Coq for natural language semantics day 2: Constructive Type Theories for NL Semantics and General Coq use for NL semantics

Stergios Chatzikyriakidis

November 23, 2017

Coq and Constructive TT semantics

- Coq implements a Dependent Type Theory
  - Even though, it is powerful enough to encode any of our favorite semantic theories, doing dependent type semantics, comes for free so to say
  - There is a very close correspondence between the systems used by authors like Ranta [Ranta(1994), Bekki(2014), Chatzikyriakidis and Luo(2017), Chatzikyriakidis and Luo(2013)] and the system implemented by Coq
- So, first, let us see some DTTs for NL semantics!

CLASP centre for linguistic theory and studies in probability

A brief intro to Modern Type Theories for NL (MTTs)

- Type Theories within the tradition of Martin Löf
  - In linguistics, this work has been initiated by pioneering work of Ranta [Ranta(1994)]
- Here, we use one such MTT, UTT, used by Luo and colleagues [Chatzikyriakidis and Luo(2014), Chatzikyriakidis and Luo(2013), Chatzikyriakidis and Luo(2017)] to the study of linguistic semantics
  - Two characteristics that are promising in using MTTs as an alternative formal semantics language:
    - ★ Consistent internal logic according to the propositions-as-types principle
    - ★ Rich type structures

CLASP centre for linguistic theory and studies in probability

Intro to MTTs-Type Many Sortedness and Rich Typing

- Many-sortedness of types
  - Use of many types to interpret CNs, *man* and *table*
  - CNs are interpreted as Types rather than as predicates $(e \rightarrow t)$
- Use of Dependent Types Π and Σ
  - When A is a type and P is a predicate over A, $\Pi x{:}A.P(x)$ is the dependent function type that stands for the universally quantified proposition $\forall x{:}A.P(x)$
  - Π for polymorphic typing: $\Pi A{:}CN.(A \rightarrow Prop) \rightarrow (A \rightarrow Prop)$
  - A is a type and B is an A-indexed family of types, then $\Sigma x{:}A.B(x)$, is a type, consisting of pairs (a, b) such that a is of type A and b is of type $B(a)$.
  - Adjectival modification as involving Σ types (Ranta, 1994; Luo, 2010): $[\![heavy\_book]\!] = \Sigma x : book.heavy(x)$

Intro to MTTs-Subtyping

- Coercive subtyping
  - Can be seen as an abbreviation mechanism
    - A is a (proper) subtype of B ($A < B$) if there is a unique implicit coercion c from type A to type B
    - An object a of type A can be used in any context $\mathfrak{C}_B[\_]$ that expects an object of type B: $\mathfrak{C}_B[a]$ is legal (well-typed) and equal to $\mathfrak{C}_B[c(a)]$.
    - For example assuming $man < human$, $John : man$ and $shout : human \rightarrow Prop$, then $shout(John)$ is well-typed.

CLASP centre for linguistic theory and studies in probability

Intro to MTTs-Universes

- Universes
  - A universe is a collection of (the names of) types into a type (Martin Löf, 1984).
  - Universes can help semantic representations. For example, one may use the universe $\mathrm{CN} : \mathit{Type}$ of all common noun interpretations and, for each type A that interprets a common noun, there is a name $\overline{A}$ in CN. For example,

    $$\overline{\mathit{man}} : \mathrm{CN} \quad \text{and} \quad T_{\mathrm{CN}}(\overline{\mathit{man}}) = \mathit{man}.$$

    In practice, we do not distinguish a type in CN and its name by omitting the overlines and the operator T_{CN} by simply writing, for instance, $\mathit{man} : \mathit{CN}$.

Modification

- Adjectival modification as involving Σ types, in line with Ranta (1994)
  - Intersective adjectives as simple predicate types and subsective as polymorphic types over the CN universe:
    - ★ $[\![black]\!] : Object \rightarrow Prop$
    - ★ $[\![small]\!] : \Pi A{:}CN.A \rightarrow Prop$ (the A argument is implicit)
    - ★ For *black man*, we have: $\Sigma m{:}\,[\![man]\!]\,.\,[\![black]\!](m) < [\![man]\!]$ (via π_1)
    - ★ $< \Sigma m{:}\,[\![human]\!]\,.\,[\![black]\!](m)$ (via subtyping propagation)
    - ★ $< [\![human]\!]$ (via π_1)
  - For *small man*:
    - ★ $\Sigma m{:}\,[\![man]\!]\,.\,[\![small]\!]\,[\![man]\!](m) < [\![man]\!]$ (via π_1)
    - ★ BUT NOT:
      $\Sigma m{:}\,[\![man]\!]\,.\,[\![small]\!]\,[\![man]\!](m) < \Sigma m{:}\,[\![animal]\!]\,.\,[\![small]\!]\,[\![man]\!](m)$
    - ★ Many instances of small: $small([\![man]\!])$ is of type $[\![man]\!] \rightarrow Prop$, $small([\![animal]\!])$ is of type $[\![animal]\!] \rightarrow Prop$

CLASP centre for linguistic theory and studies in probability

Adjectival Modification/More Advanced Issues

- Privative adjectives like *fake*
  - We follow Partee (2007) and argue that privative adjectives are actually subsective adjectives which operate on CNs with extended denotations
    - ⋆ For exaple, the denotation of *fur* is expanded to include both *real* and *fake* furs:

      (1) I don't care whether that fur is fake fur or real fur.

      (2) I don't care whether that fur is fake or real.

    - ⋆ $G = G_R + G_F$ with $inl(r){:}G_R$ and $inl(f){:}G_F$
    - ⋆ Injections as coercions: $G_R <_{inl} G \texttt{and} G_F <_{inr} G$ and we define:
      $real\_gun(inl(r)) = True \quad and \quad real\_gun(inr(f)) = False$;
      $fake\_gun(inl(r)) = False \quad and \quad fake\_gun(inr(f)) = True$.
- Non-committal adjectives like *alleged*
  - Use of TT contexts representing allegation contexts, in the same vein as Ranta for beliefs [Ranta(1994)]
    $[\![alleged\ N]\!] = \Sigma p{:}Human.\ A(p, A_N)$

CLASP centre for linguistic theory and studies in probability

Adjectival Modification/Multidimensional Adjectives

- Quantification across different dimensions
  - E.g. to be considered *healthy* one has to be healthy w.r.t a number of dimensions (blood pressure, cholesterol etc.)
    - Involves universal quantification over dimensions
  - The antonyms of these type of multidimensional adjectives existentially quantify over dimensions
    - For one to be sick, only one dimension is needed
- We formulate this idea by Sassoon (2008) as follows:
  - We define an inductive type *health*
    - $Inductive\ [\![Health]\!] : D := Heart \mid Blood\_pressure \mid Cholesterol$
  - Then we define:
    - $[\![healthy]\!] = \lambda x{:}Human.\forall h{:}Health.Healthy(h)(x)$
    - $[\![sick]\!] = \lambda x{:}Human.\neg(\forall h{:}Health.Healthy(h)(x))$

CLASP centre for linguistic theory and studies in probability

Adverbial Modification

- Typing issues: How are we going to type adverbs in a many sorted TT?
  - Two basic types
    - ★ Sentence adverbs: $Prop \rightarrow Prop$
    - ★ VP-adverbs: $\Pi A{:}CN.(A \rightarrow Prop) \rightarrow (A \rightarrow Prop)$
    - ★ Polymorphic type: Depends on the choice of A
    - ★ Given that we are talking about predicates, depends on the choice of the argument
    - ★ $[\![walk]\!] : Animal \rightarrow Prop \Rightarrow [\![ADV]\!]\,[\![walk]\!] : (Animal \rightarrow Prop)$
    - ★ $[\![drive]\!] : Human \rightarrow Prop \Rightarrow [\![ADV]\!]\,[\![drive]\!] : (Human \rightarrow Prop)$

Adverbial Modification: Veridicality

- Veridical Adverbials when applied to their argument, imply their argument
  - John opened the door quickly $\Rightarrow$ John opened the door
  - Fortunately, John is an idiot $\Rightarrow$ John is an idiot
- Non-veridical adverbs do not have this property
  - John allegedly opened the door $\not\Rightarrow$ John opened the door

CLASP centre for linguistic theory and studies in probability

Adverbial Modification: Veridicality

- We can use a similar organizational strategy as in the case with adjectives
  - Define an auxiliary object first, define the adverb as its first projection
    - ⋆ $[\![VER_{Prop}]\!] : \Pi v : Prop.\ \Sigma p : Prop.p \supset v$
    - ⋆ $[\![ADV_{ver-Prop}]\!] = \lambda v : Prop.\ \pi_1(VER_{Prop}(v))$
  - An adverb like *fortunately* will be defined:
  - $[\![fortunately]\!] = \lambda v : Prop.\ \pi_1(VER_{Prop}(v))$
- Consider the following: Fortunately, John went $\Longrightarrow$ John went
  - The second component of $(VER_{Prop}(v))$ is a proof of $[\![fortunately]\!](v) \Rightarrow v$
  - Taking v to be $[\![John\ went]\!]$, the inference follows

CLASP centre for linguistic theory and studies in probability

Adverbial Modification: Intensional/domain adverbials

- Use of TT contexts in this case as well
  - $[\![allegedly]\!] = \lambda P : Prop.\ \exists p{:}\ [\![human]\!], B_p(P)$
  - Someone has alleged that P (P is an agent's belief context (Chatzikyriakidis 2014; Chatzikyriakidis and Luo 2015))
- Introduction of intenTional contexts: Contexts including the intentions (rather than the beliefs) of an agent. We can use this idea for adverbs like intentionally:
  - $[\![Intentionally]\!] = \lambda x : [\![human]\!] .\lambda P : [\![human]\!] \rightarrow Prop.\ I_x(P(x)) \wedge \Gamma(P(x))$
- Domain adverbs, e.g. *botanically, mathematically*
  - $[\![botanically]\!] = \lambda P : Prop.\Gamma_B P$
- Intensionality without possible worlds

CLASP centre for linguistic theory and studies in probability

Anaphora: Donkey Sentences

- Ranta's influential account: using Σ projections for donkey anaphora
  - Every man who owns a donkey beats it
    - ★ Man who owns a donkey: $(\Sigma x{:}Man)(\Sigma y{:}Donkey)(\mathrm{own}(x,y))$
    - ★ Every man who owns a donkey beats it: $\Pi z{:}(\Sigma x{:}Man)(\Sigma y{:}Donkey)(\mathrm{own}(x,y))(\mathrm{beat}(\pi_1(z),\pi_1(\pi_2(z))))$

Co-predication and Individuation

- The term copredication refers to the phenomenon in which more than one predicate, representing a verb or an adjective and requiring different types of arguments, are used in coordination and applied to the 'same' CN argument.

  (3) John picked up and mastered the book.

- *Book* seems to involve both a physical and an informational component
- Interaction with Quantification

  (4) John mastered three books. (three info objects but might be one physical object, e.g. a big volume)

  (5) John picked up three books. (three physical objects but could be copies of the same book)

CLASP centre for linguistic theory and studies in probability

Co-predication and Individuation

- In co-predication, we need double distinctness

(6) John picked up and mastered three books. (three physical objects and three info objects)

- We need a theory of individuation that can take care of this
- Treating CNs as setoids rather than as simple types
  - A type plus an equivalence relation, i.e. a pair $(A_N, =_N)$
  - For example: $[\![human]\!] = (Human, =_h)$
  - CNs can inherit their identity criteria from other CNs. For example, type *Man* inherits its IC from type *Human*
    - ★ $m_1, m_2 : Man$, $m_1 =_m m_2$ is defined as $\pi_1(m_1) =_h \pi_1(m_2)$

Co-predication and Individuation

- Dot.types are complex types formed out of individual types not sharing components
  - $\textsc{Phy} \bullet \textsc{Info}$ (see [Xue and Luo(2012)] for the formal details of forming dot.types)
  - Dot.types are subtypes of their individual parts ($\textsc{Phy} \bullet \textsc{Info} < \textsc{Phy},\textsc{Info}$)
  - Book is a subtype of $\textsc{Phy} \bullet \textsc{Info}$ ($\mathit{Book} < \textsc{Phy} \bullet \textsc{Info}$)

CLASP centre for linguistic theory and studies in probability

Co-predication and Individuation

- Proper semantics for quantifier *three* in co-predication
  - Let $N = (A_N, =_N)$, $N_1 = (A_{N_1}, =_{N_1})$ and $N_2 = (A_{N_2}, =_{N_2})$ be CNs, and $A_N \leq A_{N_1} \bullet A_{N_2}$ and $P : A_{N_1} \bullet A_{N_2} \rightarrow Prop$
  - $Three^{\bullet}(N, N_1, N_2, P) =$ $\exists x, y, z{:}A_N.\ D[N_1](x, y, z)\ \&\ D[N_2](x, y, z)\ \&\ P(x) \& P(y) \& P(z)$, where $D[N_1](x, y, z) = (x \neq_{N_1} y) \& (y \neq_{N_1} z) \& (x \neq_{N_1} z)$ and similarly for $D[N_2](x, y, z)$
  - $Three^{\bullet}([\![book]\!], \textsc{Phy}, \textsc{Info}, [\![pickup\ and\ master]\!](j)) =$ $\exists x, y, z{:}\textsc{Phy} \bullet \textsc{Info}.\ D[\text{PHY}](x, y, z)\ \&\ D[\textsc{info}](x, y, z)\ \&$ $P(x) \& P(y) \& P(z) \& M(x) \& M(y) \& M(z)$

CLASP centre for linguistic theory and studies in probability

MTT semantics in Coq

- Encoding MTT semantics in Coq

MTT semantics in Coq

- Encoding MTT semantics in Coq
  - Coq is a natural toolkit to perform such a task
    - ★ The type theory implemented in Coq is an MTT in effect
    - ★ What we need, is a way to encode the various assumptions as regards linguistic semantics and then reason about them

CLASP centre for linguistic theory and studies in probability

MTT semantics in Coq

- Encoding MTT semantics based on theoretical work using Type Theory with Coercive Subtyping in Coq
  - Coq is a natural toolkit to perform such a task
    - ★ The type theory implemented in Coq is quite close to Type Theory with Coercive Subtyping
    - ★ Thus, the TT does not need to be implemented!
    - ★ What we need, is a way to encode the various assumptions as regards linguistic semantics and then reason about them

centre for linguistic theory and studies in probability

The CN universe

- Common nous in MTTs are seen as types rather than predicates
- Zhaohui Luo proposed the introduction of a universe of CN interpretations (Luo 2011, 2012 among others)
  - A collection of the names of types that interpret common nouns
  - Coq does not support universe construction
    - ★ Only the pre-defined universes can be used
    - ★ In this sense, we define *CN* to be Coq's pre-defined *Set* universe

```
Definition CN:= Set.
Parameters Man Human Animal Object:CN
```

Subtyping relations

- In order for type many-sortedness to have any advantages over more coarse grained typing (like the *e* typing in MG), a subtyping mechanism is needed
  - We have already seen the use of coercive subtyping as an adequate subtyping mechanism
  - Coq uses a similar mechanism (albeit with minor formal differences)
  - Subtyping in Coq is also based on the notion of coercion.
    - ★ An example is shown below:

```
Axiom MH: Man->Human. Coercion MH: Man>->Human.
Axiom HA: Human->Animal. Coercion HA: Human>->Animal.
```

Types for verbs

- The type of propositions is identified with *Prop*.
  - Verbs are function types returning a *Prop* type once one or more (depending on valency) arguments have been provided
    - ★ However, given type many sortedness the arguments needed for individual verbs will be dependent on the specific verb in each case
    - ★ Thus, Walk will be specified as $Animal \rightarrow Prop$ while fall as $Object \rightarrow Prop$

```
Parameter walk: Animal-> Prop.
Parameter fall: Object-> Prop.
```

CLASP centre for linguistic theory and studies in probability

Quantifiers, adjectives, adverbs

- Following work by Luo (2011, 2012) and Chatzikyriakidis and Luo (2013a,b,2014), quantifiers are given an inductive type, taking an $A{:}CN$ argument and returns the type $(A \rightarrow Prop) \rightarrow Prop$
- Adjectives are defined as simple predicates.
- VP adverbs are defined as predicate modifiers extending over the universe CN, while sentence adverbs as functions from propositions to propositions

```
Parameter some: forall A:CN, (A->Prop)->Prop
Parameter handsome: Human -> Prop
Parameter slowly: forall A:CN, (A->Prop)->(A->Prop).
Parameter fortunately: Prop ->Prop.
```

CLASP centre for linguistic theory and studies in probability

Quantifiers, adjectives, adverbs

- More must be said about the lexical semantics of all these categories.
- For example, in the case of *some* the following will be assumed
  - Same typing but has further information on the lexical semantics of some (i.e. existential quantification)

    ```
    Definition some:= fun A:CN=> fun P:(A->Prop)=> exists x: A,
    P(x).
    ```

  - More will be said about the lexical semantics as we proceed

CLASP centre for linguistic theory and studies in probability

Adjectival modification using dependent record types

- Intersective and subsective adjectival modification have been treated as involving Σ types.
- This is the analysis we follow here
  - We however follow Luo (2012) and use dependent record types instead of Σ types (which are equivalent)
    - ★ The first projection is declared as a coercion
    - ★ Thus, for *handsome man*, we get the inference *man*

```
Record handsomeman:CN:=mkhandsomeman{ c :>Man;C: handsome c }
```

CLASP centre for linguistic theory and studies in probability

Reasoning with NL

- As soon as NL categories are defined, Coq can be used to reason about them
  - In effect, we can view a valid NLI as a theorem
    - ★ Thus, we formulate NLIs as theorems
    - ★ The antecedent and consequent must be of type *Prop* in order to be used in proof mode
    - ★ Thus, the first can be formulated as a theorem, but not the second:

```
Theorem EX:(walk) John-> some Man (walk).
Theorem WA:walk -> drive.
```

Reasoning with NL

- The same tactics that can be used in proving mathematical theorems are used for NL reasoning
  - The aim is to predict correct NLIs while avoiding unwanted inferences
    - For example, given the semantics for quantifier *some*, one can formulate the following theorem and further try to prove it

```
Theorem EX:  (walk) John-> some Man (walk).
```

An NLI example

- One of the inferences we should be able to get when a proper name acts as an argument of the verb is one where an element of the same type as the proper name acts as the argument of the same verb
  - Basically, from a sentence like *John walks*, we should infer that *a man walks*
  - We formulate the theorem

```
Theorem EX:  (walk) John-> some Man (walk).
```

  - We unfold the definition for *some* and use *intro*

```
EX < intro.
1 subgoal
H : walk John
=============================
exists x : Man, walk x
```

  - We use the exists tactic to substitute x for *John*. Using *assumption* the theorem is proven

CLASP centre for linguistic theory and studies in probability

An NLI example

- To the contrary, we should not be able to prove the opposite

  ```
  Theorem EX: some Man (walk) -> (walk) John.
  ```

- Indeed, no proof can be found in this case.
  - We unfold *some* and use *intro*

    ```
    EX < intro.
    1 subgoal
    H : exists x : Man, walk x
    =============================
    walk John
    ```

  - From this point on, we can use any of the *elim, induction, case* tactics but at the end we reach a dead end

    ```
    EX < intro.
    1 subgoal
    H : exists x : Man, walk x
    x : Man
    H0 : walk x
    ```

CLASP centre for linguistic theory and studies in probability

Automation?

- From a theoretical point a view, having a system that can reason about NL semantics in such a straightforward way is already something
  - From the practical side however, in order to develop something like this into a more practical device, automation needs to be possible
    - ★ For the simple case we have been discussing, automation is possible once we unfold the definition for *some*
    - ★ The tactic *eauto* will solve the theorem in one step

```
EX< unfold some.
1 subgoal
============================
walk John -> exists x : Man, walk x
EX < eauto.
Proof completed.
```

    - ★ Still, this is not yet full automation. What can we do?

CLASP centre for linguistic theory and studies in probability

The tactic language Ltac

- Besides the predefined tactics offered by Coq or these imported by various Coq packages, Coq offers a way for the user to define his own proof-tactics
  - This is achieved by Ltac
    - ⋆ A programming language inside Coq that can be used to build new user-defined tactics
    - ⋆ Using Ltac we can define the following tactic that will fully automate the example we are interested in

      ```
      Ltac EXTAC:= cbv; eauto.
      ```

    - ⋆ The *cbv* tactic performs all possible reductions using δ, β, ζ and ι
    - ⋆ In our case, δ reduction is applied first unfolding the definition and then β reduction
    - ⋆ The tactic *compute* that embodies *cbv* can also be used

CLASP centre for linguistic theory and studies in probability

The tactic language Ltac

- What we need is automated tactics that work for a range of examples and not tactics that work on a case by case basis
  - For example, the tactic *EXTAC*, though simple enough, has the power to automate quite a few inferences
    - One can further prove:

      ```
      all Man (walk)->walk John.
      all Man (walk)->walk John->some Man (walk).
      ```

    - Also cases where subtyping is involved, like the following:

      ```
      all Animal (walk)->walk John.
      ```

CLASP centre for linguistic theory and studies in probability

The tactic language Ltac

- However, the following cannot be proven with *EXTAC*:

```
all Man (walk)-> some Man (walk).
```

  - We get the following error:

```
Coq < Theorem EX2: all Man (walk) -> some Man (walk).
1 subgoal
============================
all Man (fun x : Man => walk x) -> some Man (fun x : Man =>
walk x)
EX2< EXTAC.
No more subgoals but non-instantiated existential variables
Existential 1 = ?463 : [H : forall x : Man, walk x |- Man]
```

  - This means that *eauto* did not manage to instantiate an existential, which was then eliminated by a computation
  - The solution is to instantiate the value "manually"

CLASP centre for linguistic theory and studies in probability

The tactic language Ltac

- For example, we can substitute x for *John* using the *exists* tactic
  - We can define a similar tactic that instantiates the variable using *exists* and then calls *EXTAC*

    ```
    Ltac EXTAC1 x:= cbv; try exists x;EXTAC.
    ```
  - The command *try* + *tactic*, tries to perform the tactic, and if it fails, it moves on
  - This will suffice to prove automatically all the NL examples we have considered so far

NL Inference

- The task of determining whether an NL hypothesis H can be deduced from an NL premise P
- A central task in both theoretical and computational semantics
  - As Cooper et al. (1996) aptly put it: "inferential ability is not only a central manifestation of semantic competence but is in fact centrally constitutive of it"
    - ★ Inferential ability as the best way to test the semantic adequacy of NLP systems
    - ★ An adequate NLP system should be able to predict correct inferences like (1)-(3) without further generating unwanted inferences like (4) or (5)

(7) John walks and Mary talks $\Rightarrow$ Mary talks

(8) Some men run fast $\Rightarrow$ Some men run

(9) John walks $\Rightarrow$ Some one walks

(10) John walks and Mary talks $\nRightarrow$ If John walks, Mary talks

(11) No men run fast $\nRightarrow$ No men run

CLASP centre for linguistic theory and studies in probability

NL inference platforms: FraCas

- Platforms for NLI - The Fracas test suite
  - Came out of the FraCas consortium, a large collaboration in the 90's to create resources for computational semantics
  - Contains 349 NLIs, with one or more premises
    - ★ Categorized by semantic section: e.g. Quantifiers, adjectives temporal reference etc.
    - ★ A number of premises (usually single premised), followed by the hypothesis in the form of a question

CLASP centre for linguistic theory and studies in probability

The FraCas test suite

- Typical examples

(12) No delegate finished the report.
Did any delegate finished the report on time? [No] (quantifier section)

(13) Either Smith, Jones or Anderson signed the contract. Did John sign the contract? [UNK] (plurals)

(14) Dumbo is a large animal. Is Dumbo a small animal? [NO] (adjectives)

(15) Smith believed that ITEL had won the contract in 1992. Did ITEL win the contract in 1992? [UNK] (Attitudes)

CLASP centre for linguistic theory and studies in probability

Formulating the examples

- As already said, the examples involve a number of premises, followed by a question (h).
  - We reformulate the examples as involving declarative forms in Coq (this is a usual approach, at least with deep approaches)
    - In cases of *yes* in the FraCas test suite, we formulate a declarative hypothesis as following from the premise
    - In cases of *no*, we formulate the negation of a declarative hypothesis as following from the premise
    - In cases of *UNK*, for both the positive and the negated h, no proof should be found. If it is, we overgenerate inferences we do no want

Formulating the examples

- A *YES* example

(16) A Swede won the Nobel Prize.
Every Swede is Scandinavian.
Did a Scandinavian win the Nobel prize? [Yes, FraCas ex. 3.49]

```
Theorem SWE:(a Swede)(Won(a Nobel_Prize))->(a
Scandinavian)(Won(a Nobel_Prize)).
```

CLASP centre for linguistic theory and studies in probability

Formulating the examples

- A *NO* example

(17) A Swede did not win the Nobel Prize.
Every Swede is Scandinavian.
Did a Scandinavian win the Nobel prize? [No]

```
Theorem SWE:not((a Swede)(Won(a Nobel_Prize)))->not
(a Scandinavian)(Won(a Nobel_Prize)).
```

CLASP centre for linguistic theory and studies in probability

Formulating the examples

- An *UNK* example

(18) A Scandinavian won the Nobel Prize.
Every Swede is Scandinavian.
Did a Swede win the Nobel prize? [UNK, 3.65]

```
Theorem SWE:(a Scandinavian)(Won(a
Nobel_Prize))->(a Scandinavian)(Won(a Nobel_Prize))).

Theorem SWE:(a Scandinavian)(Won(a Nobel_Prize))->not((a
Scandinavian)(Won(a Nobel_Prize)))).
```

CLASP centre for linguistic theory and studies in probability

Evaluating against the FraCas test suite - Quantifier monotonicity

- This section involves inferences due to quantifier monotonicity
  - Upwards monotonicity on the first argument

(19) Some Irish delegates finished the survey on time
Did any delegates finish the survey on time? [YES]

★ Standard semantics for indefinites *some* and *any* (no presuppositions encoded)

```
Definition some:= fun A:CN=> fun P:A->Prop=> exists x:A, P(x)
```

CLASP centre for linguistic theory and studies in probability

Modification

- The examples we are dealing involve instance of adjectival modification
  - *Irishdelegate* in this case says that something is a *delegate* and furthermore *Irish*
  - We follow the Σ type treatment of adjectives. The first projection, $\pi 1$ is a coercion
  - We formulate it in Coq via means of dependent records

    ```
    Record Irishdelegate:CN:=mkIrishdelegate{c:> Man;
    C:Irish c}.
    ```

    - ★ With $Delegate : CN$ and $Irish : Object \to Prop$

CLASP centre for linguistic theory and studies in probability

Modification

- With these assumptions, nothing more is needed
  - The inference can be proven given the coercion of $\pi 1$
  - We formulate the theorem:

    ```
    Theorem IRISH: (some Irishdelegate(On_time(finish(the
    survey)))->(some Delegate)(On_time (finish(the survey))).

    compute.intro. elim H.intro.intro.exists x.auto.
    ```

  - Easy to prove. Subtyping does the work. Eliminating H and using intro we get an x:*Irishdelegate* that $On\_time(finish(thesurvey))(x))$ holds. Then, given subtyping, $Irishdelegate < Delegate$ via the first projection $\pi 1$, we also have that $On\_time(finish(thesurvey))(x))$ with x:*Delegate*

Subtyping again

- Other similar examples involve more direct cases of subtyping

(20) A Swede won a Nobel prize
Every Swede is a Scandinavian
Did a Scandinavian win a Nobel Prize? [YES, 3.49]

- The above is multi-premised, i.e. more than one premise
  - We first define *Swede* and *Scandinavian* as being of type *CN*
    - ⋆ This is a case of nominalized adjectives. At least in this guise they function as *CNs*. One can give a Unit type capturing both guises (more on Unit types later)
  - Note that both arguments of the verb are quantifiers
  - In order to accommodate this, we have two options
    - ⋆ The first option is to define *won* as a regular transitive (leaving tense aside for the moment since it does not play a role in proving the inference). Then, in order to perform functional application, given the higher types for quantifiers, one must directly translate to something like the following: $\exists x{:}Man, \exists y{:}Object, win(x)(y)$ (scope issues are not going to be discussed here)

centre for linguistic theory and studies in probability

Subtyping again

- Alternatively, one can follow the strategy employed by Montague and type shift the verb, thus lifting to type $((Object \to Prop) \to Prop) \to (Human \to Prop)$
  - We exemplify with both alternatives
  - The most important part in proving the inference is the declaration of a subtyping relation between *Swede* and *Scandinavian*, i.e. $Swede < Scandinavian$

```
Parameter Swede Scandinavian:CN
Won: Object->Human->Prop.
Won: ((Object->Prop)->Prop)->(Human->Prop)
Axiom ss: Swede->Scandinavian.
Coercion ss: Swede >-> Scandinavian.
Theorem SWEDE1: (a Swede)(won (a Nobel_Prize))->(a
Scandinavian (won(a Nobel_Prize)).
Theorem SWEDE2: exists x:Swede, exists y:Nobel_Prize, won(y)(x
```

CLASP centre for linguistic theory and studies in probability

The Swede example

- Formulation with the verb type-lifted

```
SWEDE22<Theorem SWEDE22: (a Swede)(Won2(a Nobel_Prize))->(
```

- We first use *cbv* to unfold the definitions. Then *intros*:

```
SWEDE22 < intros.
1 subgoal
H : exists x : Swede,
Won2 (fun P : Object -> Prop => exists x0: Nobel_Prize,
P x0) x
============================
exists x : Scandinavian,Won2 (fun P : Object -> Prop =>
exists x0 : Nobel_Prize, P x0) x
```

- Elimination (*elim*) can now be used followed by *eauto*. This suffices to prove the goal.

CLASP centre for linguistic theory and studies in probability

The Swede example

- Formulation with the verb regularly typed

```
Theorem SWEDE2: exists x:Swede, exists y:Nobel_Prize,
Won(y)(x)->exists x:Scandinavian, exists y:Nobel_Prize,
won(y)(x).
```

- There are no definitions to unfold and *intro* cannot apply.
- The natural solution is to be use *eauto*. However, this will give us the following error:

```
SWED < eauto.
No more subgoals but non-instantiated existential
variables:
Existential 1 = ?535 : [ |- Swede]
Existential 2 = ?536 : [ |- Nobel_Prize]
```

- This basically says that non-instantiated variables generated by *eapply* have been lost prior to instantiation (newer versions of Coq give a different phrasing for this!)

The Swede example

- The solution is to instantiate these variables
  - In this sense, we can introduce a number of variables (parameters) of type *Swedish* and a number of variables (parameters) of type *Nobel$_P$rize*
  - We can use one of these variables to instantiate the existentials
  - Starting with the proof, we basically instantiate both existentials
    - We then apply *eauto*, and the proof is completed (with *d*:*Swede* and *n*:*Scandinavian*.

```
SWED < exists d.
1 subgoal
============================
exists y : Nobel_Prize,
Won1 y d -> exists x : Scandinavian, exists y0 : Nobel_Prize,
Won1 y0 x
SWED < exists n.
1 subgoal
============================
Won1 n d -> exists x : Scandinavian, exists y : Nobel_Prize,
Won1 y x
SWED < eauto.
Proof completed.
```

CLASP centre for linguistic theory and studies in probability

A NO example

- Monotonicity on the first argument

(21) No delegate finished the report on time
Did any Scandinavian delegate finish the report on time?
[NO, FraCas 3.70]

- We try to prove the negation of the hypothesis

```
Theorem SCAN: (no delegate)(On_time Human(finish(the
report)))->not((some Scandinaviandelegate)(On_time Human
(finish(the report)))).
```

- We apply *cbv* to unfold the definitions followed by *intros*
- Then, the tactic jauto can be used to complete the proof
  - *jauto* is similar to *eauto* but can further open up conjunctions and existentials (what we need here)

An UNK example

- Again from the monotonicity on the first argument part of the suite

  (22) Some delegates finished the survey on time
  Did any Irish delegates finish the survey on time? [UNK, FraCas 3.71]

- Indeed the above cannot be proven given that the subtyping relation is from *Irishdelegate* $<$ *delegate* and not the other way around

CLASP centre for linguistic theory and studies in probability

An UNK example

- Basically, we end up with something like the following, and the proof cannot further continue

```
IRISH < AUTO.
H0 : exists x : delegate, On_time (finish (the survey)) x
x : delegate
H1 : On_time (finish (the survey)) x
============================
exists x0 : Irishdelegate,
On_time (finish (the survey)) (let (c0, _) := x0 in c0)
```

- Trying to substitute x for x_0 fails since the terms are of different types!

CLASP centre for linguistic theory and studies in probability

Monotonicity on the second argument

- In this section we find examples like the following:

(23) Some delegates finished the survey on time
Did some delegates finish the survey? [UNK, FraCas 3.71]

- The inference in these cases comes from the veridicality of VP-adverbials like *on time*
  - ▸ In order to capture this, we will have to see how VP veridical adverbials can be defined.
    - ★ In order to do this we first introduce the auxiliary object ADV_{ver}, for veridical VP-adverbials

(24) $ADV_{ver} : \Pi A : \textsc{cn}.\Pi v : A \rightarrow Prop.\ \Sigma p : A \rightarrow Prop.\forall x{:}A.p(x) \supset v(x)$

Monotonicity on the second argument

- Continued

(25) $ADV_{ver} : \Pi A : \textsc{cn}.\Pi v : A \rightarrow Prop.\ \Sigma p : A \rightarrow Prop.\forall x{:}A.p(x) \supset v(x)$

- Note that this is minimally different from $\forall A{:}CN, (A \rightarrow Prop) \rightarrow (A \rightarrow Prop)$, the only addition is the second part of the Σ specifying that in case $p(x)$ holds (the clause with the adverbial), then $V(x)$ also holds (the same sentence without the adverbial)
- Now, we define on time to be the first projection of this auxiliary object

(26) $on\ time = \lambda A : \textsc{cn}.\lambda v : A \rightarrow Prop.\ \pi_1(ADV(A, v))$

Monotonicity on the second argument

- We formulate these assumptions in Coq

```
Parameter ADV: forall (A : CN) (v : A -> Prop),sigT
(fun p : A -> Prop => forall x : A, p x -> v x).
Definition on_time:= fun A:CN=> fun v:A->Prop=> projT1
(ADV(v)).
```

- Let us see whether this definition suffices to prove the inference in 23.

```
IRISH2 < Theorem IRISH2: (some Delegate)(on_time
(finish(the Survey)))->(some Delegate)((finish
(the Survey))).
```

CLASP centre for linguistic theory and studies in probability

Monotonicity on the second argument

- Continued
- We unfold the definitions and use destruct for *ADV* (basically it unfolds the definition for *ADV*)

```
IRISH2 < cbv. intro. destruct ADV in H.
1 subgoal
x : Human -> Prop
f0 : forall x0 : Human, x x0 -> finish (the survey) x0
H : exists x0 : delegate, x x0
============================
exists x0 : delegate, finish (the survey) x0
```

- We apply *induction* or *elim* to *H*
  - The difference between the two is that *induction* will add the inductive hypotheses into the context while *elim* will not
  - Applying *eauto* after this, will complete the proof

A note on veridical adverbs/adverbials

- The way proposed to capture veridicality can be generalized to all VP adverbs/adverbials.
  - For example if one is interested in getting the veridicality inferences right, ignoring other issues pertaining to the lexical semantics of each adverbial, then the auxiliary object can be used in all these cases
  - Thus, adverbs like slowly, fast etc. can given a similar definition to *on_time*

(27) $adv_{ver} = \lambda A : \mathrm{CN}.\lambda v : A \rightarrow Prop.\ \pi_1(ADV_{ver}(A, v))$

  - A similar strategy can be used for veridical sentence adverbs. We first define an auxiliary object:

(28) $ADV_{Sver} : \Pi v : Prop.\ \Sigma p : Prop.p \supset V$

  - Then veridical sentence adverbs/adverbials like *fortunately, ironically* can be defined as:

(29) $adv_{Sver} = \lambda v : Prop.\ \pi_1(ADV_{Sver}(v))$

CLASP centre for linguistic theory and studies in probability

A note on veridical adverbs/adverbials

- We can check this in Coq

```
Coq < Theorem FORT: fortunately (walk John)-> walk John
1 subgoal
============================
fortunately (walk John) -> walk John
```

- We unfold the definitions and apply destruct to *ADVS*.

```
FORT < cbv. destruct ADVS.
x : Prop
w : x -> walk John
============================
x -> walk John
```

- Using *assumption* will complete the proof

CLASP centre for linguistic theory and studies in probability

Cases with more premises

- Example cases involving more than one premise

(30) Each European has the right to live in Europe
Every European is a person
Every person who has the right to live in Europe can travel freely within Europe
Can each European travel freely within Europe? [Yes, FraCas 3.20]

- For reasons of brevity some elements will be treated non-compositionally
    - But: only those that do not play any role in inference
    - Thus, to leave in Europe will be assumed as a single lexical item, since its treatment does not play any role in the specific inference
        - ★ Interesting case: Does *each european hat the right to live in Europe* imply *that each European has the right to live*?

CLASP centre for linguistic theory and studies in probability

Cases with more premises

- We assume *to_ live* to be a regular predicate. Then, we further assume that in_ Europe, freely and within_ Europe to be predicate modifiers
  - It is not difficult to give entries for prepositions *in* and *within* separately, but we will keep it simple in this case

```
Parameter in_Europe: forall A:CN, (A->Prop)->(A->Prop).
Parameter can: forall A:CN, (A->Prop)->(A->Prop).
Parameter travel: Object->Prop.
Parameter freely: forall A:CN, (A->Prop)->(A->Prop).
Parameter within_Europe: forall A:CN, (A->Prop)->(A->Prop).
```

Cases with more premises

- Let us formulate the example:
  - The first premise is straightforward
  - The second premise is encoded as a coercion and thus does not have to be present in the proof explicitly
  - The third premise is an implication relation (if a person... then)
  - Careful with the parentheses: the above two premises must imply the conclusion

    ```
    Theorem EUROPEAN: ((each European)(have
    (the righttoliveinEurope))/\forall x:person, ((have
    (the righttoliveinEurope)x)->Can (within_Europe(freely
    (travel)))x))->(each European)(Can (within_Europe(freely
    (travel)))).
    ```

  - Once formulated correctly, it is to prove
  - Using *cbv* to unfold the definitions, we can use *intuition* and complete the proof

CLASP centre for linguistic theory and studies in probability

Some toy cases of more mainstream semantics: Montague Semantics

- Note this is a shallow embedding of Montague Semantics
  - A Deep Embedding would involve defining the whole thing from scratch (the type system, the logic, models etc.)
  - This is far beyond what I could prepare for this course!
  - But even a shallow embedding will be very helpful for people trying to let us say want to type-check or verify their formal semantics accounts
  - Note of caution: I use the term Montague Semantics to refer to the whole tradition of Montagovian Semantics, so some things to be presented are not really Montague's own formulations!

CLASP centre for linguistic theory and studies in probability

Some Montague Semantics: Generalized Quantifiers

- First we need to have some set-theoretic definitions
  - This is file Set_theoretic_Defs.v
  - Let us see some definition
  - Set membership and Inclusion

```
Definition In (A:Ensemble) (x:e->Prop) : Prop := A x.
Definition Included (B C:Ensemble) : Prop :=
forall x:e->Prop, In B x -> In C x.
```

  - Both return a Proposition
    - ⋆ In takes an Ensemble (of type $(e \rightarrow Prop) \rightarrow Prop.$) and a predicate over e and returns a proposition which says the predicate is a member of the ensemble
    - ⋆ Inclusion is defined over two ensembles, and says that forall predicates over e, if they are members of the first ensemble, then they are members of the second one as well

CLASP centre for linguistic theory and studies in probability

Some Montague Semantics: Generalized Quantifiers

- Intersection and Union

```
Inductive Union (B C:Ensemble) : Ensemble :=
| Union_introl : forall x:e->Prop,
In B x -> In (Union B C) x
| Union_intror : forall x:e->Prop,
In C x -> In (Union B C) x.
Inductive Intersection (B C:Ensemble) : Ensemble :=
Intersection_intro :
forall x:e->Prop, In B x -> In C x -> In (Intersection B C
```

- Union is an inductive type: its first constructor says that forall predicates over *e*, if they are members of B (the leftmost ensemble), then they are also members of the Union of B and C
- Same with the second constructor but for the rightmost ensemble
- Intersection is easy: wanna try and tell me what this says?

CLASP centre for linguistic theory and studies in probability

Some Montague Semantics: Generalized Quantifiers

- Complement set, same set and extensionality axiom

```
Definition Complement (A:Ensemble) : Ensemble :=
fun x:e->Prop => ~ In A x.
Definition Same_set (B C:Ensemble) : Prop :=
Included B C /\ Included C B.
Axiom Extensionality_Ensembles :
forall A B:Ensemble, Same_set A B -> A = B.
```

- Complement is straightforward
- Same set is relation between two sets, one included in the other
- And lastly extensionality: if two sets are the same, then they are equal!

CLASP centre for linguistic theory and studies in probability

Some Montague Semantics: Generalized Quantifiers

- Ok, with this in place, let us do some GQ theory
- Sticking closer to notation, we define *t* to be Prop.
- For the standard quantifiers, I have kept the non set-theoretic definition. Here is *some* for example:

```
Definition some:= fun P Q: e->t=> exists x,  P x /\ Q x.
```

- Correct?
- Do *all* as an exercise!

Some Montague Semantics: Generalized Quantifiers

- Ok, let us start with *three*:

Some Montague Semantics: Generalized Quantifiers

- Ok, let us start with *three*:

```
Definition three:=  fun P Q: e->t=> some P Q /\
(CARD (intersection P Q)) = 3.
```

- I also have an existence part of *three* (makes reasoning easier!)
- What about *most*? Any ideas?

Some Montague Semantics: Generalized Quantifiers

- Ok, let us start with *three*:

```
Definition three:=  fun P Q: e->t=> some P Q /\
(CARD (intersection P Q)) = 3.
```

- I also have an existence part of *three* (makes reasoning easier!)
- What about *most*? Any ideas?

```
Definition most:=  fun P Q: e->t=> some P Q /\
gt (CARD (intersection P Q))
(CARD(complement(intersection P Q))).
```

- Exercise: define *at most five*

CLASP centre for linguistic theory and studies in probability

Some Montague Semantics: Generalized Quantifiers

- Let us do some reasoning

```
Theorem SOMEMOST: most cats walk -> some cats walk. .
```

- Trivial given the existence requirement on MOST.
    - Take some time and try to prove it!

    ```
    Theorem MOST: (exists x, cats x /\ walk x /\
    CARD(complement (intersection cats walk)) = 6
    /\ CARD(intersection cats walk) = 5)-> most cats walk.
    ```

- Properties of Qs can be defined for example:

```
Definition Conservativity:= fun Q:(e->t)->(e->t)->t =>
fun A B:e->t=>  Q A B <-> Q A (intersection A B).
```

CLASP centre for linguistic theory and studies in probability

Some Montague Semantics: Adjectives and Meaning Postulates

- The well-known classification of adjectives
  - Intersective (from Adj(N)(x) *infer* N(x))
  - Subsective (from Adj(N)(x) *infer* N(x))
  - Privative (from Adj(N)(x) *infer* $\neg$ N(x))
  - Non-committal (from Adj(N)(x) *infer* ?)

CLASP centre for linguistic theory and studies in probability

Some Montague Semantics: Adjectives and Meaning Postulates

- Let us see an adjective like *black*
  - It is intersective
  - Here is a tentative definition:

```
Parameter black: (e->t).
Definition Black:= fun Q: (e->t)=> fun x:e=> black x /\ Q x
```

  - First define a first order *black* property and then using this the higher-order adjective *Black*
  - For all other adjectives, the property is higher-order, e.g:

```
Parameter skillful: (e->t)->(e->t).
Definition Skillful:= fun Q: (e->t)=> fun x:e=> skillful Q
```

CLASP centre for linguistic theory and studies in probability

Some Montague Semantics: Adjectives and Meaning Postulates

- Defining meaning postulates as local assumptions in proofs

```
Parameter ADJ_I ADJ_S ADJ_P ADJ_NC: (e->t)->(e->t).
Parameter ADJ_i: e->t.
Variable Int: forall  Q: (e->t),
 forall  x:e,  ADJ_I Q x <->ADJ_i  x /\ Q x.
Variable Sub: forall  Q: (e->t),
forall  x:e,  ADJ_S Q x -> Q x.
Variable Pri: forall  Q: (e->t),
forall  x:e,  ADJ_P Q x -> not (Q x).
Variable NonC: forall  Q: (e->t),
forall  x:e,  ADJ_NC Q x.
```

CLASP centre for linguistic theory and studies in probability

Some Montague Semantics: Adjectives and Meaning Postulates

- Let us do some reasoning!

```
Theorem sman: ADJ_S  man John-> man John.
Theorem iman: ADJ_I  man John-> ADJ_i  John.
Theorem pman: ADJ_P  man John->not ( man John).
```

CLASP centre for linguistic theory and studies in probability

Some Montague Semantics: Type-Shifters

- Some definitions for classic type-shifters
  - The *Ident* and *Lift* shifters

```
Definition Ident:= fun x:e=>  fun y:e=>  x=y.
Definition Lift:= fun x: e=> fun P:e->t=>  P x.
```

  - Type Lowering with *BE*

```
Definition BE:= fun P:(e->t)->t=> fun x:e=>
P (fun y:e=>(y=x)).
```

centre for linguistic theory and studies in probability

Some Montague Semantics: Type-Shifters

- Polymorphic BE

```
Inductive Onebe : Set := be.
Definition BSem1 := e->e->t.
Definition BSem2 := (e->t)->e->t.
Definition be1 : BSem1:= fun x y=> x = y.
Definition be2 : BSem2:= fun P:e->t=>  fun x:e=>  P x.
Definition b1 (b:Onebe) : BSem1 := be1.
Coercion b1 : Onebe >-> BSem1.
Definition b2 (b:Onebe) : BSem2 := be2.
Coercion b2 : Onebe >-> BSem2.
```

CLASP centre for linguistic theory and studies in probability

Some Montague Semantics: Type-Shifters

- Some type checking and evaluations

```
Check Lift j.
Eval compute in Ident j.
Check BE (a man) j.
Eval compute in BE (a man) j.
Definition every:= fun P Q:e->t=> forall x, P x->Q x.
Eval compute in BE (every man) j.
(**Checking Polymorphism**)
Eval compute in (be:BSem1) (m) j.
Eval compute in (be:BSem2) (man) j.
```

- Prove the following
  - John is a man $\Rightarrow$ there is a man
  - John is every man $\Rightarrow$ John is George

CLASP centre for linguistic theory and studies in probability

D. Bekki.
Representing anaphora with dependent types.
LACL 2014, LNCS 8535, 2014.

S. Chatzikyriakidis and Z. Luo.
Adjectives in a modern type-theoretical setting.
In G. Morrill and J.M Nederhof, editors, *Proceedings of Formal Grammar 2013. LNCS 8036*, pages 159–174, 2013.

S. Chatzikyriakidis and Z. Luo.
Hyperintensionality in modern type theories.
Submitted manuscript, 2014.

S. Chatzikyriakidis and Z. Luo.
On the interpretation of common nouns: Types versus predicates.
In *Modern Perspectives in Type-Theoretical Semantics*, pages 43–70. Springer, 2017.

A. Ranta.
Type-Theoretical Grammar.
Oxford University Press, 1994.

T. Xue and Z. Luo.
Dot-types and their implementation.
Logical Aspects of Computational Linguistics (LACL 2012). LNCS 7351, 2012.

CLASP centre for linguistic theory and studies in probability