Gradability in MTT-semantics

Stergios Chatzikyriakidis⋆[1] and Zhaohui Luo⋆⋆[2]

[1] CLASP, Dept. of Philosophy, Linguistics and Theory of Science, Univ. of Gothenburg
stergios.chatzikyriakidis@gu.se
[2] Royal Holloway, University of London
zhaohui.luo@hotmail.co.uk

Abstract. In this paper, we look at the issue of gradability within formal semantics in modern type theories (MTT-semantics). Specifically, we look at both gradable adjectives and nouns, and show that the rich typing mechanisms afforded by MTT-semantics can give us a natural account of gradability. Gradable adjectives take indexed nouns as their arguments, while gradable nouns are Σ-types where their first projection is a degree parameter. Furthermore, we provide a standard polymorphic measure function applicable to all gradable adjectives and nouns. We also look at multidimensional adjectives and use enumerated types to capture multidimensionality. We formalize our account in the Coq proof assistant and check its formal correctness. Lastly, we briefly describe a recent proposal of model gradability by means of subtype universes in MTTs that can potentially give a unifying treatment of gradability for both regular gradable adjectives, but also multidimensional ones.

1 Introduction

The term gradable adjectives refers to the class of adjectives that involve some kind of grading property/parameter that allows them to be quantified according to it. For example, in the case of *small* and *large*, the grading parameter is size. Gradable adjectives have comparative and superlative forms and can be further modified by degree adverbs (e.g. *much*). Besides gradable adjectives, one also finds cases of gradable nouns, i.e. cases where the gradable element is not an adjective, but rather a noun:

(1) John is an enormous idiot/ He is a big stamp collector.

In (1), the most natural reading is not one of large physical size, but rather of the nominal holding to a high degree.

⋆ Supported by grant 2014-39 from the Swedish Research Council, which funds the Centre for Linguistic Theory and Studies in Probability (CLASP) in Dept. of Philosophy, Linguistics, and Theory of Science at the University of Gothenburg.

⋆⋆ Partially supported by EU COST Action EUTypes (CA15123, Research Network on Types).

There are furthermore adjectives that can be quantified across more than one dimension. For example in the case of *tall*, there is only one dimension involved, tallness. This is not the case for adjectives like *healthy* and *sick*, which are called multidimensional. Following [47], two different classes of multidimensional adjectives are distinguished: positive and negative. The idea is that every positive adjective has a negative counterpart, i.e. its antonym (e.g. *healthy* and *sick*). What is different between the two is the form of quantification over dimensions in each case. Positive adjectives involve universal quantification over dimensions, while negative adjectives existential quantification. For example, for someone to be considered healthy, s/he must be healthy in all dimensions, whereas for someone to be considered sick, it suffices to be sick across one dimension only. In order for this intuition to be borne out more clearly, the exception phrase headed by *except* can be used. The interesting bit here is that this phrase is only compatible with universal quantification. As seen below, 'healthy' is compatible with an 'except' phrase, while 'sick' is not:

(2) Dan is healthy except with respect to blood pressure

(3) # Dan is sick except with respect to blood pressure

In this paper, we look at both gradable adjectives/nouns and multidimensional ones from the perspective of formal semantics in modern type theories (MTT-semantics) [35, 10], arguing that MTT-semantics provides us with the mechanisms to give reasonable formal semantics accounts of these phenomena. The structure of the paper is as follows: in section 2, we give a brief introduction to MTT-semantics, concentrating on the features that are mostly relevant to the analyses in this paper. In section 3, we present our analysis of gradable and multidimensional adjectives/nouns. In section 4, we formalize our account in the Coq proof assistant and check its correctness. In section 5, we provide a brief investigation of an alternative way to deal with gradability by using recent work in Type Theory on subtype universes. Lastly, in section 6, we conclude and discuss some future work.

2 Modern Type Theories: A Brief Introductioin

Formal semantics in modern type theories (MTT-semantics) [35, 10] has been proposed as an alternative to Montague Semantics, and various semantic accounts have been given within this paradigm for a wide range of linguistic phenomena [32, 35, 11, 50, 8]. We use the term Modern Type Theories (MTTs) to refer to a class of type theories which have dependent types, inductive types and other powerful and expressive typing constructions. MTTs can be predicative, such as Martin-Löf's intensional type theory [39, 44], and impredicative, such as the Unified Theory of dependent Types (UTT) [30]. In this paper, we shall employ UTT complemented with the coercive subtyping mechanism [31, 36].

In this section we provide a brief introduction to MTTs, concentrating mostly on the features that are most relevant to this paper.

2.1 Many-Sortedness, Common Nouns as Types and Subtyping

A key difference between MTT-semantics and Montague semantics (MS) lies in the interpretation of common nouns (CNs). In [42], the underlying logic (Church's simple type theory [13]) is 'single-sorted' in the sense that there is only one type, e, of all entities. The other types such as the type of truth values, i.e. t, and the function types generated from types e and t do not stand for types of entities. Thus, no fine-grained distinctions between the elements of type e exist, and as such, all individuals are interpreted using the same type. For example, *John* and *Mary* have the same type in simple type theory, i.e. the type e of individuals. An MTT, on the other hand, can be regarded as a 'many-sorted' logical system in that it contains many types. In this respect, MTTs can make fine-grained distinctions between individuals and use those different types to interpret subclasses of individuals. For example, one can have $John : Man$ and $Mary : Woman$, where Man and $Woman$ are different types. Another very basic difference between MS and MTTs is that common nouns in MTTs (CNs) are usually interpreted as *types* [45] rather than sets or predicates (i.e., objects of type $e \rightarrow t$) as in MS. The CNs 'man, human, table' and 'book' are interpreted as types *Man, Human, Table* and *Book*, respectively. Then, individuals are interpreted as being of one of the types used to interpret CNs.

This many-sortedness has the welcome result that a number of semantically infelicitous sentences involving category mistakes, which are however syntactically well-formed, like e.g. 'he ham sandwich walks' can be explained easily. This is because a verb like 'walks' will be specified as being of type $Animal \rightarrow Prop$, while the type for 'ham sandwich' will be $Food$ or $Sandwich$:

(4) $the\ Ham\ sandwich : Food$

(5) $walk : Human \rightarrow Prop$

The idea that common nouns should be interpreted as types rather than predicates has been argued in [34] on philosophical grounds as well. There, it is claimed that the observation found in [20] according to which common nouns, in contrast to other linguistic categories, have criteria of identity that enable them to be compared, counted or quantified, has an interesting link with the constructive notion of set/type: in constructive mathematics, sets (types) are not constructed only by specifying their objects but they additionally involve an equality relation. The argument is then that the interpretation of CNs as types in MTTs is explained and justified to a certain extent. Extensions and further theoretical advances using the CNs as types approach can be found in [12] and an extension of the idea that CNs further specify their identity criteria with a case study on counting with numerical quantifiers under copredication cases is given in [9]

Interpreting CNs as types rather than predicates has also a significant methodological implication: compatibility with subtyping. For instance, one may introduce various subtyping relations by postulating a collection of subtypes (physical objects, informational objects, eventualities, etc.) of the type *Entity* [2]. It is a well-known fact that if CNs are interpreted as predicates as in traditional Montagovian settings, introducing such subtyping relations would cause problems. This is because the contravariance of function types would predict that given the subtyping relation $A \leq B$, $B \rightarrow Prop \leq A \rightarrow Prop$ would be the case (the opposite relation than the one needed). Substituting A with type Man and B with type $Human$, we come to understand why interpreting CNs as predicates is not a good idea if we want to add a subtyping mechanism.

The subtyping mechanism used in the MTT endorsed in this paper is that of coercive subtyping [31, 36]. Coercive subtyping can be seen as an abbreviation mechanism: A is a (proper) subtype of B ($A \leq B$) if there is a unique implicit coercion c from type A to type B and, if so, an object a of type A can be used in any context $\mathfrak{C}_B[\_]$ that expects an object of type B: $\mathfrak{C}_B[a]$ to be legal (well-typed) and equal to $\mathfrak{C}_B[c(a)]$.

To give an example: assume that both Man and $Human$ are base types. One may then introduce the following as a basic subtyping relation:

(6) $Man \leq Human$

2.2 Σ-types, Π-types, Indexed Types and Universes

Dependent Σ-types. One of the basic features of MTTs is the use of Dependent Types. A dependent type is a family of types that depend on some values. The constructor/operator Σ is a generalization of the Cartesian product of two sets that allows the second set to depend on values of the first. For instance, if $Human$ is a type and $Male : Human \rightarrow Prop$, then the Σ-type $\Sigma h : Human.\ Male(h)$ is intuitively the type of humans who are male.

More formally, if A is a type and B is an A-indexed family of types, then $\Sigma(A, B)$, or sometimes written as $\Sigma x : A.B(x)$, is a type, consisting of pairs (a, b) such that a is of type A and b is of type $B(a)$. When $B(x)$ is a constant type (i.e., always the same type no matter what x is), the Σ-type degenerates into product type $A \times B$ of non-dependent pairs. Σ-types (and product types) are associated projection operations π_1 and π_2 so that $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$, for every (a, b) of type $\Sigma(A, B)$ or $A \times B$.

The linguistic relevance of Σ-types can be directly appreciated once we understand that, in its dependent case, Σ-types can be used to interpret linguistic phenomena of central importance, like adjectival modification (see above for interpretation of modified CNs) [45]. For example, *handsome Man* is interpreted as a Σ-type (7), the type of handsome men (or more precisely, of those men together with proofs that they are handsome):

(7) $\Sigma m : Man\ handsome(m)$

where $handsome(m)$ is a family of propositions/types that depends on the man m.

Dependent Π-types. The other basic constructor for dependent types is Π. Π-types can be seen as a generalization of the normal function space where the second type is a family of types that might be dependent on the values of the first. A Π-type degenerates to the function type $A \to B$ in the non-dependent case. In more detail, when A is a type and P is a predicate over A, $\Pi x{:}A.P(x)$ is the dependent function type that, in the embedded logic, stands for the universally quantified proposition $\forall x{:}A.P(x)$. For example, the following sentence (8) is interpreted as (9):

(8) Every man walks.

(9) $\Pi x : Man.walk(x)$

Π-types are very useful in formulating the typings for a number of linguistic categories like VP adverbs or quantifiers. The idea is that adverbs and quantifiers range over the universe of (the interpretations of) CNs and as such we need a way to represent this fact. In this case, Π-types can be used, universally quantifying over the universe CN. (10) is the type for VP adverbs while (11) is the type for quantifiers:[3]

(10) $\Pi A : \mathsf{CN}.\ (A \to Prop) \to (A \to Prop)$

(11) $\Pi A : \mathsf{CN}.\ (A \to Prop) \to Prop$

Further explanations of the above types are given after we have introduced the concept of type universe below.

Indexed Types. An indexed type is a type of dependent type. They are families of types that are indexed by a parameter whose type is usually a simple one. Indexed types here will be used in the main analysis of gradable adjectives, as we will assume that gradable adjectives do not take simple CN types as their arguments but rather CN types indexed with a parameter. For example, we can think of the type representing humans along with their heights. We can do this using indexed types by considering the family of types $Human : Height \to Type$ indexed by heights: $Human(n)$ is the type of humans of height n.

Type Universes. An advanced feature of MTTs, which will be shown to be very relevant in interpreting NL semantics in general as well as adjectival modification specifically, is that of universes. Informally, a universe is a collection of (the names of) types put into a type [40].[4] For example, one may want to collect all the names of the types that interpret common nouns into a universe $\mathsf{CN} : Type$. The idea is that for each type A that interprets a common noun, there is a name

[3] The type for adverbs was proposed for the first time in [33].

[4] There is quite a long discussion on how these universes should be like. In particular, the debate is largely concentrated on whether a universe should be predicative or impredicative. A strongly impredicative universe U of all types (with $U : U$ and Π-types) is shown to be paradoxical [21, 19] and as such logically inconsistent. The theory UTT we use here has only one impredicative universe $Prop$ (representing the world of logical formulas) together with an infinitely many predicative universes which as such avoids Girard's paradox (see [30] for more details).

$\overline{A}$ in CN. For example,

$$\overline{Man} : \mathsf{CN} \quad \text{and} \quad T_{\mathsf{CN}}(\overline{Man}) = Man.$$

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, $Man : \mathsf{CN}$.

Having introduced the universe CN, it is now possible to explain (10) and (11). The type in (11) says that for all elements A of type CN, we get a function type $(A \to Prop) \to Prop$. The idea is that the element A is now the type used. To illustrate how this works let us imagine the case of quantifier *some* which has the typing in (11). The first argument we need, has to be of type CN. Thus *some human* is of type $(Human \to Prop) \to Prop$ given that the A here is $Human : \mathsf{CN}$ (A becomes the type $Human$ in $(Human \to Prop) \to Prop$). Then given a predicate like $walk : Human \to Prop$, we can apply *some human* to get $(some\ Human)(walk) : Prop$. Similar considerations apply for (10).

3 Gradability in MTT-semantics

In this section, we present an MTT account of a number of aspects of gradable and multidimensional adjectives.

3.1 Gradable Adjectives.

A standard assumption in the literature is that gradable adjectives involve some kind of measurement. Usually, this measurement is assumed to be a degree argument, whose presence or not, is then considered to be the main difference between gradable and non-gradable adjectives. This extra argument has been proposed to be formally encoded in the adjective's typing as in [3, 49, 23], or not as in [29, 41, 25, 48].

The account we are going to pursue here is one where the the arguments of gradable adjectives are not of simple types, but rather types indexed by degree parameters (dependent types). In MTT-semantics, the universe CN of common nouns are refined into subuniverses of CNs each of which is indexed by a degree. For example, the collection represented by the common noun *human* may be refined into the family of types indexed by heights: $Human : Height \to Type$ and $Human(n)$ is the type of humans of height n.[5] We can then define a function *height* that returns the value of the height-index of a human; i.e., $height(i, h)$ is the height of human h:

(12) $height : \Pi i : Height.\ Human(i) \to Height$

(13) $height(i, h) = i.$

[5] Informally, this family of types of humans are more refined than the type $Human$ of all humans. Formally, we'd have $HHuman(i) \leq Human$.

With these assumptions in line, we may consider the semantic interpretation of *tall* to mean that the height of the human concerned is larger than some given standard n:

(14) $tall : \Pi i : Height.\ Human(i) \to Prop$

(15) $tall(i, h) = height(i, h) \geq n$

The above definition for *tall* specifies that for any i of type $Height$, *tall* takes a human argument indexed with i and returns the proposition saying that i, the height of the human, is greater than or equal to a natural number n, which stands for the contextually restricted parameter – humans taller than n are regarded as tall. In a similar fashion, we can define the comparatives, where the RHS of (17) is the same as $i > j$:

(16) $taller\_than : \Pi i, j : Height.\ Human(i) \to Human(j) \to Prop$

(17) $taller\_than(i, j, h_1, h_2) = height(i, h_1) > height(j, h_2)$.

From this definition, we can easily prove that, for example, if $height(i, h_1) \geq height(j, h_2)$ and $tall(j, h_2)$, then $tall(i, h_1)$.

The natural question to ask at this point is the following: where does this contextual parameter come from? In what we have provided so far, it is just a number that does not depend on anything. A better and more intuitive way to refer to this contextual parameter is to make its value dependent on the noun, the adjective, and sometimes even some other contextual information. These latter three parameters in MTT-semantics are represented as a type, a predicate and a context (in type theory), respectively. In order to fornalize this idea, we use polymorphism and type dependency. First, we introduce the universe of (totally ordered) degree types, $Degree$. As examples of degrees, one would find in $Degree$ types such as $Height$, $Weight$ and $Width$, among many others. The inference rules of CN_G are given below, the second of which says that $\mathsf{CN}_{\mathrm{G}}(D)$ is a subtype of CN and the third is an example of an introduction rule for CN_{G}:

$$\frac{D : Degree}{\mathsf{CN}_{\mathrm{G}}(D) : Type} \qquad \frac{D : Degree \quad A : \mathsf{CN}_{\mathrm{G}}(D)}{A : \mathsf{CN}} \qquad \frac{i : Height}{Human(i) : \mathsf{CN}_{\mathrm{G}}(Height)}$$

We can now introduce the polymorphic standard, STND. First, for any common noun A, let $ADJ(A)$ be the type of syntactic forms of adjectives whose semantic domain is A. For instance, $TALL : ADJ(Human)$, where $TALL$ strands for the syntax of *tall*. Then, STND takes a degree D, a D-indexed common noun A and (the syntax of) an adjective whose domain is A, and returns the relevant standard for the adjective:

(18) $\mathrm{STND} : \Pi D : Degree.\ \Pi A : \mathsf{CN}_G(D).\ ADJ(A) \to D$

The next thing to consider in giving a more proper definition for *tall*, is a polymorphic type that is not restricted to $Human$ arguments with $Height$ parameters only. *Tall* can be used with types of non-humans: for example one can talk about a *tall building* or a *tall cat*. On the other hand, uses like *tall democracy* or *tall mind* do not seem to be felicitous, at least without some sort of contextual coercion. Using either $Human(i)$ as argument for *tall* or a polymorphic argument based over the universe CN will undergenerate and overgenerate respectively. One can try to use a subuniverse of CN, CN_{PHY}that basically includes all physical objects (types $\textsc{Phy}$ and its subtypes). In this respect, we can introduce the universe $\mathsf{CN}_{PHY}$with the following introduction rule:

$$\frac{A{:}\mathsf{CN}, A < \textsc{Phy}}{A : \mathsf{CN}_{PHY}}$$

With this rule and assuming that every physical object has a height, we are now in a position to upgrade the definition for *tall* (we assume that the argument A is implicit in the definition):

(19) $tall : \Pi A : \mathsf{CN}_{PHY}.\Pi i : Height.A(i) \to Prop$
(20) $tall(i, h) = height(i, h) \geq \mathrm{STND}(Height, Human, TALL)$.

Note that indexing on the noun by means of a degree gives us for free the fact that we are not talking about tallness in general but tallness with respect to the relevant class (represented by the type $Human$ in the above example). In order to understand its importance, this indexing seems to be doing the work done by using the dot combinator of [26] to compose comparison classes with adjectives in the work of [24]. To give an example, let us say that one needs to compose *tall* with its comparison class, say basketball player (represented as BB). The typings we have are as follows: $BB : e \to t$ and $tall : e \to d$. However, we need functional application to return: $BB(tall) : e \to d$. As obvious, normal functional application will not work here. Thus, the dot combinator is used to remedy this. This additional and arguably not well-motivated extra machinery is not needed in our case. Furthermore, the polymorphic STND function can be seen as a more straightforward interpretation of Kennedy's context sensitive function from measure functions (adjectives basically) to degrees [24]. Lastly, one may consider standards that are dependent on other contextual information as well: for example, whether something is regarded as an expensive car might depend on where it is considered. In that case, the $STND$ function may take an additional parameter of locations that would take this into account.

Remark 1 (CN and its subuniverses). Type universes help us in MTT-semantic formalizations. For example, we have used the universe CN as the universe that makes polymorphism over all common nouns possible and allows adequate typing for phenomena like VP-adverbs and subsective adjectives to be provided:

(21) $VP_{ADV} : \Pi A : \mathsf{CN}.\ (A \to Prop) \to (A \to Prop)$
(22) $ADJ_{SUBS} : \Pi A : \mathsf{CN}.\ A \to Prop$

Furthermore, we have used the universe CN_{PHY} in our analysis of 'tall', in order to restrict the domain of polymorphism to the subuniverse that includes all physical objects and their subtypes. Other similar useful subuniverses can be constructed in order to help us in our semantic representations. Consider for example the subsective adjective ‘skilful’. According to what we have been saying so far, it is of the type given in (22). Digging a bit deeper, one can see that ‘skilful’ is not really compatible with arguments that are not of type $Human$, or at least of type $Animal$. For example, one cannot talk about a skilful carpet or a skilful democracy. Thus, one could update the definition for ‘skilful’ taking these issues into consideration. On the assumption that ‘skilful’ is only relevant for human arguments, polymorphism is on the subuniverse CN_H, i.e. the universe including types $Human$ and its subtypes:

(23) $skilful : \Pi A : \mathsf{CN}_H.\ A \rightarrow Prop$

An important question is, of course, how can we decide what the relevant universe is in each case? Well, one way to do it is by linguistic investigation as typically done in formal linguistics, i.e. getting judgments of native speakers that will help us decide the elements of the universe to be formed. Another way to do that is to use existing lexical-semantics resources that might contain such information. For example, in [7], the authors experiment with JeuxdeMots [27], a rich lexical-semantic network constructed using GWAPs [1], in order to extract information relevant for multi-typed systems, e.g. common noun types, subtyping relations, typings for predicates etc. We believe that such connections should be explored in future work combining lexical-semantic information drawn from linguistic resources with rich formal semantics formalisms like the one we are describing in this paper.

The other question one need to answer is whether such subuniverses are formally coherent in the sense that their introduction is logically okay. One has to be careful when constructing such universes. Some universes can be formally paradoxical even though they may seem justified from a linguistic perspective.[6] Thus, a better way to put what we have been saying is the following: we construct meaningful universes based either on linguistic intuitions and/or information from lexical/semantic networks, but only when we can formally justify it, i.e. to prove meta-theoretically that the incorporation of the new universe into the original type theory is OK (e.g., logically consistent, among other properties). The universes such as CN_{PHY} and CN_H are what we call *subtype universes* studied recently by Maclean and the second author [37], see §5. □

3.2 Gradable Nouns

As already discussed in the introduction, gradable nouns concern gradability cases where the relevant gradable element is not an adjective, but rather a noun, as (24) illustrates.

(24) John is an enormous idiot. / He is a big stamp collector.

[6] For example, the type theory studied by Martin-Löf in [38] has a type U of all types (and hence, $U : U$) and has been proven to be logically inconsistent [21, 19].

Indexed types, as already mentioned, are a type of dependent types, i.e. families of types indexed by a parameter whose type is usually a simple one. We have used indexed types so far in our treatment of gradable adjectives. The question is whether we can extend the usage of indexed types to gradable nouns as well. We will argue that this is indeed possible. What we want to propose here is that the distinction between nouns and adjectives is still clear: adjectives are taken to be predicates, nouns are taken to be types. At the same time, however, we assume that gradable nouns like *idiot* and gradable adjectives like *tall* both involve a degree parameter, albeit an abstract one in the former case. A natural way to capture this idea, i.e. abstract nouns being types but still involving a degree parameter, is to use Σ-types and assume that the first projection is actually the abstract parameter. To do this, we consider the type family $IHuman : Idiocy \rightarrow Type$ indexed by idiocy degrees of type $Idiocy : Degree$, where $Idiocy$ is a type whose objects form a total order and can be compared to each other by, for example, a $\geq$-relation. Then, *idiot* can be represented by means of (25):

(25) $Idiot = \Sigma i : Idiocy.IHuman(i) \times (i \geq \mathrm{STND}(Idiocy, Human, IDIOTIC))$

An idiot is thus a triple (i, h, p) where h is a human whose idiocy degree i is larger than or equal to the standard of being an idiot. Note that this account has not only similarities with the ideas proposed in [14] but also brings out a connection with gradable adjectives in the sense that both gradable adjectives and gradable nouns involve a degree parameter. However, these two are clearly different in terms of their formal status, adjectives being predicates while nouns types.

Let us now consider *enormous idiot.* The interpretation we want to get in this case is one where someone is an idiot to a very high degree. This means that this degree must be (much) higher than the degree of idiocy needed for someone to be considered an idiot (the standard $\mathrm{STND}(Idiocy, Human, IDIOTIC)$ in (25)). In order to capture that, we first propose that *enormous* can be interpreted as having the following type, where $PHY_D : \mathsf{CN}_{\mathrm{G}}(D)$ is the type of physical objects indexed by D:

(26) $enormous : \Pi D : Degree\ \Pi A : D \rightarrow \mathsf{CN}_{\mathrm{G}}(D)\ \Pi d{:}D.\ (A(D) \rightarrow Prop)$

Then we propose the following definition for 'enormous', for $D : Degree$, $A : D \rightarrow \mathsf{CN}_{\mathrm{G}}(D)$, $d : D$, and $a : A(D)$:

(27) $Enormous(D)(A)(d)(a) = d \geq \mathrm{STND}(D, PHY_D, ENORMOUS)$

We are now ready to interpret *enormous idiot* (D and A arguments are implicit):

(28) $Enormous\ Idiot = \Sigma h : Idiot.\ enormous((\pi_1(h), \pi_2(\pi_1(h)))$
where $\mathrm{STND}(D, PHY_D, ENORMOUS) \geq \mathrm{STND}(Idiocy, Human, IDIOTIC)$

Enormous idiot is thus a pair, where the first projection consists of a proof of being an idiot h (*Idiot* itself also a Σ-type, see (25)) and the second projection requires that the standard of idiocy associated with the first projection of the second projection of h is greater than the standard for *enormous.*

3.3 Multidimensional Adjectives

Multidimensional adjectives are adjectives that can be quantified across different dimensions. Adjectives like *sick* and *healthy* fall into this category. Following [47], two different classes of multidimensional adjectives are distinguished: positive and negative. The idea is that every positive adjective has a negative counterpart, i.e. its antonym (e.g. *healthy* and *sick*). What is different between the two is the form of quantification over dimensions in each case. Positive adjectives involve universal quantification over dimensions, while negative adjectives existential quantification. For example, for someone to be considered healthy, s/he must be healthy in all dimensions, whereas sick, it suffices to be sick across one dimension only. In order for this intuition to be borne out more clearly, the exception phrase *except* can be used. The interesting bit here is that this phrase is only compatible with universal quantification. As seen below, 'healthy' is compatible with 'except', but 'sick' is not:

(29) Dan is healthy except with respect to blood pressure

(30) # Dan is sick except with respect to blood pressure

This intuition can be implemented in an MTT setting using an inductive type for multiple dimensions. Consider an adjective like *healthy.* In order for someone to be considered healthy, one must be able to universally quantify over a number of "health" dimensions: *cholesterol, blood pressure* etc. To formalize this, one can introduce the inductive type $Health$ of type $Degree$ as follows:[7]

(31) $Health : Degree = heart \mid blood\_pressure \mid cholesterol$

We assume that the adjective *healthy* is of the the following type (we use $Human$ as a simple type rather than a type-valued function as used earlier):

(32) $healthy : Health \to Human \to Prop$

We can now use this parameter as a primitive to define $Healthy$ and $Sick$ as follows:

(33) $Healthy = \lambda x : Human.\forall h : Health.\ healthy(h, x)$

(34) $Sick = \lambda x : Human.\neg(\forall h : Health.\ healthy(h, x))$

Note that, for multidimensional adjectives, each dimension may be gradable. For example, when we say that a healthy person is a person healthy in all dimensions, it basically means that each dimension surpasses a standard of healthiness. Take the dimension 'blood pressure' as an example: a child $x : Child \leq Human$ is healthy as far as blood pressure is concerned may mean that the blood pressure of x is less than some threshold with respect to children. (See §5 for formal details.)

[7] The inductive type $Health$ is a finite type (also called an enumeration type), sometimes written as $\{heart,\ blood\_pressure,\ cholesterol\}$.

Remark 2. With respect to multidimensional adjectives, there are a number of complications that need to be addressed. For example, the nature of the quantifier associated with positive adjectives does not seem to always be the universal quantifier. Sassoon and Fadlon [46] define quantificational multidimensional adjectives in the following sense:

> Quantificational adjectives like optimistic often involve counting of dimensions. As a default, entities fall under them iff they are classified under sufficiently many (e.g., some, most or all) dimensions.

Of course, this is not a problem in itself. One can modify the account w.r.t different adjectives as involving different quantificational force:

(35) $Healthy = \lambda x : Human.\ [\textsc{all},\textsc{some},\textsc{most}]\, h{:}Health.healthy(h, x)$

The choice of quantifier can be context dependent. One can assume that the quantifier quantifies over relevant dimensions in specific contexts. The definition of *Healthy* can be overloaded, picking the relevant dimensions in each case (relevant means available in that context). This is similar to the overloading technique as proposed by the second author to deal with homonymy [33].

$\square$

3.4 Multidimensional Nouns

A further interesting discussion w.r.t multidimensionality concerns multidimensional nouns. For example, a noun like 'bird', at least according to theories like Prototype and Exemplar theories,[8] is argued to involve a rich couple of dimensions, i.e. in order for something to count as a bird, a couple of different dimensions (for example, dimensions like *winged*, *small*, *can breed,* etc.) have to be taken into consideration. Then, the idea is that the conceptual structure of a noun like 'bird' will involve an ideal value for each dimension. A similarity measure is mapping entities to degrees, representing how far from the ideal dimensions of the prototype the values for the respective entities are. This is represented as a weighted sum. The important thing, skipping formal details, is summarized in the following passage:

The distances of x from the prototypical values in the different bird dimensions integrate into a unique degree in the given noun by means of averaging operations, like weighted-sums... [Sassoon, 2013]

The above passage argues that dimensions integrate (another way of putting it is collapse) into a unique degree, and, thus, are not accessible for quantification as it is the case with multidimensional adjectives. Viewing common nouns as types seems to be compatible with this claim. The idea is as follows: in order for an object to be of a CN type, the standard of membership w.r.t the weighted sum of its similarity degrees to the ideal values in the dimensions of the noun has to be exceeded.[9] Actually, [46] revises later on her view, and talks about weighted

[8] See [47] for references to the relevant literature.
[9] See [43] for more details on this approach.

products in the case of these type of nouns. Somewhere in the middle between this two types of multidimensionality, i.e. multidimensional adjectives like *healthy* and multidimensional nouns like *bird*, we find social nouns like *linguist, artist.* These seem to behave like multidimensional adjectives, in that their dimensions seem to be accessible for quantification as witness the example below:

(36) He is an artist in many respects.

Such cases are then argued to represent intermediate cases, where the dimensions are integrated into a single degree, albeit the relevant operation is one of weighted sum and not product. The argument is that these dimensions are made easier available to quantification in these cases. This might then mean that the types become more elaborate in these cases. Consider the case for *artist*, and consider the inductive type for all its dimensions (we note them here as a_1, a_2, a_3 pending a more serious discussion of what these dimensions really are):

(37) Inductive $Art : D = a_1 \mid a_2 \mid a_3$

Now, one can think that in cases where social nouns make their dimensions accessible, what happens is that some sort of quantification is at play in the form of a Σ type, where the first projection is just a type $Human$, while the second projection specifies that all dimensions of artistry hold of this human above the relevant standard. Our definition for *artist* is given below:[10]

(38) $artist = \Sigma h : Human.\forall a : Art.DIM_{CN}(h, a)$

Notice, that the above is still a type and not a predicate. One can think that the creation of such types should be in general available, as even non-social nouns, e.g. natural-kind nouns like *duck*, can be sometimes, context allowing, used in a way that seems to make their dimensions available. For example, one can imagine a context where the following is true:

(39) My dog is a cat in most respects.

Thus, it seems that the operation to turn simple types into Σ types that make their dimensions available, is a more general one, and should be restricted w.r.t context and general world-knowledge considerations.

There are far more issues to consider when one looks at multidimensional adjectives (and nouns). However, we cannot go into detail into all these issues here. This topic deserves a separate paper in its own right. We direct the interested reader to [47] and [46] for literature review and a detailed exposition of the complexity of the phenomenon in question.

4 Coq Implementation

In this section, we present a Coq implementation for the different issues we have been discussing in this paper. But first things first. What is Coq? Simplifying

[10] With $DIM_{CN} : \Pi D : Degree.Human \rightarrow D \rightarrow Prop.$)

things a bit, the main idea behind Coq can be roughly summarized as follows: you use Coq in order to see whether propositions based on statements previously pre-defined or user defined (definitions, parameters, variables) can be proven or not. Coq is a dependently typed proof assistant implementing the calculus of Inductive Constructions (CiC, see [18]). This means that the language used for expressing these various propositions is an MTT. This is a good start, at least for people using MTTs for NL semantics. Coq "speaks" so to say the language we use to interpret linguistic semantics. Given that Coq is in effect a reasoning engine, there are at least ways that can be used in studying linguistic semantics, to an extent overlapping with each other: a) as a formal checker for the semantic validity of proposed accounts in NL semantics and b) Natural Language Inference (NLI), i.e. reasoning with NL.

Remark 3 (interim note on installation). Coq can be installed easily for all platforms by visiting the system's website.[11] You can also get it using Macports, Homebrew or Nix. For mac and linux users, it is recommended to use Proof General,[12] a Coq interface for emacs that provides support for several proof assistants. □

Remark 4 (the type system implemented in Coq). The main difference between the type system that Coq implements [18] and the MTT we have been using so far (the type theory UTT [30]) is the use of coinductive types in Coq. Coinductive types are not used in any way in what we have been presenting so far, neither used in the Coq implementations. There are other minor differences between the two systems, but these are out of the scope of this paper, and play no important role in understanding the discussion in this section. □

Let us start with the formalization of gradable adjectives. We formulate the Degree universe Tarski-style in Coq:

```
(* Degree is type of names of degrees
d : Degree corresponds to type D(d) *)
(* So, Degree is a Tarski universe! *)
(* Here is an example with three degrees. *)
Require Import Omega.
Inductive Degree: Set:= HEIGHT | AGE | IDIOCY.
Definition D (d: Degree):= nat.
Definition Height := D(HEIGHT).
Definition Age:= D(AGE).
Definition Idiocy:= D(IDIOCY).
```

The code comments are enough to explain what is going on here: *Degree* is the type of names of degrees and d : *Degree* corresponds to type $D(d)$. The next step is to formalize the universe CN_{G}, and then the context dependent standard, i.e. $STND$ function:

[11] http://coq.inria.fr/download.
[12] https://proofgeneral.github.io

```
(* Universe CN_G of indexed CNs *)
Definition CN_G (_:Degree) := Set.
Parameter STND: forall d:Degree, forall A: CN_G(d), ADJ d A -> D(d).
```

Note that, in Coq, $forall$ stands for Π. With the previous parameter and definitions, *tall* can be defined:

```
Definition tall (h:Human):= ge (height h) (STND HEIGHT Human TALL).
```

With this at hand, one can define *taller*::

```
Definition taller_than (h1:Human) (h2:Human):= gt (height h2)
(height h1).
```

The next part involves formalizing gradable nouns, more specifically providing the type for *idiot* and the definitions for *enormous* and *enormous idiot*. The definitions follow closely the ones proposed in the paper. *Enormous idiot* is expressed as a Dependent Record Type: [13]

```
(**Definition for Idiot**)
Definition Idiot:=  sigT(fun x: Idiocy=> prod (IHuman x)
(ge x (STND IDIOCY Human IDIOTIC))).
Definition enormous  (d:Degree)(A:CN_G(d))(d1: D d) :=
    fun P: A =>  ge (d1)  (STND  d (PHY(d))(ENORMOUS d)).
Record enormousidiot: Set:= mkeidiot
    {h:> Idiot; EI: enormous IDIOCY
    (IHuman(projT1(h)))(projT1(h))(projT1(projT2(h)))
    /\  ge  (STND  IDIOCY (PHY(IDIOCY))(ENORMOUS IDIOCY))
    (STND IDIOCY Human IDIOTIC)}.
```

We continue with multidimensional adjectives. What we want to do in this case is implement the main idea we have been discussing in §3.3, namely the use of enumerated types in order to implement the many-dimensions aspect of multidimensional adjectives. Taking *healthy* as our example, we define the enumerated type Health that includes various health dimensions and then define adjectives *sick* and *healthy*, as involving universal quantification over the dimensions in *healthy*, and existential quantification in *sick*:

```
Definition  Degree:= Set.
Inductive Health: Degree:= Heart|Blood|Cholesterol.
Parameter Healthy: Health->Human->Prop.
Definition sick:= fun y: Human => ~ (forall x: Health, Healthy x y).
Definition healthy:= fun y: Human => forall x: Health, Healthy x y.
```

This suffices to give us the basic inferences with respect to multidimensional adjectives. For example one can prove that if John is healthy then he is healthy with respect to cholesterol, blood pressure and heart condition, if John is sick it suffices that he is not healthy across one dimension etc. These theorems, a number of other similarly relevant ones, as well as the formalization of the multidimensional noun *artist*, can be found in the Appendix A.2.

[13] Dependent Record Types in Coq are just syntactic sugar for Σ-types.

5 Modelling Gradability with Subtype Universes

Gradable adjectives and the related multidimensional cases provide challenging examples for MTT-semantics. This has led to further studies to develop type-theoretic mechanisms to formally deal with such phenomena. Recently, Maclean and the second author [37] have developed *subtype universes* for MTTs, which have interesting applications to programming and NL semantics. For the latter, they have pointed out that, employing subtype universes, one can obtain a nice semantics for gradable adjectives. We give a brief description here.

A subtype universe is a type that represents a collection of subtypes: for any type H, the universe $U(H)$ represents the collection of all subtypes of H. Such subtype universes can be specified formally by the following formation rule (U_F) and introduction rule (U_I), where $A \leq H$ is the shorthand for '$A \leq_c H$ for some coercion c' in the framework of coercive subtyping [36]:

$$(U_F)\quad \frac{\Gamma \vdash H : Type}{\Gamma \vdash U(H) : Type} \qquad\qquad (U_I)\quad \frac{\Gamma \vdash A \leq H : Type}{\Gamma \vdash A : U(H)}$$

Such type universes can be quantified over to form other propositions. For example, the proposition $\forall X{:}U(H).P(X)$ says that P holds for all subtypes of H. This, among other things, gives a nice treatment of *bounded quantification* of the form $\forall A \leq H.\ P(A)$ as proposed by Cardelli and Wegner [6], whilst avoiding the type checking issues traditionally associated with it. Also, Maclean and Luo have, for the first time, proved that extending MTTs with subtype universes preserves logical consistency [37], which is indispensable for a type theory to be used as a foundational semantic language.

Gradable adjectives such as 'tall' and 'healthy' can be modelled in MTT-semantics with the help of subtype universes. For example, let T be a type universe whose objects are base types H such as *Human* and *Building* for which the property $height : H \to Prop$ makes sense. Then, the type of *tall* can be given by means of subtype universes as in (40), which can be rewritten as (41) by means of bounded quantification as a notational abbreviation. So, *tall* is a predicate on subtypes of the base types. For instance, if $Human : T$ and $socrates : Man \leq Human$, then $tall(Human, Man, socrates)$ is a proposition. Given a threshold function $\xi : \Pi H{:}T.(U(H) \to Nat)$, one may define *tall* as $tall(H, A, x) = height(x) \geq \xi(H, A)$.

(40) $tall : \Pi H{:}T \Pi A{:}U(H).\ (A \to Prop)$

(41) $tall : \Pi H{:}T \Pi A \leq H.\ (A \to Prop)$

Note that, in modelling gradable adjectives as above, we have made use of the fact that applicability of gradable adjectives respects the usual subtyping relations (for example, if 'tall' can be applied to a type, it can be applied to any of its subtypes as well).

Multidimensional adjectives such as 'healthy' can also be modelled by means of subtype universes. For example, 'healthy' may be given the type (42) which can be rewritten as (43) in bounded quantification.

(42) $Healthy : \Pi A{:}U(Human).\ (A \to Prop)$

(43) $Healthy : \Pi A \leq Human.\ (A \to Prop)$

With healthy thresholds $\xi_i : \Pi A \leq Human.Nat$ with indexes i such as BP (for blood pressure), we have, for $A \leq Human$, $Healthy(A, x) = \bigwedge_i \chi_i(A, x)$, where χ_i's are the corresponding propositions: for instance, $\chi_{BP}(A, x) = BP(x) \leq \xi_{BP}(A)$, where A is a subtype of $Human$ examples of which include, for example, Boy and $Woman$.

Remark 5. As briefly described above, the approach to modelling gradability by means of subtype universes [37] results in simple semantic constructions and it is attractive and promising for modelling other linguistic features as well. It is worth remarking that most of the type constructions in the account in §3 are subtype universes to some extent. For example, CN_{PHY} is a subtype universe of those subtypes of PHY which are in CN as well. An in-depth comparative study would be interesting and may require further work. □

6 Conclusions and Future Work

In this paper, we have shown the use of MTT-semantics in the study of gradability. More specifically, we have shown that the rich typing mechanisms afforded by MTT-semantics can provide us with natural interpretations for both gradable and multidimensional adjectives/nouns. We have implemented the proposed accounts in the Coq proof-assistant and have checked their correctness. We have also briefly sketched an approach to modelling gradability by means of the recently studied notion of subtype universes. As mentioned, a comparative study of the two approaches to gradability is called for and left as future work.

One other issue that we have not looked at here and can be part of our future work is vagueness. In plain words, vagueness makes deciding what counts for something to be an X, where X is a gradable predicate (usually an adjective), difficult. There are three main problems associated with vagueness, the first one already mentioned and addressed in this paper: a) context dependency, b) the existence of borderline cases and c) the fact that vague adjectives (and predicates in general) give rise to the sorites Paradox. In the way our account stands, we cannot capture vagueness. We believe that this kind of problem needs to involve some kind of probabilistic reasoning. Indeed, a couple of researchers have pointed this out and have produced a body of research to this direction [22, 28, 4, 5]. At the moment, the authors do not know of any successful work in combining probability with dependent types and some new idea would be needed in order to study probabilistic type theories.[14]

[14] The work on probability in TTR (see, for example, [17]) studies probability in a set-theoretical system, because TTR [15, 16] is not a type theory, as the term is usually understood, but rather a set-theoretic notational system.

References

1. von Ahn, L., Dabbish, L.: Designing games with a purpose. Commun. ACM 51(8), 58–67 (2008)
2. Asher, N.: Lexical Meaning in Context: a Web of Words. Cambridge University Press (2012)
3. Bartsch, R., Vennermann, T.: Semantic Structures. Athenaum (1973)
4. Bernardy, J.P., Blanck, R., Chatzikyriakidis, S., Lappin, S.: A compositional bayesian semantics for natural language. In: Proceedings of the First International Workshop on Language Cognition and Computational Models. pp. 1–10 (2018)
5. Bernardy, J.P., Blanck, R., Chatzikyriakidis, S., Lappin, S., Maskharashvili, A.: Bayesian inference semantics: A modelling system and a test suite. In: Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (* SEM 2019). pp. 263–272 (2019)
6. Cardelli, L., Wegner, P.: On understanding types, data abstraction, and polymorphism. ACM Computing Surveys (4) (1985)
7. Chatzikyriakidis, S., Lafourcade, M., Ramadier, L., Zarrouk, M.: Type theories and lexical networks: Using serious games as the basis for multi-sorted typed systems (2017)
8. Chatzikyriakidis, S., Luo, Z.: Adjectival and adverbial modification: The view from modern type theories. Journal of Logic, Language and Information 26(1), 45–88 (2017)
9. Chatzikyriakidis, S., Luo, Z.: Identity criteria of common nouns and dot-types for copredication. Oslo Studies in Language 10(2) (2018)
10. Chatzikyriakidis, S., Luo, Z.: Formal Semantics in Modern Type Theories. Wiley/ISTE (2020)
11. Chatzikyriakidis, S., Luo, Z.: An account of natural language coordination in type theory with coercive subtyping. Constraint Solving and Language Processing 2012, LNCS 8114 (2013)
12. Chatzikyriakidis, S., Luo, Z., et al.: Modern perspectives in type-theoretical semantics, vol. 98. Springer
13. Church, A.: A formulation of the simple theory of types. J. Symbolic Logic 5(1) (1940)
14. Constantinescu, C.: Big eaters and real idiots: evidence for adnominal degree modification? In: Sinn und Bedeutung. vol. 17, pp. 183–200
15. Cooper, R.: Records and record types in semantic theory. Journal of Logic and Computation 2 (2005)
16. Cooper, R.: Context-Passing and Underspecification in Dependent Type Semantics. In: Chatzikyriakidis, S., Luo, Z. (eds.) Modern Perspectives in Type-Theoretical Semantics. Springer (2017)
17. Cooper, R., Dobnik, S., Lappin, S., Larsson, S.: Probabilistic type theory and natural language semantics. Linguistic Issues in Language Technology 10, 1–43 (2015)
18. The Coq Development Team: The Coq Proof Assistant Reference Manual (Version 8.1), INRIA (2007)
19. Coquand, T.: An analysis of Girard's paradox. In: Proc. 1st Ann. Symp. on Logic in Computer Science. IEEE (1986)
20. Geach, P.: Reference and Generality: An examination of some Medieval and Modern Theories. Cornell University Press (1962)

21. Girard, J.Y.: Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur. Ph.D. thesis, Université Paris VII (1972)
22. Goodman, N., Lassiter, D.: Probabilistic semantics and pragmatics: Uncertainty in language and thought. In: Lappin, S., Fox, C. (eds.) The Handbook of Contemporary Semantic Theory, Second Edition, pp. 655–686. Wiley-Blackwell, Malden, Oxford (2015)
23. Heim, I.: Degree operators and scope. In: Proceedings of SALT. vol. 10, pp. 40–64 (2000)
24. Kennedy, C.: Vagueness and grammar: The semantics of relative and absolute gradable adjectives. Linguistics and philosophy 30(1), 1–45 (2007)
25. Klein, E.: A semantics for positive and comparative adjectives. Linguistics and philosophy 4(1), 1–45 (1980)
26. Kratzer, A., Heim, I.: Semantics in generative grammar, vol. 1185. Blackwell Oxford (1998)
27. Lafourcade, M.: Making people play for lexical acquisition with the jeuxdemots prototype. In: SNLP'07: 7th international symposium on natural language processing. p. 7 (2007)
28. Lassiter, D., Goodman, N.: Adjectival vagueness in a Bayesian model of interpretation. Synthese 194, 3801–3836 (2017)
29. Lewis, D.: General semantics. Synthese 22(1), 18–67 (1970)
30. Luo, Z.: Computation and Reasoning: A Type Theory for Computer Science. Oxford University Press (1994)
31. Luo, Z.: Coercive subtyping. Journal of Logic and Computation 9(1), 105–130 (1999)
32. Luo, Z.: Type-theoretical semantics with coercive subtyping. Semantics and Linguistic Theory 20 (SALT20), Vancouver (2009)
33. Luo, Z.: Contextual analysis of word meanings in type-theoretical semantics. Logical Aspects of Computational Linguistics (LACL'2011). LNAI 6736 (2011)
34. Luo, Z.: Common nouns as types. In: Bechet, D., Dikovsky, A. (eds.) Logical Aspects of Computational Linguistics (LACL'2012). LNCS 7351 (2012)
35. Luo, Z.: Formal semantics in modern type theories with coercive subtyping. Linguistics and Philosophy 35(6), 491–513 (2012)
36. Luo, Z., Soloviev, S., Xue, T.: Coercive subtyping: theory and implementation. Information and Computation 223, 18–42 (2012)
37. Maclean, H., Luo, Z.: Subtype uiniverses. Post-proceedings of the 26th Inter. Conf. on Types for Proofs and Programs (TYPES20). Leibniz International Proceedings in Informatics, Vol. 188 (2021)
38. Martin-Löf, P.: An intuitionistic theory of types (1971), Manuscript.
39. Martin-Löf, P.: An intuitionistic theory of types: predicative part. In: H.Rose, J.C.Shepherdson (eds.) Logic Colloquium'73 (1975)
40. Martin-Löf, P.: Intuitionistic Type Theory. Bibliopolis (1984)
41. McConnell-Ginet, S.M.: Comparative constructions in English: A syntactic and semantic analysis (1973)
42. Montague, R.: Formal Philosophy. Yale University Press (1974), collected papers edited by R. Thomason
43. Murphy, G.: The big book of concepts (2004)
44. Nordström, B., Petersson, K., Smith, J.: Programming in Martin-Löf's Type Theory: An Introduction. Oxford University Press (1990)
45. Ranta, A.: Type-Theoretical Grammar. Oxford University Press (1994)

46. Sassoon, G.W., Fadlon, J.: The role of dimensions in classification under predicates predicts their status in degree constructions. Glossa: a journal of general linguistics 2(1) (2017)
47. Sassoon, G.: A typology of multidimensional adjectives. Journal of semantics (2012)
48. Van Benthem, J.: The logic of time: a model-theoretic investigation into the varieties of temporal ontology and temporal discourse, vol. 156. Springer Science & Business Media (2012)
49. Von Stechow, A.: Comparing semantic theories of comparison. Journal of semantics 3(1), 1–77 (1984)
50. Xue, T., Luo, Z.: Dot-types and their implementation. Logical Aspects of Computational Linguistics (LACL 2012). LNCS 7351 (2012)

A Coq code

A.1 Gradable Adjectives

```
(* Degree is type of names of degrees --*)
(*d: Degree corresponds to type D(d) *)
(* So, Degree is a Tarski universe! *)
(* Here is an example with three degrees. *)
Require Import Omega.
Inductive Degree: Set:= HEIGHT | AGE | IDIOCY |
Definition D (d: Degree):= nat.
Definition Height:= D(HEIGHT).
Definition Age:= D(AGE).
Definition Idiocy:= D(IDIOCY).

(* Universe CN_G of indexed CNs *)
Definition CN_G (_:Degree) := Set.
Parameter Human: CN_G(HEIGHT).
Parameter John Mary Kim : Human.
Parameter height: Human->Height.

(** Type of physical objects indexed with a degree**)
Parameter PHY : forall d: Degree, CN_G(d).

(* ADJ(D,A) of syntax of adjectives whose domain is A : CN_G(d) *)
Parameter ADJ: forall d:Degree, CN_G(d)->Set.
Parameter TALL SHORT: ADJ HEIGHT Human.
Parameter IDIOTIC: ADJ IDIOCY Human.
Parameter ENORMOUS: forall d: Degree,  ADJ d  (PHY(d)).

(* STND *)
Parameter STND: forall d:Degree, forall A:CN_G(d), ADJ d A -> D(d).

(* semantics of tall, taller_than *)
```

```
Definition tall (h:Human):= ge (height h) (STND HEIGHT Human TALL).
Definition taller_than (h1:Human) (h2:Human) := gt (height h2) (height h1).


Theorem TALLER:
taller_than Mary John /\ height Mary =
170 -> gt (height John) 170.
cbv. intro. omega. Qed.
Theorem trans:
taller_than Mary John /\ taller_than Kim Mary ->
taller_than Kim John.
cbv. intro. omega. Qed.


(**Definition for Idiot**)
Parameter IHuman : Idiocy -> CN_G(IDIOCY).
Definition Idiot:=
sigT(fun x: Idiocy=> (sigT (fun y: (IHuman x)
=> (ge x (STND IDIOCY Human IDIOTIC))))).
Definition enormous  (d:Degree)(A:CN_G(d))(d1: D d)
:= fun P: A =>  ge (d1)  (STND  d (PHY(d))(ENORMOUS d)).
Record enormousidiot: Set:= mkeidiot
{h1:> Idiot; EI1: enormous IDIOCY
(IHuman(projT1(h1)))(projT1(h1))(projT1(projT2(h1)))
/\  ge  (STND  IDIOCY (PHY(IDIOCY))(ENORMOUS IDIOCY))(STND IDIOCY Human IDIOTIC) }.

(*From enormous idiot it follows that there exists an idiot such
that their standard of idiocy is higher or equal to
the standard for  idiotic humans*)
Theorem ENORMOUS1:
enormousidiot -> exists H: Idiot,
projT1(H) >= STND IDIOCY (PHY IDIOCY) (ENORMOUS IDIOCY).
cbv. firstorder. Qed.

(*From enormous idiot it follows that there exists an idiot such
that their standard of idiocy is higher or equal to both
the standard for enormous idiots and the standard for idiotic
humans*)
Theorem ENORMOUS2:
enormousidiot -> exists H: Idiot,
projT1(H) >= STND IDIOCY (PHY IDIOCY) (ENORMOUS IDIOCY)
/\ projT1(H) >= (STND IDIOCY Human IDIOTIC).
cbv. firstorder. unfold Idiot in h2. exists h2. firstorder.
unfold enormous in H. firstorder. elim h2. intros. destruct p.
omega. Qed.
```

```
(*From enormous idiot it follows that there exists an idiot such
that their standard of idiocy is higher or equal to the standard
for enormous idiots and idiotic humans and also the standard for
enormous idiots is higher than that for idiotic humans*)
Theorem ENORMOUS3:
enormousidiot -> exists H: Idiot,
projT1(H) >= STND IDIOCY (PHY IDIOCY) (ENORMOUS IDIOCY)
/\ projT1(H) >= (STND IDIOCY Human IDIOTIC)
/\ STND IDIOCY (PHY IDIOCY) (ENORMOUS IDIOCY)
>= (STND IDIOCY Human IDIOTIC).
cbv. firstorder. unfold Idiot in h2. exists h2. firstorder.
unfold enormous in H. firstorder. elim h2. intros.
destruct p.
omega. Qed.
```

A.2 Multidimensional Adjectives

```
(*Dealing with multidimensional adjectives Health as an inductive
type where  the dimensions are enumerated. This is just an enumerated
type*)
Definition Degree:= Set.
Parameter Human: CN.
Parameter John: Human.
Inductive Health: Degree:= Heart|Blood|Cholesterol.
Parameter Healthy: Health -> Human -> Prop.
Definition sick:=fun y: Human => ~ (forall x : Health, Healthy x y).
Definition healthy:= fun y: Human => forall x: Health, Healthy x y.

Theorem HEALTHY:
healthy John -> Healthy Heart John /\ Healthy Blood John
/\ Healthy Cholesterol John.
cbv. intros. split. apply H.
split. apply H. apply H. Qed.

Theorem HEALTHY2:
healthy John -> not (sick John).
cbv. firstorder. Qed.

Theorem HEALTHY3:
(exists x: Health, Healthy x John) -> healthy John.
cbv. firstorder. Abort.

Theorem HEALTHY4:
(exists x: Health, not (Healthy x John)) -> healthy John.
```

```
cbv. firstorder. Abort.

Theorem HEALTHY5:
(exists x: Health, not (Healthy x John))  -> sick John.
cbv. firstorder.  Qed.

(*Multidimensional noun Artist*)
Inductive Art: Degree:= a1|a2|a3.
Set Implicit Arguments.
Parameter  DIM_CN : forall  D: Degree,   Human ->  D  -> Prop.
Record Artist: Set:= mkartist
{h:> Human; EI:  forall a: art,
(DIM_CN h a)}.
```