Valid formulas, games and network protocols

Jean-Louis Krivine, Yves Legrandgérard

Valid formulas, games and network protocols

Jean-Louis Krivine & Yves Legrandgérard
Paris VII University, C.N.R.S.

November 14, 2007

Introduction

We describe a remarkable relation between a fundamental notion of mathematical logic – that is *valid formula of predicate calculus* – and the specification of network protocols. We explain here in detail several simple examples : the acknowledgement of one or two packets, and then of an arbitrary number. We show that, using this method, it is possible to specify the composition of protocols.

We tried to write a self-contained paper, as far as possible, in what concerns the basic notions of the calculus of predicates. In particular, the notion of *valid formula* is defined with the help of the tools introduced in the present paper (specifically, the game associated with the formula). The equivalence with the usual definition of this notion in logic is explained in the appendix, but is never used in the paper.

Logical framework

The language we use is described below. It is the well known *predicate calculus*, fundamental in mathematical logic ; important restriction : the only allowed logical symbols are $\to, \perp, \forall$, respectively read as " implies ", " false ", " for all ". In fact, every other logical symbol can be defined with them (see below). This restriction is therefore only syntactic, but not semantic.

We suppose given an infinite set of *variables* : $\{x, y, \ldots\}$, an infinite set of *constants* : $\mathcal{C} = \{a, b, \ldots\}$ and some *predicate symbols* $P, Q, R, \ldots$; each of them has an *arity* which is an integer ≥ 0.

Atomic formulas are of the form $\perp$ (read *false*) or $Pt_1 \ldots t_k$ (denoted also as $P(t_1, \ldots, t_k)$) where P is a predicate symbol of arity k and $t_1, \ldots, t_k$ are variables or constants.
Formulas of the predicate calculus are built with the following rules :

- An atomic formula is a formula.
- If F and G are formulas, then $F \to G$ is a formula (read « F implies G »).
- If F is a formula and x is a variable, then $\forall x\, F$ is a formula (read " for all x, F ").

Remark. *Propositional calculus* is contained in predicate calculus : it has the only logical symbols $\to$ and $\perp$, and only predicate symbols of arity 0, usually called *propositional variables.*

We shall systematically use the notation $A, B \to C$ for $A \to (B \to C)$ and, more generally $A_1, A_2 \ldots, A_n \to B$ for $A_1 \to (A_2 \to (\cdots (A_n \to B) \cdots))$.
Usual connectives $\neg, \wedge, \vee, \leftrightarrow$ of propositional calculus are considered as abbreviations, and defined as follows :
$\neg F$ is $F \to \bot$; $F \wedge G$ is $(F, G \to \bot) \to \bot$; $F \vee G$ is $\neg F, \neg G \to \bot$;
$F \leftrightarrow G$ is $(F \to G) \wedge (G \to F)$ that is $((F \to G), (G \to F) \to \bot) \to \bot$.

Remark. The connective `XOR`, usual in computer science and often denoted as $F\,\check{}\,G$, can be defined as $\neg F \leftrightarrow G$. This abbreviation is not used in the formulas of predicate calculus.

The existential quantifier $\exists$ (read " there exists ") is also considered as an abbreviation : $\exists x\, F$ is defined as $\neg \forall x \neg\, F$ that is $\forall x(F \to \bot) \to \bot$.

The notation $\vec{x}$ will denote a finite sequence of variables $x_1, \ldots, x_n$.
Therefore, we shall write $\forall \vec{x}$ for $\forall x_1 \ldots \forall x_n$ and the same with $\exists$.

In a formula such as $\forall x\, A$, the subformula A is called the *scope* of the quantifier $\forall x$. An *occurrence* of a variable x in a formula F is called *bounded* if it is in the scope of a quantifier $\forall x$; otherwise, this occurrence is called *free*. Given a bounded occurrence of x, the quantifier which bounds it, is by definition, the nearest quantifier $\forall x$ which has this occurrence in its scope.
For instance, in the formula $\forall x[\forall x(Rx \to Ry) \to \forall y(Ry \to Rx)]$ there are a bound and a free occurrence of the variable y and two bounded occurrences of the variable x. These two occurrences of x are not bounded by the same quantifier.
A variable x is called *free* in the formula F if there is at least one free occurrence of x. The formula F is called *closed* if it contains no free variable.
The notation $F[x_1, \ldots, x_n]$ (or $F[\vec{x}]$) will mean that the free variables of the formula F are *amongst* $x_1, \ldots, x_n$. Then, the formula $\forall x_1 \ldots \forall x_n\, F[x_1, \ldots, x_n]$ (or $\forall \vec{x}\, F[\vec{x}]$) is closed.

In any formula F, we can rename the *bounded* variables in an arbitrary way, provided that no *capture of variable* occurs. This means that no free occurrence becomes bound ; and that any bound occurrence must remain bounded *by the same quantifier*. Any formula G, obtained from F in this way is considered as identical with F.
For instance, $\forall z[\forall y(Rx \to Ry) \to Rz]$ is identified with $\forall y[\forall y(Rx \to Ry) \to Ry]$.

For any formula $F[x_1, \ldots, x_k] \equiv F[\vec{x}]$ and constants $a_1, \ldots, a_k$, we denote by $F[a_1, \ldots, a_k] \equiv F[\vec{a}]$ the closed formula we obtain by replacing each *free* occurrence of x_i with a_i $(1 \leq i \leq k)$.

Remark. Any *atomic closed* formula $\not\equiv \bot$ has the form $Pa_1 \ldots a_k$, where P is a predicate symbol of arity k and $a_1, \ldots, a_k$ are constants. In the interpretation in terms of network protocols which is given below, such a formula represents a *packet*, the predicate symbol P represents the *datas* and $a_1, \ldots, a_k$ represent the *header fields* of the packet. When $k = 0$, i.e. when P is a propositional variable, P represents a pure data packet.

Normal form of a formula

A formula is said to be *in normal form* or *normal*, if it can be obtained by means of the following rules :
• an atomic formula A is normal ;

• if $\Phi_1, \ldots, \Phi_n$ are normal, if A is atomic and if $\vec{x} = (x_1, \ldots, x_k)$ is a finite sequence of variables, then $\forall \vec{x}(\Phi_1, \ldots, \Phi_n \to A)$ is a normal formula.
If $n = 0$, by definition, this formula is $\forall \vec{x}\, A$.
In the same way, if $k = 0$ this formula is $\Phi_1, \ldots, \Phi_n \to A$.

For instance, if R is a unary predicate symbol, the formula $\forall x\, Rx \to \forall x\, Rx$ is not a normal form ; but $\forall y(\forall x\, Rx \to Ry)$ is a normal form.

With any formula F, we associate its normal form $\widehat{F}$, which is obtained as follows :
• if F is atomic, $\widehat{F} \equiv F$;
• if F is $\forall x\, G$, then $\widehat{F}$ is $\forall x\, \widehat{G}$;
• if F is $G \to H$, we write $\widehat{H} \equiv \forall \vec{x}(\Phi_1, \ldots, \Phi_n \to A)$. We first rename the (bounded) variables $\vec{x}$ so that they become not free in G (a good method is to use variables that do not appear in G) ; then, $\widehat{F}$ is $\forall \vec{x}[\widehat{G}, \Phi_1, \ldots, \Phi_n \to A]$.

Remark. Obviously, F and $\widehat{F}$ have the same free variables. In particular, if F is closed, then $\widehat{F}$ is also closed.
Note that any formula of the propositional calculus is in normal form.

For instance, the normal form of the formula $(Rx \to \forall x\, Rx) \to \forall x\, Rx$ is :
$\forall z[\forall y(Rx \to Ry) \to Rz]$ or $\forall y[\forall y(Rx \to Ry) \to Ry]$.

The game associated with a closed formula

Given a closed formula F, we define a two players' game ; the players will be called $\exists$*loise* and $\forall$*belard* or, more briefly, $\exists$ and $\forall$ (the same notation as the quantifiers, but no confusion is possible). $\exists$loise is also called the " player " or the " defender " and $\forall$belard is called the " opponent ".
Intuitively, the player $\exists$ defends the formula F, i.e. pretends this formula is " true " and the opponent $\forall$ attacks it, i.e. pretends it is " false ".
Be careful, there is no symmetry between the players, as it will be seen by the rule of the game. To make the intuitive idea more precise, we can say that $\exists$ pretends the formula F is " always true " and that $\forall$ pretends it is " sometimes false ".

Now, we assume that the closed formula F has been put in normal form.
Here is the rule of the game associated with this formula [jlk] :

We have three finite sets of normal closed formulas, denoted by $\mathcal{U}, \mathcal{V}, \mathcal{A}$, which change during the play. The elements of the set $\mathcal{A}$ are closed *atomic* formulas. The sets $\mathcal{U}$ and $\mathcal{A}$ *increase* during the play. At the beginning of the play, we have $\mathcal{U} = \{F \to \bot\}$, $\mathcal{V} = \{F\}$ and $\mathcal{A} = \{\bot\}$ (one-element sets). The first move is done by the opponent $\forall$.
Consider now, during the play, a moment when the opponent $\forall$ must play.
If, at this moment, the set $\mathcal{V}$ is empty, the game stops and $\forall$ has lost.
Otherwise, he chooses a formula $\Phi \equiv \forall \vec{x}(\Psi_1[\vec{x}], \ldots, \Psi_m[\vec{x}] \to A[\vec{x}])$ which is in $\mathcal{V}$ and a sequence $\vec{a}$ of constants, of the same length as $\vec{x}$.
Then *he adds* the formulas $\Psi_1[\vec{a}], \ldots, \Psi_m[\vec{a}]$ to the set $\mathcal{U}$ and also the atomic formula $A[\vec{a}]$ to the set $\mathcal{A}$. Then the defender $\exists$ must play.
She chooses, in the set $\mathcal{U}$, a formula $\Psi \equiv \forall \vec{y}(\Phi_1[\vec{y}], \ldots, \Phi_n[\vec{y}] \to B[\vec{y}])$; she chooses also a sequence $\vec{b}$ of constants, of the same length as $\vec{y}$, *in such a way that* $B[\vec{b}] \in \mathcal{A}$; this is always possible, since she can, at least, choose $F \to \bot$ which is in $\mathcal{U}$.

Then, *she replaces* the content of the set $\mathcal{V}$ with $\{\Phi_1[\vec{b}], \ldots, \Phi_n[\vec{b}]\}$.
Then $\forall$ must play, and so on.

Remarks. We observe that the opponent $\forall$ wins if, and only if, the play is infinite. The play ends after a finite time if, and only if, $\mathcal{V}$ becomes empty (and then, the player $\exists$ wins). Just before, the player $\exists$ has chosen an atomic formula which is in $\mathcal{U} \cap \mathcal{A}$.
The intuitive meaning of the rule of this game is as follows : at each moment, the defender $\exists$ pretends that one of the formulas of $\mathcal{U}$ is false and that every formula of $\mathcal{V}$ is true. On the other hand, the opponent $\forall$ pretends that every formula of $\mathcal{U}$ is true and that one of the formulas of $\mathcal{V}$ is false. Now, both agree on the fact that every formula of $\mathcal{A}$ is false.

In the examples below, we shall interpret a play of this game as a session of communication following a certain protocol. In this interpretation, the opponent $\forall$ is the *sender* and the defender $\exists$ is the *receiver*.

The disymmetry of the game is well expressed by a celebrated sentence of Jon Postel (known as “ Postel's law ” [jp]) : “ Be conservative in what you send, be liberal in what you receive ”.

Examples

1) $F \equiv P \to P$. We can describe an arbitrary play by the following table :

$\mathcal{U}$	$\mathcal{V}$	$\mathcal{A}$	
$(P \to P) \to \bot$	$P \to P$	$\bot$	$\forall$ has no choice
$(P \to P) \to \bot, P$	unchanged	$\bot, P$	$\exists$ chooses $(P \to P) \to \bot$
unchanged	unchanged	unchanged	$\forall$ has no choice
unchanged	unchanged	unchanged	$\exists$ chooses $(P \to P) \to \bot$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
unchanged	unchanged	unchanged	$\exists$ chooses P
unchanged	$\emptyset$	unchanged	$\exists$ wins

The different possible plays depend only on the number n of times when $\exists$ chooses the formula $(P \to P) \to \bot$. If n is infinite, $\exists$ loses.

2) $F \equiv P \to Q$

$\mathcal{U}$	$\mathcal{V}$	$\mathcal{A}$	
$(P \to Q) \to \bot$	$P \to Q$	$\bot$	$\forall$ has no choice
$(P \to Q) \to \bot, P$	unchanged	$\bot, Q$	$\exists$ cannot choose P
unchanged	unchanged	unchanged	$\forall$ has no choice
unchanged	unchanged	unchanged	$\exists$ cannot choose$P$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

There is only one possible play and $\forall$ wins, since this play is infinite.

3) The reader is invited to study by himself the following two examples :
$((Q \to Q) \to P) \to P$; $((P \to Q) \to P) \to P$ (Peirce's law).

4) $F \equiv \forall x\, Px \to \forall x\, Px$. The normal form of F est $G \equiv \forall y(\forall x\, Px \to Py)$.

$\mathcal{U}$	$\mathcal{V}$	$\mathcal{A}$	
$\neg G$	G	$\perp$	$\forall$ chooses b_0
$\neg G, \forall x\, Px$	unchanged	$\perp, Pb_0$	$\exists$ chooses $\neg G$
unchanged	unchanged	unchanged	$\forall$ chooses b_1
unchanged	unchanged	$\perp, Pb_0, Pb_1$	$\exists$ chooses $\neg F$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
unchanged	unchanged	$\perp, Pb_0, \ldots, Pb_n$	$\exists$ chooses $\forall x\, Px$ and b_i
unchanged	$\emptyset$	unchanged	$\exists$ wins

Like in example 1, the play only depends on the moment when the player $\exists$ chooses the formula $\forall x\, Px$ and one of the b_i's already chosen by the opponent $\forall$.

We can give the following interpretation, in terms of network : the player $\forall$ sends the data packet P with the headers b_0, then b_1, ... The acknowledgement by the receiver $\exists$ only happens at the n-th step, and it is the packet Pb_i that is acknowledged. Then the play, that is to say the session, stops immediately. From the network point of view, this means that the acknowledgement cannot be lost ; in other words, that the channel from the receiver $\exists$ to the sender $\forall$ is reliable.
In the following section, we treat a particularly important example : the acknowledgement of a packet in a channel which is not reliable.

The formula $\exists x(Px \to \forall y\, Py)$

Let us call F the normal form of this formula, i.e. $\forall x(\forall y(Px \to Py) \to \perp) \to \perp$. For the sake of clarity, let us put $G[x] \equiv \forall y(Px \to Py)$; thus, we have $F \equiv \neg\forall x \neg\, G[x]$.

The tables I and II below represent what happens during a play, in the (very particular) case when $\exists$ plays in such a way as to win as quickly as possible. There are two possibilities, following what the opponent $\forall$ plays at line 3 :

Table I

	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{A}$	
1	$\neg F$	F	$\perp$	$\forall$ chooses F
2	$\neg F, \forall x\, \neg G[x]$	unchanged	unchanged	$\exists$ chooses $\forall x\, \neg G[x]$ and a
3	unchanged	$G[a] \equiv \forall y(Pa \to Py)$	unchanged	$\forall$ chooses b, with $b \neq a$
4	$\neg F, \forall x\, \neg G[x], Pa$	unchanged	$\perp, Pb$	$\exists$ chooses $\forall x \neg G[x]$ and b
5	unchanged	$G[b] \equiv \forall y(Pb \to Py)$	unchanged	$\forall$ chooses c
6	$\neg F, \forall x\, \neg G[x], Pa, Pb$	unchanged	$\perp, Pb, Pc$	$\exists$ chooses Pb
7	unchanged	$\emptyset$	unchanged	$\exists$ wins

Table II

	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{A}$	
1	$\neg F$	F	$\perp$	$\forall$ chooses F
2	$\neg F, \forall x\, \neg G[x]$	unchanged	unchanged	$\exists$ chooses $\forall x\, \neg G[x]$ and a
3	unchanged	$G[a] \equiv \forall y(Pa \to Py)$	unchanged	$\forall$ chooses a
4	$\neg F, \forall x\, \neg G[x], Pa$	unchanged	$\perp, Pa$	$\exists$ chooses Pa
5	unchanged	$\emptyset$	unchanged	$\exists$ wins

But this is only a particular case. The game we are considering presents, in fact, a great variety of possible plays. We shall see that these various plays correspond exactly to the various possibilities which may happen during the acknowledgement of a packet. The play which is described in table I represents the case when the communication occurred in the best possible way. We can interpret it as follows : the receiver $\exists$ begins the session by sending the header a (line 3) ; then the sender $\forall$ sends the packet Pb (line 4) ; $\exists$ receives the packet Pb and sends the acknowledgement (line 5) ; then $\forall$ correctly receives this acknowledgement and sends a signal Pc to terminate the session (line 6).

Several variants are possible :
i) The player $\exists$ can, at each moment, choose the formula $\neg F$. This corresponds to a re-initialisation of the session.
ii) She can also choose the formula $\forall x \neg\, G[x]$ with an arbitrary header a', which corresponds to no acknowledgement. Then, the opponent $\forall$ must send the packet again. This situation corresponds to the loss of the acknowledgement.
iii) In this case, the sender $\forall$ has the possibility of sending Pa' again, which gives to the receiver $\exists$ the possibility of finishing the session immediately by choosing precisely the formula Pa' (since it is now both in $\mathcal{U}$ and $\mathcal{A}$). This corresponds to the case when the sender asks to finish the session. This may happen at the very beginning : it is the case in the play which is described in table II (line 3 : $\forall$ chooses a) ; this corresponds to a refusal of opening the session ; then $\exists$ can only close the session, by choosing Pa (again, it is what happens in table II) or to re-initialise it (by choosing $\neg F$ or $\forall x \neg\, G[x]$).
iv) The player $\exists$ can terminate the play by choosing the formula Pb, where b is any of the headers sent by $\forall$. This corresponds to a successfull communication session, perhaps after some loss of acknowledgements.

Any session is a combination of an arbitrary number of such variants.

Sending several packets

We consider now the case of the acknowledgement of a fixed number n of packets, n being a previously given integer ; the order of the packets must be preserved. The associated formula F_n is defined by recurrence :
$F_1 \equiv \exists x \forall y (P_1 x \to P_1 y)$; $F_{n+1} \equiv \exists x \forall y ((F_n \to P_{n+1} x) \to P_{n+1} y)$; F_n is in normal form.
For the sake of simplicity, we consider only the case $n = 2$. We have the formula :
$F' \equiv \exists x \forall y ((F \to Px) \to Py)$ with $F \equiv \exists x \forall y (Qx \to Qy)$.
We put $G[x] \equiv \forall y ((F \to Px) \to Py)$, $H[x] \equiv \forall y (Qx \to Qy)$;
thus, we have $F' \equiv \forall x \neg G[x] \to \bot$ and $F \equiv \forall x \neg H[x] \to \bot$.
The table below describes once more what happens during a play where $\exists$ finishes in the quickest possible way. For the sake of clarity, in the columns $\mathcal{U}$ and $\mathcal{A}$, we shall put, at each line, only the *new* formulas.

	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{A}$	
1	$\neg F'$	F'	$\perp$	$\forall$ has no choice
2	$\forall x\,\neg G[x]$	unchanged	unchanged	$\exists$ chooses $\forall x\,\neg G[x]$ and a
3	unchanged	$G[a] \equiv \forall y((F \to Pa) \to Py)$	unchanged	$\forall$ chooses b, with $b \neq a$
4	$F \to Pa$	unchanged	Pb	$\exists$ chooses $\forall x \neg G[x]$ and b
5	unchanged	$G[b] \equiv \forall y((F \to Pb) \to Py)$	unchanged	$\forall$ chooses c
6	$F \to Pb$	unchanged	Pc	$\exists$ chooses Pb
7	unchanged	F	unchanged	$\forall$ has no choice
		.... *acknowledgement of the first packet*		
8	$\forall x\,\neg H[x]$	unchanged	unchanged	$\exists$ chooses $\forall x\,\neg H[x]$ and d
9	unchanged	$H[d] \equiv \forall y(Qd \to Qy)$	unchanged	$\forall$ chooses e, with $e \neq d$
10	Qd	unchanged	Qe	$\exists$ chooses $\forall x \neg H[x]$ and e
11	unchanged	$H[e] \equiv \forall y(Qe \to Qy)$	unchanged	$\forall$ chooses f
12	Qe	unchanged	Qf	$\exists$ chooses Qe
13	unchanged	$\emptyset$	unchanged	$\exists$ wins
		.... *acknowledgement of the second packet*		

In this particular case, we essentially get twice the table I of the previous example. Of course, we may get all the variants already described. But new variants may appear : indeed, after the acknowledgement of the first packet (lines 8, 10 and 12), the player $\exists$ can, for instance, come back to line 4, that is to say ask again for the first packet. Thus the receiver may lose a packet, even after having correctly acknowledged it. It is interesting to notice that she has not to acknowledge it again.

Strategies and valid formulas

Let us consider the game associated with a normal closed formula F. A *strategy* for $\exists$ in this game is, by definition, a function $\mathcal{S}$, which takes as an argument a finite sequence of triples $(\mathcal{U}_i, \mathcal{V}_i, \mathcal{A}_i)_{0 \leq i \leq n}$ ($\mathcal{U}_i, \mathcal{V}_i$ are finite sets of normal closed formulas and $\mathcal{A}_i$ is a finite set of atomic closed formulas) and gives as a result an ordered pair $(\Psi, \vec{b})$ with $\Psi \in \mathcal{U}_n$, $\Psi \equiv \forall \vec{y}(\Phi_1[\vec{y}], \ldots, \Phi_k[\vec{y}] \to B[\vec{y}])$, $\vec{b}$ has the same length as $\vec{y}$ and $B[\vec{b}] \in \mathcal{A}_n$.
Intuitively, a strategy $\mathcal{S}$ for $\exists$loise is a general method which, each time she must play, chooses for her a possible play, given all the moves already played.
The strategy $\mathcal{S}$ is called a *winning strategy* if $\exists$ wins every play following this strategy, whatever be the choices of $\forall$.
We could define in the same way the winning strategies for $\forall$.

A normal closed formula F is called *valid* if there exists a winning strategy for $\exists$, in the game associated with F. Valid formulas are exactly those which correspond to network protocols.

Games associated with a conjunction or a disjunction.
Given two formulas F, G, the game which is associated with the formula $F \wedge G$, i.e. $(F, G \to \perp) \to \perp$ consists essentially in the following (this is easily checked) :
The opponent $\forall$ chooses one of these two formulas and the game goes on, following the chosen formula ; however, the player $\exists$ can, at every moment, decide to start again the

play from the beginning.
With the formula $F \vee G$, it is the player $\exists$ who chooses the formula.

Composition of protocols

Let us consider two valid formulas F and G, which correspond respectively to the "protocols" (i.e. games) $\mathcal{P}_F$ and $\mathcal{P}_G$; we propose now to build a valid formula H such that the associated protocol $\mathcal{P}_H$ is : $\mathcal{P}_F$ then $\mathcal{P}_G$.

Let $A = P(x_1, \dots, x_n)$ (or $\perp$) be an atomic formula and F a normal formula. An "occurrence" of A in F is simply one of the places, in F, where A appears.
Each occurrence of an atomic formula A in F appears at the end of a subformula of F, of the form $\forall\vec{x}(\Psi_1, \dots, \Psi_k \to A)$; k will be called the *number of hypothesis* of this occurrence of A.

Each occurrence of an atomic formula A in F is either *positive* or *negative*. This property is defined in the following way, by recurrence on the length of F :

- If F is atomic, then $F \equiv A$ and the occurrence of A in F is positive.
- If $F \equiv G \to H$, the occurrence of A in F that we consider, is either in G, or in H. If it is in H, its sign is the same in F as in H. If it is in G, it has opposite signs in F and in G.
- If $F \equiv \forall x\, G$, the occurrence of A we consider, has the same sign in F and in G.

An atomic occurrence A in F, which is *negative and without hypothesis*, will be called a *final atomic occurrence*. Indeed, it corresponds to the end of a play.

Now, we can build the formula H we are looking for : *it is obtained by replacing, in F, each final atomic occurrence A with $G \to A$.*

Remark. It is easy to show that, if F and G are valid, then the formula H defined in this way is also valid (see the appendix).

Example. Take the formula $F \equiv \forall x[\forall y(Px \to Py) \to \perp] \to \perp$ which corresponds to the sending and the acknowledgement of a packet.
Then, we get : $H \equiv \forall x[\forall y((G \to Px) \to Py) \to \perp] \to \perp$.
Indeed, there are, in F, two atomic negative occurrences, which are Px and the first occurrence of $\perp$. The only atomic occurrence without hypothesis is Px.
In particular, if we take $G \equiv \forall x[\forall y(Qx \to Qy) \to \perp] \to \perp$, we get the protocol which corresponds to the sending of two packets (see above).

Formulas and protocols using integer variables

We now consider formulas written with a new type of variables : the " integer type " ; to denote variables of this type, we shall use the letters i, j, k, l, m, n. Thus, there are now two types of variables : the type " integer " and the type already defined, which we shall call the type " acknowledgement " ; for the variables of this type, we use as before the letters x, y, z.
Moreover, we have function symbols *on the integer type* (they denote functions from integers to integers), in particular the constant 0 and the successor s (which represents the function $n \mapsto n + 1$). Each function symbol f has an arity $k \in \mathbb{N}$ and represents a

well determined function from $\mathbb{N}^k$ to $\mathbb{N}$ which is also denoted by f. We define the *terms* of integer type by the following rules :

• an integer variable or a function symbol of arity 0 (integer constant) is a term of integer type.
• if f is a function symbol of arity k and $t_1, \ldots, t_k$ are terms of integer type, then $f(t_1, \ldots, t_k)$ is a term of integer type.

We note that a term of integer type without variable (closed term) represents an integer.

Predicate symbols are also typed. For example, Pnx or $P(n, x)$ (the first argument of P is of integer type, the second is of type acknowledgement).

Definition of formulas.
• Atomic formulas : $Pt_1 \ldots t_k$; t_i is a constant or a variable of type acknowledgement if the i-th place of P is of this type; a term of type integer, if the i-th place of P is of integer type.
• If F, G are formulas, $F \to G$ is also a formula.
• If F is a formula, $\forall x\, F$ and $\forall n\, F$ are also formulas.
• If F is a formula and t, u are terms of integer type, then $t = u \to F$ is also a formula.

Be careful, the expression $t = u$ alone *is not a formula.*

Normal forms.
They are defined as follows :
• An atomic formula is normal.
• If F is normal, $\forall x\, F$ and $\forall n\, F$ are normal.
• If A is atomic and $\Phi_1, \ldots, \Phi_k$ are normal formulas or expressions of the form $t = u$, then $\Phi_1, \ldots, \Phi_k \to A$ is a normal formula.

We put the formulas under normal form exactly as in the previous case.

Game associated with a closed formula under normal form.
We indicate here only the additions to the game rule which has been already given :

i) when one of the players has chosen a formula $\forall \vec{\xi}(\Phi_1, \ldots, \Phi_n \to A)$, ($\vec{\xi} = (\xi_1, \ldots, \xi_n)$ where ξ_i is a variable of type integer or acknowledgement) :

- first, he or she chooses some values $\vec{a}$ for $\vec{\xi}$.
- then, he or she computes the (boolean) expressions Φ_j of the form $t_j = u_j$.
- if all of them are true, we get rid of them and the play goes on as before with the (simpler) formula obtained in this way.
- if any of them is false :
  if $\exists$ is playing, she must choose other values $\vec{a}$ or another formula (which is always possible, as we already saw).
  if the opponent $\forall$ is playing, then he has lost and the play stops.

ii) as in the previous game, when the player $\exists$ chooses, in the set $\mathcal{U}$, a formula :
$\Psi \equiv \forall \vec{y}(\Phi_1[\vec{y}], \ldots, \Phi_n[\vec{y}] \to B[\vec{y}])$ and a sequence $\vec{b}$ of constants, of the same length as $\vec{y}$, she have to check that $B[\vec{b}] \in \mathcal{A}$, i.e. to check that two atomic closed formulas are identical. These formulas may contain closed terms of type integer and *these terms must be computed before comparing them.*

ω-valid formulas.
A normal closed formula F will be called *ω-valid* if there exists a winning strategy for $\exists$,

in the game associated with F. The ω-valid formulas are exactly those which correspond to network protocols (as before, with valid formulas).

Example.
First, $\forall$ send an integer n; after that, acknowledgement of n packets. The formula is :
$F \equiv \forall j\{\forall i[j = si \to \exists x\forall y((Ui \to Pix) \to Piy)] \to Uj\} \to \forall n\, Un.$
U is a predicate symbol with one argument of integer type; P has two arguments, the first is of integer type, the second of type acknowledgement.

Put $G \equiv \forall j\{\forall i[j = si \to \exists x\forall y((Ui \to Pix) \to Piy)] \to Uj\}$ and
$H[i, x] \equiv \forall y((Ui \to Pix) \to Piy)$.
We put the formula F in normal form, so it is written as $F \equiv \forall n(G \to Un)$.

The following table shows the particular session in which every packet is acknowledged in the quickest possible way.

	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{A}$	
1	$\neg F$	F	$\perp$	$\forall$ chooses n_0
2	G	unchanged	Un_0	$\exists$ chooses G and n_0
3	unchanged	$\forall i(n_0{=}si{\to}\exists xH[i,x])$	unchanged	$\forall$ can only choose $n_0 - 1$
4	$\forall x\neg H[n_0{-}1, x]$	unchanged	unchanged	$\exists$ chooses a_0
5	unchanged	$H[n_0{-}1, a_0]$	unchanged	$\forall$ chooses $b_0 \neq a_0$
6	$U(n_0{-}1){\to}P(n_0{-}1, a_0)$	unchanged	$P(n_0{-}1, b_0)$	$\exists$ chooses $\forall x\neg H[n_0{-}1, x]$ and b_0
7	unchanged	$H[n_0{-}1, b_0]$	unchanged	$\forall$ chooses $b_1 \neq a_0, b_0$
8	$U(n_0{-}1){\to}P(n_0{-}1, b_0)$	unchanged	$P(n_0{-}1, b_1)$	$\exists$ chooses $U(n_0{-}1){\to}P(n_0{-}1, b_0)$
9	unchanged	$U(n_0{-}1)$	unchanged	$\forall$ can only choose $U(n_0{-}1)$
....	....	*acknowledgement of packet* $n_0{-}1$	....	....
10	unchanged	unchanged	$U(n_0{-}1)$	$\exists$ chooses G and $n_0{-}1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
....	....	*acknowledgement of packet* 0	....	....
	unchanged	unchanged	$U0$	$\exists$ chooses G and 0
	unchanged	$\forall i(0{=}si{\to}\exists xH[i,x])$	unchanged	$\forall$ has lost

In order to avoid a supplementary complication, we did not ask that the integer n (the number of packets to transmit) which is sent by $\forall$, be acknowledged. But if we want this integer to be acknowledged, we must add a field " acknowledgement " to the predicate symbol U, which therefore becomes binary. In this case, we write the following formula :
$F \equiv \forall n\exists x'\forall y'((\neg\, G[x'] \to Unx') \to Uny')$ with
$G[x'] \equiv \forall j\{\forall i[j = si \to \exists x\forall y((Uix' \to Pix) \to Piy)] \to Ujx'\}$

The following table shows the beginning of a communication session and the acknowledgement of the integer n.
We put $H[n, x'] \equiv \forall y'((\neg\, G[x'] \to Unx') \to Uny')$.

	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{A}$	
1	$\neg F$	F	$\perp$	$\forall$ chooses n_0
2	$\forall x' \neg H[n_0, x']$	unchanged	unchanged	$\exists$ chooses $\forall x' \neg H[n_0, x']$ and x'_0
3	unchanged	$H[n_0, x'_0]$	unchanged	$\forall$ chooses y'_0
4	$\neg G[x'_0] \to Un_0x'_0$	unchanged	$Un_0y'_0$	$\exists$ chooses $\forall x' \neg H[n_0, x']$ and y'_0
5	unchanged	$H[n_0, y'_0]$	unchanged	$\forall$ chooses y'_1
6	$\neg G[y'_0] \to Un_0y'_0$	unchanged	$Un_0y'_1$	$\exists$ chooses $\neg G[y'_0] \to Un_0y'_0$
7	unchanged	$\neg\, G[y'_0]$	unchanged	$\forall$ has no choice
8	$G[y'_0]$	unchanged	unchanged	$\exists$ chooses $G[y'_0]$ and n_0
	$\vdots$	$\vdots$	$\vdots$	$\vdots$

From now on, the play continues as in the previous example (with the formula $G[y'_0]$ instead of the formula G).

Remark. A somewhat simpler formula for the same protocol can be written as :
$\forall n \exists x' \forall y' (G[x'] \to Uny')$ with, as before :
$G[x'] \equiv \forall j \{\forall i [j = si \to \exists x \forall y ((Uix' \to Pix) \to Piy)] \to Ujx'\}$.
The reader will check this easily.

Appendix

Valid formulas.
The usual definition of a valid formula of the predicate calculus uses the notion of *model.* The interested reader will find it, for example in [rcdl] or [jrs]. A formula is called valid if it is satisfied in every model. A fundamental theorem of logic, known as the *completeness theorem*, says that a formula is valid if, and only if, it is provable by means of the deduction rules of “ pure logic ”.
This notion of validity is equivalent to that introduced in the present paper, which is given in terms of strategies (see a proof in [jlk]).
It is often much easier to check the validity of a formula with the help of models. For example, it is immediately seen in this way that the formula $F \equiv \exists x (Px \to \forall y\, Py)$ is valid : indeed, either the model we consider satisfies $\forall y\, Py$ and therefore also F, either it satisfies $\exists x \neg\, Px$ and thus again F.

Let us consider two valid formulas F and G, and let H be the formula defined above, such that the protocol $\mathcal{P}_H$ associated with H is obtained by composing the protocols associated with F and G. Then, it is easy to show that H is valid. Indeed, we obtained the formula H by replacing, in F, some subformulas A with $G \to A$. But A and $G \to A$ are obviously equivalent, since G is valid. Thus, we get finally a formula H wich is equivalent to F, and therefore a valid formula.

For the formulas with two types (integer and acknowledgement), the situation is a bit more complex. The ω-valid formulas are the formulas which are satisfied in every *ω-model*, that is to say the models in which the integer type has its standard interpretation. Again, in this case, it is often much easier to use this definition in order to check that a formula is ω-valid.
For instance, it is not difficult to show that the formula (that we have already used before)

$F \equiv G \to \forall n\, Un$, with $G \equiv \forall j\{\forall i[j = si \to \exists x \forall y((Ui \to Pix) \to Piy)] \to Uj\}$ is ω-valid.
Indeed, we assume G and we show Un by recurrence on the integer n.
Proof of $U0$. We put $j = 0$ in G; since $0 = si$ is false, we get $U0$.
Proof of $Un \to U(n+1)$. We put $j = n+1$ in G. Then, it suffices to show :
$\forall i[n+1 = si \to \exists x \forall y((Ui \to Pix) \to Piy)]$ with Un as an hypothesis. Since $n+1 = si$ is equivalent to $i = n$, we now need to show $\exists x \forall y((Un \to Pnx) \to Pny)$, that is to say $\exists x \forall y(Pnx \to Pny)$ (because we assume Un). But this last formula is already shown.

With some simple changes, the same proof works for the formulas :
$\forall n \exists x' \forall y'((\neg G[x'] \to Unx') \to Uny')$ and $\forall n \exists x' \forall y'(G[x'] \to Uny')$
with $G[x'] \equiv \forall j\{\forall i[j = si \to \exists x \forall y((Uix' \to Pix) \to Piy)] \to Ujx'\}$.
Indeed, you only need to show the first one, since the second is trivially weaker.

Determination of games.
A game is called *determined* if one of the players has a winning strategy. It is always the case for the games considered in this paper (Gale-Stewart theorem).
Sketch of proof. Suppose that $\exists$ has no winning strategy. Then, the following strategy is winning for the opponent $\forall$: to play, at each step, in such a way that the player $\exists$ has no winning strategy from this step. Then the play lasts infinitely long, and $\forall$ wins.

References

[rcdl] René Cori & Daniel Lascar. *Mathematical logic.* Oxford University Press (2001).
[jlk] Jean-Louis Krivine. *Realizability : a machine for analysis and set theory.* Geocal06, Marseille (2006).
`http://cel.archives-ouvertes.fr/cel-00154509`. More recent version at :
`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).