ISSN 2186-7437

NII Shonan Meeting Report

No. 2015-2

Logical Analysis of Descriptions and their Representations

—a Computational Logic Approach—

Yoshiki Kinoshita
Bengt Nordström

January 26–29, 2015

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

Logical Analysis of Descriptions and their Representations

—a Computational Logic Approach—

Organisers:
Yoshiki Kinoshita (Kanagawa University)
Bengt Nordström (Chalmers University of Technology)

January 26–29, 2015

Contents

1 Overview

Aim of the workshop

Our aim is to investigate whether logical methods are applicable to analysis of descriptions in, for instance,

- system specification,
- formal mathematics,
- system assurance,
- juris-informatics (law) and
- industrial standardisation.

In these areas, we have descriptions of some objects and relations between them. These descriptions are either formal or informal (in English or some graph language without precise semantics). We want to conduct logical, computational and mathematical studies that clarify the structure of such descriptions. Aspects of this has been studied in discourse representation theory, constructive type theory and informal logic. Results of earlier studies include object role modelling (ORM), state charts, Toulmin diagrams, goal structuring notation (GSN), grammatical frameworks (GF) and various proof assistants. Some applications may be found in the above areas.

There has been an emphasis on reasoning aspects when applying mathematical logic in Computer Science and Informatics. In this workshop, we want to investigate the importance of ontological (Here we use the word ontology as a description of a domain with objects and relations and properties of these.) aspects of mathematical logic. These are more important than deductive aspects in, for instance, validation of information systems. We are interested in description languages of varying level of formality, where the formal part makes some mechanical checking and manipulation possible.

The combination of deduction and ontology can also be found in mathematics where the concepts have matured during hundreds of years. Nowadays, a similar combination of ontological and deductive aspects can be found in descriptions of information systems, laws and rules, etc. We are convinced that studies of informal mathematics will be an important contribution to a deeper understanding of these issues. This is an attempt to put together people from different fields to find a common analysis of logical description languages. We need to agree about the major research issues. The time is not yet ripe to agree about techniques.

Study area

- Constructive Type Theory
- Informal logic
- Discourse Representation Theory
- Ontology in information science

Possible application area

- System specification
- Formal mathematics
- System assurance (verification and validation)
- Juris-informatics
- Industrial standardisation

Related methods

- Proof assistants: editors for formal proofs like COQ, HOL, Agda.
- Toulmin model of argument
- Goal Structuring Notation
- Grammatical frameworks

- Object Role Modelling
- State charts

Research questions

- How to describe a modelling language (syntax, semantics, well formalness)How to use a modelling language
- Analysis of how models are evolved
- Examples of models
- General requirement of modelling languages
- Different persons want to have different knowledge/perspective of the same object
- One description can have many presentations (different natural languages, graphs, pictures)

2 Summary

The record of the workshop, including the program and the slides used for presentation, is put on the following URL:

`https://sites.google.com/site/logicalanalysisofdescriptions/`

At the very end of the workshop, a short wrap-up session was held to clarify the understanding shared by the participants. This section summerises the discussion in the wrap-up session.

Terminology

Although it seems natural to use the notion of formal systems in mathematical logic for our analysis of description, the objects to be used for description, are not necessarily sequences of symbols but diagrams consisting of nodes and edges. Formal systems in Logic appear to use sequence of symbols but they often use a tree structure behind the sequence, so the objects for description in formal systems are really trees, rather than sequences, of symbols.

So, it is assumed hereafter that a system for description is provided by a set of sentences and its subset of wellformed (acceptable) sentences. We call the system for description *formal* if there is a decision procedure for the set of wellformed sentences (i.e., if there is a terminating program that tells whether a given sentence is acceptable or not), and call it *informal* otherwise. In the case that the set of objects consist of sequences of symbols, a formal grammar may work to serve as the decision system, thus formal systems in the sense of mathematical logic is formal in the sense here.

Abstract syntax is a representation of a description by means of a mathematical object (giving the essential syntactic structure expressing how a description is built and what the parts are), and *concrete syntax* is a presentation of a description for a human being such as a sequence of characters, pixels on a screen, sequence of sounds etc.

Representation and presentation

Through the discussion of this workshop, it was made clear that there are three approaches to abstract syntax, and two to concrete syntax.

The three approaches to abstract syntax are

AS1 *trees*

AS2 *directed acyclic graphs*

AS3 *general graphs*

The first approaches can be regarded as more restricted than the later since a tree is a directed acyclic graph.

The two approaches to concrete syntax are

CS1 an approach to use textual presentation and

CS2 an approach to use graphical presentation.

The approach to use the language Agda for assurance cases, for instance, may be classified as mixture of AS1 and CS1. Goal Structured Notation may be classified as AS2 and CS2. Sato's L-expressions may be regarded as an attempt to find a good textual presentation (CS1) for the representation of lambda terms by means of trees (AS1).

Justification and demonstration

We try to apply logic because we need to talk about validity. The validity is not only of the claims, but also of the formal system itself.

Formal proofs express demonstrations by means of a valid derivation of claims in a given formal system, but they do not tell us anything about the validity or appropriateness of the formal system itself. On the other hand, *argumentations*, used in the assurance case community, are justifications to cover the validity of both claims and the formal system itself.

It may be said that formal proofs are for *verification*, while argumentations are for *validation*. In systems and software engineering, verification and validation are distinguished carefully. ISO 15288 *System life cycle processes*, for instance, defines these terms as follows:

> **verification:** confirmation, through the provision of objective evidence, that specified requirements have been fulfilled
>
> Note 1 to entry: Verification is a set of activities that compares a system or system element against the required characteristics. This includes, but is not limited to, specified requirements, design description and the system itself. The system was built right.
>
> **validation:** confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled
>
> Note 1 to entry: A system is able to accomplish its intended use, goals and objectives (i.e., meet stakeholder requirements) in the intended operational environment. The right system was built.

Changes

Systems usually change over time and it is difficult to synchronise this with the argumentation of the system. It was agreed that is a serious problem. No decisive solution was proposed in the workshop, though some techniques, including Kido's work on defeasible GSN, was discussed.

3 Abstracts of the talks

Recurrent patterns in bringing together natural language-based descriptions and formal representations

Serge Autexier, DFKI Bremen

In this talk I present work done in four different application areas to go from natural language based descriptions to semi-formal and formal logic representations and back. The areas are mathematical documents, hardware system specifications, law texts, and smart home assistance processes. Repetitive patterns are

(1) natural language input being mapped in more than one target semi-formal or formal representations,

(2) not only mapping from natural language to the increasingly formal representations is necessary, but also the way back is important,

(3) checking occurs at all different representations and

(4) maintaining relationships between fragments of the representations at the different levels is important for bidirectional incremental processing and analysis as well as change management and change effect propagation.

In the talk we will sketch for each of the four applications areas the formalisms and techniques used or that we plan to use for (1)-(4) in order to identify commonalities for a uniform methodology.

Safety and assurance cases: formality and expressiveness in Claims, Argument and Evidence

Robin Bloomfield, Adelard and London City

In this talk I will describe the role of claims, arguments and evidence (CAE) in safety and assurance cases. As practitioners, we have been able to review a variety of safety and assurance cases to capture the expressiveness that is needed to describe these engineering arguments. From this we have defined a number of CAE fragments, Blocks, of reasoning. These provide a mechanism for experimenting with different semantics and I will speculate on whether we can have a framework that encompasses the informal to formal and allow different levels of formal rigour.

Language Support for Assurance Cases

Ewen Denney, SGT / NASA Ames

Assurance arguments are inherently heterogenous, aggregating both deductive and inductive reasoning and diverse evidence. Clearly, assurance arguments can sometimes be formalized and there are times when this is worth doing, at least for part of the argument. Equally, however, there are times when an argument is best left informal. We argue that despite this mixed formality, assurance arguments can still be usefully given formal underpinnings.

We contend that for assurance cases to be accepted as first-class engineering artifacts and to improve their credibility, they need a rigorous basis. Additionally, we believe that tool support is crucial for the wider acceptance of structured safety arguments during the certification of safety-critical products.

In this talk we describe several aspects of such a formal basis, which have been implemented in the AdvoCATE toolset.

Analysing Ambiguity in the Grammatical Framework

Ramona Enache, Göteborg

Controlling complexity is the essence of computer programming

Johan Georg Granström

The software industry has formally been in a crisis since the 1968 NATO Software Engineering Conference in Garmisch. This means that the majority of today's practitioners only have experienced an industry in crisis, and consider this the normal state. The popularity and prevalence of lightweight practices are a symptom of this crisis: the only constant is rapid change.

The title of this talk is "controlling complexity is the essence of computer programming", which is a quote from the book Software Tools written by Kernighan and Plauger and published in 1976. I think this quote expresses a fundamental truth. The technological advances and engineering feats since 1968 speak for themselves; but, in terms of controlling complexity, few advances have been made. Complexity cannot be controlled unless it is first made visible and this is where visual formal languages can make a big difference. Complexity is too often hidden deep in code repositories understandable only by a select few engineers. A visual formal language must have a rigorous formal semantics and an intuitive visual representation, without semantic gap between the two. Moreover, to be useful in practical applications, it has to be integrated into the toolchain and the formal semantics has to be the source of truth. In this presentation, I will discuss what it means for the formal semantics to be the source of truth for visual formal languages dealing with models, workflows, and deployment.

Dynamic Reasoning for Safety Assurance

Ibrahim Habli, York

My talk will discuss the concept of dynamic safety cases as a basis for representing and supporting through-life safety assurance. In particular, I will explore how dynamic safety cases can be used as a way for proactively assessing and evolving the safety reasoning not only during initial development but also after deployment based on operational data.

Decentralized SNS for Collaborative Authoring of Logical Structures

Koiti Hasida, Tokyo

Empirical evidences corroborate that graphs representing logical structures are not only easier to understand and compose but also better in quality than traditional linear texts. However, graphical authoring has not yet spread to many users so far. It may be possible to commoditize graphical authoring by implementing it on a decentralized SNS. Potential use cases and business models are discussed.

The logic of infinitival complement constructions

Lauri Karttunen, Stanford

The topic of my talk is part of a larger enterprise of 'Natural Logic that tries to provide systems of reasoning in natural language without translation to a logical formula by rules that act on structures that a parser would produce to yield inferences that are obviously valid because of the lexical meaning of the words.

For example, "John forgot to close the door" implies that John did not close the door. That is a logical entailment because if John actually closed the door the sentence is false.

But there is another aspect of the meaning of "NP forgot to VP" that is more difficult to pin down. The speaker could mean that John had intended to close the door but forgot to do it, or she could mean hat John was supposed to close the door but forgot his responsibilities rather than his own intentions.

The speaker of "John forgot to close the door" is committed to both to the entailment that John did not close the door and the suggestion that he intended or was supposed to close the door.

But these commitments are of different types. If you turn the example into a question "Did John forget to close the door?" what is being questioned is whether John closed the door but the implication that he intended or was supposed to do it is still there.

This is a long-winded introduction to the distinction that linguists have made since the 1970s between entailment and presupposition. There are few computational systems that try to deal with these issues. I hope to interest the GF community to take up the challenge.

Balancing the Formal and Informal in Safety Case Arguments

Tim Kelly, York

In many safety-critical industries, developers and operators are required to construct and present well reasoned arguments that their systems achieve acceptable levels of safety. These arguments (together with supporting evidence) are typically referred to as a safety case. Safety arguments historically have been communicated through narrative text, leading often to problems of comprehension and communication. Over the last twenty years, there has been increasing interest in using structured argumentation notations such as GSN (Goal Structuring Notation) or CAE (Claims-Argument-Evidence) to communicate the structure of the argument. Whilst such arguments are structured, they remain informal. There is increasing interest in exploring how these informal arguments may be modelled in formal logic, potentially opening up benefits of forms of analysis not possible with informally recorded argument. This talk will discuss a a number of considerations in balancing the role of informal and formal logic in modelling safety case arguments.

Encouraging Critical Communication Using Defeasible Goal Structuring Notation

Hiroyuki Kido, Tokyo

Confirmation bias and clarification of a rationale for changes are two important issues associated with the use of safety, assurance and dependability cases. In this presentation, I would like to talk about an attempt to address these issues with a defeasible goal structuring notation, i.e., one possible extension of a goal structuring notation.

Formal assurance case

Yoshiki Kinoshita, Kanagawa, organiser

Starting from the widely accepted observation that assurance cases cannot, and should not, fully be formalised, and assuming formal systems are not Occam's razor, I still wish to discuss how much formal systems can help writing and evaluating assurance cases.

Formal systems are necessary to enable manipulation of assurance cases by mechanical means, like consistency checking (in various settings), traceability checking and all sorts of translations. They also help, however, provide a framework for assurance cases so as to give guidance for writing assurance cases, as well as criteria for evaluating them.

The Hard Problems Are Outside The Formalisms

John Knight, Virginia

All forms of logic reduce to symbol manipulation. For logic to be useful in application, the logic requires an interpretation, and interpretation has to be in terms of natural language. The semantics of all forms of natural language reduce to an agreed shared meaning. Thus the results of using logic are inherently limited by the accuracy and completeness of an interpretation that has to be informal.

The need for analysis in areas such as safety assurance is well illustrated by the continued though infrequent occurrence of serious accidents. Could formalism reduce or perhaps eliminate this problem? In this presentation, I will argue that there is no possibility and there never will be of analyzing properties such as system safety from within a purely mathematical framework. I will review the fundamental limitations of logic and natural language, and illustrate the difficulties using the analyses of two major accidents. Finally, I will show that the hard problems remain outside the sphere of any form of analysis.

Broad-Coverage Semantic Analysis using Big Data on the Web

Gerard de Melo, Tsinghua

High-quality semantic analysis of natural language text has been a long-standing goal in computer science. While we are still far from solving this problem, we are now able draw on unprecedented amounts of Big Data on the Web to develop systems that are more robust and cover a larger number of concepts and phenomena than those of the past. In this talk, I present a series of results on how information mined from the Web can enable better semantic analyses.

The first one is UWN/MENTA, one of the largest multilingual semantic hierarchies, describing millions of names and words in over 200 languages. A second part focuses on efforts for reasoning with such knowledge, such as YAGO-SUMO and SPASS-XDB, which connect entities to axioms.

Other examples include NomLex-BR and our FrameNet-based system for better semantic abstraction.

Finally, I conclude with our efforts to better model human-style commonsense knowledge and reasoning, e.g. in the WebChild and ImpLex projects. For more information about these projects, please refer to http://gerard.demelo.org.

Definitions and declarations

Bengt Nordström, Chalmers, organiser

It is a truism that formal reasoning will not solve the problem of avoiding errors in the design of programs and other systems.

But I think that formal reasoning can help us to construct such systems. We can have a notion of syntactical correctness and use editors to construct designs which are correct by their constructions.

In order to give a modular description of formalizations it is important how names and definitions are introduced. I will give a first sketch to such an analysis.

Interlingua to Interlinguas

Aarne Ranta, Göteborg

In 1629, Descartes proposed a “language of true philosophy”, which would “enumerate all thoughts and order them”, in the same way as “one can learn in one day the names of all numbers up to infinity and write them in an unknown language”. This proposal was a response to a naive approach that someone was marketing at the time, based on a superficial list of words.

Descartess proposal was one inspiration for translation based on interlingua - in modern terms, a formal semantic description of meaning expressed in different languages, which must be preserved in translation. In addition to defining what translation must preserve, interlingua has an advantage of scalability: a translation system for N languages can be built from 2N+1 components, as opposed to N(N+1) if each language pair is treated separately.

However, constructing a "language of true philosophy is a demanding task. No approach to automatic translation that depends on the existence of such an interlingua can be expected to deliver results in any foreseeable future. GF (Grammatical Framework) is an approach meant to cope with this problem. GF implements interlingual translation, but instead of assuming one universal interlingua, it provides a framework for defining different ones, inspired by logical frameworks (LF) based on type theory. Thus there is a GF translator for mathematical exercises, another one for tourist phrasebooks, and so on.

Interlinguas in typical GF applications are domain-specific. They are much like domain ontologies in the semantic web, as opposed to universal ontologies. They enable high-quality meaning-preserving translation. But unlike main-stream tools like Google translate, they are not able to translate everything. In recent GF work, however, attention has also been turned to the main-stream task. There we need a universal interlingua that can translate everything. But again, we cannot hope to have just a single one. Instead, we use the framework to define a layered system of interlinguas, where the highest layer represents semantics (preserving meaning), the middle layer represents superficial syntax (preserving grammatical structure), and the lowest layer represents chunks of words (preserving some of the main ideas of the source text).

Following this idea, we are building an interlingual wide-coverage translation system, which currently addresses 12 languages. This is an on-going open-ended project, where contributions are welcome, as regards both target languages and interlingual descriptions.

On The Interpretation and Evaluation of Assurance Case Arguments

John Rushby, SRI

An assurance case justifies certain claims about a system by means of an argument based on evidence about the system, its design, and construction. In contrast, standards and guidelines for assurance typically specify just the evidence required and lack an explicit argument.

An assurance case argument decomposes the top claim into sub claims that are recursively decomposed until we reach sub claims directly supported by evidence. The world is uncertain and our knowledge may be imperfect, so the overall argument is necessarily inductive: that is, it strongly suggests that the top claim is true but cannot guarantee it. So we have doubts, but where in the argument may they be located? Put another way: does the necessarily inductive nature of the overall argument sanction each step being inductive?

Arguments may be normalized (by introducing additional sub claims if necessary) into a simple form in which each step comprises a claim supported either by a collection of subclaims or by a collection of evidence, but not by a mixture of the two; we refer to the former as reasoning steps and the latter as evidential steps.

Evidential steps are the bridge between the real world and the world of our concepts, expressed as claims: they are interpreted epistemically and are necessarily inductive. The evidence supplied in an evidential claim is "weighed" to ensure it crosses some threshold of credibility so that its claim may be treated as a "settled fact." The weighing can be quantified using subjective probabilities, formalized using ideas from Bayesian analysis, and mechanised using Bayesian Belief Nets (BBNs). Not all the evidence in a step need directly support its claim: some evidence may be used to increase confidence in other evidence, and BBN analysis can accommodate this.

Reasoning steps are about just that–reasoning–and are interpreted in logic. The conjunction of the subclaims to a reasoning step should deductively entail its claim: that is, truth of the subclaims in a step must guarantee truth of its claim. Confidence items have no part in deductive reasoning steps; allowing reasoning steps to be inductive and to include confidence items sets too low and too fungible a bar, and one that is difficult, if not impossible, to evaluate.

Deductive reasoning steps can be evaluated by actively seeking and then dealing with defeaters: defeater are circumstances in which the claim may be false even though the subclaims are true; they are to arguments what hazards are to systems: safety is ensured by aggressively searching for them and addressing any that are discovered. Evaluation of bespoke arguments is delicate and costly; there is much to be said for basing assurance case arguments on templates that have been subjected to extensive review.

Not all systems are equally critical so it is desirable to allow less costly assurance for less risky systems: this is called graduated assurance. But how can we modify an argument template to make it weaker without destroying its credibility?

An argument whose evidential steps cross the boundary established for settled facts, and whose reasoning steps are all deductive is said to be sound. Graduated assurance cannot remove or change sub claims without destroying soundness (or requiring a reworked argument), but it can lower the boundary required for evidential steps. Although it is not acceptable to evaluate reasoning steps probabilistically (because that takes us back to assurance as a collection of evidence and vitiates the point of an assurance case argument) it is feasible to propagate probabilistic estimates of evidential strength through the reasoning steps and thereby derive a crude measure of the overall strength of a sound argument, and this can guide the evidential weakening performed for graduated assurance.

The L-calculus - A natural extension to -calculus

Masahiko Sato, Kyoto

Closing the Validation Gap

Anton Setzer, Swansea

Verification is the process of guaranteeing that a program fulfils its specification. Validation is the processes of checking that a program fulfils the requirements or that the specification is sufficient to guarantee the requirements. Specification can be fully formal, and we can in many frameworks prove that a programs fulfils the specification. Requirements are informal, and express what the program is supposed to do. Therefore validation cannot be carried out fully formally, and we cannot guarantee that a specification is sufficient to guarantee the requirements. Therefore there is a validation gap between the specification and the requirements. There are examples where the specification is much more complicated than the program to be verified, and it might be easier to see that the program fulfils the requirements rather than the specification guaranteeing it.

One way around this problem is to introduce a specification which is as close as possible to the requirements. One obtains then two specifications: A program specification which is used to verify the correctness of programs, and a requirements specification, which is close to the requirements. Now it should be possible to prove that the program specification implies the requirements specification.

The verification of a program against the requirements specification is therefore carried out in two steps: One step is to verify that the program fulfils the program specification. This step might be carried out using automated theorem proving. The second step is to verify that the program specification implies the requirement specification. This step might be carried out using interactive theorem proving.

We will demonstrate this distribution by referring to a PhD project with Karim Kanso, in which railway interlocking systems were verified in this way in the interactive theorem prover Agda. This verification used an integration of automated theorem proving into Agda.

References: Karim Kanso, Anton Setzer: A light-weight integration of automated and interactive theorem proving. Math. Struct. in Comp. Science, Online First, pp. 1 - 25, 2014. Doi 10.1017/S0960129514000140.

Formal Assurance Case in Agda, FACIA

Makoto Takeyama, Kanagawa

We present a notion of formal assurance case, which makes explicit the dependency of a system assurance argument on its ontological basis and presumptions. The ontology is formulated as a formal theory and the argument as a proof in it. Both are described in the programming-and-proof language Agda based on Type Theory. Routine integrity-checking on arguments is done by Agda type checker, so that the human reviewer can concentrate on their judgment on contents. The module system and other programming language features of Agda provides the structuring and abstraction mechanisms for assurance cases necessary for higher readability and maintainability in the large.

Formal Symbolic Learning with Logic and its Application to Test Case Generation

Akihiro Yamamoto, Kyoto

Recently machine learning is attracted much attention for analyzing large scale data. The relation between machine learning and logic programming languages was first indicated by Gordon Plotkin. In this talk we review the relation on the viewpoint of recent machine learning research. Briefly speaking, machine learning is categorized into two types, supervised and unsupervised. For both types logic programming languages have nice properties which are useful for learning from discrete structural data and could be related to program development.