RAPPORT DE MISSION - UMR 8163 « SAVOIRS, TEXTES, LANGAGE »

NOM Prénom : De Mol, Liesbeth

Nature de la mission (séjour de recherche, participation à un colloque…) : organisation d'un symposium: *HaPoC Symposium: Computations, Proofs and Models* lors du 15ième conférence Logic, Methodology and Philosophy of Science (CLMPS 2015)

Lieu et date : CLMPS, Helsinki, 3-8 Aout, 2015

Frais de mission attribués par le laboratoire :

Frais attribués: 775 EUR

Description de la mission (par ex. résumé de l'intervention proposée/activités de recherche réalisées au cours de la mission…) :

La 15ième conférence Logic, Methodology and Philosophy of Science (CLMPS 2015) a été organisée de 2-8 Aout, 2015. Cette conférence est la conférence primordiale de la *Division for Logic, Methodology and Philosophy of Science* et est organisé tous les 4 ans. La commission de la DHST (*Division History of Science and Technology*) History and Philosophy of Computing (HaPoC), dont je suis présidente, voudrait proposé HaPoC comme commission entre les deux divisions DHST et DLMPS pendant l'assemblée générale de la DLMPS alors que la commission soit vraiment une commission sur l'histoire *et* la philosophie du « calcul ». C'était pour cette raison que la DLMPS nous a invité d'organiser un symposium alors que HaPoC peut se présenter.

Avec Giuseppe Primiero (Middlesex University), le vice-président de HaPoC, j'ai animé un symposium dont le but était de réfléter sur des interactions entre la logique et l'informatique d'un point de vue interdisciplinaire. Nous avons invité plusieurs orateurs internationals qui réprésentent le caractère interdisciplinair de la commission. Le programme était comme suit:

Liesbeth De Mol (CNRS, UMR 8163, STL Université de Lille 3), *Introduction*

Stephanie Dick (Harvard University), *Putting Mathematics into the Computer: Implementation and Epistemology in Early Automated Logic*

Dale Miller (Inria/Saclay and Lix) *Defining the semantics of proof evidence*

Peter Koepke (University of Bonn) *Formalism and Computations*

Oron Shagrir (University of Jerusalem), *The Church-Turing Theses*

Matti Tedre (Stockholm University), *Competing Claims to Computing as a Discipline*

Edgar Daylight (Universiteit Utrecht) *Using History to Make Software More Tangible*

Gilles Dowek (Inria/Deducteam and Mooc Lab), *How do we know that a statement true in Computer Science?*

Discussion

Edgar Daylight (Universiteit Utrecht), ***Using History to Make Software More Tangible***

While computer museums are successful in exhibiting restored computing machinery, they face the challenge of finding ways to make *software* accessible to a general audience. One possible approach, which I will discuss, is based on the observation that several software tools are built in fundamentally different ways, even if the tools under scrutiny are designed to solve similar problems. By examining the principles underlying these software tools with pluralistic, historically-shaded glasses, the historian can help the non-expert to obtain a firmer grip on what software entails today.

For example, Marvin Minsky's 1967 book, `Computation: Finite and Infinite Machines,' can be scrutinized with epistemological lenses in the following manner. Minsky used the word "program" on page 25 to refer to data and instructions that fit in a real, finitely large memory of a physical computer. On page 153, by contrast, he used the very same word to refer to a mathematical object of "unlimited storage capacity." Fifteen years later, the word "program" had several meanings:

+ for Actor A, the term referred solely to a technological artefact;

+ for Actors B and C, the term referred to a mathematical object of finite size and infinite capacity, respectively; and

+ for Actor D, a computer program was *not* a logico-mathematical construction but a model (of the real world) that cannot be understood by solely resorting to logic and strict rules.

Computer programmers thus viewed software technology from different angles, and these different schools of thought are apparent in present-day software technologies as well.

It is by contrasting a particular computing concept or phrase (such as the words "computer program") from the viewpoints of multiple historical actors, and by doing this exercise for several concepts and phrases (such as "universal Turing machine," "recursive procedure," and "computer science"), that children and other non-experts will gradually acquire a better understanding of, and an appreciation for, the constituents of present-day software.

Stephanie Dick (Harvard University), ***Putting Mathematics into the Computer: Implementation and Epistemology in Early Automated Logic***

There is no automation without invention. Especially in the early decades, actually getting programs to run on computers was no small feat. Models and algorithms had to be translated into a form that computers could execute. Significant hurdles related to the allocation and management of memory had to be overcome. The affordances of computing machines had to be accommodated. Models and algorithms had to be *implemented.* The work of implementation spans multiple media - from paper to transistor - and involves many practices - from diagramming to coding. In implementing programs practitioners had to craft many new tools, both abstract and material. Implementation is where abstraction and materiality meet. It is the site where we see practitioners rethinking their objects of interest, their disciplines, their theories, their questions, through the lens of computing machines. And yet, where histories of software even exist, implementation has not been a dominant focus. Historians have, for the most part, emphasized high level description over low-level detail, taking for granted the complex division between hardware and software.

This paper focuses on implementation. In particular, I track how one logician - Chinese American Hao Wang - took a highly abstract piece of mathematical logic and translated it into an actionable set of computer tools for mathematical research. I propose that in doing so, Wang introduced new tools and practices to mathematics. The resulting program, "The Program P" developed in the late1950s, also reveals the

epistemological significance of implementation: in making it, Wang came to know new things about logic and to know them in quite new ways.

Gilles Dowek (Inria), ***How do we know that a statement true in Computer Science?***

Different sciences such as natural sciences, mathematics, ..., use different notions of truth. In this talk I will ask the question of the nature of truth in informatics. I will defend the idea that different notions of truth are simultaneously at work, and that informatics articulates them in a specific way.

Dale Miller (Inria Saclay and Lix, École polytechnique), ***Defining the semantics of proof evidence***

If automatic and interactive theorem provers record completed proofs, they can do so using a range of proof structures, such as natural deduction, tableaux, resolution, and winning strategies. Technology-based proof scripts are also commonly used. I will outline how recent advances in proof theory, particularly, focused proof systems for intuitionistic and classical logics, can be used to formally define a wide range of proof evidence. Interpreters of such formal definitions become proof checkers. In order to make it possible to elide many formal details in proofs, such proof checkers are expected to perform significant proof reconstruction via deterministic and non-deterministic computations. I will discuss some of the ramifications of employing this framework on the ability of machines and humans to trust and communicate formal proofs.

Oron Shagrir (University of Jerusalem), ***The Church-Turing theses***

The Church-Turing thesis (CTT) places an upper bound on the computational power of computing agents; the bound is that of Turing-machine computability. My aim is to discuss different versions of CTT and to show that they reflect different notions of computing agents. I start with the original theses by Church and Turing that are arguably about human computers. I argue, however, that we can distinguish between different notions of human computers, which I call cognitive and non-cognitive. I then discuss several machine versions of CTT, on which the computing agent is an algorithmic machine, a physical machine, or any computing machine. I close with the so-called strong CTT that is about computational complexity.

Bénéfice de la mission (pour le chercheur/l'enseignant-chercheur, pour le laboratoire) :

Le symposium était bien visité par les autres participants de CLMPS et a été bien récipié. Les différents exposés ont donné lieu à des discussions parfois intenses. En général, les participants ont bien apprécié l'interdisciplinarité du symposium ce qui est très important étant donné la mission interdiscplinaire de la commission HaPoC. Ceci est aussi confirmé par l'approbation de l'AG de la DLMPS d'une commission HaPoC entre DLMPS et DHST. Pratiquement ca veut dire que la commission s'engage d'organiser des symposia HaPoC dans les conférences CLMPS des prochaînes années.