telle que $u \in |A|_e$ et $\pi' \in [B]_e$. $k_\pi * u \cdot \pi'$ se réduit alors en $u * \pi$, or $\pi \in A^{\perp\!\!\!\perp}$ donc $u * \pi$ est élément de $\perp\!\!\!\perp$, et par anti-réduction on a bien $k_\pi * u \cdot \pi' \in \perp\!\!\!\perp$, d'où $k_\pi \in [A \to B]_e$, et donc $\kappa x.t \in |A|_e$;

quantification supposons $\Gamma \vdash t : A[X]$ dérivable, X n'étant pas parmi les variables libres de Γ. Par hypothèse d'induction, on prouve donc que pour tout environnement e', toute pile π de $[A[X]]_{e'}$ et tous termes $t'_i \in |A_i|_{e'}$, on a $t[t'_1/x_1, \ldots, t'_n/x_n] * \pi \in \perp\!\!\!\perp$. Soit alors une pile π de $\left[\forall X\, A[X]\right]_e$. Il existe donc une partie $\mathcal{Z}$ de Π telle que π soit élément de $[A[X]]_{e[\mathcal{Z}/X]}$. Comme X n'apparaît pas dans les $|A_i|$, ceux-ci sont indépendants de la valeur de $e(X)$, et chaque terme t_i est élément de $|A_i|_{e[\mathcal{Z}/X]}$, donc l'hypothèse d'induction peut s'appliquer et on a bien $\bar{t} * \pi \in \perp\!\!\!\perp$

Enfin, le cas de l'élimination du quantificateur est trivial. ■

En corollaire, on a donc le cas particulier suivant, dont on remarque qu'il ne s'applique qu'à des formules closes :

Lemme 2 (*lemme d'adéquation*)

Si $\vdash t : A$ est dérivable, alors $t \Vdash A$ est vérifié pour tout $\perp\!\!\!\perp$.

1.2.3 Problème de la spécification

Le problème de la spécification consiste à chercher le comportement que *spécifie* un type donné, c'est-à-dire le comportement commun à tous les termes d'un type particulier.

La réalisabilité est un outil très puissant pour aborder le problème de la spécification. On peut par exemple l'utiliser pour prouver facilement le résultat suivant :

Proposition 3 (*spécification de l'identité*)

Si $\vdash t : \forall X(X \to X)$ est dérivable, alors pour tout terme a et tout pile π, $t * a \cdot \pi \succ a * \pi$.

Démonstration : Appelons I le type $\forall X(X \to X)$. Soit t un terme tel que $\vdash t : I$ soit dérivable. D'après le lemme d'adéquation, on sait donc que t réalise I, c'est-à-dire que pour toute valeur de $\perp\!\!\!\perp$ et toute pile π de $I^{\perp\!\!\!\perp}$, l'exécutable $t * \pi$ est dans $\perp\!\!\!\perp$. Soient alors a un terme quelconque et π une pile quelconque. Si l'on définit $\perp\!\!\!\perp$ comme la clôture par anti-réduction de $\{a * \pi\}$, l'ensemble $I^{\perp\!\!\!\perp}$ contient $|X| \cdot X^{\perp\!\!\!\perp}$ pour toute valuation de $X^{\perp\!\!\!\perp}$. Si l'on choisit $X^{\perp\!\!\!\perp} = \{\pi\}$, comme $a * \pi \in \perp\!\!\!\perp$ on a $a \in |X|$, d'où $a \cdot \pi \in I^{\perp\!\!\!\perp}$, et $t * a \cdot \pi \in \perp\!\!\!\perp$, et par définition de $\perp\!\!\!\perp$ ceci signifie que $t * a \cdot \pi$ se réduit en $a * \pi$. ■

La méthode employée dans la preuve est en fait assez générale : on définit $\perp\!\!\!\perp$ comme l'ensemble des exécutables qui se réduisent de la façon voulue (donc ici qui se réduisent vers une valeur particulière), et on prouve que tous les termes qui réalisent le type considéré se comportent *bien* lorsqu'ils sont appliqués à des données de la bonne forme.

Chapitre 2

Tautologies disjonctives

On se pose ici le problème de la spécification pour des tautologies classiques, c'est-à-dire explicitement non intuitionnistes. Le passage de la logique intuitionniste à la logique classique dans le typage a nécessité l'introduction d'une notion de contrôle dans le calcul, en conséquence les tautologies purement classiques vont spécifier des *structures de contrôle*, faisant apparaître des mécanismes voisins de ceux que l'on rencontre dans les langages de programmation [1, 2].

2.1 Loi de Peirce

Le cas de la loi de Peirce, par exemple, a été évoqué dans l'équivalence entre le $\lambda\kappa$-calcul et le λc-calcul, en montrant que le terme cc la réalisait. En notant $P = \forall A(((A \to \bot) \to A) \to A)$, on peut en fait prouver que tout $\lambda\kappa$-terme qui réalise P se comporte comme cc. L'énoncé précis est le suivant :

Proposition 4 (*spécification de la loi de Peirce*)

Soit c tel que $\vdash c : P$ soit dérivable. Quels que soient la pile π et les termes f et a tels que pour tout α on ait $f * \alpha \cdot \pi \succ \alpha * a \cdot \pi'$ pour une certaine pile π', l'exécutable $c * f \cdot \pi$ se réduit en $a * \pi$.

Démonstration : Soient un terme c, une pile π et deux termes f et a vérifiant les conditions ci-dessus. Posons pour $\perp\!\!\!\perp$ la clôture par anti-réduction de $\{a * \pi\}$ et pour $\mathcal{A}$ le singleton $\{\pi\}$. Par définition, on a alors $a \in |\mathcal{A}|$. Pour tout élément α de $|\mathcal{A} \to \bot|$, par hypothèse $f * \alpha \cdot \pi$ se réduit en un $\alpha * a \cdot \pi'$. Comme $(\mathcal{A} \to \bot)^{\perp\!\!\!\perp} = |\mathcal{A}| \cdot \Pi$, cet exécutable est dans $\perp\!\!\!\perp$, donc $f * \alpha \cdot \pi$ également, d'où $f \in |(\mathcal{A} \to \bot) \to \mathcal{A}|$. Comme $\pi \in \mathcal{A}$ par définition, la pile $f * \pi$ est élément de $(((\mathcal{A} \to \bot) \to \mathcal{A}) \to \mathcal{A})^{\perp\!\!\!\perp}$, et donc de $P^{\perp\!\!\!\perp}$. Comme c est élément de $|P|$ en vertu du lemme d'adéquation, $c * f \cdot \pi$ est alors élément de $\perp\!\!\!\perp$, ce qui signifie qu'il se réduit en $a * \pi$. ■

Ceci signifie en fait que si la fonction f met son argument en tête sur une pile non vide (ce que garantit le typage), $c * f \cdot \pi$ se comporte comme $f * k_\pi \cdot \pi$, or si f ne met pas son argument en tête, le résultat est trivial, donc cette proposition a bien le sens que l'on veut lui donner.

2.2 Le tiers-exclu

Le tiers-exclu, sous forme disjonctive, s'écrit $\forall A(\neg A \vee A)$. Sa reformulation dans notre fragment du système F s'écrit alors :

$$T = \forall A \forall B\ (\neg A \to B) \to (A \to B) \to B$$

où $\neg A$ est synonyme de $A \to \bot$ et donc de $A \to \forall X\, X$.

Un terme de type $\neg A$ est un terme qui, appliqué à un terme de type A, donne un objet de n'importe quel type. Il est facile de voir que, puisque le système de typage utilisé est cohérent, aucun terme pur ne peut avoir cette propriété dès que le type A est habité (et donc vrai d'un point de vue logique). Par conséquent, un terme de type $\neg A \to B$ ne peut être appliqué qu'à un objet contenant une continuation, c'est-à-dire un objet produit en utilisant un κ.

2.2.1 Réalisation par un $\lambda\kappa$-terme

Le type T est une proposition prouvable en logique classique, il existe donc des termes qui la réalisent. Considérons par exemple :

$$M_T \equiv \lambda f.\lambda h.\kappa x.f(\lambda y.x(hy))$$

Ce terme a T pour type, selon la preuve suivante où $\Gamma = x : B \to C, h : A \to B, f : (A \to C) \to B$:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash f : (A \to C) \to B} \quad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : B \to C} \quad \dfrac{\dfrac{}{\Gamma \vdash h : A \to B} \quad \dfrac{}{\Gamma, y : A \vdash y : A}}{\Gamma, y : A \vdash hy : B}}{\Gamma, y : A \vdash x(hy) : C}}{\Gamma \vdash \lambda y.x(hy) : A \to C}}{\Gamma \vdash f(\lambda y.x(hy)) : B}}{h : A \to B, f : (A \to C) \to B \vdash \kappa x.f(\lambda y.x(hy)) : B}}{f : (A \to C) \to B \vdash \lambda h.\kappa x.f(\lambda y.x(hy)) : (A \to B) \to B}}{\vdash \lambda f.\lambda h.\kappa x.f(\lambda y.x(hy)) : ((A \to C) \to B) \to (A \to B) \to B}$$

Étudions le comportement de ce terme M_T. Pour tous termes f et h et tout pile π, on a

$$M_T * f \cdot h \cdot \pi \succ f * \alpha_{h,\pi} \cdot \pi$$

en posant $\alpha_{h,\pi} = \lambda y.(k_\pi(hy))$. Dans le cas où la réduction met ensuite $\alpha_{h,\pi}$ en tête, le typage garantit que la pile sur laquelle il se trouve n'est pas vide, et la règle qui s'applique est

$$\alpha_{h,\pi} * a \cdot \pi' \succ h * a \cdot \pi$$

En considérant ce comportement avec un œil de programmeur, on reconnaît une sorte de structure d'exception. On commence par évaluer f (le corps) avec en argument $\alpha_{h,\pi}$ (l'exception). Si cette évaluation met $\alpha_{h,\pi}$ en tête (lève l'exception) sur une pile avec un terme a au sommet (la donnée passée à l'exception), on évalue h (le gestionnaire d'exception) avec a en argument sur la pile π (dans le contexte de départ). La différence avec les exceptions telles qu'on les trouve par exemple dans Caml est qu'ici le mécanisme est correctement typé (en effet, l'opérateur **raise**, qui a le type absurde exn $\to \alpha$, est un artifice qui sert à permettre le typage des exceptions), assurant que toute exception est rattrapée car il s'agit d'un mécanisme local. Ceci dit, la comparaison avec Caml est à considérer avec prudence, car ce langage fonctionne en appel par valeur, alors que notre calcul est à appel par nom.

2.2.2 Spécification

On peut montrer que ce que spécifie le type T est en fait le comportement de M_T ou un comportement symétrique :

Proposition 5 (*spécification du tiers-exclu*)

Soit c un terme tel que $\vdash c : T$ soit dérivable. Pour tous termes f, h et a et pour toutes piles π, π_f et π_h pour lesquels
- *pour tout α, $f * \alpha \cdot \pi \succ \alpha * a \cdot \pi_f$,*
- *pour tout α, $h * \alpha \cdot \pi \succ \alpha * \pi_h$,*

$c * f \cdot h \cdot \pi$ se réduit en $a * \pi_h$.

Démonstration : Supposons c, f, h, a, π, π_f et π_h définis selon l'énoncé. Définissons $\perp\!\!\!\perp$ comme la clôture de $\{a * \pi_h\}$, et posons $\mathcal{A} = \{\pi_h\}$ et $\mathcal{B} = \{\pi\}$. Pour tout α de $|\mathcal{A}|$, $\alpha * \pi_h \in \perp\!\!\!\perp$ implique $h * \alpha \cdot \pi \in \perp\!\!\!\perp$, donc h est élément de $|\mathcal{A} \to \mathcal{B}|$. De même, pour tout α de $|\neg\mathcal{A}|$ on a $\alpha * a \cdot \pi_g \in \perp\!\!\!\perp$ car $a \in |\mathcal{A}|$, d'où $f * \alpha \cdot \pi \in \perp\!\!\!\perp$, ce qui prouve $f \in |\neg\mathcal{A} \to \mathcal{B}|$. Par conséquent, comme c est élément de $|T|$ par adéquation, $c * f \cdot h \cdot \pi$ est élément de $\perp\!\!\!\perp$, donc il se réduit en $a * \pi_h$. ■

Ceci dit, les termes de type T peuvent se comporter différemment de M_T. On peut en effet donner une définition symétrique :

$$N_T \equiv \lambda f.\lambda h.\kappa x.h(\kappa y.x(fy))$$

Le comportement de N_T est alors symétrique dans la mesure où il évalue h avant f, selon les règles

$$\begin{aligned} N_T * f \cdot h \cdot \pi &\succ h * \beta_{f,\pi} \cdot \pi \\ \beta_{f,\pi} * \pi' &\succ f * k_{\pi'} \cdot \pi \end{aligned}$$

C'est-à-dire qu'au lieu de passer une donnée de f à h, on passe une continuation de h à f.

On peut tenter d'expliquer intuitivement ce comportement en disant que N_T évalue h avec en argument $\beta_{f,\pi}$ (sorte de co-exception). Si h met $\beta_{f,\pi}$ en tête (demande une valeur), f est évaluée dans le contexte initial avec en argument la pile où se trouvait $\beta_{f,\pi}$ (le point où l'évaluation de h s'est interrompue). f peut alors soit ignorer cette continuation, soit l'utiliser pour envoyer une donnée à h. On retrouve d'ailleurs ici le passage de donnée de f à h.

Le calcul est manifestement différent, or la proposition 5 signifie que $M_T * f \cdot h \cdot \pi$ et $N_T * f \cdot h \cdot \pi$ doivent aboutir au même résultat. Que doit-on en penser ? Ceci signifie simplement que si f et g mettent tous deux leur argument (quel qu'il soit) en tête, alors le résultat sera le même, ce dont on peut se convaincre aisément.

2.2.3 Le combinateur C_T

Reprenons les réductions de M_T. On avait donc

$$\begin{aligned} M_T * f \cdot h \cdot \pi &\succ f * \alpha_{h,\pi} \cdot \pi \\ \alpha_{h,\pi} * a \cdot \pi' &\succ h * a \cdot \pi \end{aligned}$$

Comme le terme a est issu de la réduction de f sur une pile contenant $\alpha_{h,\pi}$, il peut lui aussi contenir $\alpha_{h,\pi}$. La réduction de $h * a \cdot \pi$ peut donc le mettre en tête, et dans ce cas on a les réductions suivantes :

$$h * a \cdot \pi \succ a * \pi_1 \succ \alpha_{h,\pi} * a' \cdot \pi_2$$

La règle sur $\alpha_{h,\pi}$ s'applique alors à nouveau et on a

$$\alpha_{h,\pi} * a' \cdot \pi_2 \succ h * a' \cdot \pi \succ a' * \pi_1[a'/a]$$

où la deuxième réduction est justifiée par le fait que la réduction de $h * a \cdot \pi$ est indépendante de a jusqu'à ce que celui-ci arrive en tête, dans l'exécutable $a * \pi_1$.

On peut donc prédire toute la réduction qui mène de $h * a' \cdot \pi$ à $a' * \pi_1[a/a']$. On pourrait l'éviter en réduisant directement $\alpha_{h,\pi} * a' \cdot \pi_2$ en $a' * \pi_1[a'/a]$, ce qui serait la cas si $\alpha_{h,\pi}$ avait été remplacé par $\lambda x.(k_{\pi_1[x/a]}x)$ dans a, mais ce terme contient une substitution à l'intérieur de la continuation k_{π_1}, ce que n'autorise pas le calcul tel que nous l'avons défini.

Introduisons alors artificiellement dans l'algèbre des termes une constante C_T ainsi qu'une famille de termes $\alpha_{t,\pi}$, avec des règles de réduction qui rendent compte de ce processus. Ici, on va en fait ignorer la substitution qui a lieu dans la continuation k_{π_1}, en introduisant les règles suivantes :

$$\begin{array}{lrcl} \text{[try]} & C_T * f \cdot h \cdot \pi & \succ & f * \alpha_{h,\pi} \cdot \pi \\ \text{[catch]} & \alpha_{h,\pi} * a \cdot \pi' & \succ & h * \kappa x.a[x/\alpha_{h,\pi}] \cdot \pi \end{array}$$

où la variable x est naturellement supposée libre dans a. Ceci revient donc à remplacer $\alpha_{h,\pi}$ par k_{π_1}, avec les notations précédentes. On n'effectue pas de substitution dans ce k_{π_1} parce qu'une telle opération pose des problèmes de définition ; par exemple, on ne doit pas substituer x à tous les sous-termes égaux à a, mais uniquement ceux qui sont issus de la réduction de $f * \alpha_{h,\pi} \cdot \pi$.

Dans ce nouveau système de réductions, le combinateur synthétique C_T ne se comporte pas de la même façon que M_T. Cependant, il réalise bien T, si l'on suppose quelques conditions supplémentaires sur $\perp\!\!\!\perp$:

Définition 6 (*séquentialité*)

L'ensemble d'observables $\perp\!\!\!\perp$ est dit séquentiel *s'il vérifie les conditions suivantes :*

1. *$\perp\!\!\!\perp$ est clos par réduction,*
2. *si $e[t/x] \in \perp\!\!\!\perp$ et si la réduction de e ne place jamais x en tête, alors $e[u/x] \in \perp\!\!\!\perp$ pour tout u,*
3. *$\perp\!\!\!\perp$ ne contient aucun exécutable de la forme $\lambda x.t * \rho$.*

Il reste encore à supposer que toute réduction dans $\perp\!\!\!\perp$ termine. Ce n'est pas fondamentalement nécessaire, on pourrait énoncer une condition moins restrictive.

Proposition 6

Pour tout $\perp\!\!\!\perp$ séquentiel, le combinateur C_T réalise T.

Démonstration : Considérons un $\perp\!\!\!\perp$ vérifiant les conditions de séquentialité et une pile $f \cdot h \cdot \pi$ de l'ensemble $T^{\perp\!\!\!\perp}$. Il existe donc des ensembles de piles $\mathcal{A}$ et $\mathcal{B}$ tels que l'on ait $f \in |(\mathcal{A} \to \perp) \to \mathcal{B}|$, $h \in |\mathcal{A} \to \mathcal{B}|$ et $\pi \in \mathcal{B}$. Pour prouver que $C_T * f \cdot h \cdot \pi$ est élément de $\perp\!\!\!\perp$, il suffit alors de prouver que $f * \alpha_{h,\pi} \cdot \pi$ est dans $\perp\!\!\!\perp$, ce qui est assuré si $\alpha_{h,\pi}$ est élément de $|\mathcal{A} \to \perp|$. Considérons donc une pile $a \cdot \pi'$ élément de $(\mathcal{A} \to \perp)^{\perp\!\!\!\perp} = |\mathcal{A}| \cdot \Pi$. L'exécutable $\alpha_{h,\pi} * a \cdot \pi'$ se réduit en $h * \kappa x.a[x/\alpha_{h,\pi}] \cdot \pi$, or par hypothèse on a $h \in |\mathcal{A} \to \mathcal{B}|$ et $\pi \in \mathcal{B}$, donc pour conclure il suffit de prouver que $\kappa x.a[x/\alpha_{h,\pi}] \in |\mathcal{A}|$, sachant que $a \in |\mathcal{A}|$ par hypothèse.

Soit π_1 une pile de $\mathcal{A}$. La réduction de $\kappa x.a[x/\alpha_{h,\pi}] * \pi_1$ donne $a[k_{\pi_1}/\alpha_{h,\pi}] * \pi_1$, puis deux cas se présentent :

- soit la réduction de $a * \pi_1$ ne place pas $\alpha_{h,\pi}$ en tête, auquel cas la deuxième condition sur $\perp\!\!\!\perp$ prouve que $a[k_{\pi_1}/\alpha_{h,\pi}] * \pi_1$ est élément de $\perp\!\!\!\perp$
- soit $a * \pi_1$ se réduit en $\alpha_{h,\pi} * b \cdot \pi_2$ avec $b \in |\mathcal{A}|$, alors $a[k_{\pi_1}/\alpha_{h,\pi}] * \pi_1$ se réduit en $k_{\pi_1} * b[k_{\pi_1}/\alpha_{h,\pi}] \cdot \pi_2$ puis en $b[k_{\pi_1}/\alpha_{h,\pi}] * \pi_1$

Dans le deuxième cas, on se retrouve à vouloir prouver sur b ce que l'on voulait prouver sur a. Si l'on suppose que la réduction de $a[k_{\pi_1}/\alpha_{h,\pi}] * \pi_1$ termine, on a une induction bien fondée et la preuve est terminée. ■

Moyennant un certain nombre de restrictions sur le choix des $\perp\!\!\!\perp$, on a donc prouvé que le combinateur C_T, qui n'est pas exprimable par un $\lambda\kappa$-terme, était cohérent avec le reste du calcul et du typage. Il faut cependant se demander ce qui a été perdu en restreignant la forme de $\perp\!\!\!\perp$ de cette façon.

Dans les preuves évoquées ici, on n'a jamais besoin d'utiliser de $\perp\!\!\!\perp$ qui ne vérifie explicitement pas ces conditions, donc les résultats restent vrais dans le calcul enrichi, si on les interprète correctement. Par exemple, dans la preuve de la proposition 3 on définissait $\perp\!\!\!\perp$ comme la clôture par anti-réduction du singleton $\{a * \pi\}$, il faut donc maintenant considérer à la place le plus petit $\perp\!\!\!\perp$ séquentiel contenant ce singleton. Le résultat que l'on démontre est alors le suivant :

Proposition 7 (*3 revisité*)

Si $\vdash t : \forall X(X \to X)$ est dérivable, alors pour tout terme a et toute pile π tels que $a * \pi$ ne soit pas de la forme $\lambda x.t * \rho$, il existe un exécutable en lequel se réduisent $t * a \cdot \pi$ et $a * \pi$.

La formulation est plus lourde, mais elle signifie essentiellement que $t * a \cdot \pi$ et $a * \pi$ ont (intuitivement) la même valeur dans le $\lambda\kappa$-calcul enrichi du combinateur C_T.

2.3 Le tiers-exclu symétrique

À titre d'exemple, étudions maintenant une autre variante du tiers-exclu. On considère la formule qui s'écrit $\forall A \forall B(A \to B) \vee (B \to A)$ sous forme disjonctive. Sa formulation dans les types est donc

$$G = \forall A \forall B \forall C\, ((A \to B) \to C) \to ((B \to A) \to C) \to C$$

Cette forme est plus symétrique, et le comportement le sera aussi.

2.3.1 Spécification

Le type G spécifie un comportement un peu plus complexe que T :

Proposition 8 (*spécification du tiers-exclu symétrique*)

Soit c un terme tel que $\vdash c : G$ soit dérivable. Pour tous termes f, g, a et b, et pour toutes piles π, π_f et π_g pour lesquels on a
- *pour tout α, $f * \alpha \cdot \pi \succ \alpha * a \cdot \pi_f$,*
- *pour tout β, $g * \beta \cdot \pi \succ \beta * b \cdot \pi_g$,*

$c * f \cdot g \cdot \pi$ se réduit soit en $b * \pi_f$ soit en $a * \pi_g$.

Démonstration : Supposons définis c, f, g, a, b, ainsi que π, π_f et π_g selon l'énoncé. On choisit pour $\perp\!\!\!\perp$ la clôture par anti-réduction de $\{b * \pi_f, a \cdot \pi_g\}$. On pose ensuite $\mathcal{A} = \{\pi_g\}$, $\mathcal{B} = \{\pi_f\}$, et $\mathcal{C} = \{\pi\}$. On a donc $a \in |\mathcal{A}|$ et $b \in |\mathcal{B}|$. Par conséquent, pour tout $\alpha \in |\mathcal{A} \to \mathcal{B}|$, on a $\alpha * a \cdot \pi_f \in \perp\!\!\!\perp$, d'où $f * \alpha \cdot \pi \in \perp\!\!\!\perp$, soit $f \in |(\mathcal{A} \to \mathcal{B}) \to \mathcal{C}|$. De même, pour tout $\beta \in |\mathcal{B} \to \mathcal{A}|$, on a $\beta * b \cdot \pi_g \in \perp\!\!\!\perp$, et par conséquent $g * \beta \cdot \pi \in \perp\!\!\!\perp$, d'où $g \in |(\mathcal{B} \to \mathcal{A}) \to \mathcal{C}|$. L'adéquation montre que c est élément de $|((\mathcal{A} \to \mathcal{B}) \to \mathcal{C}) \to ((\mathcal{B} \to \mathcal{A}) \to \mathcal{C}) \to \mathcal{C}|$, par conséquent l'exécutable $c * f \cdot g \cdot \pi$ est élément de $\perp\!\!\!\perp$, ce qui signifie qu'il se réduit en $b * \pi_f$ ou en $a * \pi_g$. ■

Ici encore, on peut construire un terme qui a G pour type. Par exemple :

$$M_G \equiv \lambda f.\lambda g.\kappa x.f(\lambda a.x(g(\lambda b.a)))$$

Bien entendu, ce terme ne traite pas f et g de façon symétrique. On pourrait définir un terme ayant le comportement inverse, c'est-à-dire qui applique g à un terme contenant la continuation courante.

Ce terme M_G réalise en fait plus que G, puisqu'il réalise la formule $\forall A \forall B \forall C\,(A \to B) \vee (C \to A)$. Dans un contexte ou plusieurs exécutables évoluent en parallèle, on peut réaliser G de façon plus stricte (et plus intuitive) en implémentant un mécanisme de synchronisation entre deux processus. On pourrait en fait réaliser G ici par un mécanisme qui simulerait l'exécution parallèle de deux processus.

2.3.2 Le combinateur C_G

Selon le même principe que pour la construction de C_T à partir du tiers exclu, on peut aussi ajouter au langage un combinateur C_G, avec des règles de réduction associées, pour obtenir une réalisation de G. On introduit en plus de C_G une famille de termes $\beta_{t,\pi}$, et on définit les règles suivantes :

$$\begin{array}{rcl} C_G * f \cdot g \cdot \pi & \succ & f * \beta_{g,\pi} \cdot \pi \\ \beta_{g,\pi} * a \cdot \pi & \succ & g * \lambda x.\kappa y.a[y/\beta_{g,\pi}] \cdot \pi \end{array}$$

Cette extension du calcul est construite à partir du terme M_G de la même façon que l'extension avec le combinateur C_T avait été construite à partir du terme M_T, en exploitant le fait que certaines réductions sont prévisibles, et le résultat est effectivement similaire.

2.4 Généralisation

On a donc mis en évidence sur trois exemples de tautologies classiques d'une part une spécification de leur comportement, d'autre part une façon de les réaliser à l'aide de combinateurs synthétiques. On peut maintenant tenter de généraliser ces résultats.

Ces trois exemples sont en fait des tautologies *disjonctives* en ce sens qu'elles sont de la forme générale suivante :

Définition 7 (*tautologie disjonctive*)

Une formule purement disjonctive est une formule de la forme

$$A = \bigvee_{i=1}^{n} A_i \qquad \text{avec} \qquad A_i = B_{i,1} \to \cdots \to B_{i,n_i} \to C_i$$

où les $B_{i,j}$ et les C_i sont des variables propositionnelles. On appelle ensemble de vérité *d'une telle formule A l'ensemble*

$$tr(A) = \{(i,j,k) \mid B_{i,j} = C_k\}$$

Lorsque $tr(A)$ n'est pas vide, A est une tautologie, et on parle alors de tautologie disjonctive.

On parle de formule *disjonctive* parce que de telles formules, en appliquant la transformation de $A \to B$ en $\neg A \vee B$, se mettent sous la forme $\bigvee_{i=1}^{n} \left(\bigvee_{j=1}^{n_i} \neg B_{i,j} \right) \vee C_i$, qui est une disjonction de

littéraux. Dans le cadre des types du $\lambda\kappa$-calcul, on transformera plutôt ces formule en implications de la forme

$$\big((B_{1,1} \to \cdots \to B_{1,n_1} \to C_1) \to D\big) \to \cdots \to \big((B_{n,1} \to \cdots \to B_{n,n_n} \to C_n) \to D\big) \to D$$

Il apparaît alors que les deux exemples précédents se mettent sous cette forme (ce n'est pas le cas de la loi de Peirce). En effet :

- pour le tiers exclu, $A_1 = A \to B$ et $A_2 = A$ conduisent bien à

$$\big((A \to B) \to C\big) \to \big(A \to C\big) \to C\,;$$

- pour le tiers exclu symétrique, $A_1 = A \to B$ et $A_2 = B \to A$ conduisent à

$$\big((A \to B) \to C\big) \to \big((B \to A) \to C\big) \to C\,.$$

2.4.1 Spécification

Ce que spécifient ces formules dans le $\lambda\kappa$-calcul est connu [1], et s'énonce de la façon suivante :

Théorème 9 (*spécification des tautologies disjonctives*)

Soit A une formule purement disjonctive et soit c un terme tel que $\vdash c : A$ soit dérivable. Soient des termes f_i et $b_{i,j}$ et des piles π et π_i tels que pour terme a et pour tout i on ait $f_i * a \cdot \pi \succ a * b_{i,1} \cdots b_{i,n_i} \cdot \pi_i$ (en utilisant les notations de la définition 7), alors il existe un (i,j,k) dans $tr(A)$ tel que $c * f_1 \cdots f_n \cdot \pi$ se réduise en $b_{i,j} * \pi_k$.

Notons que cette proposition ne suppose pas que A est une tautologie. En effet, si la formule A est fausse, par adéquation elle n'est tout simplement pas réalisable, ce qui rend la proposition trivialement vraie.

Démonstration : Supposons que A est vraie, et donc que $tr(A)$ n'est pas vide. Soient alors des familles de termes f_i et $b_{i,j}$, une pile π et une famille de piles π_i vérifiant les conditions de la proposition. Définissons $\perp\!\!\!\perp$ comme la clôture de $\{b_{i,j} * \pi_k \mid (i,j,k) \in tr(A)\}$ par anti-réduction, et définissons les valeurs de vérité des variables par $[D] = \{\pi\}$ et $[C_i] = \{\pi_k \mid C_k = C_i\}$. Ceci peut ne pas définir la valeur de chaque variable intervenant dans A ; si une variable X n'apparaît pas parmi les C_i, on pose $[X] = \varnothing$ et donc $|X| = \Lambda$.

Avec ces définitions, chaque $b_{i,j}$ appartient au $|B_{i,j}|$ associé, soit parce que $B_{i,j}$ est trivial, soit parce qu'il existe k tel que $(i,j,k) \in tr(A)$ et donc $[B_{i,j}] = [C_k]$ et $b_{i,j} * \pi_k \in \perp\!\!\!\perp$. Par conséquent, pour tout i, pour tout $\alpha \in |A_i|$ et pour toute pile $\pi_i \in [C_i]$, l'exécutable $\alpha \cdot b_{i,1} \cdots b_{i,n_i} \cdot \pi$ est élément de $\perp\!\!\!\perp$ puisque $[A_i] = |B_{i,1}| \cdots |B_{i,n_i}| \cdot [C_i]$, et donc $f_i * \alpha \cdot \pi$ est aussi dans $\perp\!\!\!\perp$ par anti-réduction, d'où $f_i \in |A_i \to D|$. Comme c réalise le type A par hypothèse, on a donc $c * f_1 \cdots f_n \cdot \pi \in \perp\!\!\!\perp$, ce qui signifie que cet exécutable se réduit en l'un des $b_{i,j} * \pi_k$ avec $(i,j,k) \in tr(A)$. ■

La formulation de cette proposition est un peu lourde, mais on voit facilement que les propositions 5 et 8 en sont des cas particuliers.

2.4.2 Combinateurs synthétiques

Selon la même méthode que précédemment, on peut aussi définir des combinateurs artificiels pour réaliser ces tautologies. On peut même à nouveau introduire une liaison sur un terme pour optimiser le calcul.

Considérons donc une tautologie disjonctive A, avec les notation de la définition 7. Considérons un terme t qui réalise A, en utilisant un triplet particulier $(i,j,k) \in tr(A)$. Ce qu'exprime la

proposition 9 est que si f_i et f_k utilisent leur argument selon les règles

$$f_i * a \cdot \pi \succ a * b_{i,1} \cdots b_{i,n_i} \cdot \pi_i$$
$$f_k * a \cdot \pi \succ a * b_{k,1} \cdots b_{k,n_k} \cdot \pi_k$$

quel que soit a, alors $t * f_1 \cdots f_n \cdot \pi$ se réduit en $b_{i,j} * \pi_k$. Le terme t se comporte donc comme le combinateur (non optimisé) suivant :

$$C_A^{i,j,k} * f_1 \cdots f_n \cdot \pi \succ f_i * \alpha \cdot \pi$$
$$\alpha * b_1 \cdots b_{n_i} \cdot \pi' \succ f_k * \lambda c_1 \cdots \lambda c_{n_k} b_j \cdot \pi$$

Pour construire un combinateur optimisé, étudions alors la chaîne de réductions suivante :

$C_A^{i,j,k} * f_1 \cdots f_n \cdot \pi \succ$	$f_i * \alpha \cdot \pi$	par définition
$\succ$	$\alpha * b_1 \cdots b_{n_i} \cdot \pi'$	par hypothèse
$\succ$	$f_k * \lambda c_1 \cdots \lambda c_{n_k} b_j \cdot \pi$	par définition
$\succ$	$\lambda c_1 \cdots \lambda c_{n_k} b_j * c_1 \cdots c_{n_k} \cdot \pi_k$	par hypothèse
$\succ$	$b_j * \pi_k$	
supposons $\succ$	$\alpha * d_1 \cdots d_{n_i} \cdot \pi''$	
alors $\succ$	$f_k * d_j \cdot \pi$	par définition
puis $\succ$	$d_j * \big(c_1 \cdots c_{n_k} \cdot \pi_k\big)[d_j/b_j]$	car cette réduction est indépendante de b_j

Donc si à la sixième étape on remplaçait α par un opérateur correspondant intuitivement à

$$\lambda \vec{d}\big((k_{\pi_k} b_j)[d_j/b_j]\big) \qquad \text{avec } \vec{d} \text{ d'arité } n_i,$$

la réduction serait la même mais en effectuant les deux dernières chaînes de réductions en une seule étape. À nouveau, la substitution $[d_j/b_j]$ pose un problème de définition, mais si l'on remplace la règle de réduction de α par une version optimisée en ignorant cette substitution, on obtient les règles

$$C_A^{i,j,k} * f_1 \cdots f_n \cdot \pi \succ f_1 * \alpha \cdot \pi$$
$$\alpha * b_1 \cdots b_{n_i} \cdot \pi' \succ f_k * \lambda \vec{c}.\kappa x.\big(b_j[\lambda \vec{d}(x\, d_j)/\alpha]\big) \cdot \pi$$

avec $\vec{c}$ d'arité n_k et $\vec{d}$ d'arité n_i. Ce combinateur, dans le nouveau système de réductions, est *valide* sous les mêmes conditions que dans le cas particulier du tiers-exclu évoqué plus haut :

Théorème 10 (*validité des combinateurs synthétiques*)

Soit A une tautologie disjonctive, soit (i, j, k) un élément de l'ensemble de vérité $tr(A)$. Pour tout $\perp\!\!\!\perp$ séquentiel, le combinateur $C_A^{i,j,k}$ réalise A.

Démonstration : Considérons un $\perp\!\!\!\perp$ vérifiant les conditions de séquentialité et une pile $f_1 \cdots f_n \cdot \pi$ de l'ensemble $A^{\perp\!\!\!\perp}$. Il existe donc une valuation des variables de A telle que l'on ait $f_p \in |A_p \to D|$ pour chaque p et $\pi \in [D]$. Pour prouver que $C_A^{i,j,k} * \vec{f} \cdot \pi$ est élément de $\perp\!\!\!\perp$, il suffit alors de prouver que $f_i * \alpha \cdot \pi$ est dans $\perp\!\!\!\perp$, ce qui est assuré si α est élément de $|A_i|$, c'est-à-dire si

$$\alpha \in |B_{i,1} \to \cdots \to B_{i,n_i} \to C_i|.$$

Considérons donc une pile $b_1 \cdots b_{n_i} \cdot \pi'$ élément de $[A_i] = |B_{i,1}| \cdots |B_{i,n_i}| \cdot [C_i]$. On a donc la réduction

$$\alpha * b_1 \cdots b_{n_i} \cdot \pi' \succ f_k * \lambda \vec{c}.\kappa x.(b_j[\lambda \vec{d}(x d_j)/\alpha]) \cdot \pi$$

or par hypothèse on a $f_k \in |A_k \to D|$ et $\pi \in [D]$, donc pour conclure il suffit de prouver que $\lambda\vec{c}.\kappa x.(b_j[\lambda\vec{d}(xd_j)/\alpha])$ est élément de $|A_k|$, donc que

$$\lambda\vec{c}.\kappa x.(b_j[\lambda\vec{d}(xd_j)/\alpha]) \in |B_{k,1} \to \cdots \to B_{k,n_k} \to C_k|,$$

sachant que $b_j \in |B_{i,j}|$, donc $b_{i,j} \in |C_k|$, par hypothèse.

Soit $c_1 \cdots c_{n_k} \cdot \pi_1$ une pile de $[A_k]$. La réduction de $\lambda\vec{c}.\kappa x.b_j[\lambda\vec{d}(xd_j)/\alpha]$ sur cette pile donne $b_j[\lambda\vec{d}(k_{\pi_1}d_j)/\alpha] * \pi_1$, puis deux cas se présentent :

- soit la réduction de $b_j * \pi_1$ ne place pas α en tête, auquel cas la deuxième condition sur $\perp\!\!\!\perp$ prouve que $b_j[\lambda\vec{d}(k_{\pi_1}d_j)/\alpha] * \pi_1$ est élément de $\perp\!\!\!\perp$
- soit $b_j * \pi_1$ se réduit en un $\alpha * d_1 \cdots d_{n_i} \cdot \pi_2$ avec $d_p \in |B_{i,p}|$ pour chaque p, alors $b_j[\lambda\vec{d}(k_{\pi_1}d_j)/\alpha] * \pi_1$ se réduit en $k_{\pi_1} * d_j[\lambda\vec{d}(k_{\pi_1}d_j)/\alpha] \cdot \pi_2$ puis en $d_j[\lambda\vec{d}(k_{\pi_1}d_j)/\alpha] * \pi_1$

Dans le deuxième cas, on veut donc prouver pour d_j ce que l'on voulait prouver sur b_j. On a alors une induction bien fondée si l'on suppose que la réduction de $b_j[\lambda\vec{d}(k_{\pi_1}d_j)/\alpha] * \pi_1$ termine, ce qui permet de conclure. ■

Chapitre 3

Formes disjonctives

Étudions à présent une classe de formules différente de la précédente : les disjonctions de conjonctions de littéraux. D'un point de vue purement logique, toute formule est équivalente à une formule de cette classe, à savoir sa forme normale disjonctive, cependant ce n'est pas ici la valeur de vérité des formules qui est importante (puisqu'on ne considère en fait que des tautologies), mais bien leur structure.

Définition 8 (*forme disjonctive*)

On appelle forme disjonctive *une formule A de la forme*

$$A = \bigvee_{i=1}^{n} A_i \qquad \textit{avec} \qquad A_i = \bigwedge_{j=1}^{n_i} \neg N_{i,j} \wedge \bigwedge_{j=1}^{p_i} P_{i,j}$$

où les $N_{i,j}$ et $P_{i,j}$ sont des variables propositionnelles.

La reformulation en implications d'une telle formule s'écrit alors

$$\begin{array}{rcl} A & = & \big(\neg N_{1,1} \to \cdots \to \neg N_{1,n_1} \to P_{1,1} \to \cdots \to P_{1,p_1} \to X\big) \to \\ & & \qquad\qquad \vdots \\ & & \big(\neg N_{n,1} \to \cdots \to \neg N_{n,n_n} \to P_{n,1} \to \cdots \to P_{n,p_n} \to X\big) \to X \end{array}$$

Pour interpréter cette formule comme un type, on la considère quantifiée universellement en toutes les variables qui y apparaissent, comme on le fait dans la théorie des types simples. Notons que ces formules sont toujours fausses dans le cas intuitionniste, sauf dans le cas extrême où l'une des clauses est vide (et donc vraie) ; elle ne peuvent donc spécifier que des structures de contrôle manipulant des continuations.

3.1 Exemples

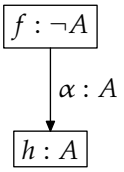
Avant de tenter d'énoncer des résultat généraux sur cette classe de formules très vaste, étudions quelques exemples simples.

3.1.1 Retour sur le tiers-exclu

L'exemple le plus simple est celui du tiers exclu, déjà rencontré dans le cadre des tautologies disjonctives. Il s'exprime en effet $(\neg A \to X) \to (A \to X) \to X$, donc avec $A_1 = \neg A$ et $A_2 = A$. Ici, on sait qu'un comportement possible est défini par

$$\begin{array}{r} C * f \cdot h \cdot \pi \succ f * \alpha \cdot \pi \\ \alpha * a \cdot \pi' \succ h * a \cdot \pi \end{array}$$

que l'on représentera par



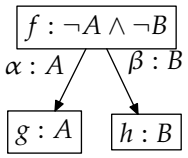
Le sens de ce diagramme est le suivant : les deux nœuds représentent les deux clauses A_1 et A_2, auxquelles correspondent les deux arguments f et h qu'attend le combinateur que l'on spécifie. La flèche du nœud $\neg A$ au nœud A (étiquetée par α et le type A) représente le comportement de α qui, s'il est placé en tête par la réduction de $f * \alpha \cdot \pi$, passe à la réduction de h, sur un terme de type A fourni par la réduction de f.

3.1.2 Tiers-exclu à deux variables

Le tiers exclu appliqué à deux variables propositionnelles distinctes peut s'énoncer $(\neg A \wedge \neg B) \vee A \vee B$. Il est sous forme disjonctive, mais il n'est pas exprimable par une formule du type précédent. On peut lui faire correspondre un comportement correspondant au traitement de deux exceptions A et B distinctes, ce qui correspond aux règles

$$\begin{array}{r} C * f \cdot g \cdot h \cdot \pi \succ f * \alpha \cdot \beta \cdot \pi \\ \alpha * a \cdot \pi' \succ g * a \cdot \pi \\ \beta * b \cdot \pi' \succ h * b \cdot \pi \end{array}$$

que l'on représentera par



Le sens de ce diagramme est similaire au précédent : La réduction de $C * f \cdot g \cdot h \cdot \pi$ commence par placer f en tête, avec deux « exceptions » α et β. Si f place α en tête sur un terme de type A, la réduction reprend sur la pile initiale avec g appliqué à ce terme. Si β est placé en tête, le comportement est similaire.

Il est facile de prouver que le combinateur C ainsi défini réalise le type voulu, c'est-à-dire

$$((A \to \bot) \to (B \to \bot) \to C) \to (A \to C) \to (B \to C) \to C,$$

et il serait encore possible d'optimiser ce calcul en remplaçant a et b par $\kappa\alpha.a$ et $\kappa\beta.b$ respectivement dans les deuxième et troisième règles.

D'autre part, on peut prouver sans grande difficulté que ce type spécifie bien un comportement comme celui de C. Plus précisément :

Proposition 11 (*spécification du tiers-exclu à deux variables*)

Soit c un $\lambda\kappa$-terme tel que $\vdash c : \forall A \forall B (\neg A \wedge \neg B) \vee A \vee B$ soit dérivable. Pour tous termes f, g, h, a et b et toutes piles π, π_f, π_g et π_h tels que
- *pour tout α, $g * \alpha \cdot \pi \succ \alpha \cdot \pi_g$,*
- *pour tout α, $h * \alpha \cdot \pi \succ \alpha \cdot \pi_h$.*

On a alors :
- *si pour tous α et β, $f * \alpha \cdot \beta \cdot \pi \succ \alpha * a \cdot \pi_f$, alors $c * f \cdot g \cdot h \cdot \pi \succ a * \pi_g$;*
- *si pour tous α et β, $f * \alpha \cdot \beta \cdot \pi \succ \beta * b \cdot \pi_f$, alors $c * f \cdot g \cdot h \cdot \pi \succ b * \pi_h$.*

Démonstration : Soit c un terme de type $(\neg A \to \neg B \to C) \to (A \to C) \to (B \to C) \to C$. Supposons les termes f, g, h, a et b et les piles π, π_f, π_g et π_h définies selon les conditions énoncées. Définissons $\perp\!\!\!\perp$ comme la clôture par anti-réduction de $\{a * \pi_g, b * \pi_h\}$ et posons $[A] = \{\pi_g\}$, $[B] = \{\pi_h\}$ et $[C] = \{\pi\}$. Pour tout $\alpha \in |A|$, on a $\alpha * \pi_g \in \perp\!\!\!\perp$ donc $g * \alpha \cdot \pi$ est dans $\perp\!\!\!\perp$, ce qui prouve que g est élément de $|A \to C|$. On prouve de même que h est élément de $|B \to C|$.

Dans le premier cas, pour tous α et β, $f * \alpha \cdot \beta \cdot \pi$ se réduit en $\alpha * a \cdot \pi_f$. Comme a est élément de $|A|$ par définition de $\perp\!\!\!\perp$, pour tout α de $|\neg A|$ on a $\alpha * a \cdot \pi_f \in \perp\!\!\!\perp$ d'où $f * \alpha \cdot \beta \cdot \pi \in \perp\!\!\!\perp$ pour tout β, et donc $f \in |\neg A \to \neg B \to C|$, et par conséquent $c * f \cdot g \cdot h \cdot \pi$ est dans $\perp\!\!\!\perp$, ce qui signifie qu'il se réduit en $a * \pi_g$ ou $b * \pi_h$. Dans ce cas de figure, b est absolument quelconque (contrairement à a qui découle de la réduction de f), donc si $c * f \cdot g \cdot h \cdot \pi$ se réduisait en $b * \pi_h$, on prouverait en remplaçant b par n'importe quel autre terme que cet exécutable se réduit aussi en $t * \pi_h$ quel que soit t, ce qui est absurde puisque la réduction est déterministe. Par conséquent $c * f \cdot g \cdot h \cdot \pi$ se réduit en $a * \pi_g$.

La preuve du cas symétrique où pour tous α et β, $f * \alpha \cdot \beta \cdot \pi$ se réduit en $\beta * b \cdot \pi_f$ se mène de façon identique. ■

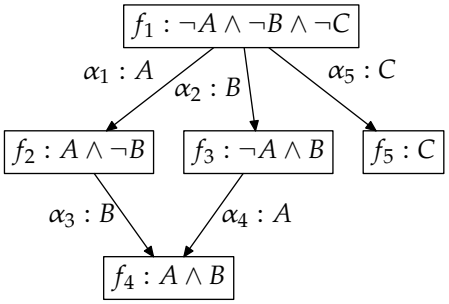
Ce cas du tiers exclus à deux variables est significativement plus complexe que le cas à une seule variable. En particulier, on remarque qu'il est nécessaire d'avoir une hypothèse sur la réduction de g et de h même si le diagramme pourrait laisser croire que seuls deux clauses sont effectivement utilisées, soit $\neg A \wedge \neg B$ et A, soit $\neg A \wedge \neg B$ et B. En revanche, ce résultat s'étend immédiatement à un nombre arbitraire de variables, donnant une spécification similaire pour les tautologies de la forme $(\neg A_1 \wedge \cdots \wedge \neg A_n) \vee A_1 \vee \cdots \vee A_n$.

3.1.3 Un exemple plus touffu

On peut maintenant s'essayer à interpréter un type plus complexe, afin d'avoir une intuition plus précise des comportements associés aux formes disjonctives. Prenons par exemple trois variables propositionnelles A, B et C, et cinq clauses :

$$A = (\neg A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B) \vee (\neg A \wedge B) \vee (A \wedge B) \vee C$$

On peut représenter cette formule par le diagramme suivant, duquel on déduit un système de réductions définissant un combinateur C_A :



$$\begin{aligned}
C_A * f_1 \cdot f_2 \cdot f_3 \cdot f_4 \cdot f_5 \cdot \pi &\succ f_1 * \alpha_1 \cdot \alpha_2 \cdot \alpha_5 \cdot \pi \\
\alpha_1 * a \cdot \pi' &\succ f_2 * a \cdot \alpha_3^a \cdot \pi \\
\alpha_2 * b \cdot \pi' &\succ f_3 * \alpha_4^b \cdot b \cdot \pi \\
\alpha_3^a * b \cdot \pi' &\succ f_4 * a \cdot b \cdot \pi \\
\alpha_4^b * a \cdot \pi' &\succ f_4 * a \cdot b \cdot \pi \\
\alpha_5 * c \cdot \pi' &\succ f_5 * c \cdot \pi
\end{aligned}$$

Remarquons ici que si f_1 place son premier argument (l'exception α_1) en tête sur une pile commençant par a, et si f_2 place en tête sont second argument sur une pile commençant par b, alors cet opérateur conduit à l'évaluation de f_4 sur une pile contenant à la fois a et b.

Dans ce cadre, « lever une exception », pour reprendre le vocabulaire déjà employé, correspond à emprunter une arête du diagramme et reprendre l'évaluation avec une hypothèse supplémentaire. Par exemple, le fait que $f_1 * \alpha_1 \cdot \alpha_2 \cdot \alpha_5$ place α_1 en tête sur un terme a de type A signifie que f_1 réfute l'hypothèse $\neg A$ en fournissant a qui est une preuve de A ; la règle de réduction de

α_1 conduit ensuite à la clause $A \wedge \neg B$, d'où l'on ne peut atteindre aucune clause où A apparaît négativement.

Une fois de plus il est facile de prouver que le combinateur C_A ainsi produit réalise le type A. Pour ce qui est d'en définir une version optimisée à base de liaison dynamique des α_i, si l'opération est possible, en revanche sa rédaction deviendrait rapidement trop lourde pour être d'un réel intérêt.

3.2 Généralisations

Les tautologies de cette forme sont toujours des formes plus ou moins élaborées du tiers-exclu. L'idée que suggèrent les quelques diagrammes qui précèdent est que les formes disjonctives spécifient en quelque sorte des mécanismes de preuve interactive : le combinateur commence par évaluer l'un de ses arguments f_1, supposé prouver X sous l'hypothèse $\neg N_{i,1} \wedge \cdots \wedge \neg N_{i,n_1} \wedge P_{i,1} \wedge \cdots \wedge P_{i,p_i}$. Si f_1 ne place aucun de ses $n_i + p_i$ arguments en tête, c'est que X est effectivement prouvé ; à l'inverse, si l'un des arguments arrive en tête, c'est que f_1 fournit la réfutation de l'une des hypothèses, et le processus continue avec une autre clause, en utilisant cette réfutation.

3.2.1 Synthèse de combinateurs

Intuitivement, il est possible de construire un diagramme de contrôle pour chaque tautologie sous forme disjonctive, et d'en déduire la définition d'un combinateur. Pour formaliser cette intuition, il est nécessaire de définir précisément les diagrammes.

Une tentative de définition pourrait être la suivante. Considérons une formule disjonctive $A = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{n_i} B_{i,j}$, où les $B_{i,j}$ sont des littéraux. Posons ensuite

$$\mathcal{L} = \{(i, j) \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant n_i\}$$

pour désigner l'ensemble des indices des littéraux. On dira qu'un diagramme de contrôle est un graphe orienté $\mathcal{G} = (\mathcal{L}, \mathcal{E})$ sur cet ensemble qui vérifie les deux conditions suivantes :

couplage : pour tout $(i, j) \in \mathcal{L}$,

- si $B_{i,j}$ est un littéral négatif $\neg X$, alors il existe une unique arête $((i, j), (i', j')) \in \mathcal{E}$ et elle vérifie $i \neq i'$ et $B_{i',j'} = X$,
- si $B_{i,j}$ est un littéral positif X, alors tout arête $((i', j'), (i, j)) \in \mathcal{E}$ vérifie $i \neq i'$ et $B_{i,j} = \neg X$,

c'est-à-dire que les arêtes associent chaque littéral à un littéral opposé ;

mémorisation : notons $\rightarrow$ la relation sur $\{1, \ldots, n\}$ définie par

$$i \rightarrow i' \quad \text{si et seulement si} \quad \exists j \in \{1, \ldots, n_i\}\, \exists j' \in \{1, \ldots, n_{i'}\}\, ((i, j), (i', j')) \in \mathcal{E},$$

c'est-à-dire que $\rightarrow$ est la relation $\mathcal{E}$ quotientée par l'égalité de la première coordonnée, alors pour tout $B_{i,j}$ positif, pour tout $B_{i',j'}$ tel que $i \rightarrow^* i'$, $B_{i',j'} \neq \neg B_{i,j}$.

Les clauses sont les parties de $\mathcal{L}$ de première coordonnée constante. L'intuition calculatoire est la suivante : le diagramme doit correspondre à un combinateur C qui, appliqué à une pile $f_1 \cdots f_n \cdot \pi$, implémente une sorte de schéma d'exceptions. L'existence d'un arête $((i, j), (i', j'))$ signifie que si l'évaluation d'un f_i place en tête un argument (une exception) α de type négatif $B_{i,j} = \neg X$, l'effet de C est de transférer l'évaluation vers $f_{i'}$, passant à celui-ci la valeur (de type X) sur laquelle f_i a appliqué α. On se place dans un cadre où toutes les exceptions sont « directes », c'est-à-dire d'un type négatif, et donc qu'il n'y a pas de co-exception (voir section 2.2.2 pour une explication de cette notion).

Dans cette interprétation, la condition de couplage signifie que toute exception doit être rattrapée (existence d'une arête sortante) et que toute valeur (d'un type positif) doit être fournie par le rattrapage d'une exception (nature des arêtes entrantes). La condition de mémorisation signifie, dans l'intuition, que si une exception $\neg X$ a été rattrapée une fois (fournissant, d'un point de vue logique, une réfutation de l'hypothèse $\neg X$), alors le diagramme ne peut pas mener à une clause dans laquelle X apparaît à nouveau en négatif. On peut remarquer que ces deux conditions garantissent au graphe d'être sans cycle.

On conjecture que l'existence, d'un diagramme vérifiant ces deux conditions, pour la formule A, garantit que A est une tautologie. En interprétant les arêtes du graphes comme expliqué, on peut déduire un système de règles de réduction, qui se déduit facilement du diagramme, et dont on conjecture qu'il réalise le type A.

Malgré tout, il ne serait pas surprenant que la définition proposée ici ne soit pas exactement celle recherchée. En effet, la formalisation de cette méthode, qui s'avère être assez lourde, n'a pas été menée à bout. Il va sans dire que la construction de combinateurs optimisés, comme on l'a fait dans la section 2.4.2, si elle semble possible, est elle aussi largement plus complexe ; cette possibilité n'a pas pu dépasser le stade de l'intuition.

3.2.2 Une approche « déclarative »

Une autre approche a été explorée pour associer des comportements calculatoires aux formes disjonctives. Considérons à nouveau une telle formule que l'on notera $A = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{n_i} B_{i,j}$. On appellera *section* tout élément du produit $\{1, \dots, n_1\} \times \cdots \times \{1, \dots, n_n\}$, c'est-à-dire qu'une section s est le choix, dans chaque clause A_i, d'un littéral $B_{i,s(i)}$ particulier. On peut déjà caractériser les tautologies à l'aide des sections :

Proposition 12

Soit $A = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{n_i} B_{i,j}$ une formule disjonctive. A est une tautologie si et seulement si, pour toute section s de A, il existe i et j tels que $B_{i,s(i)} = \neg B_{j,s(j)}$.

Démonstration : Supposons qu'il existe une section s ne vérifiant pas cette propriété. Tous les littéraux $B_{i,s(i)}$ sont donc faits de variables distinctes. On peut alors définir une valuation v des variables telle que $v(B_{i,s(i)})$ soit vraie si et seulement si $B_{i,s(i)}$ est négative. Si une variable n'est pas parmi ces littéraux, on lui associe une valeur arbitraire. Il est alors clair que, dans cette valuation, chaque clause est fausse, puisque l'un de ses littéraux est faux, et donc A n'est pas une tautologie.

Supposons maintenant que A n'est pas une tautologie. Il existe donc une valuation v des variables qui rend A fausse, donc qui rend fausses toutes les clauses. Dans cette valuation, chaque clause A_i étant fausse, il existe un littéral $B_{i,s(i)}$ pour lequel $v(B_{i,s(i)})$ est faux, ce qui définit une section s. Comme tous les littéraux $B_{i,s(i)}$ sont faux dans la valuation v, il ne peut y en avoir deux d'opposés, donc la section s ne vérifie pas la condition voulue, ce qui prouve la réciproque. ■

Si A est une tautologie, on peut alors définir une fonction de synchronisation S qui, à chaque section s, associe un couple (i, j) pour lequel $B_{i,s(i)} = \neg B_{j,s(j)}$. On peut en déduire un combinateur C_A^S auquel on passe en argument une famille $(f_1, \dots, f_n)$ de fonctions et une pile π, et qui a l'effet (intuitif) de lancer en parallèle l'évaluation de tous les f_i sur une pile $\alpha_{i,1} \cdots \alpha_{i,n_i} \cdot \pi$ contenant n_i exceptions, $\alpha_{i,j}$ correspondant au type $B_{i,j}$. Si l'évaluation s'arrête pour chaque processus i en mettant en tête un $\alpha_{i,s(i)}$, on obtient une section s. La synchronisation donne alors un couple $(i, j) = S(s)$ pour lequel on sait que $B_{i,s(i)} = \neg B_{j,s(j)}$. On sait donc que le processus i est de la forme $\alpha_{i,s(i)} * a \cdot \pi_i$ et que le processus j est de la forme $\alpha_{j,s(j)} * \pi_j$. La réalisabilité garantit alors que a réalise $B_{j,s(j)}$ et que π_j est dans l'orthogonal de ce type. Le terme a et la pile π_j sont donc compatibles, et on peut remplacer les deux processus par $a * \pi_j$.

Pour une meilleure compréhension de ce genre de mécanisme de synchronisation, on pourra se reporter à l'article de Vincent DANOS et Jean-Louis KRIVINE [1] qui contient une étude plus détaillée de systèmes de synchronisation basés sur la spécification, dans le cas des tautologies disjonctives étudiées dans la section 2. À partir de ce schéma de synchronisation, on peut ensuite déduire des mécanismes de contrôle séquentiels, mais ceci n'a pas été fait en détail ici. On soupçonne toutefois que cette approche, après une étude plus poussée, peut conduire à des choses similaires à ce que suggèrent les idées de la section 3.2.1.

3.2.3 Spécification

Le problème de la spécification, dans le cadre des formes disjonctives en général, pose lui aussi des problèmes de formalisation. On peut toujours à force d'intuition aboutir à une spécification pour un cas plus ou moins général, mais un véritable théorème portant sur les formes disjonctives sans restrictions reste à découvrir. Un obstacle principal dans la formalisation des preuves de spécification est en particulier l'abondance de notations : un cas, encore relativement simple, comme celui du tiers-exclu à plusieurs variables (proposition 11), nécessite déjà l'introduction d'un grand nombre de cinq termes et quatre piles. Même si l'outillage de la réalisabilité rend la preuve essentiellement immédiate, l'avalanche de notation et de quantifications laisse augurer des preuves difficiles à manipuler, en particulier dès qu'il s'agit de mémoriser des termes comme on l'a vu dans l'exemple de la section 3.1.3.

Mon intuition est, malgré tout, que ces formules spécifie un comportement qui devrait se décrire correctement en utilisant des diagrammes de contrôle : les ensembles de vérité (analogues au $tr(A)$ des tautologies disjonctives) seraient alors des chemins dans un tel diagramme, représentant une sorte de preuve interactive comme évoqué plus haut. Je ne suis pas parvenu à des résultats significatifs sur le sujet, mais il semblerait intéressant de commencer par s'intéresser à un cas plus restreint, par exemple des formules à n variables et 2^n clauses (toutes les combinaisons de n littéraux). Cette voie reste en cours d'exploration.

Conclusion

La théorie de la réalisabilité, par son approche sémantique du typage, fournit donc une méthode relativement systématique pour construire des preuves de spécification, même s'il est bien entendu nécessaire de découvrir les spécifications d'un type donné par l'intuition. Dans le cadre des tautologies étudiées ici, on obtient ainsi la description formelle d'une famille de structures de contrôle.

À partir de la spécification d'une structure de contrôle tirée d'une de ces tautologies, on dispose aussi d'une méthode pour déduire la définition d'un combinateur qui enrichit le langage, en effectuant des réductions qui n'auraient pas été possibles avec la base du $\lambda\kappa$-calcul. L'introduction de combinateurs qui enrichissent réellement le calcul nécessite des restriction dans le choix des observables utilisés dans les preuves ; bien que cela soit une restriction, toute preuve (menée dans le $\lambda\kappa$-calcul pur) respectant les conditions supplémentaires restera vraie dans le calcul étendu, ce qui est un argument de modularité important.

Le λ-calcul typé fournit un fondement théorique fort aux langages de programmation fonctionnels, et ses extensions avec contrôle permettent d'avoir un fondement théorique pour les langages fonctionnels qui permettent la manipulation des continuations. Dans ce contexte, le fait que l'étude de tautologies classiques en tant que spécifications fasse apparaître des structures de contrôle qui ressemblent à des schémas de traitement d'exceptions donne une justification sérieuse à ces modes de programmation. De plus, cela indique des façons de concevoir des structures fortement typées, et donc sûres, contrairement aux fonctionnalités ajoutées empiriquement aux langages existants.

Bibliographie

[1] V. DANOS AND J.-L. KRIVINE, *Disjunctive tautologies as synchronisation schemes*, in Proceedings of CSL'00, P. Clote and H. Schwichtenberg, eds., no. 1862 in Lecture Notes in Computer Science, Fischbachau, 2000, Springer Verlag, pp. 292–301.

[2] ——, *Classical realizability and new forms of control*. slides, Mar. 2002.

[3] T. G. GRIFFIN, *A formulae-as-types notion of control*, in 17th Symposium on Principles of Programming Languages, ACM, Jan. 1990, pp. 47–58.

[4] J.-L. KRIVINE, *Typed lambda-calculus in classical Zermelo-Frænkel set theory*, Archive in Mathematical Logic, 40 (2001), pp. 189–205.

[5] M. PARIGOT, *$\lambda\mu$-calculus : an algorithmic interpretation of classical natural deduction*, in Proceedings of Conference on Logic Programming and Automated Reasoning, vol. 624 of Lecture Notes in Computer Science, Springer Verlag, 1992, pp. 190–201.