Formules valides, jeux et protocoles réseau

Jean-Louis Krivine, Yves Legrandgérard

Formules valides, jeux et protocoles réseau

Jean-Louis Krivine & Yves Legrandgérard

Université Paris VII, C.N.R.S.

14 novembre 2007

Introduction

On décrit ici une relation remarquable entre une notion fondamentale de logique mathématique – celle de *formule valide du calcul des prédicats* – et la spécification de protocoles réseau. On explique ensuite en détail quelques exemples simples, comme l'acquittement d'un ou deux paquets, puis d'un nombre quelconque. On montre comment, par cette méthode, il est possible de spécifier la composition de protocoles.

On a essayé de rendre l'article le plus « auto-suffisant » possible en ce qui concerne les notions indispensables de calcul des prédicats. En particulier, la notion de *formule valide* est définie à l'aide des outils introduits (le jeu associé à la formule). L'équivalence avec la définition usuelle en logique est indiquée en appendice, mais n'est pas utilisée dans l'article.

Cadre logique

Le langage utilisé, décrit ci-dessous, est le *calcul des prédicats* bien connu en logique ; restriction importante : les seuls symboles logiques autorisés sont $\to, \bot, \forall$, qui se lisent respectivement « implique », « faux », « pour tout ». Il se trouve que tous les autres symboles logiques peuvent être définis à partir de ceux-là (voir plus bas). La restriction est donc seulement syntaxique et non sémantique.

On se donne un ensemble infini de *variables* : $\{x, y, \ldots\}$, un ensemble infini $\mathscr{C} = \{a, b, \ldots\}$ de *constantes* et des *symboles de prédicat* $P, Q, R, \ldots$ qui ont chacun une *arité* (entier naturel).

Une *formule atomique* s'écrit $\bot$ (qui se lit *faux*) ou bien $P t_1 \ldots t_k$ (noté aussi $P(t_1, \ldots, t_k)$) où P est un symbole de prédicat d'arité k et $t_1, \ldots, t_k$ sont des variables ou des constantes. On construit ensuite les formules en appliquant les règles suivantes :

- Une formule atomique est une formule.
- Si F et G sont des formules, alors $F \to G$ est une formule (lire « F implique G »).
- Si F est une formule et x une variable, alors $\forall x\, F$ est une formule (lire « pour tout x, F »).

Remarque. Le *calcul propositionnel* est la partie du calcul des prédicats dans laquelle il n'y a comme symboles logiques que $\to$ et $\bot$, et seulement des symboles de prédicat d'arité 0, habituellement appelés *variables propositionnelles*.

On utilisera systématiquement la notation $A, B \to C$ pour $A \to (B \to C)$ et, plus généralement $A_1, A_2 \ldots, A_n \to B$ pour $A_1 \to (A_2 \to (\cdots (A_n \to B) \cdots))$.

Les connecteurs usuels $\neg, \wedge, \vee, \leftrightarrow$ du calcul propositionnel sont considérés comme des abréviations et définis comme suit :
$\neg F$ est $F \to \bot$; $F \wedge G$ est $(F, G \to \bot) \to \bot$; $F \vee G$ est $\neg F, \neg G \to \bot$;
$F \leftrightarrow G$ est $(F \to G) \wedge (G \to F)$ c'est-à-dire $((F \to G), (G \to F) \to \bot) \to \bot$.

Remarque. Le connecteur `XOR`, usuel en informatique et souvent noté $F \hat{} G$, peut être défini par $\neg F \leftrightarrow G$. Cette abréviation n'est pas utilisée dans les formules du calcul des prédicats.

Le quantificateur $\exists$ (il existe) est aussi considéré comme une abréviation : $\exists x F$ est, par définition, $\neg \forall x \neg F$ ou, en développant : $\forall x (F \to \bot) \to \bot$.

La notation $\vec{x}$ désignera une suite de variables $x_1, \ldots, x_n$.
On écrira donc $\forall \vec{x}$ pour $\forall x_1 \ldots \forall x_n$ et de même avec $\exists$.

Dans une formule de la forme $\forall x A$, la formule A est appelée la *portée* du quantificateur $\forall x$. Une *occurrence* d'une variable x dans une formule F est dite *liée* si elle est dans la portée d'un quantificateur $\forall x$; sinon, cette occurrence est dite *libre*. Etant donnée une occurrence liée de x, le quantificateur qui la lie est, par définition, le quantificateur $\forall x$ dans la portée duquel cette occurrence se trouve qui est le plus proche.
Par exemple, dans la formule $\forall x [\forall x (Rx \to Ry) \to \forall y (Ry \to Rx)]$ il y a une occurrence libre et une occurrence liée pour la variable y et deux occurrences liées pour la variable x. Les deux occurrences de x ne sont pas liées par le même quantificateur.
Une variable x est dite *libre* dans la formule F si elle y a au moins une occurrence libre. La formule F est dite close si elle n'a pas de variable libre.
On écrira $F[x_1, \ldots, x_n]$ (ou $F[\vec{x}]$) pour indiquer que les variables libres de la formule F se trouvent *parmi* $x_1, \ldots, x_n$. La formule $\forall x_1 \ldots \forall x_n F[x_1, \ldots, x_n]$ (ou $\forall \vec{x} F[\vec{x}]$) est alors close.

Dans une formule F, on peut renommer les variables *liées* de façon quelconque, à condition d'éviter les *captures de variables*. Cela veut dire qu'une occurrence libre ne doit pas devenir liée ; et qu'une occurrence liée doit rester liée *par le même quantificateur*. Une formule G obtenue de cette façon à partir de F est considérée comme identique à F.
Par exemple, $\forall z [\forall y (Rx \to Ry) \to Rz]$ est identifiée avec $\forall y [\forall y (Rx \to Ry) \to Ry]$.

Etant données une formule $F[x_1, \ldots, x_k] \equiv F[\vec{x}]$ et des constantes $a_1, \ldots, a_k$, on désigne par $F[a_1, \ldots, a_k] \equiv F[\vec{a}]$ la formule close obtenue en substituant a_i à chaque occurrence *libre* de x_i $(1 \leq i \leq k)$.

Remarque. Une formule *atomique close* $\not\equiv \bot$est de la forme $P a_1 \ldots a_k$, où P est un symbole de prédicat d'arité k et $a_1, \ldots, a_k$ des constantes. Dans l'interprétation qui est faite plus loin en termes de protocoles réseau, une telle formule représente un *paquet*, le symbole de prédicat P représente les *données* et $a_1, \ldots, a_k$ représentent les *champs de l'en-tête* du paquet. Lorsque $k = 0$, c'est-à-dire lorsque P est une variable propositionnelle, P représente un paquet de données pures.

Forme normale d'une formule

Une formule est dite *sous forme normale* si on peut l'obtenir par les règles suivantes :
• une formule atomique A est normale ;
• si $\Phi_1, \ldots, \Phi_n$ sont normales, si A est atomique et si $\vec{x} = (x_1, \ldots, x_k)$ est une suite de variables, alors $\forall \vec{x} (\Phi_1, \ldots, \Phi_n \to A)$ est une formule normale.

Par convention, si $n = 0$, la formule se réduit à $\forall \vec{x}\, A$.
De façon analogue, si $k = 0$ la formule se réduit à $\Phi_1, \ldots, \Phi_n \to A$.

Par exemple, R étant un symbole de prédicat unaire, la formule $\forall x\, Rx \to \forall x\, Rx$ n'est pas sous forme normale et $\forall y(\forall x\, Rx \to Ry)$ est sous forme normale.

A chaque formule F est associée sa forme normale $\widehat{F}$ obtenue de la façon suivante :
• si F est atomique, $\widehat{F} \equiv F$;
• si F est $\forall x\, G$, alors $\widehat{F}$ est $\forall x\, \widehat{G}$;
• si F est $G \to H$, on écrit $\widehat{H} \equiv \forall \vec{x}(\Phi_1, \ldots, \Phi_n \to A)$. On doit éventuellement renommer les variables (liées) $\vec{x}$ de façon qu'aucune ne soit libre dans G et alors $\widehat{F}$ est $\forall \vec{x}[\widehat{G}, \Phi_1, \ldots, \Phi_n \to A]$.

Remarque. Il est clair que F et $\widehat{F}$ ont les mêmes variables libres. En particulier, si F est close, $\widehat{F}$ l'est aussi.
Notons que toute formule du calcul propositionnel est sous forme normale.

Par exemple, la forme normale de la formule $(Rx \to \forall x\, Rx) \to \forall x\, Rx$ est :
$\forall z[\forall y(Rx \to Ry) \to Rz]$ ou $\forall y[\forall y(Rx \to Ry) \to Ry]$.

Jeu associé à une formule close

A chaque formule close F, on associe un jeu à deux joueurs traditionnellement appelés ∃*loïse* et ∀*bélard*. Dans la suite on les désignera par ∃ et ∀ (bien entendu, à ne pas confondre avec les quantificateurs). On appelle aussi ∃loïse la « joueuse » et ∀bélard l'« opposant ».
Intuitivement, la joueuse ∃ défend la formule F, c'est-à-dire prétend qu'elle est « vraie » et l'opposant ∀ l'attaque, c'est-à-dire prétend qu'elle est « fausse ».
Attention, il n'y a aucune symétrie entre les deux joueurs, comme on va le voir par la règle du jeu. Pour préciser l'idée intuitive, on peut dire que ∃ prétend que la formule F est « toujours vraie » et que ∀ prétend qu'elle est « parfois fausse ».

On suppose que la formule close F a été mise sous forme normale.
Voici maintenant la règle du jeu associé à cette formule [jlk] :

On a trois ensembles finis de formules closes normales notés $\mathscr{U}, \mathscr{V}, \mathscr{A}$, qui évoluent au cours de la partie. L'ensemble $\mathscr{A}$ est formé de formules *atomiques* closes. Les ensembles $\mathscr{U}$ et $\mathscr{A}$ croissent au cours de la partie. Au début de la partie, on a $\mathscr{U} = \{F \to \bot\}$, $\mathscr{V} = \{F\}$ et $\mathscr{A} = \{\bot\}$.
C'est le joueur ∀ qui commence.
On se place donc maintenant, en cours de partie, à un moment où c'est à ∀ de jouer.
Si, à ce moment, l'ensemble $\mathscr{V}$ est vide, le jeu s'arrête et ∀ a perdu la partie.
Sinon, il choisit une formule $\Phi \equiv \forall \vec{x}[\Psi_1(\vec{x}), \ldots, \Psi_m(\vec{x}) \to A(\vec{x})]$ qui est dans $\mathscr{V}$ et une suite $\vec{a}$ de constantes, de même longueur que $\vec{x}$.
Il *ajoute* les formules $\Psi_1(\vec{a}), \ldots, \Psi_m(\vec{a})$ à l'ensemble $\mathscr{U}$ et la formule atomique $A(\vec{a})$ à l'ensemble $\mathscr{A}$. C'est alors le tour de la joueuse ∃.
Cette dernière choisit arbitrairement, dans l'ensemble $\mathscr{U}$, une formule :
$\Psi \equiv \forall \vec{y}[\Phi_1(\vec{y}), \ldots, \Phi_n(\vec{y}) \to B(\vec{y})]$; elle choisit aussi une suite $\vec{b}$ de constantes, de même longueur que $\vec{y}$, *de façon que* $B(\vec{b}) \in \mathscr{A}$; c'est toujours possible, puisqu'elle peut, au moins, choisir $F \to \bot$ qui est dans $\mathscr{U}$.
Ensuite, *elle remplace* le contenu de l'ensemble $\mathscr{V}$ par $\{\Phi_1(\vec{b}), \ldots, \Phi_n(\vec{b})\}$.
C'est alors à ∀ de jouer, et ainsi de suite.

Remarques. On voit que le joueur ∀ gagne si et seulement si la partie est infinie. La partie se termine en un temps fini si et seulement si $\mathscr{V}$ devient vide (∃ gagne). Au coup d'avant, la joueuse ∃ a choisi une formule atomique qui est dans $\mathscr{U} \cap \mathscr{A}$.
La signification intuitive de la règle du jeu est la suivante : à chaque instant, la joueuse ∃ prétend que l'une des formules de $\mathscr{U}$ est fausse et que toutes les formules de $\mathscr{V}$ sont vraies. De son côté, le joueur ∀ prétend que toutes les formules de $\mathscr{U}$ sont vraies et que l'une des formules de $\mathscr{V}$ est fausse. Par contre, ils sont tous deux d'accord sur le fait que toutes les formules de $\mathscr{A}$ sont fausses.

Dans les exemples qui suivent, nous interprétons une partie dans ce jeu, comme une session de communication suivant un certain protocole. Le joueur ∀ est alors l'*émetteur* et la joueuse ∃ est la *destinataire*.

La dissymétrie du jeu est bien exprimée dans la phrase célèbre de Jon Postel (dite « loi de Postel » [jp]) : « Be conservative in what you send, be liberal in what you receive ».

Exemples

1) $F \equiv P \to P$. On peut décrire le déroulement d'une partie par le tableau suivant :

$\mathscr{U}$	$\mathscr{V}$	$\mathscr{A}$	
$(P \to P) \to \bot$	$P \to P$	$\bot$	∀ n'a pas le choix
$(P \to P) \to \bot, P$	inchangé	$\bot, P$	∃ choisit $(P \to P) \to \bot$
inchangé	inchangé	inchangé	∀ n'a pas le choix
inchangé	inchangé	inchangé	∃ choisit $(P \to P) \to \bot$
⋮	⋮	⋮	⋮
inchangé	inchangé	inchangé	∃ choisit P
inchangé	$\emptyset$	inchangé	∃ a gagné

Les différentes parties possibles ne dépendent que du nombre n de fois où ∃ choisit la formule $(P \to P) \to \bot$. Si n est infini, ∃ perd.

2) $F \equiv P \to Q$

$\mathscr{U}$	$\mathscr{V}$	$\mathscr{A}$	
$(P \to Q) \to \bot$	$P \to Q$	$\bot$	∀ n'a pas le choix
$(P \to Q) \to \bot, P$	inchangé	$\bot, Q$	∃ ne peut pas choisir P
inchangé	inchangé	inchangé	∀ n'a pas le choix
inchangé	inchangé	inchangé	∃ ne peut pas choisir P
⋮	⋮	⋮	⋮

Il n'y a qu'une seule partie possible et elle est gagnée par ∀, car elle est infinie.

3) Le lecteur est invité à étudier lui-même les deux exemples suivants :
$((Q \to Q) \to P) \to P$; $((P \to Q) \to P) \to P$ (loi de Peirce).

4) $F \equiv \forall x\, Px \to \forall x\, Px$. La forme normale de F est $G \equiv \forall y(\forall x\, Px \to Py)$.

$\mathscr{U}$	$\mathscr{V}$	$\mathscr{A}$	
$\neg G$	G	$\bot$	$\forall$ choisit b_0
$\neg G, \forall x\, Px$	inchangé	$\bot, Pb_0$	$\exists$ choisit $\neg G$
inchangé	inchangé	inchangé	$\forall$ choisit b_1
inchangé	inchangé	$\bot, Pb_0, Pb_1$	$\exists$ choisit $\neg F$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
inchangé	inchangé	$\bot, Pb_0, \ldots, Pb_n$	$\exists$ choisit $\forall x\, Px$ et b_i
inchangé	$\emptyset$	inchangé	$\exists$ a gagné

Comme dans l'exemple 1, le déroulement de la partie ne dépend que du moment où la joueuse $\exists$ choisit $\forall x\, Px$ et l'un des b_i précédemment choisis par le joueur $\forall$.

On peut donner l'interprétation suivante en termes de réseau : le joueur $\forall$ expédie le paquet de données P avec les en-têtes b_0, puis $b_1, \ldots$ L'acquittement par la destinataire $\exists$ n'a lieu qu'à l'étape n, et c'est le paquet Pb_i qui est acquitté. La partie, c'est-à-dire la session, s'arrête alors immédiatement. Cela veut dire, du point de vue réseau, que l'acquittement ne peut pas être perdu, autrement dit, que le canal du destinataire $\exists$ vers l'expéditeur $\forall$ est fiable.

Dans la section suivante, on traite un exemple particulièrement important : l'acquittement d'un paquet dans un canal non fiable.

La formule $\exists x(Px \to \forall y\, Py)$

On appelle F la forme normale de cette formule, à savoir $\forall x(\forall y(Px \to Py) \to \bot) \to \bot$. Pour alléger l'écriture, on pose $G[x] \equiv \forall y(Px \to Py)$ et on a donc $F \equiv \neg\forall x \neg G[x]$.

Les tableaux I et II suivants représentent le déroulement d'une partie, dans le cas (très particulier) où $\exists$ joue de façon à gagner le plus vite possible. Il y a deux possibilités, suivant ce que joue $\forall$ à la ligne 3 :

Tableau I

	$\mathscr{U}$	$\mathscr{V}$	$\mathscr{A}$	
1	$\neg F$	F	$\bot$	$\forall$ choisit F
2	$\neg F, \forall x \neg G[x]$	inchangé	inchangé	$\exists$ choisit $\forall x \neg G[x]$ et a
3	inchangé	$G[a] \equiv \forall y(Pa \to Py)$	inchangé	$\forall$ choisit b, avec $b \neq a$
4	$\neg F, \forall x \neg G[x], Pa$	inchangé	$\bot, Pb$	$\exists$ choisit $\forall x \neg G[x]$ et b
5	inchangé	$G[b] \equiv \forall y(Pb \to Py)$	inchangé	$\forall$ choisit c
6	$\neg F, \forall x \neg G[x], Pa, Pb$	inchangé	$\bot, Pb, Pc$	$\exists$ choisit Pb
7	inchangé	$\emptyset$	inchangé	$\exists$ a gagné

Tableau II

	$\mathscr{U}$	$\mathscr{V}$	$\mathscr{A}$	
1	$\neg F$	F	$\bot$	$\forall$ choisit F
2	$\neg F, \forall x \neg G[x]$	inchangé	inchangé	$\exists$ choisit $\forall x \neg G[x]$ et a
3	inchangé	$G[a] \equiv \forall y(Pa \to Py)$	inchangé	$\forall$ choisit a
4	$\neg F, \forall x \neg G[x], Pa$	inchangé	$\bot, Pa$	$\exists$ choisit Pa
5	inchangé	$\emptyset$	inchangé	$\exists$ a gagné

Mais il ne s'agit là que de cas particuliers. Le jeu considéré présente, en effet, toute une variété de parties possibles. On va voir que ces diverses parties correspondent exactement

aux diverses possibilités qui se présentent lors de l'acquittement d'un paquet.
La partie décrite dans le tableau I représente le cas où l'échange s'est passé parfaitement. L'interprétation est la suivante : la destinataire $\exists$ initie la session avec l'envoi de l'en-tête a (ligne 3) ; l'expéditeur $\forall$ envoie le paquet Pb (ligne 4) ; $\exists$ reçoit le paquet Pb et envoie l'acquittement (ligne 5) ; $\forall$ a bien reçu cet acquittement et envoie la fin de session Pc (ligne 6).

Plusieurs variantes possibles :
i) La joueuse $\exists$ peut, à tout moment, choisir la formule $\neg F$. Cela correspond à une réinitialisation de la session.
ii) Elle peut aussi choisir la formule $\forall x \neg G[x]$ avec un en-tête a' quelconque, ne correspondant à aucun acquittement. Le joueur $\forall$ doit alors réexpédier le paquet. Cette situation correspond à la perte de l'acquittement.
iii) Dans ce cas, l'expéditeur $\forall$ a la possibilité de renvoyer Pa', ce qui donne à $\exists$ la possibilité de mettre fin immédiatement à la session en choisissant justement la formule Pa' (car elle se trouve alors à la fois dans $\mathscr{U}$ et $\mathscr{A}$). Cela correspond au cas où l'expéditeur demande une fin de session. Cela peut arriver dès le début : c'est ce qui se passe dans la partie décrite dans le tableau II (ligne 3 : $\forall$ choisit a) ; cela correspond à un refus d'ouverture de session ; $\exists$ ne peut alors que clore la session, en choisissant Pa (c'est encore ce qui se passe dans le tableau II) ou la réinitialiser (en choisissant $\neg F$ ou $\forall x \neg G[x]$).
iv) La joueuse $\exists$ peut terminer la partie en choisissant la formule Pb, b étant l'un quelconque des en-têtes envoyés par $\forall$. Cela correspond à une session couronnée de succès, après d'éventuelles pertes d'acquittements.

Une session quelconque est une combinaison d'un nombre arbitraire de ces variantes.

L'envoi de plusieurs paquets

On considère ici le cas de l'acquittement d'un nombre n de paquets, n étant fixé à l'avance et l'ordonnancement étant préservé. La formule associée F_n est définie par récurrence : $F_1 \equiv \exists x \forall y (P_1 x \to P_1 y)$; $F_{n+1} \equiv \exists x \forall y ((F_n \to P_{n+1} x) \to P_{n+1} y)$; F_n est sous forme normale.

Pour fixer les idées, on va considérer le cas $n = 2$, soit donc la formule :
$F' \equiv \exists x \forall y ((F \to Px) \to Py)$ avec $F \equiv \exists x \forall y (Qx \to Qy)$.
On pose $G[x] \equiv \forall y ((F \to Px) \to Py)$, $H[x] \equiv \forall y (Qx \to Qy)$;
on a donc $F' \equiv \forall x \neg G[x] \to \bot$ et $F \equiv \forall x \neg H[x] \to \bot$.
Le tableau ci-dessous décrit encore le déroulement d'une partie où $\exists$ termine le plus vite possible. Pour alléger l'écriture, dans les colonnes $\mathscr{U}$ et $\mathscr{A}$, on ne mettra à chaque ligne que les *nouvelles* formules.

	$\mathscr{U}$	$\mathscr{V}$	$\mathscr{A}$	
1	$\neg F'$	F'	$\bot$	$\forall$ n'a pas le choix
2	$\forall x \neg G[x]$	inchangé	inchangé	$\exists$ choisit $\forall x \neg G[x]$ et a
3	inchangé	$G[a] \equiv \forall y((F \to Pa) \to Py)$	inchangé	$\forall$ choisit b, avec $b \neq a$
4	$F \to Pa$	inchangé	Pb	$\exists$ choisit $\forall x \neg G[x]$ et b
5	inchangé	$G[b] \equiv \forall y((F \to Pb) \to Py)$	inchangé	$\forall$ choisit c
6	$F \to Pb$	inchangé	Pc	$\exists$ choisit Pb
7	inchangé	F	inchangé	$\forall$ n'a pas le choix
.....		*acquittement du premier paquet*		.....
8	$\forall x \neg H[x]$	inchangé	inchangé	$\exists$ choisit $\forall x \neg H[x]$ et d
9	inchangé	$H[d] \equiv \forall y(Qd \to Qy)$	inchangé	$\forall$ choisit e, avec $e \neq d$
10	Qd	inchangé	Qe	$\exists$ choisit $\forall x \neg H[x]$ et e
11	inchangé	$H[e] \equiv \forall y(Qe \to Qy)$	inchangé	$\forall$ choisit f
12	Qe	inchangé	Qf	$\exists$ choisit Qe
13	inchangé	$\emptyset$	inchangé	$\exists$ a gagné
.....		*acquittement du deuxième paquet*		.....

Dans ce cas particulier, on a essentiellement deux fois le tableau I de l'exemple précédent. On peut, bien entendu, avoir toutes les variantes déjà décrites. Mais de nouvelles variantes apparaissent : en effet, après l'acquittement du premier paquet (lignes 8, 10 et 12), la joueuse $\exists$ peut, par exemple, revenir à la ligne 4, c'est-à-dire redemander le premier paquet. Autrement dit, la destinataire peut perdre un paquet, même après l'avoir correctement acquitté. Il est intéressant de noter qu'elle n'a plus à l'acquitter à nouveau.

Stratégies et formules valides

Considérons le jeu associé à une formule F normale close. Une *stratégie* pour $\exists$ dans ce jeu est, par définition, une fonction $\mathscr{S}$, qui prend comme argument une suite finie de triplets $(\mathscr{U}_i, \mathscr{V}_i, \mathscr{A}_i)_{0 \leq i \leq n}$ ($\mathscr{U}_i, \mathscr{V}_i$ sont des ensembles finis de formules closes normales et $\mathscr{A}_i$ un ensemble fini de formules atomiques closes) et rend un couple $(\Psi, \vec{b})$ avec $\Psi \in \mathscr{U}_n$, $\Psi \equiv \forall \vec{y}[\Phi_1(\vec{y}), \ldots, \Phi_k(\vec{y}) \to B(\vec{y})]$, $\vec{b}$ a la même longueur que $\vec{y}$ et $B(\vec{b}) \in \mathscr{A}_n$.
Intuitivement, une stratégie $\mathscr{S}$ pour $\exists$loïse est une méthode générale qui, à chaque fois que c'est à elle de jouer, choisit un coup possible à sa place, en fonction de tout ce qui a été joué jusqu'alors.
La stratégie $\mathscr{S}$ est appelée *stratégie gagnante* si $\exists$ gagne toute partie jouée en suivant cette stratégie, quels que soient les choix de $\forall$.
On définirait de façon analogue les stratégies gagnantes pour $\forall$.

Une formule F close normale est dite *valide* s'il existe une stratégie gagnante pour $\exists$, dans le jeu associé à F. Les formules valides sont exactement celles qui correspondent à des protocoles réseau.

Jeux associés à une conjonction ou une disjonction.

Etant données deux formules F, G, le jeu associé à la formule $F \wedge G$, c'est-à-dire $(F, G \to \bot) \to \bot$ consiste essentiellement en ceci (vérification immédiate) :

le joueur $\forall$ choisit l'une des deux formules et le jeu se poursuit suivant la formule choisie (toutefois, la joueuse $\exists$ peut, à tout moment, décider de recommencer la partie).
Avec la formule $F \vee G$, c'est la joueuse $\exists$ qui choisit.

Composition de protocoles

Considérons deux formules valides F et G, correspondant respectivement aux « protocoles » (autrement dit aux jeux) $\mathscr{P}_F$ et $\mathscr{P}_G$; on se propose de construire une formule valide H telle que le protocole associé $\mathscr{P}_H$ soit : $\mathscr{P}_F$ puis $\mathscr{P}_G$.

Soit $A = P(x_1, \ldots, x_n)$ (ou $\bot$) une formule atomique et F une formule normale. Une « occurrence » de A dans F est simplement l'un des endroits, dans F, où apparaît A.
Chaque occurrence d'une formule atomique A de F apparaît à la fin d'une sous-formule de F, de la forme $\forall \vec{x}(\Psi_1, \ldots, \Psi_k \to A)$; k sera appelé le *nombre d'hypothèses* de l'occurrence atomique A considérée.

Chaque occurrence d'une formule atomique A dans F est soit *positive*, soit *négative*. Cette propriété est définie, par récurrence sur la longueur de F, de la façon suivante :

- Si F est atomique, alors $F \equiv A$ et l'occurrence de A dans F est positive.
- Si $F \equiv G \to H$, l'occurrence considérée de A dans F se trouve soit dans G, soit dans H. Si elle est dans H, elle a le même signe dans F que dans H. Si elle est dans G, elle a des signes opposés dans F et dans G.
- Si $F \equiv \forall x\, G$, l'occurrence de A considérée a le même signe dans F et dans G.

Une occurrence atomique A dans F, qui est *en position négative et sans hypothèse*, sera appelée *occurrence atomique terminale*. Elle correspond, en effet, à la fin d'une partie.

On peut maintenant construire la formule H cherchée : *elle est obtenue en remplaçant, dans F, chaque occurrence atomique terminale A par $G \to A$.*

Remarque. Il est facile de montrer que, si F et G sont valides, la formule H ainsi définie est également valide (voir l'appendice).

Exemple. Prenons $F \equiv \forall x[\forall y(Px \to Py) \to \bot] \to \bot$ qui correspond à l'envoi et l'acquittement d'un paquet. On a alors : $H \equiv \forall x[\forall y((G \to Px) \to Py) \to \bot] \to \bot$.
En effet, il y a, dans F, deux occurrences atomiques négatives qui sont Px et la première occurrence de $\bot$. La seule occurrence atomique sans hypothèse est Px.
En particulier, si on prend $G \equiv \forall x[\forall y(Qx \to Qy) \to \bot] \to \bot$, on obtient le protocole correspondant à l'envoi de deux paquets (voir ci-dessus).

Formules et protocoles utilisant des variables d'entiers

On considère maintenant des formules écrites avec un nouveau type de variables : le type « entier » ; on utilisera, pour les variables de ce type, les lettres i, j, k, l, m, n. Il y a donc maintenant deux types de variables : le type entier et le type utilisé jusqu'ici, que nous appellerons type « acquittement » ; on utilisera, pour ce type, les lettres x, y, z.
On a, en outre, des symboles de fonction *sur le type entier*, en particulier 0 et s (pour la fonction successeur, c'est-à-dire $sn = n+1$). Chaque symbole de fonction f a une arité $k \in \mathbb{N}$ et représente une fonction bien déterminée notée également $f : \mathbb{N}^k \to \mathbb{N}$. On peut définir les *termes* de type entier par les règles suivantes :

• une variable ou un symbole de fonction d'arité 0 (constante d'entier) est un terme de type entier.
• si f est un symbole de fonction d'arité k et $t_1, \ldots, t_k$ sont des termes de type entier, alors $f(t_1, \ldots, t_k)$ est un terme de type entier.

Notons qu'un terme de type entier sans variable (terme clos) représente un entier.

Les prédicats sont également typés. Par exemple Pnx ou $P(n, x)$ (la première place de P est de type entier, la deuxième est de type acquittement).

Définition des formules.
• Formules atomiques : $Pt_1 \ldots t_k$; t_i est une constante ou une variable de type acquittement si la i-ième place est de ce type ; un terme de type entier, si la i-ième place est de type entier.
• Si F, G sont des formules, $F \to G$ aussi.
• Si F est une formule, $\forall x\, F$ et $\forall n\, F$ aussi.
• Si F est une formule et t, u sont deux termes de type entier, alors $t = u \to F$ est une formule.

Attention, l'expression $t = u$ toute seule n'est pas une formule.

Formes normales.
Elles sont définies comme suit :
• Une formule atomique est normale.
• Si F est normale, $\forall x F$, $\forall n F$ sont normales.
• Si A est atomique, si $\Phi_1, \ldots, \Phi_k$ sont des formules normales ou des expressions de la forme $t = u$, alors $\Phi_1, \ldots, \Phi_k \to A$ est une formule normale.

La mise sous forme normale d'une formule se fait exactement comme dans le cas précédent.

Jeu associé à une formule close sous forme normale.
On indique ici seulement les ajouts à la règle du jeu précédemment écrite :

i) quand l'un des joueurs a choisi une formule $\forall \vec{\xi}(\Phi_1, \ldots, \Phi_n \to A)$, ($\vec{\xi} = (\xi_1, \ldots, \xi_n)$ où ξ_i est une variable de type entier ou acquittement) :
- il choisit d'abord des valeurs $\vec{a}$ pour $\vec{\xi}$.
- il évalue ensuite les expressions Φ_j de la forme $t_j = u_j$.
- si elles sont toutes vraies, on les enlève et on continue la partie comme précédemment avec la formule ainsi obtenue.
- si l'une d'elles est fausse :
  si c'est l'opposant $\forall$ qui joue, il a perdu.
  si c'est la joueuse $\exists$, elle doit choisir d'autres valeurs $\vec{a}$ ou une autre formule (ce qui est toujours possible, comme on l'a déjà vu).

ii) comme dans le jeu précédent, lorsque la joueuse $\exists$ choisit, dans l'ensemble $\mathscr{U}$, une formule $\Psi \equiv \forall \vec{y}[\Phi_1(\vec{y}), \ldots, \Phi_n(\vec{y}) \to B(\vec{y})]$ et une suite $\vec{b}$ de constantes, de même longueur que $\vec{y}$, elle doit vérifier que $B(\vec{b}) \in \mathscr{A}$, c'est-à-dire vérifier que deux formules atomiques closes sont identiques. Ces formules peuvent comporter des termes (clos) de type entier et on doit *calculer ces termes avant de les comparer.*

Formules ω-valides.
Une formule F close normale sera dite *ω-valide* s'il existe une stratégie gagnante pour $\exists$, dans le jeu associé à F. Les formules ω-valides sont exactement celles qui correspondent à des protocoles réseau (comme dans le cas précédent).

Exemple.
$\forall$ envoie un entier n ; puis acquittement de n paquets. La formule est :
$F \equiv \forall j\{\forall i[j = si \to \exists x \forall y((Ui \to Pix) \to Piy)] \to Uj\} \to \forall n\, Un$.
U est un symbole de prédicat à une place de type entier ; P est à deux places, la première de type entier, la seconde de type acquittement.

Posons $G \equiv \forall j\{\forall i[j = si \to \exists x \forall y((Ui \to Pix) \to Piy)] \to Uj\}$ et
$H[i,x] \equiv \forall y((Ui \to Pix) \to Piy)$.
Une fois mise sous forme normale, la formule F s'écrit $F \equiv \forall n(G \to Un)$.

Le tableau suivant représente la session particulière dans laquelle tous les paquets sont acquittés le plus rapidement possible.

	$\mathscr{U}$	$\mathscr{V}$	$\mathscr{A}$	
1	$\neg F$	F	$\bot$	$\forall$ choisit n_0
2	G	inchangé	Un_0	$\exists$ choisit G et n_0
3	inchangé	$\forall i(n_0{=}si{\to}\exists x H[i,x])$	inchangé	$\forall$ ne peut que choisir n_0-1
4	$\forall x \neg H[n_0{-}1,x]$	inchangé	inchangé	$\exists$ choisit a_0
5	inchangé	$H[n_0{-}1,a_0]$	inchangé	$\forall$ choisit $b_0 \neq a_0$
6	$U(n_0{-}1){\to}P(n_0{-}1,a_0)$	inchangé	$P(n_0{-}1,b_0)$	$\exists$ choisit $\forall x \neg H[n_0{-}1,x]$ et b_0
7	inchangé	$H[n_0{-}1,b_0]$	inchangé	$\forall$ choisit $b_1 \neq a_0, b_0$
8	$U(n_0{-}1){\to}P(n_0{-}1,b_0)$	inchangé	$P(n_0{-}1,b_1)$	$\exists$ choisit $U(n_0{-}1){\to}P(n_0{-}1,b_0)$
9	inchangé	$U(n_0{-}1)$	inchangé	$\forall$ ne peut que choisir $U(n_0{-}1)$
.....		*acquittement du paquet* $n_0{-}1$		.....
10	inchangé	inchangé	$U(n_0{-}1)$	$\exists$ choisit G et $n_0{-}1$
⋮	⋮	⋮	⋮	⋮
.....		*acquittement du paquet* 0		.....
	inchangé	inchangé	$U0$	$\exists$ choisit G et 0
	inchangé	$\forall i(0{=}si{\to}\exists x H[i,x])$	inchangé	$\forall$ a perdu

Pour éviter une complication supplémentaire, on n'a pas demandé que l'entier n (nombre de paquets à transmettre), envoyé par $\forall$, soit acquitté. Si on veut, de plus, que cet entier soit acquitté, il faut ajouter un champ « acquittement » au prédicat U, qui devient donc binaire. On écrit alors la formule suivante :
$F \equiv \forall n \exists x' \forall y'((\neg G[x'] \to Unx') \to Uny')$ avec
$G[x'] \equiv \forall j\{\forall i[j = si \to \exists x \forall y((Uix' \to Pix) \to Piy)] \to Ujx'\}$

Le tableau suivant donne le début d'une session et montre l'acquittement de l'entier n.
On a posé $H[n,x'] \equiv \forall y'((\neg G[x'] \to Unx') \to Uny')$.

	$\mathscr{U}$	$\mathscr{V}$	$\mathscr{A}$	
1	$\neg F$	F	$\bot$	$\forall$ choisit n_0
2	$\forall x' \neg H[n_0, x']$	inchangé	inchangé	$\exists$ choisit $\forall x' \neg H[n_0, x']$ et x'_0
3	inchangé	$H[n_0, x'_0]$	inchangé	$\forall$ choisit y'_0
4	$\neg G[x'_0] \to Un_0 x'_0$	inchangé	$Un_0 y'_0$	$\exists$ choisit $\forall x' \neg H[n_0, x']$ et y'_0
5	inchangé	$H[n_0, y'_0]$	inchangé	$\forall$ choisit y'_1
6	$\neg G[y'_0] \to Un_0 y'_0$	inchangé	$Un_0 y'_1$	$\exists$ choisit $\neg G[y'_0] \to Un_0 y'_0$
7	inchangé	$\neg G[y'_0]$	inchangé	$\forall$ n'a pas le choix
8	$G[y'_0]$	inchangé	inchangé	$\exists$ choisit $G[y'_0]$ et n_0
	$\vdots$	$\vdots$	$\vdots$	$\vdots$

A partir de là, la partie continue comme dans l'exemple précédent (la formule $G[y'_0]$ remplace la formule G).

Remarque. On peut écrire une formule un peu plus simple pour le même protocole, à savoir $\forall n \exists x' \forall y' (G[x'] \to Uny')$ avec, comme ci-dessus :
$G[x'] \equiv \forall j \{\forall i [j = si \to \exists x \forall y ((Uix' \to Pix) \to Piy)] \to Ujx'\}$.
Le lecteur le vérifiera aisément.

Appendice

Formules valides.
La définition habituelle des formules valides du calcul des prédicats utilise la notion de *modèle*. Le lecteur intéressé la trouvera, par exemple, dans [rcdl] ou dans [jrs]. Une formule est dite valide si elle est satisfaite dans tous les modèles. Un théorème fondamental en logique, le *théorème de complétude*, affirme qu'une formule est valide si et seulement si elle est démontrable avec les règles de logique pure.
Cette notion de validité est équivalente à celle introduite dans le présent article, au moyen des stratégies (voir la preuve dans [jlk]).
Il est souvent plus facile de vérifier qu'une formule est valide, à l'aide des modèles. Par exemple, on voit ainsi immédiatement que la formule $F \equiv \exists x (Px \to \forall y Py)$ est valide : en effet, ou bien le modèle considéré satisfait $\forall y Py$ et donc aussi F, ou bien il satisfait $\exists x \neg Px$ et donc encore F.

Considérons deux formules valides F et G, et soit H la formule définie plus haut, telle que le protocole $\mathscr{P}_H$ associé à H soit le composé des protocoles associés à F et G. On montre alors aisément que H est valide. En effet, on a obtenu H en remplaçant, dans F, certaines sous-formules A par $G \to A$. Or A et $G \to A$ sont trivialement équivalentes, puisque G est valide. On obtient donc finalement une formule H équivalente à F, et donc valide.

Formules avec type entier
Pour les formules à deux types (entier et acquittement), la situation est un peu plus complexe. Les formules ω-valides sont les formules satisfaites dans tous les *ω-modèles*, c'est-à-dire les modèles dans lesquels le type entier a son interprétation standard. Dans ce cas encore, il est souvent plus facile d'utiliser cette définition pour vérifier qu'une formule est ω-valide.
Par exemple, on vérifie facilement que la formule (étudiée plus haut) $F \equiv G \to \forall n Un$, avec

$G \equiv \forall j\{\forall i[j = si \to \exists x \forall y((Ui \to Pix) \to Piy)] \to Uj\}$ est ω-valide.
En effet, on fait l'hypothèse G et on montre Un par récurrence sur l'entier n.
Preuve de $U0$. On fait $j = 0$ dans G ; comme $0 = si$ est faux, on en tire $U0$.
Preuve de $Un \to U(n+1)$. On fait $j = n+1$, dans G. Il suffit de montrer :
$\forall i[n+1 = si \to \exists x \forall y((Ui \to Pix) \to Piy)]$ sous l'hypothèse Un. Comme $n+1 = si$ équivaut à $i = n$, on est ramené à montrer $\exists x \forall y((Un \to Pnx) \to Pny)$, c'est-à-dire $\exists x \forall y(Pnx \to Pny)$ (puisqu'on suppose Un). Cette dernière formule est déjà démontrée.

Avec quelques modifications simples, cette preuve montre aussi les formules :
$\forall n \exists x' \forall y'((\neg G[x'] \to Unx') \to Uny')$ et $\forall n \exists x' \forall y'(G[x'] \to Uny')$
avec $G[x'] \equiv \forall j\{\forall i[j = si \to \exists x \forall y((Uix' \to Pix) \to Piy)] \to Ujx'\}$.
En fait, il suffit de montrer la première, car elle implique trivialement la seconde.

Détermination des jeux.
Un jeu est dit *déterminé* si l'un des deux joueurs a une stratégie gagnante. C'est toujours le cas pour les jeux considérés dans cet article (théorème de Gale-Stewart).
Idée de la preuve. Supposons que $\exists$ n'ait pas de stratégie gagnante. Alors la stratégie suivante est gagnante pour le joueur $\forall$: jouer à chaque étape de façon à ce que la joueuse $\exists$ n'ait toujours pas de stratégie gagnante. Le jeu dure alors indéfiniment, donc $\forall$ gagne.

Références

[rcdl] René Cori & Daniel Lascar. *Logique mathématique.* Masson (1994).
[jlk] Jean-Louis Krivine. *Realizability : a machine for Analysis and set theory* . Geocal'06, Marseille (2006).
`http://cel.archives-ouvertes.fr/cel-00154509`. Version plus récente à :
`http://www.pps.jussieu.fr/~krivine/articles/Mathlog07.pdf`
[jp] Jon Postel. *RFC 793 - Transmission Control Protocol specification.* (1981).
[jrs] Joseph R. Schoenfield. *Mathematical logic.* Addison Wesley (1967).