Formal Semantics in MTTs: Playing Around with the Coq Proof Assistant

Stergios Chatzikyriakidis
CLASP, Department of Philosophy, Linguistics and Theory of Science
University of Gothenburg

June 29, 2018

CLASP centre for linguistic theory and studies in probability

Proof Assistants: A brief history

- Started in the early 60s

CLASP centre for linguistic theory and studies in probability

Proof Assistants: A brief history

- Started in the early 60s
  - The need for formally verified proofs

CLASP centre for linguistic theory and studies in probability

Proof Assistants: A brief history

- Started in the early 60s
  - The need for formally verified proofs
  - The AUTOMATH project. Late 60's (see [de Bruijn(1980)] for a survey)
    - ⋆ Aim: a system for the mechanic verification of mathematics

Proof Assistants: A brief history

- Started in the early 60s
  - The need for formally verified proofs
  - The AUTOMATH project. Late 60's (see [de Bruijn(1980)] for a survey)
    - Aim: a system for the mechanic verification of mathematics
    - Several AUTOMATH systems have been implemented

CLASP centre for linguistic theory and studies in probability

Proof Assistants: A brief history

- Started in the early 60s
  - The need for formally verified proofs
  - The AUTOMATH project. Late 60's (see [de Bruijn(1980)] for a survey)
    - Aim: a system for the mechanic verification of mathematics
    - Several AUTOMATH systems have been implemented
    - The first system to practically exploit the Curry-Howard isomorphism

CLASP centre for linguistic theory and studies in probability

Proof Assistants: A brief history

- Proof-assistant technology has gone a long way since then

centre for
linguistic theory
and studies in probability

Proof Assistants: A brief history

- Proof-assistant technology has gone a long way since then
  - Proliferation of proof assistants implementing various logical frameworks

CLASP centre for linguistic theory and studies in probability

Proof Assistants: A brief history

- Proof-assistant technology has gone a long way since then
  - Proliferation of proof assistants implementing various logical frameworks
    - ⋆ Classical logics/set theory (Mizar, Isabelle)

CLASP centre for linguistic theory and studies in probability

Proof Assistants: A brief history

- Proof-assistant technology has gone a long way since then
  - Proliferation of proof assistants implementing various logical frameworks
    - Classical logics/set theory (Mizar, Isabelle)
    - Constructive Type Theories (MTTs, Coq, Lego, Plastic, Agda among other things)
  - Important verified proofs

CLASP centre for linguistic theory and studies in probability

Proof Assistants: A brief history

- Proof-assistant technology has gone a long way since then
  - Proliferation of proof assistants implementing various logical frameworks
    - Classical logics/set theory (Mizar, Isabelle)
    - Constructive Type Theories (MTTs, Coq, Lego, Plastic, Agda among other things)
  - Important verified proofs
    - Four Colour Theorem ([Gonthier(2005)], Coq)

CLASP centre for linguistic theory and studies in probability

Proof Assistants: A brief history

- Proof-assistant technology has gone a long way since then
  - Proliferation of proof assistants implementing various logical frameworks
    - Classical logics/set theory (Mizar, Isabelle)
    - Constructive Type Theories (MTTs, Coq, Lego, Plastic, Agda among other things)
  - Important verified proofs
    - Four Colour Theorem ([Gonthier(2005)], Coq)
    - Jordan curve theorem ([Korniłowicz(2007), Hales(2007)], Mizar and HOL respectively)

CLASP centre for linguistic theory and studies in probability

Proof Assistants: A brief history

- Proof-assistant technology has gone a long way since then
  - Proliferation of proof assistants implementing various logical frameworks
    - Classical logics/set theory (Mizar, Isabelle)
    - Constructive Type Theories (MTTs, Coq, Lego, Plastic, Agda among other things)
  - Important verified proofs
    - Four Colour Theorem ([Gonthier(2005)], Coq)
    - Jordan curve theorem ([Korniłowicz(2007), Hales(2007)], Mizar and HOL respectively)
    - The prime number theorem ([Avigad et al.(2007)Avigad, Donnelly, Gray, and Raff], Isabelle)

CLASP centre for linguistic theory and studies in probability

Proof Assistants: A brief history

- Proof-assistant technology has gone a long way since then
  - Proliferation of proof assistants implementing various logical frameworks
    - Classical logics/set theory (Mizar, Isabelle)
    - Constructive Type Theories (MTTs, Coq, Lego, Plastic, Agda among other things)
  - Important verified proofs
    - Four Colour Theorem ([Gonthier(2005)], Coq)
    - Jordan curve theorem ([Korniłowicz(2007), Hales(2007)], Mizar and HOL respectively)
    - The prime number theorem ([Avigad et al.(2007)Avigad, Donnelly, Gray, and Raff], Isabelle)
    - Feit-Thompson theorem ([Gonthier et al.(2013)Gonthier, Asperti, Avigad, Bertot, Cohen, Garillot, I Coq (170.000 lines of code!))

CLASP centre for linguistic theory and studies in probability

The Coq proof assistant

- INRIA project
  - Started in 1984 as an implementation of Coquand's Calculus of Constructions (CoC)
  - Extension to the Calculus of Inductive Constructions (CiC) in 1991

The Coq proof assistant

- INRIA project
  - Started in 1984 as an implementation of Coquand's Calculus of Constructions (CoC)
  - Extension to the Calculus of Inductive Constructions (CiC) in 1991
  - Coq offers a program specification and mathematical higher-level language called *Gallina* based on CiC

CLASP centre for linguistic theory and studies in probability

The Coq proof assistant

- INRIA project
  - Started in 1984 as an implementation of Coquand's Calculus of Constructions (CoC)
  - Extension to the Calculus of Inductive Constructions (CiC) in 1991
  - Coq offers a program specification and mathematical higher-level language called *Gallina* based on CiC
  - CiC combines both expressive higher-order logic as well as a richly typed functional programming language
- Winner of the 2013 ACM software system award
- A collection of 100 mathematical theorems proven in Coq: http://perso.ens-lyon.fr/jeanmarie.madiot/coq100/

CLASP centre for linguistic theory and studies in probability

The Coq proof assistant

- An ideal tool for formal verification

centre for
linguistic theory
and studies in probability

The Coq proof assistant

- An ideal tool for formal verification
  - Powerful and expressive logical language

CLASP centre for linguistic theory and studies in probability

The Coq proof assistant

- An ideal tool for formal verification
  - Powerful and expressive logical language
  - Consistent embedded logic

CLASP centre for linguistic theory and studies in probability

The Coq proof assistant

- An ideal tool for formal verification
  - Powerful and expressive logical language
  - Consistent embedded logic
  - Built-in proof tactics that help in the development of proofs

CLASP centre for linguistic theory and studies in probability

The Coq proof assistant

- An ideal tool for formal verification
  - Powerful and expressive logical language
  - Consistent embedded logic
  - Built-in proof tactics that help in the development of proofs
  - Equipped with libraries for efficient arithmetics in N, Z and Q, libraries about lists, finite sets and finite maps, libraries on abstract sets, relations and classical analysis among others

CLASP centre for linguistic theory and studies in probability

The Coq proof assistant

- An ideal tool for formal verification
  - Powerful and expressive logical language
  - Consistent embedded logic
  - Built-in proof tactics that help in the development of proofs
  - Equipped with libraries for efficient arithmetics in N, Z and Q, libraries about lists, finite sets and finite maps, libraries on abstract sets, relations and classical analysis among others
  - Built-in automated tactics that can help in the automation of all or part of the proof process

CLASP centre for linguistic theory and studies in probability

The Coq proof assistant

- An ideal tool for formal verification
  - Powerful and expressive logical language
  - Consistent embedded logic
  - Built-in proof tactics that help in the development of proofs
  - Equipped with libraries for efficient arithmetics in N, Z and Q, libraries about lists, finite sets and finite maps, libraries on abstract sets, relations and classical analysis among others
  - Built-in automated tactics that can help in the automation of all or part of the proof process
  - Allows the definition of new proof tactics by the user
    - ⋆ The user can develop automated tactics by using this feature

Installing Coq

- Easy to install (http://coq.inria.fr/download)
- Use the installer or can get Coq via Macports or HomeBrew
- There is an interface for emacs, Proof General (provides support for a number of proof assistants incl. Coq, Isabelle, HOL among others)
  - Get Proof-general here: `https://proofgeneral.github.io/`
  - Customize your emacs .init file according to the instructions in there

Relevance to NL semantics?

- Ok, how is this relevant to NL semantics?

centre for linguistic theory and studies in probability

Relevance to NL semantics?

- Ok, how is this relevant to NL semantics?
  - This is a valid question

CLASP centre for linguistic theory and studies in probability

Relevance to NL semantics?

- Ok, how is this relevant to NL semantics?
  - This is a valid question
- The way we see it. Three main points:

CLASP centre for linguistic theory and studies in probability

Relevance to NL semantics?

- Ok, how is this relevant to NL semantics?
  - This is a valid question
- The way we see it. Three main points:
  1. Proof assistants implement constructive type theories (e.g. Coq, Agda)

CLASP centre for linguistic theory and studies in probability

Relevance to NL semantics?

- Ok, how is this relevant to NL semantics?
  - This is a valid question
- The way we see it. Three main points:
  1. Proof assistants implement constructive type theories (e.g. Coq, Agda)
  2. Proof assistants are extremely powerful reasoning engines

Relevance to NL semantics?

- Ok, how is this relevant to NL semantics?
  - This is a valid question
- The way we see it. Three main points:
  1. Proof assistants implement constructive type theories (e.g. Coq, Agda)
  2. Proof assistants are extremely powerful reasoning engines
  3. Constructive type theories as an alternative language for formal semantics

CLASP centre for linguistic theory and studies in probability

Relevance to NL semantics?

- Ok, how is this relevant to NL semantics?
  - This is a valid question
- The way we see it. Three main points:
  1. Proof assistants implement constructive type theories (e.g. Coq, Agda)
  2. Proof assistants are extremely powerful reasoning engines
  3. Constructive type theories as an alternative language for formal semantics

CLASP centre for linguistic theory and studies in probability

Relevance to NL semantics?

- Given these three points, two main uses:

centre for linguistic theory and studies in probability

Relevance to NL semantics?

- Given these three points, two main uses:
    1. Natural Language Reasoners

CLASP centre for linguistic theory and studies in probability

Relevance to NL semantics?

- Given these three points, two main uses:
  1. Natural Language Reasoners
  2. Formal Checkers of the validity of semantic accounts

CLASP centre for linguistic theory and studies in probability

Relevance to NL semantics?

- Given these three points, two main uses:
  1. Natural Language Reasoners
  2. Formal Checkers of the validity of semantic accounts

centre for linguistic theory and studies in probability

An example of a simple proof

- Transitivity of implication: $(P \to Q) \to (Q \to R) \to (P \to R)$ (file Oslo_basics.v)
  - Important note: all examples discussed in the talk can be found here: **Github repository**
- What is needed before we get into proof mode
  - Declaring P, Q, R as propositional variables

    ```
    Variables P Q R:Prop.
    ```

  - With this declaration at hand, we can get into proof mode:

    ```
    Theorem trans: (P->Q)->(Q->R)->(P->R)
    ```

CLASP centre for linguistic theory and studies in probability

Proof tactics

- Some of the basic predefined Coq tactics (some examples in files Oslo_basics.v and Oslo_basics_1b.v)
  - Conjunction
    - *elim*: Use of the elimination rule
    - *split*: Splits the conjunction into two subgoals
    - Examples:

      ```
      Theorem conj: A/\B->A.
      Theorem conj: B/\(A/\C)->A/\B.
      ```
  - Disjunction
    - *Elim:* elimination rule
    - *Left,Right:* deals with one of the two disjuncts

      ```
      Theorem disj: (B\/(B\/C))/\(A\/B)->A\/B.
      ```
  - *Implication* ($\Rightarrow$) and *Forall*
    - *intro(s)*
    - *apply*

centre for linguistic theory and studies in probability

Proof tactics

- Existential
  - *exists t*: instantiates an existential variable
- Equality (=)
  - *reflexivity, symmetry, transitivity*: the usual properties of equality
  - *congruence*: used when a goal is solvable after a series of rewrites
  - *rewrite, subst*: rewrites an element of the equation with the other element of the equation. *Subst* is used when one of the terms is a variable

Proof tactics - exists, elim

- Imagine we want to prove the following:

```
Parameter P: nat -> Prop.
Theorem EXISTS: P 5-> exists n: nat, P n.
```

- We can use the tactic *exists* to substitute 5 for *n* and prove the goal (example in Oslo_basics_1b.v)

Formal Checking/Reasoning

- The idea is simple: formalize your semantic account and check that is correct (type-checks, correct entailments etc.)

centre for
linguistic theory
and studies in probability

Formal Checking/Reasoning

- The idea is simple: formalize your semantic account and check that is correct (type-checks, correct entailments etc.)
  - Coq speaks an MTT, so MTT accounts can be easily implemented without having to define the theory

Formal Checking/Reasoning

- The idea is simple: formalize your semantic account and check that is correct (type-checks, correct entailments etc.)
  - Coq speaks an MTT, so MTT accounts can be easily implemented without having to define the theory
  - In principle, all semantic theories can be implemented in Coq (the system is expressive enough)

CLASP centre for linguistic theory and studies in probability

Formal Checking/Reasoning

- The idea is simple: formalize your semantic account and check that is correct (type-checks, correct entailments etc.)
  - Coq speaks an MTT, so MTT accounts can be easily implemented without having to define the theory
  - In principle, all semantic theories can be implemented in Coq (the system is expressive enough)
    - ⋆ Shallow vs Deep embedding
- Some toy illustrative examples

Formal Checking/Reasoning

- The idea is simple: formalize your semantic account and check that is correct (type-checks, correct entailments etc.)
  - Coq speaks an MTT, so MTT accounts can be easily implemented without having to define the theory
  - In principle, all semantic theories can be implemented in Coq (the system is expressive enough)
    - Shallow vs Deep embedding
- Some toy illustrative examples
  - Montagovian Type-shifters (file type_shifters.v)
  - Some toy TTR examples (Records.v)
  - Retoré's dot-types and polymorphic conjunction (file MontagovianLexiconToy.v)

CLASP centre for linguistic theory and studies in probability

Formal Checking/Reasoning

- The idea is simple: formalize your semantic account and check that is correct (type-checks, correct entailments etc.)
  - Coq speaks an MTT, so MTT accounts can be easily implemented without having to define the theory
  - In principle, all semantic theories can be implemented in Coq (the system is expressive enough)
    - Shallow vs Deep embedding
- Some toy illustrative examples
  - Montagovian Type-shifters (file type_shifters.v)
  - Some toy TTR examples (Records.v)
  - Retoré's dot-types and polymorphic conjunction (file MontagovianLexiconToy.v)
  - Champollion's coordination paper (formalized part of the account as a test case to check correctness (it works!)) (file Champollion.v)

CLASP centre for linguistic theory and studies in probability

Formal Checking/Reasoning

- The idea is simple: formalize your semantic account and check that is correct (type-checks, correct entailments etc.)
  - Coq speaks an MTT, so MTT accounts can be easily implemented without having to define the theory
  - In principle, all semantic theories can be implemented in Coq (the system is expressive enough)
    - ★ Shallow vs Deep embedding
- Some toy illustrative examples
  - Montagovian Type-shifters (file type_shifters.v)
  - Some toy TTR examples (Records.v)
  - Retoré's dot-types and polymorphic conjunction (file MontagovianLexiconToy.v)
  - Champollion's coordination paper (formalized part of the account as a test case to check correctness (it works!)) (file Champollion.v)

CLASP centre for linguistic theory and studies in probability

Formal Checking/Reasoning

- Some more elaborate examples in MTTs
- MTT fragment that deals with entailment cases from the FraCaS (file MTT_fragment_for_FraCaS.v)
- Identity criteria (an older version of the theory presented on Monday but still works!) (file individuationnew.v)

CLASP centre for linguistic theory and studies in probability

The FraCoq system

- Bernardy and Chatzikyriakidis (2017) (file FraCoq.v)

centre for
linguistic theory
and studies in probability

The FraCoq system

- Bernardy and Chatzikyriakidis (2017) (file FraCoq.v)
  - Leverages two well-studied tools

CLASP centre for linguistic theory and studies in probability

The FraCoq system

- Bernardy and Chatzikyriakidis (2017) (file FraCoq.v)
  - Leverages two well-studied tools
    - Grammatical Framework [Ranta(2011)]

centre for
linguistic theory
and studies in probability

The FraCoq system

- Bernardy and Chatzikyriakidis (2017) (file FraCoq.v)
  - Leverages two well-studied tools
    - ⋆ Grammatical Framework [Ranta(2011)]
    - ⋆ Coq
  - Uses the GF FraCaS treebank

CLASP centre for linguistic theory and studies in probability

The FraCoq system

- Bernardy and Chatzikyriakidis (2017) (file FraCoq.v)
  - Leverages two well-studied tools
    - Grammatical Framework [Ranta(2011)]
    - Coq
  - Uses the GF FraCaS treebank
  - Then, every syntactic construction is mapped to a (compositional) semantics

CLASP centre for linguistic theory and studies in probability

The FraCoq system

- Bernardy and Chatzikyriakidis (2017) (file FraCoq.v)
  - Leverages two well-studied tools
    - Grammatical Framework [Ranta(2011)]
    - Coq
  - Uses the GF FraCaS treebank
  - Then, every syntactic construction is mapped to a (compositional) semantics
  - Reasoning is performed

The FraCoq system

- We use Ljünglof's FraCaS treebank and take these trees to their semantic counterparts
- The structure of the semantic representation
    1. Every GF syntactic category C is mapped to a Coq *Set*, noted $[\![C]\!]$.
    2. GF Functional types are mapped compositionally : $[\![A \to B]\!] = [\![A]\!] \to [\![B]\!]$
    3. Every GF syntactic construction function f:X is mapped to a function $[\![f]\!]$ such that $[\![f]\!] : [\![X]\!]$.
    4. GF function applications are mapped compositionally: $[\![t(u)]\!] = [\![t]\!]([\![u]\!])$.

CLASP centre for linguistic theory and studies in probability

The FraCoq system

- Sentences
  - We interpret sentences as propositions: $[\![S]\!] = \mathit{Prop}$.
  - To verify that P entails H, we prove the proposition $[\![P]\!] \to [\![H]\!]$.

```
Definition S := Prop.
```

- Common Nouns
  - Predicates over an abstract object type

```
Parameter object : Set.
Definition CN := object->Prop.
```

CLASP centre for linguistic theory and studies in probability

The FraCoq system

- Verb phrases
  - Parameterize over the *noun* of the subject (using Π types)

```
Definition VP := forall (subjectClass : CN)
object -> Prop.
```

CLASP centre for linguistic theory and studies in probability

The FraCoq system

- Adjectives
  - Functions from cn to cn (predicates to predicates)

    ```
    Definition A := CN -> CN.
    ```

  - Different classes of adjectives are captured using coercions (subtyping). All special classes of adjectives are subtypes of A.

    ```
    Definition IntersectiveA := object -> Prop.
    Definition wkIntersectiveA : IntersectiveA -> A
    := fun a cn (x:object) => a x /\ cn x.
    Coercion wkIntersectiveA : IntersectiveA >-> A.
    ```

  - Provision is made for intersective, subsective, privative and non-committal adjectives
- For a tutorial of how the system works, see here: **tutorial**

CLASP centre for linguistic theory and studies in probability

The FraCoq system

- Covers almost half of the suite (174 examples)

CLASP centre for linguistic theory and studies in probability

The FraCoq system

- Covers almost half of the suite (174 examples)
  1. Quantifiers
  2. Plurals
  3. Adjectives
  4. Comparatives
  5. Attitudes
  - Interesting to note that no complete run of the suite has been made yet!

CLASP centre for linguistic theory and studies in probability

The FraCoq system

- Covers almost half of the suite (174 examples)
  1. Quantifiers
  2. Plurals
  3. Adjectives
  4. Comparatives
  5. Attitudes
  - Interesting to note that no complete run of the suite has been made yet!

CLASP centre for linguistic theory and studies in probability

Some sample FraCaS examples

(1) A Swede won the Nobel Prize.
Every Swede is Scandinavian.
Did a Scandinavian win the Nobel prize? [Yes, FraCas 049]

centre for linguistic theory and studies in probability

Some sample FraCaS examples

(3) A Swede won the Nobel Prize.
Every Swede is Scandinavian.
Did a Scandinavian win the Nobel prize? [Yes, FraCas 049]

(4) No delegate finished the report on time..
Did any Scandinavian delegate finish the report on time? [No, FraCaS 070]

CLASP centre for linguistic theory and studies in probability

Evaluation

- The following table presents the results (Ours) as well as a comparison with the approach in Mineshima et al. (MINE, 2015), Bos (Nut, 2008) and Abzianidze (Langpro, 2015)

	Section	# examples	Ours	MINE	Nut	Langpro
1	Quantifiers	75	.96	.77	.53	.93 (44)
2	Plurals	33	.76	.67	.52	.73 (24)
3	Adjectives	22	.95	.68	.32	.73 (12)
4	Comparatives	31	.56	.48	.45	-
5	Attitudes	13	.85	.77	.46	.92 (9)
6	Total	174 (181)	0.83	0.69	0.50	0.85

- The approach by Abzianidze has an accuracy of 0.85 without involving the comparative section. If this section is taken out, our system's accuracy rises to 0.88

CLASP centre for linguistic theory and studies in probability

Conclusions/Future Research

- MTTs as foundational languages for formal

Conclusions/Future Research

- MTTs as foundational languages for formal
  - Formally, well-studied

centre for
linguistic theory
and studies in probability

Conclusions/Future Research

- MTTs as foundational languages for formal
  - Formally, well-studied
  - Expressively adequate

CLASP centre for linguistic theory and studies in probability

Conclusions/Future Research

- MTTs as foundational languages for formal
  - Formally, well-studied
  - Expressively adequate
  - Proof-theoretically specified, supporting effecting reasoning

CLASP centre for linguistic theory and studies in probability

Conclusions/Future Research

- MTTs as foundational languages for formal
  - Formally, well-studied
  - Expressively adequate
  - Proof-theoretically specified, supporting effecting reasoning
- State of maturity of both MTT semantics and proof assistant technology
  - Use proof assistant technology and MTTs for formal verification and inference

CLASP centre for linguistic theory and studies in probability

N.G. de Bruijn.
A survey of the project AUTOMATH.
In J. Hindley and J. Seldin, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980.

Georges Gonthier.
A computer-checked proof of the Four Colour Theorem.
2005.
URL
http://research.microsoft.com/~{}gonthier/4colproof.pdf.

Artur Korniłowicz.
A proof of the jordan curve theorem via the brouwer fixed point theorem.
2007.

Thomas C Hales.
The jordan curve theorem, formally and informally.
American Mathematical Monthly, 114(10):882–894, 2007.

CLASP centre for linguistic theory and studies in probability

Jeremy Avigad, Kevin Donnelly, David Gray, and Paul Raff.
A formally verified proof of the prime number theorem.
ACM Transactions on Computational Logic (TOCL), 9(1):2, 2007.

Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell OConnor, Sidi Ould Biha, et al.
A machine-checked proof of the odd order theorem.
In *Interactive Theorem Proving*, pages 163–179. Springer, 2013.

A. Ranta.
Grammatical Framework: Programming with Multilingual Grammar.
CSLI Publications, 2011.