Toward protological foundations for logic

Jean-Baptiste Joinet

Université Paris 1 (Panthéon-Sorbonne), UFR de Philosophie

Cirphles, École Normale Supérieure, Dép. de Philosophie
(USR 3308, CNRS & ENS Paris)

21 july 2011

Introduction

- Contemporary results in Proof theory and Computing theory : new lights on foundational questions in Logic.
- Logic as a foundation for versus foundations for logic itself
- General idea : foundations of logic upon a computational protologic
- Foundational ideas and computationalist views are not easy to defend in the contemporary philosophical debate.
- So, to begin with : spend time about the notion of computation itself.

Introduction

- Contemporary results in Proof theory and Computing theory : new lights on foundational questions in Logic.
- Logic as a foundation for versus foundations for logic itself
- General idea : foundations of logic upon a computational protologic
- Foundational ideas and computationalist views are not easy to defend in the contemporary philosophical debate.
- So, to begin with : spend time about the notion of computation itself.

Introduction

- Contemporary results in Proof theory and Computing theory : new lights on foundational questions in Logic.
- Logic as a foundation for versus foundations for logic itself
- General idea : foundations of logic upon a computational protologic
- Foundational ideas and computationalist views are not easy to defend in the contemporary philosophical debate.
- So, to begin with : spend time about the notion of computation itself.

Introduction

- Contemporary results in Proof theory and Computing theory : new lights on foundational questions in Logic.
- Logic as a foundation for versus foundations for logic itself
- General idea : foundations of logic upon a computational protologic
- Foundational ideas and computationalist views are not easy to defend in the contemporary philosophical debate.
- So, to begin with : spend time about the notion of computation itself.

Introduction

- Contemporary results in Proof theory and Computing theory : new lights on foundational questions in Logic.
- Logic as a foundation for versus foundations for logic itself
- General idea : foundations of logic upon a computational protologic
- Foundational ideas and computationalist views are not easy to defend in the contemporary philosophical debate.
- So, to begin with : spend time about the notion of computation itself.

Plan

Computing and programming languages

Notions of program and of programming language

Omnipresent in the present times.

But little philosophical attention and low level of esteem.

Computing and programming languages

Notions of program and of programming language

Omnipresent in the present times.

But little philosophical attention and low level of esteem.

Computing and programming languages

Notions of program and of programming language

Omnipresent in the present times.

But little philosophical attention and low level of esteem.

Computing and programming languages

Philosophical attention to the notions of "program" and "programming language" suffer from two prejudices:

Prejudice 1: inferior notions

Only technical, applied notions.

Belong to engineers vocabulary.

Prejudice 2: connivance with reductionism

Connivance with "Computational mecanicism" is imputed.

Computational reductionism :

- reduces thought, or even world, to discrete, digital computing.
- results in limiting human freedom or world indeterminacy.

Computing and programming languages

Philosophical attention to the notions of "program" and "programming language" suffer from two prejudices:

Prejudice 1: inferior notions

Only technical, applied notions.

Belong to engineers vocabulary.

Prejudice 2: connivance with reductionism

Connivance with "Computational mecanicism" is imputed.

Computational reductionism :

- reduces thought, or even world, to discrete, digital computing.
- results in limiting human freedom or world indeterminacy.

Computing and programming languages

Philosophical attention to the notions of "program" and "programming language" suffer from two prejudices:

Prejudice 1: inferior notions

Only technical, applied notions.

Belong to engineers vocabulary.

Prejudice 2: connivance with reductionism

Connivance with "Computational mecanicism" is imputed.

Computational reductionism :

- reduces thought, or even world, to discrete, digital computing.
- results in limiting human freedom or world indeterminacy.

Computing and programming languages

Philosophical attention to the notions of "program" and "programming language" suffer from two prejudices:

Prejudice 1: inferior notions

Only technical, applied notions.

Belong to engineers vocabulary.

Prejudice 2: connivance with reductionism

Connivance with "Computational mecanicism" is imputed.

Computational reductionism :

- reduces thought, or even world, to discrete, digital computing.
- results in limiting human freedom or world indeterminacy.

Computing and programming languages

Philosophical attention to the notions of "program" and "programming language" suffer from two prejudices:

Prejudice 1: inferior notions

Only technical, applied notions.

Belong to engineers vocabulary.

Prejudice 2: connivance with reductionism

Connivance with "Computational mecanicism" is imputed.

Computational reductionism :

- reduces thought, or even world, to discrete, digital computing.
- results in limiting human freedom or world indeterminacy.

Overcoming prejudice 1 : philosophically minor notions ?

Programming language are axiomatic systems

A programming language = a set of axioms for elementary acts.

To combine these axioms (writing down a program) = to describe:

- a possible complete generic action,
- ready to be actualized when confronted to the next environment looming up.

Axiomatizing interaction

In general, such an environment is itself composed by acting processes so that, rather than an action, the computational process must be thought as confrontation of actions: interaction.

Programming languages = systems of axioms for elementary kinds of interaction.

Overcoming prejudice 1 : philosophically minor notions ?

Programming language are axiomatic systems

A programming language = a set of axioms for elementary acts.

To combine these axioms (writing down a program) = to describe:

- a possible complete generic action,
- ready to be actualized when confronted to the next environment looming up.

Axiomatizing interaction

In general, such an environment is itself composed by acting processes so that, rather than an action, the computational process must be thought as confrontation of actions: interaction.

Programming languages = systems of axioms for elementary kinds of interaction.

Overcoming prejudice 1 : philosophically minor notions ?

Programming language are axiomatic systems

A programming language = a set of axioms for elementary acts.

To combine these axioms (writing down a program) = to describe:

- a possible complete generic action,
- ready to be actualized when confronted to the next environment looming up.

Axiomatizing interaction

In general, such an environment is itself composed by acting processes so that, rather than an action, the computational process must be thought as confrontation of actions: interaction.

Programming languages = systems of axioms for elementary kinds of interaction.

Overcoming prejudice 1 : philosophically minor notions ?

Programming language are axiomatic systems

A programming language = a set of axioms for elementary acts.

To combine these axioms (writing down a program) = to describe:

- a possible complete generic action,
- ready to be actualized when confronted to the next environment looming up.

Axiomatizing interaction

In general, such an environment is itself composed by acting processes so that, rather than an action, the computational process must be thought as confrontation of actions: interaction.

Programming languages = systems of axioms for elementary kinds of interaction.

Overcoming prejudice 1 : philosophically minor notions ?

Programming language are axiomatic systems

A programming language = a set of axioms for elementary acts.

To combine these axioms (writing down a program) = to describe:

- a possible complete generic action,
- ready to be actualized when confronted to the next environment looming up.

Axiomatizing interaction

In general, such an environment is itself composed by acting processes so that, rather than an action, the computational process must be thought as confrontation of actions: interaction.

Programming languages = systems of axioms for elementary kinds of interaction.

Overcoming prejudice 1 : philosophically minor notions ?

Programming language are axiomatic systems

A programming language = a set of axioms for elementary acts.

To combine these axioms (writing down a program) = to describe:

- a possible complete generic action,
- ready to be actualized when confronted to the next environment looming up.

Axiomatizing interaction

In general, such an environment is itself composed by acting processes so that, rather than an action, the computational process must be thought as confrontation of actions: interaction.

Programming languages = systems of axioms for elementary kinds of interaction.

Overcoming prejudice 1 : philosophically minor notions ?

Programming language are axiomatic systems

A programming language = a set of axioms for elementary acts.

To combine these axioms (writing down a program) = to describe:

- a possible complete generic action,
- ready to be actualized when confronted to the next environment looming up.

Axiomatizing interaction

In general, such an environment is itself composed by acting processes so that, rather than an action, the computational process must be thought as confrontation of actions: interaction.

Programming languages = systems of axioms for elementary kinds of interaction.

Overcoming prejudice 1 : philosophically minor notions ?

Programming language are axiomatic systems

A programming language = a set of axioms for elementary acts.

To combine these axioms (writing down a program) = to describe:

- a possible complete generic action,
- ready to be actualized when confronted to the next environment looming up.

Axiomatizing interaction

In general, such an environment is itself composed by acting processes so that, rather than an action, the computational process must be thought as confrontation of actions: interaction.

Programming languages = systems of axioms for elementary kinds of interaction.

Overcoming prejudice 1: philosophically minor notions ?

Computation from an axiomatic point of view :

- an **abstract** notion: computing considered per se :
  - independently of any computer (human or not),
  - so in particular independently of any computational intention or aim.
- a **general** notion: synonymous with interactional process

Overcoming prejudice 1: philosophically minor notions ?

Computation from an axiomatic point of view :

- an **abstract** notion: computing considered per se :
  - independently of any computer (human or not),
  - so in particular independently of any computational intention or aim.
- a **general** notion: synonymous with interactional process

Overcoming prejudice 1: philosophically minor notions ?

Computation from an axiomatic point of view :

- an **abstract** notion: computing considered per se :
  - independently of any computer (human or not),
  - so in particular independently of any computational intention or aim.
- a **general** notion: synonymous with interactional process

Overcoming prejudice 1: philosophically minor notions ?

Computation from an axiomatic point of view :

- an **abstract** notion: computing considered per se :
  - independently of any computer (human or not),
  - so in particular independently of any computational intention or aim.
- a **general** notion: synonymous with interactional process

Overcoming prejudice 2: computation against reductionism

Program

- An **open** notion: axioms have to be chosen
- A **plural** notion: any choice for axioms determines a given kind of dynamic, with its own complexity, its own properties

Computation

- A **plastic** notion :
  - interactions could be performed in parallel or not
  - interactions could be performed following various strategies
  - an elementary interaction itself could be non deterministic
- A **wild** phenomenon :
  - possibly infinite
  - possibly non deterministic
  - possibly inconsistent

Overcoming prejudice 2: computation against reductionism

Program

- An **open** notion: axioms have to be chosen
- A **plural** notion: any choice for axioms determines a given kind of dynamic, with its own complexity, its own properties

Computation

- A **plastic** notion :
  - interactions could be performed in parallel or not
  - interactions could be performed following various strategies
  - an elementary interaction itself could be non deterministic
- A **wild** phenomenon :
  - possibly infinite
  - possibly non deterministic
  - possibly inconsistent

Overcoming prejudice 2: computation against reductionism

Program

- An **open** notion: axioms have to be chosen
- A **plural** notion: any choice for axioms determines a given kind of dynamic, with its own complexity, its own properties

Computation

- A **plastic** notion :
  - interactions could be performed in parallel or not
  - interactions could be performed following various strategies
  - an elementary interaction itself could be non deterministic
- A **wild** phenomenon :
  - possibly infinite
  - possibly non deterministic
  - possibly inconsistent

Overcoming prejudice 2: computation against reductionism

Program

- An **open** notion: axioms have to be chosen
- A **plural** notion: any choice for axioms determines a given kind of dynamic, with its own complexity, its own properties

Computation

- A **plastic** notion :
  - interactions could be performed in parallel or not
  - interactions could be performed following various strategies
  - an elementary interaction itself could be non deterministic
- A **wild** phenomenon :
  - possibly infinite
  - possibly non deterministic
  - possibly inconsistent

Overcoming prejudice 2: computation against reductionism

Program

- An **open** notion: axioms have to be chosen
- A **plural** notion: any choice for axioms determines a given kind of dynamic, with its own complexity, its own properties

Computation

- A **plastic** notion :
  - interactions could be performed in parallel or not
  - interactions could be performed following various strategies
  - an elementary interaction itself could be non deterministic
- A **wild** phenomenon :
  - possibly infinite
  - possibly non deterministic
  - possibly inconsistent

Overcoming prejudice 2: computation against reductionism

Program

- An **open** notion: axioms have to be chosen
- A **plural** notion: any choice for axioms determines a given kind of dynamic, with its own complexity, its own properties

Computation

- A **plastic** notion :
  - interactions could be performed in parallel or not
  - interactions could be performed following various strategies
  - an elementary interaction itself could be non deterministic
- A **wild** phenomenon :
  - possibly infinite
  - possibly non deterministic
  - possibly inconsistent

Overcoming prejudice 2: computation against reductionism

Program

- An **open** notion: axioms have to be chosen
- A **plural** notion: any choice for axioms determines a given kind of dynamic, with its own complexity, its own properties

Computation

- A **plastic** notion :
  - interactions could be performed in parallel or not
  - interactions could be performed following various strategies
  - an elementary interaction itself could be non deterministic
- A **wild** phenomenon :
  - possibly infinite
  - possibly non deterministic
  - possibly inconsistent

Overcoming prejudice 2: computation against reductionism

Program

- An **open** notion: axioms have to be chosen
- A **plural** notion: any choice for axioms determines a given kind of dynamic, with its own complexity, its own properties

Computation

- A **plastic** notion :
  - interactions could be performed in parallel or not
  - interactions could be performed following various strategies
  - an elementary interaction itself could be non deterministic
- A **wild** phenomenon :
  - possibly infinite
  - possibly non deterministic
  - possibly inconsistent

Overcoming prejudice 2: computation against reductionism

Program

- An **open** notion: axioms have to be chosen
- A **plural** notion: any choice for axioms determines a given kind of dynamic, with its own complexity, its own properties

Computation

- A **plastic** notion :
  - interactions could be performed in parallel or not
  - interactions could be performed following various strategies
  - an elementary interaction itself could be non deterministic
- A **wild** phenomenon :
  - possibly infinite
  - possibly non deterministic
  - possibly inconsistent

Overcoming prejudice 2: computation against reductionism

Program

- An **open** notion: axioms have to be chosen
- A **plural** notion: any choice for axioms determines a given kind of dynamic, with its own complexity, its own properties

Computation

- A **plastic** notion :
  - interactions could be performed in parallel or not
  - interactions could be performed following various strategies
  - an elementary interaction itself could be non deterministic
- A **wild** phenomenon :
  - possibly infinite
  - possibly non deterministic
  - possibly inconsistent

Overcoming prejudice 2: computation against reductionism

Program

- An **open** notion: axioms have to be chosen
- A **plural** notion: any choice for axioms determines a given kind of dynamic, with its own complexity, its own properties

Computation

- A **plastic** notion :
  - interactions could be performed in parallel or not
  - interactions could be performed following various strategies
  - an elementary interaction itself could be non deterministic
- A **wild** phenomenon :
  - possibly infinite
  - possibly non deterministic
  - possibly inconsistent

Overcoming prejudice 2: computation against reductionism

.

Computation, thought, world

Computational reductionism :

thought (or world) reduced to computation.

Which kind of links between:

- computation as axiomatized and thought?
- computation as axiomatized and world?

As usual with axioms, the answer depends on the axioms' status:

are "elementary interaction axioms"
descriptive or normative?

Overcoming prejudice 2: computation against reductionism

.

Computation, thought, world

Computational reductionism :

thought (or world) reduced to computation.

Which kind of links between:

- computation as axiomatized and thought?
- computation as axiomatized and world?

As usual with axioms, the answer depends on the axioms' status:

are "elementary interaction axioms"
descriptive or normative?

Overcoming prejudice 2: computation against reductionism

.

Computation, thought, world

Computational reductionism :

thought (or world) reduced to computation.

Which kind of links between:

- computation as axiomatized and thought?
- computation as axiomatized and world?

As usual with axioms, the answer depends on the axioms' status:

are "elementary interaction axioms"
descriptive or normative?

Overcoming prejudice 2: computation against reductionism

.

Computation, thought, world

Computational reductionism :

thought (or world) reduced to computation.

Which kind of links between:

- computation as axiomatized and thought?
- computation as axiomatized and world?

As usual with axioms, the answer depends on the axioms' status:

are "elementary interaction axioms"
descriptive or normative?

Overcoming prejudice 2: computation against reductionism

"Computation and world":
whenever axioms are descriptive

Computation simulating, modelizing informational interactions "in the world". Notably:

- **Linguistic** interaction (LOCI® research program)
- **Biological** informational interaction at the level of cells (bio-computing research programs)
- **Economical**, transactional interaction (see for instance of D. Porello and U Endriss papers).

No "reductionism" here, because there is no project to reduce anything to computing.

"Computation and world" whenever axioms are normative

- The question is : in which respect the world impose conditions for that normativity.
- At the more elementary level, informational interactions rest on **physics**: laws for information in space and time (transmission, duplicabiliy, localization, identity, non determinism, . . .).
- Questions typically addressed by :
  - Gandy's "physical version" of Turing-Church's thesis
  - Quantum computing
  - Girard's "Geometry of interaction" program (to some respects)
  - Feasability approaches

Overcoming prejudice 2: computation against reductionism

"Computation and thought": whenever axioms are descriptive

- To describe the thought processing : methodologically problematic.
- Who wants so, may imagine that those (however open, plural, plastic, wild) processes offer a model of the processes of thought. But:
- this is not a consequence of the definitions;
- this would not be as worst as the ordinary logical "intuitionistic" reduction (rather simplistic conception of rational thought processing as concatenation of judgments);
- anyway, constructing from computation simulation of semantical phenomena would then be a crucial experience.

Overcoming prejudice 2: computation against reductionism

"Computation and thought": whenever axioms are descriptive

- To describe the thought processing : methodologically problematic.
- Who wants so, may imagine that those (however open, plural, plastic, wild) processes offer a model of the processes of thought. But:
- this is not a consequence of the definitions;
- this would not be as worst as the ordinary logical "intuitionistic" reduction (rather simplistic conception of rational thought processing as concatenation of judgments);
- anyway, constructing from computation simulation of semantical phenomena would then be a crucial experience.

Overcoming prejudice 2: computation against reductionism

"Computation and thought": whenever axioms are descriptive

- To describe the thought processing : methodologically problematic.
- Who wants so, may imagine that those (however open, plural, plastic, wild) processes offer a model of the processes of thought. But:
- this is not a consequence of the definitions;
- this would not be as worst as the ordinary logical "intuitionistic" reduction (rather simplistic conception of rational thought processing as concatenation of judgments);
- anyway, constructing from computation simulation of semantical phenomena would then be a crucial experience.

Overcoming prejudice 2: computation against reductionism

"Computation and thought": whenever axioms are descriptive

- To describe the thought processing : methodologically problematic.
- Who wants so, may imagine that those (however open, plural, plastic, wild) processes offer a model of the processes of thought. But:
- this is not a consequence of the definitions;
- this would not be as worst as the ordinary logical "intuitionistic" reduction (rather simplistic conception of rational thought processing as concatenation of judgments);
- anyway, constructing from computation simulation of semantical phenomena would then be a crucial experience.

Overcoming prejudice 2: computation against reductionism

"Computation and thought": whenever axioms are normative

- The question is now : which "internal" conditions for interaction axioms normativity ?
- Could be reformulated: under which condition **semantic** can grows up from such axiomatic accounts (Girard's "transcendantal syntax" program).
- Notably: computational consistency (separation properties), data representation possibility, ... Semantics of programs (or semantics for actions : proto-semantics).

Plan

Actional semantic: general question

Interaction as the semantical source (proto-semantics)

The meaning of the text of a program does not first depend on the usual semantical question :

"what is that text refering to ?", "what does it denote ?",

but depends on a preliminary question :

"what does this program ?", "how does it act ?"

Programs = acting texts = texts whose meaning is paradigmatically actional

Semantically: acting is primitive.

Actional semantic: general question

Interaction as the semantical source (proto-semantics)

The meaning of the text of a program does not first depend on the usual semantical question :

"what is that text refering to ?", "what does it denote ?",

but depends on a preliminary question :

"what does this program ?", "how does it act ?"

Programs = acting texts = texts whose meaning is paradigmatically actional

Semantically: acting is primitive.

Actional semantic: general question

Interaction as the semantical source (proto-semantics)

The meaning of the text of a program does not first depend on the usual semantical question :

"what is that text refering to ?", "what does it denote ?",

but depends on a preliminary question :

"what does this program ?", "how does it act ?"

Programs = acting texts = texts whose meaning is paradigmatically actional

Semantically: acting is primitive.

Actional semantic: "Sense / Reference" distinction disturbed

Extension of the "definition domain" of Sense

- The bursting of the notion of program into semantics, pushes back the artifical limits (extensionality) of the "definition domain" of Sense.

  Pratically, it is only through one particular (among various) algorithms that, for instance, an arithmetical function is given to us.
- A given operation allows for multiple "modes of being given" (various algorithmes), alias for multiple "Senses".
- Semantically, the bursting of the notion of program/action thus corresponds to a desirable **extension of the domain of Sense**.

Actional semantic: "Sense / Reference" distinction disturbed

Complete reversion of the Sense/Reference relationship

- Fregean (realist) context: the reference of terms is primitive and independant of sense (a sense = only as a way to designate the reference).
- Actional semantics = radical reversion of the fregean relationship between Sense and Reference. Indeed:
- the computational evaluation = but **a transformation of Sense**. For instance : $2+2$ and 4 (same reference but not the same sense). The computing process converts the first sense (2+2) into the second one (4).
- The reference = the invariant of that transformation of sense.
- The Sense now comes first. This opens us toward a non trivial re-construction of referents as second.

Actional semantic: "Sense / Reference" distinction disturbed

Complete reversion of the Sense/Reference relationship

- Fregean (realist) context: the reference of terms is primitive and independant of sense (a sense = only as a way to designate the reference).
- Actional semantics = radical reversion of the fregean relationship between Sense and Reference. Indeed:
- the computational evaluation = but **a transformation of Sense**. For instance : $2+2$ and 4 (same reference but not the same sense). The computing process converts the first sense (2+2) into the second one (4).
- The reference = the invariant of that transformation of sense.
- The Sense now comes first. This opens us toward a non trivial re-construction of referents as second.

Actional semantic: "Sense / Reference" distinction disturbed

Complete reversion of the Sense/Reference relationship

- Fregean (realist) context: the reference of terms is primitive and independant of sense (a sense = only as a way to designate the reference).
- Actional semantics = radical reversion of the fregean relationship between Sense and Reference. Indeed:
- the computational evaluation = but **a transformation of Sense**. For instance : $2+2$ and 4 (same reference but not the same sense). The computing process converts the first sense (2+2) into the second one (4).
- The reference = the invariant of that transformation of sense.
- The Sense now comes first. This opens us toward a non trivial re-construction of referents as second.

Actional semantic: "Sense / Reference" distinction disturbed

Complete reversion of the Sense/Reference relationship

- Fregean (realist) context: the reference of terms is primitive and independant of sense (a sense = only as a way to designate the reference).
- Actional semantics = radical reversion of the fregean relationship between Sense and Reference. Indeed:
- the computational evaluation = but **a transformation of Sense**. For instance : $2+2$ and 4 (same reference but not the same sense). The computing process converts the first sense (2+2) into the second one (4).
- The reference = the invariant of that transformation of sense.
- The Sense now comes first. This opens us toward a non trivial re-construction of referents as second.

Actional semantic: "Sense / Reference" distinction disturbed

Complete reversion of the Sense/Reference relationship

- Fregean (realist) context: the reference of terms is primitive and independant of sense (a sense = only as a way to designate the reference).
- Actional semantics = radical reversion of the fregean relationship between Sense and Reference. Indeed:
- the computational evaluation = but **a transformation of Sense**. For instance : $2+2$ and 4 (same reference but not the same sense). The computing process converts the first sense (2+2) into the second one (4).
- The reference = the invariant of that transformation of sense.
- The Sense now comes first. This opens us toward a non trivial re-construction of referents as second.

Actional semantic: conditions for reference

Conditions for a survival of reference to the Sense transformation and its semantical exploitation

- (at least partial) termination = notion of results (our extensional screen)
- computational consistency (survival of results diversity to computation and separation properties).

Actional semantic: conditions for reference

Conditions for a survival of reference to the Sense transformation and its semantical exploitation

- (at least partial) termination = notion of results (our extensional screen)
- computational consistency (survival of results diversity to computation and separation properties).

Actional semantic: conditions for reference

Conditions for a survival of reference to the Sense transformation and its semantical exploitation

- (at least partial) termination = notion of results (our extensional screen)
- computational consistency (survival of results diversity to computation and separation properties).

Plan

Toward protological foundations for logic

Proofs/Programs correspondance

- Typed lambda-calculus : typing rules = (with help of a second level grammar) rules to select a subset of programs, hence a sub-dynamic.
- Curry-Howard correspondence : typed programs = proofs.
- Logic is selecting a subset from a computational "protologic" (the considered set of programs)
- Logic = taming of wild processes (control of the dynamic of interaction).

Toward protological foundations for logic

Proofs/Programs correspondance

- Typed lambda-calculus : typing rules = (with help of a second level grammar) rules to select a subset of programs, hence a sub-dynamic.
- Curry-Howard correspondence : typed programs = proofs.
- Logic is selecting a subset from a computational "protologic" (the considered set of programs)
- Logic = taming of wild processes (control of the dynamic of interaction).

Toward protological foundations for logic

Proofs/Programs correspondance

- Typed lambda-calculus : typing rules = (with help of a second level grammar) rules to select a subset of programs, hence a sub-dynamic.
- Curry-Howard correspondence : typed programs = proofs.
- Logic is selecting a subset from a computational "protologic" (the considered set of programs)
- Logic = taming of wild processes (control of the dynamic of interaction).

Toward protological foundations for logic

Proofs/Programs correspondance

- Typed lambda-calculus : typing rules = (with help of a second level grammar) rules to select a subset of programs, hence a sub-dynamic.
- Curry-Howard correspondence : typed programs = proofs.
- Logic is selecting a subset from a computational "protologic" (the considered set of programs)
- Logic = taming of wild processes (control of the dynamic of interaction).

Toward protological foundations for logic

Curry-Howard correspondence: foundational consequences ?

- Epistemologically : which justification for that "taming" rules ?
- The semantical success of the enterprise : strong normalization (= only results); computational semantical conditions happen to survive.
- But the taming is due to an external and primitive norm: no proper explanation of rules (dogmatic)

Toward protological foundations for logic

Curry-Howard correspondence: foundational consequences ?

- Epistemologically : which justification for that "taming" rules ?
- The semantical success of the enterprise : strong normalization (= only results); computational semantical conditions happen to survive.
- But the taming is due to an external and primitive norm: no proper explanation of rules (dogmatic)

Toward protological foundations for logic

Curry-Howard correspondence: foundational consequences ?

- Epistemologically : which justification for that "taming" rules ?
- The semantical success of the enterprise : strong normalization (= only results); computational semantical conditions happen to survive.
- But the taming is due to an external and primitive norm: no proper explanation of rules (dogmatic)

Toward protological foundations for logic

Curry-Howard correspondence: foundational consequences ?

- Epistemologically : which justification for that "taming" rules ?
- The semantical success of the enterprise : strong normalization (= only results); computational semantical conditions happen to survive.
- But the taming is due to an external and primitive norm: no proper explanation of rules (dogmatic)

Toward protological foundations for logic

Duality as algorithmic interaction

- Interactional duality (negation) will organize itself the taming
- At the proofs level, this would be interaction between:
  proofs of A and proofs of $\neg A$

Obstacle

- Gödel's Completeness theorem = establishes duality between
  Proofs of A and Models of $\neg A$
  - The syntax/semantic duality faces heterogeneous entities:
    Proofs (algorithms) and Structures
  - To face algorithms/proofs inhabiting type A, one also needs algorithms/"proofs" inhabiting type $\neg A$. Bad proofs, for sure. In logical terms: argumentations (rather than proofs), correct or not correct. Belonging to the computational "protologic".

Toward protological foundations for logic

Duality as algorithmic interaction

- Interactional duality (negation) will organize itself the taming
- At the proofs level, this would be interaction between:
  proofs of A and proofs of $\neg A$

Obstacle

- Gödel's Completeness theorem = establishes duality between
  Proofs of A and Models of $\neg A$
  - The syntax/semantic duality faces heterogeneous entities:
    Proofs (algorithms) and Structures
  - To face algorithms/proofs inhabiting type A, one also needs algorithms/"proofs" inhabiting type $\neg A$. Bad proofs, for sure. In logical terms: argumentations (rather than proofs), correct or not correct. Belonging to the computational "protologic".

Toward protological foundations for logic

Duality as algorithmic interaction

- Interactional duality (negation) will organize itself the taming
- At the proofs level, this would be interaction between:
  proofs of A and proofs of $\neg A$

Obstacle

- Gödel's Completeness theorem = establishes duality between
  Proofs of A and Models of $\neg A$
  - The syntax/semantic duality faces heterogeneous entities:
    Proofs (algorithms) and Structures
  - To face algorithms/proofs inhabiting type A, one also needs algorithms/"proofs" inhabiting type $\neg A$. Bad proofs, for sure. In logical terms: argumentations (rather than proofs), correct or not correct. Belonging to the computational "protologic".

Toward protological foundations for logic

Duality as algorithmic interaction

- Interactional duality (negation) will organize itself the taming
- At the proofs level, this would be interaction between:
  proofs of A and proofs of $\neg A$

Obstacle

- Gödel's Completeness theorem = establishes duality between
  Proofs of A and Models of $\neg A$
  - The syntax/semantic duality faces heterogeneous entities:
    Proofs (algorithms) and Structures
  - To face algorithms/proofs inhabiting type A, one also needs algorithms/"proofs" inhabiting type $\neg A$. Bad proofs, for sure. In logical terms: argumentations (rather than proofs), correct or not correct. Belonging to the computational "protologic".

Toward protological foundations for logic

Duality as algorithmic interaction

- Interactional duality (negation) will organize itself the taming
- At the proofs level, this would be interaction between:
  proofs of A and proofs of $\neg A$

Obstacle

- Gödel's Completeness theorem = establishes duality between
  Proofs of A and Models of $\neg A$
  - The syntax/semantic duality faces heterogeneous entities:
    Proofs (algorithms) and Structures
  - To face algorithms/proofs inhabiting type A, one also needs algorithms/"proofs" inhabiting type $\neg A$. Bad proofs, for sure. In logical terms: argumentations (rather than proofs), correct or not correct. Belonging to the computational "protologic".

Toward protological foundations for logic

Toward an auto-taming of the interaction

- Broaden the logical horizon: from proofs to argumentations (proofs and para-proofs)
- Interactional duality between argumentations processes m and p: agreement between them along the deliberation ($m \perp p$)
- Generalize the duality notion to sets of argumentations
- Types = set of processes with a common behavior with respect to duality (a type = a set which is dual to some set)
- Typing (logical) rules = operations over types preserving the property to be a type

Toward protological foundations for logic

Toward an auto-taming of the interaction

- Broaden the logical horizon: from proofs to argumentations (proofs and para-proofs)
- Interactional duality between argumentations processes m and p: agreement between them along the deliberation ($m \perp p$)
- Generalize the duality notion to sets of argumentations
- Types = set of processes with a common behavior with respect to duality (a type = a set which is dual to some set)
- Typing (logical) rules = operations over types preserving the property to be a type

Toward protological foundations for logic

Toward an auto-taming of the interaction

- Broaden the logical horizon: from proofs to argumentations (proofs and para-proofs)
- Interactional duality between argumentations processes m and p: agreement between them along the deliberation ($m \perp p$)
- Generalize the duality notion to sets of argumentations
- Types = set of processes with a common behavior with respect to duality (a type = a set which is dual to some set)
- Typing (logical) rules = operations over types preserving the property to be a type

Toward protological foundations for logic

Toward an auto-taming of the interaction

- Broaden the logical horizon: from proofs to argumentations (proofs and para-proofs)
- Interactional duality between argumentations processes m and p: agreement between them along the deliberation ($m \perp p$)
- Generalize the duality notion to sets of argumentations
- Types = set of processes with a common behavior with respect to duality (a type = a set which is dual to some set)
- Typing (logical) rules = operations over types preserving the property to be a type

Toward protological foundations for logic

Toward an auto-taming of the interaction

- Broaden the logical horizon: from proofs to argumentations (proofs and para-proofs)
- Interactional duality between argumentations processes m and p: agreement between them along the deliberation ($m \perp p$)
- Generalize the duality notion to sets of argumentations
- Types = set of processes with a common behavior with respect to duality (a type = a set which is dual to some set)
- Typing (logical) rules = operations over types preserving the property to be a type

Toward protological foundations for logic

Toward an auto-taming of the interaction

- Broaden the logical horizon: from proofs to argumentations (proofs and para-proofs)
- Interactional duality between argumentations processes m and p: agreement between them along the deliberation ($m \perp p$)
- Generalize the duality notion to sets of argumentations
- Types = set of processes with a common behavior with respect to duality (a type = a set which is dual to some set)
- Typing (logical) rules = operations over types preserving the property to be a type

Toward protological foundations for logic: Internal versus External completeness

"External" completeness (traditional completeness, K. Gödel)

$\forall A$ (formula):

either $\exists p$ s.t. $p \vdash A$ (p a proof/algorithm) or $\exists m$ s.t. $m \nvDash A$ (m a structure)

Internal completeness (strong completeness, J-Y. Girard)

$\forall A$ (formula), $\forall p$ (algorithm/argumentation):

either $p \vdash A$ (p an algorithm/argumentation) or $\exists m$ s.t. $m \in A^{\perp}$ and $m \not\perp p$ (m an algorithm/argumentation)

In other terms: m is an argumentation against A and the interaction between the algorithms/argumentations m and p fails, meaning that m "detects" an argumentative "error" in p or leads p into an infinite "debate".

Toward protological foundations for logic: Internal versus External completeness

"External" completeness (traditional completeness, K. Gödel)

$\forall A$ (formula):

either $\exists p$ s.t. $p \vdash A$ (*p* a proof/algorithm) or $\exists m$ s.t. $m \nvDash A$ (*m* a structure)

Internal completeness (strong completeness, J-Y. Girard)

$\forall A$ (formula), $\forall p$ (algorithm/argumentation):

either $p \vdash A$ (*p* an algorithm/argumentation) or $\exists m$ s.t. $m \in A^{\perp}$ and $m \not\perp p$ (*m* an algorithm/argumentation)

In other terms: *m* is an argumentation against *A* and the interaction between the algorithms/argumentations *m* and *p* fails, meaning that *m* "detects" an argumentative "error" in *p* or leads *p* into an infinite "debate".

Toward protological foundations for logic: Internal versus External completeness

"External" completeness (traditional completeness, K. Gödel)

$\forall A$ (formula):

either $\exists p$ s.t. $p \vdash A$ (*p* a proof/algorithm) or $\exists m$ s.t. $m \nvDash A$ (*m* a structure)

Internal completeness (strong completeness, J-Y. Girard)

$\forall A$ (formula), $\forall p$ (algorithm/argumentation):

either $p \vdash A$ (*p* an algorithm/argumentation) or $\exists m$ s.t. $m \in A^{\perp}$ and $m \not\perp p$ (*m* an algorithm/argumentation)

In other terms: *m* is an argumentation against *A* and the interaction between the algorithms/argumentations *m* and *p* fails, meaning that *m* "detects" an argumentative "error" in *p* or leads *p* into an infinite "debate".

Toward protological foundations for logic

Making logic emerge from protologic

- Allows generating logical rules (types construction) from the algorithmical protologic.
- Architectonic foundational value of negation (algorithmic duality):

Toward protological foundations for logic

Making logic emerge from protologic

- Allows generating logical rules (types construction) from the algorithmical protologic.
- Architectonic foundational value of negation (algorithmic duality):

Conclusion

- Logic depends on the protologic:

  ~~absolute foundations~~
  relative fondations

- What is A protologic ?

- From Logic/Psychology to Logic/Physics.

Conclusion

- Logic depends on the protologic:

  ~~absolute foundations~~
  relative fondations

- What is A protologic ?

- From Logic/Psychology to Logic/Physics.

Conclusion

- Logic depends on the protologic:

  ~~absolute foundations~~
  relative fondations

- What is A protologic ?

- From Logic/Psychology to Logic/Physics.

The end

Thanks to LOCI team. . .