An Account of Natural Language Coordination in Type Theory with Coercive Subtyping*

Stergios Chatzikyriakidis[1] and Zhaohui Luo[2]

[1] Dept. of Computer Science, Royal Holloway, Univ. of London
Egham, Surrey TW20 0EX, U.K., Open University of Cyprus
stergios.chatzikyriakidis@cs.rhul.ac.uk
[2] Dept. of Computer Science, Royal Holloway, Univ. of London
Egham, Surrey TW20 0EX, U.K.
zhaohui.luo@hotmail.co.uk

Abstract. We discuss the semantics of NL coordination in modern type theories (MTTs) with coercive subtyping. The issue of conjoinable types is handled by means of a type universe of linguistic types. We discuss quantifier coordination, arguing that they should be allowed in principle and that the semantic infelicity of some cases of quantifier coordination is due to the incompatible semantics of the relevant quantifiers. Non-Boolean collective readings of conjunction are also discussed and, in particular, treated as involving the vectors of type $Vec(A, n)$, an inductive family of types in an MTT. Lastly, the interaction between coordination and copredication is briefly discussed, showing that the proposed account of coordination and that of copredication by means of dot-types combine consistently as expected.

1 Introduction

The literature on NL coordination dates back to [1] and a number of proposals have been put forth within the Montagovian tradition since then. However, a number of central issues as regards NL coordination have not been clarified yet. In this paper we depart from single-sorted versions of type theory found in Montague's work (as well as in most of the subsequent work within the same tradition) and employ a many-sorted modern type theory (MTT),[1] as proposed and studied for NL semantics in [6,7,8], to deal with two central issues in NL coordination. These issues concern the notion of conjoinable types, in effect the question of which NL elements can be coordinated, and non-Boolean conjunction, where a collective rather than the expected Boolean distributive reading of *and* arises.[2] The difference between collective and distributive readings is exemplified

* This work is supported by the research grant F/07-537/AJ of the Leverhulme Trust in the U.K.

[1] Examples of modern type theories include Martin-Löf's type theory (MLTT) [2,3], the Unifying Theory of dependent Types (UTT) [4] and the type theory implemented in the Coq proof assistant (pCIC) [5].

[2] We will use the term 'collective coordination' to refer to non-Boolean conjunction as the latter is described in analyses like [9] and [10].

D. Duchier and Y. Parmentier (Eds.): CSLP 2012, LNCS 8114, pp. 31–51, 2013.

in the examples below, where the same conjoined NP is interpreted distributively in (1) but collectively in (2):

(1) John and Mary came to the Party.

(2) John and Mary met at the Party.

We shall investigate how collective readings can be interpreted by means of the inductive family of types of vectors in an MTT.

We further discuss the interaction between dot-types for coordinated NPs. Dot-types have been proposed by Pustejovsky [11,12] for lexical interpretations of inherently polysemous words in phenomena such as co-predication (see, for example, [13]).[3] For example, *book* according to [11] can be represented with the dot-type PHY • INFO, a type whose objects have both a physical and an informational aspect. Dot-types have been formally introduced into MTTs with coercive subtyping [7,8] and a computational implementation of this account in Plastic[4] has also been done [17]. What we want to look at in this paper is the interaction between these types and coordination, i.e. examples of the following sort:

(3) The book and my lunch were sent by mistake to someone else.

(4) John picked up the newspaper and the book from the floor.

Given that the dot-types of the coordinated phrases are different and assuming that the NL coordination operate on the same types, we will have to explain how coordination is possible in these cases. The problem that arises in examples like (3) and (4) is that the individual NPs of the conjunction (e.g. *the book* and *my lunch* in (3) have different types (PHY • INFO for book and EVENT • PHY for lunch). The challenge is to account for the possibility of coordination in these cases by, at the same time, retaining the assumption that coordination operates on elements of the same type. As we shall see, the coercive subtyping mechanism actually allows us to combine the proposed typing for NL coordinations and the account with dot-types in a rather straightforward way.

2 Type Theory with Coercive Subtyping

In this paper, we employ modern type theories (MTTs) as the language for formal semantics. A brief introduction to the relevant features of MTTs are given here.

[3] See also [14] for a critique of the flaws in the various formalizations of dot-types in their original formulation as well as in much of the later work based on that.

[4] Plastic [15] is a *proof assistant*, an implementation of the modern types theory UTT [4] on the computer for formalised proof development. In the context of linguistic semantics, type theory based proof assistants such as Agda [16], Coq [5] and Plastic can be used to formalise and reason about the formal semantics based on MTTs.

An MTT has a number of differences when compared to Church's simple type theory as employed in Montague semantics [18,19]. One of the most important differences between an MTT and the simple type theory, is that the former can be regarded as *many-sorted* while the latter single-sorted. MTTs use many types to interpret Common Nouns (CN) such as $[\![man]\!]$ and $[\![table]\!]$, while single-sorted type theories use only one type (e) for the type of all entities (and another type t for logical truth values), with CNs being interpreted as predicates of type $e \to t$.

In Montague semantics, an Intransitive Verb (IV) is interpreted as a function from entities to truth values ($e \to t$), a type which is shared with CNs and intersective adjectives, and a quantified NP as of the type from properties to truth values ($(e \to t) \to t$).

In an MTT, types ('sorts') are used to interpret the domains to be represented. Some of them are:

- the *propositional types* (or logical propositions),
- the *inductive types* such as the type of natural numbers and Σ-types of dependent pairs,
- the inductive families of types such as the types $Vec(A, n)$ of vectors (or lists) of length n, and
- other more advanced type constructions such as *type universes*.

For example, within such a many-sorted logical system CNs are not interpreted as predicates as in the Montagovian tradition but rather as Types.[5] Theoretical motivation behind such a proposal has been provided by the second author based on the notion of identity criteria that CNs have according to [21].[6] Then given the interpretation of CNs as types, adjectives are interpreted as a predicate over the type interpreting the domain of the adjective. For example, the adjective *handsome* is interpreted as $[\![handsome]\!] : [\![man]\!] \to Prop$, with *Prop* being the type of logical propositions.[7] Modified CNs are then interpreted as Σ-types, the types that intuitively represent subset types but contain explicit proof objects.[8]

One of the important features of MTTs is the use of dependent types. Two examples of basic constructors for dependent types are Π and Σ. The Π-type corresponds to universal quantification in the dependent case and implication in

[5] In this paper we only deal with count and not mass nouns. For a first attempt at a treatment of mass nouns within the framework presented in this paper, see [20]. The treatment presented there seems compatible with the account presented in this paper. However, more careful examination is needed in order to see whether this is indeed true.

[6] See [20] for the exact details of this proposal and information on the identity criteria.

[7] MTTs have consistent internal logics based on the propositions-as-types principle [22,23]. For example, in a predicative type theory such as Martin-Löf's type theory, the logical proposition $A\&B$ corresponds to the product type $A \times B$ (a special case of Σ-type – see below) and a pair of a proof of A and a proof of B corresponds to an object of the product type. In an impredicative types theory such as UTT, logical propositions are similarly constructed as types but, furthermore, there is the type $Prop$ – a totality of logical propositions.

[8] See [6,7] for more details on this.

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 $\forall x{:}A.P(x)$. A Π-type degenerates to the function type $A \to B$ in the non-dependent case. In the case of Σ, 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 $A \times B$.

Coercive subtyping is an adequate subtyping mechanism for MTTs [24,25] and, in particular, it avoids a problem associated with the ordinary notion of subtyping (subsumptive subtyping), namely violation of canonicity [7]. Basically, coercive subtyping is an 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 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: $\mathfrak{C}_B[a]$ is legal (well-typed) and equal to $\mathfrak{C}_B[c(a)]$. For instance, one may introduce $[\![man]\!] < [\![human]\!]$. Then, if we assume that $[\![John]\!] : [\![man]\!]$ and $[\![shout]\!] : [\![human]\!] \to Prop$, the interpretation (6) of (5) is well-typed, thanks to the coercive subtyping relation between $[\![man]\!]$ and $[\![human]\!]$:[9]

(5) John shouts.

(6) $[\![shout]\!]([\![John]\!])$

Ending our discussion on the preliminaries of TTCS, we mention one further more advanced feature of the theory, that of universes. A universe is a collection of (the names of) types into a type [2]. This can be seen as a reflection principle where the universe basically reflects the types whose names are its objects. Universes are extremely useful in accounts of lexical semantics using MTTs. Specifically, universes can help semantic representations. To give an example, one may use the universe CN : $Type$ 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,

$$\overline{[\![man]\!]} : \text{CN} \quad \text{and} \quad T_{\text{CN}}(\overline{[\![man]\!]}) = [\![man]\!] .$$

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, $[\![man]\!]$.

Summarizing, we can say that the use of TTCS in interpreting NL semantics has given a number of interesting results and insights. These include an increased type granularity when compared to Montague Semantics given its type richness as well as an adequate subtyping mechanism.[10] Furthermore the interpretation

[9] It is important to note that function types are contravariant. Thus, even though $Man < Human$, it does not hold that $Man \to Prop < Human \to Prop$ but rather that $Human \to Prop < Man \to Prop$.

[10] This subtyping mechanism is however in line with canonicity and as such computationally more attractive [7].

of CNs as Types rather than predicates seems to be closer to the idea according to which the distinguishing feature of CNs, when compared to other parts of speech, is that only the former have what Geach called, criteria of identity [21]. The work presented in [20] provides strong arguments for supporting the non-predicate view on CNs based on Geach's identity criteria. The successful formalization [7] and subsequent implementation in Plastic [17] of *dot.types* is another achievement of this line of research given that no proper formalization of *dot.types* existed up to that point. The use of universes has been also proven fruitful in looking at alternative ways for defining the types for quantifiers and adverbs among others. Lastly, parts of the various proposals made in the aforementioned papers have been tested using the Coq interactive theorem prover. Some first results can be seen in [8] as well as in this paper. Current work of the first author concentrates on the use of Coq to prove valid NL theorems[11] as well as building universes relevant to NL semantics (e.g. CN, $LType$) in Plastic.[12]

3 Conjoinable Types

The issue of defining which NL types are conjoinable is of very high importance to all accounts of coordination proposed so far. Most of the accounts that have been proposed in the Montagovian tradition argue that conjoinable types are either of type t or of a function type that ends with t. The formalization might be different in individual cases but the core of the proposal is pretty much the same. The definition as given by Winter [26] is given below (using the term t-reducible):[13]

(7) τ is a t-reducible type iff $\tau = t$ or $\tau = \tau_1\tau_2$, where τ_1 is any type and τ_2 is a t-reducible type.

Such type of formulation allows coordination of categories ending in type t only, with type e conjunction not being possible. Thus, in these accounts proper name coordination is either assumed to involve type-lifting to quantifier type or proper names are assumed to be quantifiers in all cases. However, Partee & Rooth [10] propose a definition of e conjoinable types to deal with collective reading cases. Similar proposals have been made by Hoeksema [27]. Of course, an inductive definition of an e-conjoinable type does not make much sense given that at least in standard Montagovian semantics, the only e conjoinable types are the type of individual concepts, of type $s \to e$, i.e the type from indices to individuals, so the definition basically covers just one case.

Moving away from the simple type theory in Montague Grammar and using many-sorted MTTs, the first question to ask ourselves is how conjoinable categories can be defined. Well, the first question to be asked is which linguistic

[11] An example of this type is the following: if John and Mary met then John met Mary. Such theorems can be proved if the correct semantics are given in each case.

[12] This is not possible in Coq.

[13] We follow the notation as this is given in [26]. As such, $\tau_1\tau_2$ should be taken to mean $\tau_1 \to \tau_2$.

types can be conjoined? Surprisingly (or not) it seems that all linguistic categories can be conjoined. We first note the obvious cases of sentence and predicate coordination (8 and 9) to CN coordination (10):

(8) John walks and Mary talks.

(9) John walks and talks.

(10) A friend and colleague came.

Then, quantified NP coordination (11), quantifier coordination (12) and proper name (PN) coordination are possible (13):

(11) Every student and every professor arrived.

(12) Some but not all students got an A.

(13) John and Mary went to Italy.

Adverb conjunction(14), preposition conjunction(15), PP conjunction (16)

(14) I watered the plant in my bedroom but it still died slowly and agonizingly.

(15) I can do with or without you.

(16) The book is on the table and next to the chair.

Lastly, coordination of subordinate connectives is also possible (17):

(17) When and if he comes, you can ask him.

3.1 Universe of Linguistic Types

In this section we will propose a way to handle the flexibility NL coordination exhibits by using a MTT. The key idea behind the account we are going to propose is the notion of a universe.

A universe, as we have already mentioned at the end of §2, is a collection of (the names of) types into a type [2]. In the case of coordination, the universe CN of the types that we have used to interpret common nouns is far too small to capture the generality of the phenomenon. Given that all linguistic categories can be coordinated, the universe we need, has to be far more general than CN.

The idea is to introduce a type universe $LType$ of *Linguistic Types*. Intuitively, $LType$ contains (the names of) all types that are employed in linguistic semantics. Of course, in doing so, we will have to specifically say what we consider a linguistic type to be. Even though a thorough discussion of meticulously constructing the universe of linguistic types is out of the scope of this paper, we shall indicate positively what types may have names in the universe $LType$.[14] Figure 1 contains some of the introduction rules for $LType$, where we have used the so-called Russell-style formulation of a universe to omit the names of its objects. The informal explanations of the rules in Figure 1 are given below.

[14] We leave this most thorough and complete discussion of the universe $LType$ for future work. Besides the theoretical work behind such a proposal, we plan to implement the universe $LType$ in *Plastic*, which contrary to Coq, allows construction of universes.

$$\frac{}{PType : Type} \qquad \frac{}{Prop : PType} \qquad \frac{A : LType \quad P(x) : PType\ [x{:}A]}{\Pi x : A.P(x) : PType}$$

$$\frac{}{LType : Type} \qquad \frac{}{\text{CN} : LType} \qquad \frac{A : \text{CN}}{A : LType} \qquad \frac{A : PType}{A : LType}$$

Fig. 1. Some (not all) introduction rules for $LType$

- The type $Prop$ of logical propositions is a linguistic type. (It is of type $PType$[15] by the first rule and hence of type $LType$ by the last rule.)
- If A are linguistic types and P is an A-index family of types in $PType$, so is the Π-type $\Pi x{:}A.P(x)$. In particular, in the non-dependent case, if A_i are linguistic types, so is the arrow type $A_1 \to ... \to A_n \to Prop$. (It is of type $PType$ by repeated uses of the third rule and hence of type $LType$ by the last rule.)
- The universe CN (of types that interpret common nouns) is an object of type $LType$.
- If A interprets a common noun, then A is a linguistic type in $LType$. For example, the Σ-types that interpret modified CNs are in $LType$.

Other example types in $LType$ include the type of VP adverbs (18), quantifiers (19), argument-introducing and adjunct-introducing prepositions (20 and 21 respectively):

(18) $\Pi A : \text{CN}.\ (A \to Prop) \to (A \to Prop)$

(19) $\Pi A : \text{CN}.\ (A \to Prop) \to Prop$

(20) $\Pi A : \text{CN}.\ A \to (A \to Prop) \to (A \to Prop)$

(21) $\Pi A : \text{CN}.\ A$

To our knowledge, almost all types needed for representing the semantics of NLs can be handled with these rules. However, we shall leave the universe $LType$ *open* in the sense that we may introduce new types into it in the future.[16]

Having described the universe of linguistic types, we can now use it to describe the type of coordinators: every (binary) coordinator is of the following type:

(22) $\Pi A : LType.\ A \to A \to A$

For instance, the coordinator *and* is of the above type.

To give an example of how this type works, let us imagine three cases of coordination: PN coordination (John and George), propositional coordination (John runs and Mary drives) and VP coordination (John cycles and drives).

[15] $Ptype$ can be thought of as the universe of predicates. It is an intermediate universe used to build $LType$.

[16] For example, an extra introduction rule will be needed for vector types. See Remark 1 for such a rule. Formally, openness of a universe would imply that we do not impose an elimination rule for it. We omit the technical details here.

In the first case, *John* and *George* are of type $[\![Man]\!]$, so the A in this case is of type $[\![Man]\!]$ which is in $LType$ given that it is of type CN. In the case of propositional coordination, our A is of type $Prop$, which being a $PType$ is also an $LType$. In the third case our A is of type $[\![Man]\!] \rightarrow Prop$ which is also in $LType$. Similar considerations apply to all the cases from (8) to (17). Thus, this type captures the flexibility associated with coordination.[17] It is not difficult to see that all examples of coordination from (8) to (17) are predicted via the type given for coordination above.[18] However, what we need to discuss is examples where the rule proposed in (47) might seem to overgenerate or departs from the standard assumptions as these are made in the formal semantic literature.

3.2 Quantifier Coordination

The type for coordination we have proposed might be argued to overgenerate for cases involving coordination of two quantifiers like the ones shown below:

(23) # Some and every man came

(24) # No and some boy read

The above sentences seem to be generated via the rule we have proposed for coordination. Note, that this problem applies to all coordination accounts proposed. Given that quantifiers involve a function type ending in t, they should be conjoinable according to the accounts proposed in the Montagovian literature. No explicit discussion has been made of how cases like these are disallowed, so it would be good to see in more detail what is going on in these cases.

The basic problem is that some quantifiers seem to be able to be coordinated and some others do not. Between the cases of quantifiers that cannot be coordinated there are cases where adding a modal adverb between the coordinator and the second conjunct make a difference in acceptability. For example, adding the modal adverb *possibly* in (23) but not in (26) makes the sentence semantically felicitous:[19]

(25) Some and possibly every man came

(26) # No and possibly some boy read

[17] Of course, there are cases discussed in the literature where coordination of different categories seems to be possible. One such example is discussed in [28], where an adjective is coordinated with a NP: *John is either stupid or a liar*. We will not pursue an account here but we could note that an account in a similar vein to the one proposed by [29] where coordination even in this case operates on like and not on unlike categories is possible.

[18] All the examples have been checked using the Coq theorem prover [5]. The code can be found in the Appendix.

[19] As one of the reviewers correctly notes, cases of quantifier coordination that are always felicitous include cases involving a proper noun and a quantifier, e.g. *John and every girl left the party*. Obviously, if one wants to retain the assumption that similar types are conjoined, the proper noun is a generalized quantifier in the previous example.

For the rest of the cases, whether such sentences will be semantically felicitous depends largely on the type of coordination in each case (cf. the two examples below):

(27) # One and two of my students came to the party.

(28) One or two of my students came to the party.

Thus, it seems that in principle, coordination of quantifiers should be allowed, given that there are clear cases where this is possible. However, allowing coordination of quantifiers to be in principle possible, we will have to explain semantically infelicitous cases like (27). A way that can help us rule out a number of infelicitous semantic cases is to look at the semantics of the individual quantifiers in combination with the coordinator in each case. Let us take the example of the following sentence:

(29) # Some and no men arrived.

The quantifiers in the above example can be coordinated via the rule we have proposed. However, the semantics we get for the coordinated NP *some and no man* are the following, in effect a contradiction:

(30) $\exists x : [\![man]\!] .P(x) \wedge \sim \exists x : [\![man]\!] .P(x)$

We can quite plausibly assume that the contradictory semantics is the reason the conjunction is infelicitous in (33), especially when uttered out of the blue without any context. However, imagine the following situation: someone is lying and has stated that *no men arrived* on one occasion and that *some men arrived* on another. Then, the hearer might spot this contradiction and utter the following *some and no men arrived?*. In this case, *some and no men* is perfectly felicitous.[20] Moving on to cases of disjunction of the same quantifiers, we notice that no special context is needed in order for these to be felicitous. Examples of this quantifier combination are quite frequently found in NL:

(31) People with some or no academic training.

(32) This license may grant the customer the ability to configure some or no parts of the software themselves.

The semantics of *some or no x* in contrast to the semantics of *some and no x* do not give rise to a contradiction. To the contrary, they are always true under any interpretation. The example below depicts the semantics of *some or no men*:[21]

(33) $\exists x : [\![man]\!] \, P(x) \vee \sim \exists x : [\![man]\!] .P(x)$

[20] The same kind of example can be devised for cases like (25).

[21] This is the case assuming a logical interpretation of *some*. If the quantity implicature is taken into consideration, the quantifier combination is not always true.

Further examples of quantifiers that do not need a special context in order to be felicitous are *some but not all*, *more than three and less than five*. It might then be the case that quantifier combinations that are always false (in effect contradictions) need a special context in order to be felicitous, while quantifier combinations that do not fall into this category do not. Of course, there are obvious counterexamples to such a proposal, for example cases like *some and all* or *most and all*, which are of course infelicitous in the absence of any special context contrary to what we would expect in case what we say is true. However, quantifiers like *some* and *most* in NL carry a quantity implicature (see e.g. [30], [31] and [32] among others). The idea is that a speaker uttering *some* and not the stronger *all*, does that because he believes that substitution for the stronger value cannot be done *salva veritate*. For if the latter was true, he would have uttered the stronger *all*. A quantifier combination like *some and all* cancels out this implicature, so this might be the reason for the infelicitousness of this quantifier combination when uttered out of context. The same can be argued for the case of *most and all*. The issue requires more careful examination in order to see whether what we have argued is true or not. In particular, one has to check whether cases of quantifier combinations that are always false need the aid of some special context in order to be felicitous. Then, cases where infelicitousness arises unexpectedly must be shown to arise from other independent factors (like the quantity implicature for example). We believe that what we have proposed can produce a fruitful line of research as regards quantifier coordination but at least for this paper, we will not examine the issue any further. What is rather uncontroversial, no matter the assumptions we make as regards the interplay of quantifier coordination and the use of context, is that we need a rule for coordination that will in principle allow quantifier coordination. The rule we have proposed in (26) suffices for this reason.

Recapitulating, we propose a general rule for coordination which extends over a universe that contains all linguistic types, the universe $LType$. This rule is general enough to allow all types of coordination we find in NL. The rule might seem to overgenerate in the case of quantifier coordination, but as we have seen, in principle quantifier coordination should be allowed. The infelicitous cases (when uttered out of the blue) are attributed to the semantics of the individual quantifiers under the coordinator involved in each case.

3.3 Collective Coordination

The first thing we have to see is what is the prediction our typing rule proposed for coordination makes for these cases. But before doing this, we first have to discuss the typing of predicates like *meet* in their collective interpretation. Such predicates can be seen as one place predicates that take a plural argument and return a logical proposition (something in $Prop$ in our case), an assumption

already made in a number of the accounts within the Montagovian tradition (e.g. [26,33]). The plausible question is how plural arguments are going to be represented.

An interesting account of plurality within an MTT is presented by [34], where Martin-Löf's type theory is used for linguistic semantics. In this account, plural count nouns are interpreted using the type $List(A)$. Such an account is shown to give a uniform treatment of both singular and plural anaphora, being compatible with the classical type-theoretic treatment of anaphora, as this is given by [35].[22] In MLTT, $List(A)$ corresponds to the set of lists of elements of a set A. We will keep the intuition regarding the need to represent lists of objects but instead of using the inductive type $List(A)$, we will use the inductive family of types $Vec, n.\ Vec(A, n)$ and $List(A)$ are very much alike, with the difference being mainly that $Vec(A, n)$ involves an additional argument n of type Nat, which counts the number of the elements in a list (that is why they are called vectors):

(34) $Vec : (A : Type)(n : Nat)Type$

Intuitively, $Vec(A, n)$ is the type of lists $[a_1, ..., a_n]$ of length n, with $a_i : A$ $(i = 1, ..., n)$.[23]

Now, collective predicates can be given types in a more appropriate way. For example, the collective predicate *meet* can be given the following type:

(35) $\Pi n : Nat.\ Vec(\llbracket human \rrbracket, n + 2) \rightarrow Prop$

Please note that, as $n \geq 0$, an object of type $Vec(\llbracket human \rrbracket, n + 2)$ has length of at least 2 or longer – this means that *meet* can only be applied to at least two people, but not less. Such more exact requirements are captured in typing by means of the inductive families like $Vec(A, n)$.[24]

Now, let us explain how to interpret sentences like (36):

(36) John and Mary met.

[22] Another interesting account of plurals is given by [36] using Girard's system F. It is shown that the basic properties of plurals can be effectively accounted for by using a second-order system like Girard's system F.

[23] See, for example, Chapter 9 of [4] for the formal definition of $Vec(A, n)$. We omit the formal details here.

[24] As suggested by an anonymous reviewer, one might consider using finite types (see, for example, Appendix B of [20]) instead of the vector type. However, a lot of the cases discussed here (e.g. the respectively) involve ordering of elements, in which case vector types seem more well suited (at least for these cases). However, for other cases of plurality where ordering is not needed (or needs to be avoided), using finite types seems to be a good suggestion and as such will be considered for a more general account of plurality in TTCS.

The above typing of *meet* assumes that the typing for humans can distinguish the number for the plural cases. We propose the following rule for collective *and* should be given the following type:[25]

(37) $\Pi A : LType.\ \Pi n, m : Nat.\ Vec(A, n) \to Vec(A, m) \to Vec(A, n + m)$

Assuming that $J : Vec([\![human]\!], 1)$ and $M : Vec([\![human]\!], 1)$, then J *and* M is of type $Vec([\![human]\!], 2)$. In order to type phrases like *John and Mary*, we need to introduce the following coercions, for every type A:

$$A <_c Vec(A, 1)$$

where the coercion c maps any a to $[a]$, the vector with only one object a. With $John : [\![man]\!] < [\![human]\!]$ and $Mary : [\![woman]\!] < [\![human]\!]$, we have that *John and Mary* is interpreted as of type $Vec([\![human]\!], 2)$ and therefore, the above sentence (36) gets interpreted as intended.

However, we are not done yet with collective predication, given that we have not yet discussed cases involving quantifiers. Such a case is shown below:

(38) Three men and five women met.

If we assume that the numeral quantifiers 1, 2, 3, ... are given the following type:

$$n \ : \ \Pi A{:}\textsc{cn}.\ (Vec(A, n) \to Prop) \to Prop,$$

then we have[26]

$$\begin{aligned}
[\![three\ man]\!] &= [\![three]\!]([\![man]\!]) \\
&: (Vec([\![man]\!], 3) \to Prop) \to Prop \\
&< (Vec([\![human]\!], 3) \to Prop) \to Prop \\
[\![five\ women]\!] &= [\![five]\!]([\![women]\!]) \\
&: (Vec([\![women]\!], 5) \to Prop) \to Prop \\
&< (Vec([\![human]\!], 5) \to Prop) \to Prop
\end{aligned}$$

and

$$\begin{aligned}
&[\![three\ man\ and\ five\ women]\!] \\
&= [\![and]\!]([\![three]\!]([\![man]\!]),\ [\![five]\!]([\![women]\!])) \\
&: (Vec([\![human]\!], 8) \to Prop) \to Prop
\end{aligned}$$

This can now be applied to the semantics of meet (more precisely, $[\![meet]\!](6)$) to form the semantics of (38).

[25] Note that A extends over *Ltype* since different vector types might be needed and not just vector cases with $A : CN$. See the following discussion on quantifiers and *respectively*. It is not clear whether this rule is too strong or not. This is an issue that needs to be further researched. In any case, if we want to provide a typing for collective *and* which covers all the cases we are interested in and furthermore avoids overloading, we need a bigger universe than CN. Whether the universe needed in this case is *LType* or something considerably smaller is a subject open to further inquiry and debate.

[26] Here, we have assumed that, if $A <_c B$, then $Vec(A, n) <_{map(c} Vec)(B, n)$, where $map(c)$ maps $[a_1, ..., a_n]$ to $[c(a_1), ..., c(a_n)]$.

Remark 1 (Vector types and LType). The meticulous reader might have noticed that the introduction rules we proposed for $LType$ do not include vector types. Since vector types are employed in NL semantics (at least in our account), then these types should be included in $Ltype$. For example, one could consider the following introduction rule:

$$\frac{A : LType \; n{:}Nat}{Vec(A, n) : LType}$$

Under this rule, vector types are now part of $LType$. Again, we note that the task of meticulously constructing the universe $LType$ will not be undertaken in this paper. However, such a rule is indicative of how vector types can be included in the constructed $LType$ universe.

One further welcoming extension of the account proposed is a straightforward explanation of the way the reciprocal *each other* functions in English. Verbs like *meet* are reciprocal predicates in the sense that they do not need an overt reciprocal to give rise to a reciprocal reading (basically what we have been calling the collective reading so far). For non-reciprocal predicates, there is the possibility of getting these readings via the use of *each other*. The idea is that *each other* in English turns a transitive predicate into an intransitive one whose sole argument is a vector whose length is at least 2:

(39) $[\![each\ other]\!] : \Pi A : \text{CN}, \Pi n : Nat.\ (A \to A \to Prop) \to (Vec(A, n + 2) \to Prop)$

Lastly collective coordination cases involving *respectively* can also be accounted for by using a similar reasoning. In particular, one might view *respectively* as big functor which takes two vector arguments and returns a proposition. Specifically, the following typing seems to be appropriate for *respectively*:

(40) $[\![respectively]\!] : \Pi A : \text{CN}.\ Vec(A, n) \to Vec((A \to Prop), n) \to Prop$

Basically, *respectively* takes one $VecA$ argument and one $Vec(A \to Prop)$ argument. Such a typing is able to deal with most of the cases involving *respectively* shown below:

(41) Stergios and Zhaohui cycle and drive respectively.

(42) Stergios and Zhaohui are Greek and Chinese respectively.

(43) Stergios and Zhaohui prepared and taught a course respectively.

In (41), the second vector argument is the result of coordination of the predicates *cycle* and *drive* of type $[\![Human]\!] \to Prop$. With this rule for collective *and*, let us get back to the examples in (41) to (43). In (41) and (42), the second vector argument is of type $Vec(A \to Prop)$ given that both intransitive verbs and adjectives are defined as predicates $(A \to Prop)$. This is formed from coordinating intransitive verbs and adjectives *respectively*. In (43), things are a little bit more complicated, since some functional application is needed first in order for the desired type

to be reached. Specifically, we need to somehow apply *prepared* and *taught* to *a course* and then coordinate the result of these two applications (*prepared a course* and *taught a course*) to get the $Vec(A \rightarrow Prop)$ argument. The exact details of how this is done will not be fleshed out in this paper. The same goes for more complicated respectively-structures involving ditransitive verbs like the ones shown below:

(44) Stergios and Zhaohui sent a paper to Robin and Tao respectively.

(45) Stergios and Zhaohui sent Robin beer and wine respectively.

Remark 2. A further interesting issue involves inferencing arising from collective coordination. The following examples constitute classic NLI cases associated with collective coordination:

(46) Stergios and Zhaohui met ⇒ Stergios met Zhaohui and Zhaohui met Stergios

(47) Stergios and Zhaohui hit each other ⇒ Stergios hit Zhaohui and Zhaohui hit Stergios

(48) Stergios and Zhaohui are Greek and Chinese respectively ⇒ Stergios is Greek

In order to deal with these inferences, the definitions provided for collective *meet* and *each other*, besides typing information must provide information that will somehow allow inferences like the above to arise. For example, assuming a transitive version of *meet*, defined as being of type $[\![human]\!] \rightarrow [\![human]\!] \rightarrow Prop$, we can use a definition for collective *meet* that besides typing information will further provide information about its relation with transitive *meet*. The same idea is applied to the definition of *each other*, the only difference being that *each other* basically defines a relation over an arbitrary P of type $[\![human]\!] \rightarrow [\![human]\!] \rightarrow Prop$ and not a specific predicate as in the case of collective *meet*. Similarly, *respectively* takes two vector arguments and applies each element of the $Vec(A)$ argument to each element of the $Vec(A \rightarrow Prop)$. This kind of approach can be found in the appendix where the definitions for *meet*, *each other* and *respectively* needed to capture inferences like the ones in (46), (47) and (48) are shown. The inferences discussed are also shown to be derivable using Coq in the form of theorems. The idea is to treat NLIs as valid Coq theorems. The proof steps needed for proving the inferences in (46), (47) and (48) are also given in the appendix. It has to be noted that this line of research, i.e. using an interactive theorem prover like Coq in order to deal with NLIs, seems to us quite promising for at least two reason: a) checking the predications that an MTT like the one presented here makes with respect to NLIs and b) explore ways in which an interactive theorem prover can help automatic theorem proving when dealing with NLIs. This last issue, we believe, is of vital importance, given that interactive theorem proving might reveal strategies for proving certain NLIs that can otherwise go unnoticed. In the long run, these collection of strategies gathered from the 'experience' of interactive theorem proving can be potentially used to make automation more effective. However, a proper discussion of NLIs and their implementation as Coq theorems are out of the scope of this paper.

4 Interaction of Coordination and Copredication

Dot-types have been successfully formalized in MTTs with coercive subtyping [7,8], and an implementation of them in the proof assistant Plastic also exists [17]. We first summarize the account proposed for dot-types and then proceed and discuss the interaction between dot-types and coordination. We will use *book* as our prototypical example in presenting the account.

Book is assumed to be a dot-type having both a physical and an informational aspect. The type-theoretic formalization of this intuition proceeds as follows. Let PHY and INFO be the types of physical objects and informational objects, respectively. One may consider the dot-type PHY•INFO as the type of the objects with both physical and informational aspects. A *dot-type* is then a subtype of its constituent types: PHY • INFO < PHY and PHY • INFO < INFO. A book may be considered as having both physical and informational aspects, reflected as:

$$(*) \qquad [\![book]\!] < \text{PHY} \bullet \text{INFO}.$$

[27]

Now, consider the following sentence:

(49) John picked up and mastered the book.

We assume the following typing for *pick-up* and *master* respectively:

$$[\![pick\ up]\!] : [\![human]\!] \to \text{PHY} \to Prop$$
$$[\![master]\!] : [\![human]\!] \to \text{INFO} \to Prop$$

Because of the above subtyping relationship $(*)$ (and contravariance of subtyping for the function types), we have

$$\begin{aligned} [\![pick\ up]\!] \ &: \ [\![human]\!] \to \text{PHY} \to Prop \\ &< \ [\![human]\!] \to \text{PHY} \bullet \text{INFO} \to Prop \\ &< \ [\![human]\!] \to [\![book]\!] \to Prop \end{aligned}$$

$$\begin{aligned} [\![master]\!] \ &: \ [\![human]\!] \to \text{INFO} \to Prop \\ &< \ [\![human]\!] \to \text{PHY} \bullet \text{INFO} \to Prop \\ &< \ [\![human]\!] \to [\![book]\!] \to Prop \end{aligned}$$

Therefore, $[\![pick\ up]\!]$ and $[\![master]\!]$ can both be used in a context where terms of type $[\![human]\!] \to [\![book]\!] \to Prop$ are required and the interpretation of the sentence (49) can proceed as intended.

The first case of interaction has already been introduced and involves examples like (49). It is easy to see how this is going to be predicted given what we have said.[28]

[27] See [17] for a detailed exposition of how dot-types are formalized in MTT with coercive subtyping.

[28] See [7,8] for an account of this.

The next step is to take a look at examples where two words with dot-types are coordinated. Such an example is shown below:

(50) The book and my lunch were sent by mistake to someone else.

In the above example we have two dot-types involved, PHY • INFO and PHY • EVENT, representing the types for *book* and *lunch* respectively. Let us see whether the rule for coordination we have along with the treatment of dot-types will give us the correct results.

We need to coordinate the two NPs:

$$[\![the\ book]\!] : [\![book]\!] \quad \text{and} \quad [\![my\ lunch]\!] : [\![lunch]\!] .$$

Furthermore, the passive *send* is of the following type:

$$[\![send_{pass}]\!] : \text{HUMAN} \rightarrow \text{PHY} \rightarrow Prop.$$

Now, because

$$[\![book]\!] < \text{PHY} \bullet \text{INFO} < \text{PHY}$$

$$[\![lunch]\!] < \text{PHY} \bullet \text{EVENT} < \text{PHY}$$

the above sentence (50) can be interpreted as intended. In other words, the coercive subtyping mechanism interacts with that for coordination correctly.

5 Conclusions

In this paper we presented an account of NL coordination using Type Theory with Coercive Subtyping. The issue of conjoinable types was taken care of by proposing an inductive type for coordination which extends over the universe of Linguistic Types, called *LType*. This type has been shown to be sufficient to explain the flexibility of NL coordination. We argued that a rule for NL coordination should in principle allow quantifier coordination and showed that the infelicitous quantifier combination cases are due to the inherent semantics of the quantifier combination under the coordinator in each case, along with general pragmatic implicatures associated with quantifiers (e.g. the quantity implicature for quantifiers *some* and *most*). Collective coordination was accounted for, assuming that collective predicates take one vector argument representing plurality. A second rule for collective *and* was proposed which takes two vector arguments of n and m length and produces a vector type of length $n + m$. Furthermore, the issue of inferences arising from collective predication was briefly discussed and a proposal on how to define collective predicates as well as the reflexive *each other* and the adverb *respectively* was proposed. These proposals have been implemented in Coq and a proof of these NLIs within in the form of theorems is possible. Lastly, the interaction of *dot.types* with coordination was briefly discussed. It was shown that the coordination account proposed in combination with the co-predication account as this was given in [8] gives the correct predictions.

Future work includes the further refinement of the universe $LType$. The Coq proof assistant does not allow one to introduce new universes and, as a consequence, we had to use some existing universe instead, which is not quite faithful. Some other proof assistants, like Plastic [15], allow one to introduce new universes. We are going to use such systems for this work so that the universe of linguistic types may get implemented properly.

Acknowledgement. Thanks go to the reviewers of the paper for useful comments of its earlier version.

References

1. Montague, R.: The proper treatment of quantification in ordinary English. In: Hintikka, J., Moravcsik, J., Suppes, P. (eds.) Approaches to Natural Languages (1973)
2. Martin-Löf, P.: Intuitionistic Type Theory. Bibliopolis (1984)
3. Nordström, B., Petersson, K., Smith, J.: Programming in Martin-Löf's Type Theory: An Introduction. Oxford University Press (1990)
4. Luo, Z.: Computation and Reasoning: A Type Theory for Computer Science. Oxford Univ Press (1994)
5. The Coq Development Team: The Coq Proof Assistant Reference Manual (Version 8.3), INRIA (2010)
6. Ranta, A.: Type-Theoretical Grammar. Oxford University Press (1994)
7. Luo, Z.: Type-theoretical semantics with coercive subtyping. Semantics and Linguistic Theory 20 (SALT20), Vancouver (2010)
8. Luo, Z.: Contextual analysis of word meanings in type-theoretical semantics. In: Pogodalla, S., Prost, J.-P. (eds.) LACL 2011. LNCS, vol. 6736, pp. 159–174. Springer, Heidelberg (2011)
9. Keenan, E., Faltz, L.: Logical Types for Natural Language. In: Department of Linguistics. UCLA (1978)
10. Partee, B., Rooth, M.: Generalized conjunction and type ambiguity. In: Bauerle, R., Schwarze, C., von Stechow, A. (eds.) Meaning, use, and Interpretation of Language. Mouton De Gruyter (1983)
11. Pustejovsky, J.: The Generative Lexicon. MIT (1995)
12. Pustejovsky, J.: Meaning in Context: Mechanisms of Selection in Language. Cambridge Press (2005)
13. Asher, N.: Lexical Meaning in Context: a Web of Words. Cambridge University Press (2012)
14. Bassac, C., Mery, B., Retoré, C.: Towards a type-theoretical account of lexical semantics. Journal of Logic Language and Information 19, 229–245 (2010)
15. Callaghan, P., Luo, Z.: An implementation of LF with coercive subtyping and universes. Journal of Automated Reasoning 27, 3–27 (2001)
16. The Agda proof assistant (version 2), `http://appserv.cs.chalmers.se/users/ulfn/wiki/agda.php?` (2008)
17. Xue, T., Luo, Z.: Dot-types and their implementation. In: Béchet, D., Dikovsky, A. (eds.) LACL 2012. LNCS, vol. 7351, pp. 234–249. Springer, Heidelberg (2012)
18. Church, A.: A formulation of the simple theory of types. J. Symbolic Logic 5 (1940)
19. Montague, R.: Formal Philosophy. Yale University Press (1974)

20. Luo, Z.: Common nouns as types. In: Béchet, D., Dikovsky, A. (eds.) LACL 2012. LNCS, vol. 7351, pp. 173–185. Springer, Heidelberg (2012)
21. Geach, P.: Reference and Generality: An examination of some Medieval and Modern Theories. Cornell University Press (1962)
22. Curry, H., Feys, R.: Combinatory Logic, vol. 1. North-Holland (1958)
23. Howard, W.A.: The formulae-as-types notion of construction. In: Hindley, J., Seldin, J. (eds.) To H. B. Curry: Essays on Combinatory Logic. Academic Press (1980)
24. Luo, Z.: Coercive subtyping. Journal of Logic and Computation 9, 105–130 (1999)
25. Luo, Z., Soloviev, S., Xue, T.: Coercive subtyping: theory and implementation (2012) (submitted manuscript)
26. Winter, Y.: A unified semantic treatment of singular NP coordination. Linguistics and Philosophy 19, 337–391 (1996)
27. Hoeksema, J.: The semantics of non-boolean "and". Journal of Semantics 6, 19–40 (1998)
28. Moortgat, M.: Categorial type logics. In: van Benthem, J., ter Meulen, A. (eds.) Handbook of Logic and Language. Elsevier/Mit press (1997)
29. Morril, G.: Type Logical Grammar: Categorial Logic of Signs. Kluwer Academic Publishers (1994)
30. Horn, L.: A Natural History of Negation. University of Chicago Press (1989)
31. Horn, L.: The border wars: a neo-Gricean perspective. In: von Heusinger, K., Turner, K. (eds.) Where Semantics Meets Pragmatics, pp. 21–46. Elsevier, Amsterdam (2006)
32. Geurts, B.: Quantity Implicatures. Cambridge University Press (2010)
33. Winter, Y.: Flexibility Principles in Boolean Semantics. MIT Press, New York (2002)
34. Boldini, P.: The reference of mass terms from a type-theoretical point of view. In: Paper from the 4th International Workshop on Computational Semantics (2001)
35. Sundholm, G.: Proof theory and meaning. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic III: Alternatives to Classical Logic. Reidel (1986)
36. Retoré, C.: Variable types for meaning assembly: a logical syntax for generic noun phrases introduced by 'most'. Recherches Linguistiques de Vincennes 41, 83–102 (2012)

A Implementations in Coq

We use Coq's predifined Type Universe instead of $LType$. Bvector is needed for vectors (Require Import Bvector). The coercion $A <_c Vec(A, 1)$ for proper nouns is not possible in Coq, so we have to introduce the coercions as separate entries.

A.1 Conjoinable Types

```
(* Categories*)
Definition CN := Set.
Parameters Bank Institution Human Man Woman Object Animal OObject: CN.
Parameter John Stergios Zhaohui : Man.
Parameter Mary: Woman.
```

```
Axiom mh : Man->Human.  Coercion mh : Man >-> Human.
Axiom wh : Woman->Human.  Coercion wh : Woman >-> Human.
Axiom ha: Human-> Animal. Coercion ha: Human>-> Animal.
Axiom ao: Animal->Object. Coercion ao:  Animal>-> Object.
Axiom ooo: OObject-> Object. Coercion ooo: OObject>->Object.
Parameter walk: Animal ->Prop.
Parameter talk cycle drive: Human->Prop.
Parameter attack killed: Animal -> Animal -> Prop.
Parameter If when: Prop-> Prop-> Prop.
Parameter the some most all: forall A:CN, (A->Prop)->Prop.
Parameter die: OObject-> Prop.
Parameter slowly agonizingly: forall A:CN, (A->Prop)->(A->Prop).
Parameter And: forall A:Type, A->A->A. (*Predefined Type universe
 instead of LType*)
(*Cases to check*)
Check And Man (Stergios)(Zhaohui)
Check And Man (Stergios)(Mary) (*does not go through because Mary:Woman*)
Check And Human (Stergios)(Mary) (*this is fine given Woman Man<Human*)
Check And ((Human->Prop)->Prop) (some Man)(some Woman) (*Quantified NP*)
Check And (forall A: CN, (A->Prop)->Prop) (some)(all).(*Quantifier*)
Check And (Human->Prop) (cycle)(drive) (*VP*)
Check And (forall A:CN, (A->Prop)->(A->Prop))(slowly)(agonizingly).
(*VP adverb*)
Check And (Prop->Prop->Prop) (If)(when) (*subordinate conjunction*)
```

A.2 Collective Coordination

```
Require Import Bvector.
Variables n m: nat.
Parameter meetc:forall n:nat, vector Human(n+2)->Prop.(*collective meet*).
Parameter John1 George1: vector Human 1.(*coercions do not work with
vectors so we use Human instead of Man here*)
(*Unit type for collective And*)
Inductive OneAndc : Set := Andc.
Definition AndSem1 :=  forall A: CN,forall n:nat,forall m:nat, vector (A)(n)
->vector(A)(m)->vector(A)(n+m).
Definition AndSem2 :=forall A: CN,forall n:nat,forall m:nat, ((vector A n)->Prop)
->Prop->
((vector A m)->Prop)->Prop->((vector A (n+m))->Prop).
Parameter Andc1 : AndSem1.
Parameter Andc2 : AndSem2.
Definition a1 (a:OneAndc) : AndSem1 := Andc1.  Coercion a1 : OneAndc >-> AndSem1.
Definition a2 (a:OneAndc) : AndSem2 := Andc2.  Coercion a2 : OneAndc >-> AndSem2.
*Some interesting cases to check*
Check meetc 0 ((Andc:AndSem1 (Human)(1)(1)(John1)(George1)) (*John and George met,
with both George and John of lower type*)
```

A.3 Co-predication

```
(* Phy dot Info *)
Parameter Phy Phy1 Info : CN.(*Phy1 should be taken to be the same as Phy*)
Record PhyInfo : CN := mkPhyInfo { phy :> Phy; info :> Info }.
Parameter Book: PhyInfo.
Parameter Event : CN.
Record EventPhy : CN := mkEventPhy { event :> Event; phy1 :> Phy1}.
(*Phy1 is used  because Phy cannot be used twice*)
Parameter lunch: EventPhy.
Axiom po: Phy->Object. Coercion po:Phy>->Object.
Axiom pp: Phy1->Phy. Coercion pp: Phy1>->Phy.(*We introduce this coercion
to mean that the two Phy and Phy1 are the same.*)
Parameter was_given_to_someone_else: Object->Prop.
*Interesting case to check*)
Check was_given_to_someone_else (And(Object)(Book)(lunch)).
```

A.4 Collective Coordination Inferences

```
Parameter hit meettr: Human->Human->Prop.
Parameter meettr: Human -> Human -> Prop
Definition meetc:= fun v : vector Human 2 =>exists x: Human, exists y:Human,
meettr x y /\ meettr y x /\ v=(Human::x)1((Human::y)0(Vnil Human))
Definition each_other:=fun x:vector Human 2,fun P:Human->Human->Prop=>
exists k : Human,exists l: Human,P k l/\P l k/\(Human::k)1((Human :: l)0
(Vnil Human))=x.
Definition respectively:=fun A:CN,fun x:vector A 2,fun P:vector(A ->Prop)
2=>exists k:A,exists l:A,exists Q:A->Prop,exists R:A->Prop,Q k/\R l/\(A::k)
1((A::l)0(Vnil A))=x/\((A -> Prop)::Q)1(((A->Prop)::R)0(Vnil(A->Prop)))=P.

*Proof of Stergios and Zhaohui hit each other -> Stergios hit Zhaohui*
Theorem each_other1:each_other((Human::Stergios)1((Human::Zhaohui)0(Vnil
Human))) hit -> hit(Stergios)(Zhaohui).
intros.
unfold each_other in H.
destruct H.
destruct H.
destruct H.
destruct H0.
inversion H1.
replace Stergios with x.
replace Zhaohui with x0.
assumption.

*Proof of Stergios and Zhaohui met -> Stergios met Zhaohui*
Theorem MEET1:meetc ((Human::Stergios)1((Human::Zhaohui)0(Vnil Human))) ->
meettr(Stergios)(Zhaohui).
intros.
unfold meetc in H.
```

```
destruct H.
destruct H.
destruct H.
destruct H0.
inversion H1.
replace Stergios with x .
replace Zhaoui with x0 .
assumption.

*Proof of John and Stergios hit Zhaohui and George respectively->Stergios hit
Zhaohui*
Coq< Theorem resp: respectively Human((Human:: John)1((Human::Stergios)0(Vnil
Human)))(((Human->Prop)::(hit Zhaohui))1(((Human->Prop)::(hit George))0(Vnil
(Human->Prop))))-> hit Zhaohui John.
intros.
destruct H.
destruct H.
destruct H.
destruct H.
destruct H.
destruct H0.
destruct H1.
inversion H1.
inversion H2.
rewrite <- H4.
rewrite <- H6.
assumption.
```