Istituto per la Ricerca Scientifica e Tecnologica
I 38100 Trento – Loc. Panté di Povo – tel. 0461–814444
Telex 400874 ITCRST – Telefax 0461–810851

A Many Sorted Natural Deduction

Alessandro Cimatti
Fausto Giunchiglia
Richard Weyhrauch

April 1994
Technical Report # 9404-08

Publication Notes: Presented at the ECAI-94 workshop "From Theorem Provers to Mathematical Assistants: Issues and Possible Solutions", August 9, 1994, Amsterdam, The Netherlands.

Istituto Trentino di Cultura

A many sorted natural deduction*

A. Cimatti[1], F. Giunchiglia[1,2], R. W. Weyhrauch[3]

cx@irst.itc.it fausto@irst.itc.it rww@ibuki.com

[1]IRST, Povo, 38050 Trento, Italy
[2]DISA, University of Trento, Via Inama 5, 38100, Trento, Italy
[3]IBUKI, PO Box 1627, Los Altos, Ca 94022, USA

Abstract

The goal of this paper is to motivate and define yet another sorted logic, called SND. All the previous sorted logics which can be found in the Artificial Intelligence literature have been designed to be used in (completely) *automated deduction*. SND has been designed to be used in *interactive theorem proving*. Because of this shift of focus, SND has been designed to satisfy three innovative design requirements; that is: it is defined on top of a natural deduction calculus, and in a way to be a definitional extension of such calculus; and it is implemented on top of its implementation. In turn, because of this fact, SND has various innovative technical properties; among them: it allows us to deal with free variables, it has no notion of wellsortedness and of wellsortedness being a prerequisite of wellformedness, its implementation is such that, in the default mode, the system behaves exactly as with the original unsorted calculus.

*The formal system presented here was originally defined by R.W. Weyhrauch and implemented in the original version of `FOL` [Wey77]. The first and second author have re-implemented it in the new version of `FOL`. All the authors have contributed to the formal analysis done in this paper. The authors thank Luciano Serafini for his unvaluable technical support. Alex Simpson and Christoph Weidenbach are also thanked for the useful feedback and discussions.

1 Introduction

In interactive theorem proving, the system is used as a proof assistant, more than as an inference engine able to solve difficult problems on its own. The user engages in a two way conversation with the system, where he suggests proof strategies, the system listens to his suggestions, acts on their basis, and then returns to him with an answer. Examples of answers are: the assertion of a new theorem or of a new set of subgoals; a loop, possibly stopped because of some resource limitation; a suggestion for some new search strategy; and so on. Most often, the potentially combinatorial explosive decisions are left to the judgement of the user, while the system performs all the mechanical steps, which can also be very complex, and book-keeping. This interaction can be very long and tedious, and for long proofs, like those needed in, e.g., hardware or software verification, can last days or even months.

A crucial issue becomes therefore that of defining a meaningful and natural interaction, where the problem can be stated and solved according to the user's intuitions. This makes it easier for the user to understand the internal workings of the system, to provide the most effective suggestions, and, ultimately, to save huge amounts of time. A solution to this problem involves many factors and can be faced at many different levels. Following [GPT94], inside an interactive reasoning system, we can distinguish three major architectural components, namely the logic, the control, and the interaction, where the logic defines the basic inference steps, the control defines how the basic inference steps can be put together to obtain possibly very complex strategies, and the interaction defines how the system interacts with the user.

The core component of an interactive reasoning system is the logic. Everything else is implemented on top of it, and defined in terms of it. A wrong choice of the logic may make the rest useless syntactic sugar. Consider for instance resolution. Resolution uses formulas in a specific normal form thus avoiding the use of many logical connectives; and has only one inference rule. This makes it very uniform and suited for the mechanization of automated theorem provers. However it also makes it not suited for interactive theorem proving: the proofs are too uniform, have little structure, and are therefore hard to understand. A far more suited calculus is Natural Deduction (ND from now on) or some sequent presentation of ND [Gen69; Pra65]. ND allows for the use of all the logical connectives (which in [Wey77] have been extended with `if` term constructors and `if` formula constructors), and for a very natural style of deduction where each connective is manipulated by an intro-

duction and an elimination rule. It is a fact that the most successful interactive theorem provers (and most of those used in practical applications) use (a sequent presentation of) ND, often with the addition of some decision procedures for some particularly interesting classes, e.g. propositional logic, the $\forall\exists$ class, linear arithmetic, Presburger arithmetic (see for instance [Wey77; GMW79; Pau79; CAB+86; Giu92; ORSvH95]). The use of decision procedures is motivated by the desire to have bigger inference steps and/or to automate hard reasoning steps.

The aim of the work described in this paper is to improve on this picture and define a sorted logic, called SND, where sorts are used to obtain a calculus more natural than ND, and more suitable for implementation inside an interactive theorem prover. In particular, SND has been implemented inside the `FOL` system [Wey80; Wey77]; and used to prove many theorems in mathematical logic, theory of computation, program synthesis, and commonsense reasoning. This fact has motivated some choices made below, i.e. the specific calculus and the fact that we use first order classical ND. However, as the rest of the paper makes clear, the ideas and formal treatment can be generalized to other calculi.

2 A sorted logic for interactive theorem proving

A huge amount of work can be found in the Artificial Intelligence literature which studies and defines sorted calculi (see for instance [Wal87; Coh87; SS89; BHP+92; Wei93a; Fri91]). SND shares a lot of motivations and intuitions with this work. In particular, in SND, like in any other sorted logic, the sort information is encoded in a way to allow

- for shorter problem statements and bigger inference steps, and therefore for shorter proofs; and
- for a smaller search space, and therefore for more efficient reasoning.

At the same time, however, SND has been designed to be used in interactive theorem proving, while, as far as we know, *all* the previous work has been developed with the aim of using sorts in (completely) automated deduction, no matter what the application domain was (e.g. knowledge representation, like in [BHP+92]; theorem proving, like in [Wal87; Coh87; Wei93a]; commonsense reasoning like in [RC89]).

This shift of interest has had more impact than one would have expected. Even the features SND has in common with the previous calculi and that have been listed above, have different motivations. Thus, in SND the first feature is motivated by the desire to lower the rate of user intervention, and the second by the desire to lower the user waiting time. However, these are only second level differences which have had no impact in the design of SND. There are other design choices which make SND quite innovative with respect to the state of the art, also from a technical point of view.

A first major difference is that SND is defined on top of ND, contrarily to what is the case with all the previous work we are aware of. Most of this work is based on resolution (see for instance [Coh87; Wal87; SS89; Wei93a]); an exception is the work on semantic tableaux described in [SW90]. The work by [Fri91], though introduced in a very general way, is then again developed for (closed) formulas in (Skolem) normal form. Considering ND instead of resolution adds the further complication of having to deal with free variables, a problem which is notoriously not trivial [1].

A second major difference is that SND has been designed and implemented *incrementally* on top of (the existing `FOL` implementation of) ND. The implementation of SND has required only some minor recoding of some crucial inference rules (i.e. the quantifier rules, see section 4.3). More importantly, the decision procedures developed for the original `FOL` unsorted logic (and described in [Wey77; AG93; GAP95]) can be used in SND with no recoding at all. This feature is very important because developing decision procedures is very costly (taking, at least, various man months), and because, as already pointed out in section 1, most interactive theorem provers make extensive use of decision procedures, all developed for the unsorted calculus. A further advantage is that the current version of `FOL` has a default mode which feels exactly as the original implementation of ND (see example in section 6). Because of this all the examples developed for ND and, in particular all those developed before the addition of sorts, can be run unmodified.

[1] For instance, in [Ros53], Rosser points out that giving a formal characterization of free variables in the presence of sorts is a serious problem, because "*... the intuitive meaning does not furnish a satisfactory guide*". Rosser's conclusion is that "*... in symbolic logic perhaps it is best not to give any special significance to x in $A(x)$ when it occurs free*". In this paper we show that SND provides a precise formalization of the intuitive meaning which Rosser claims to be difficult to formalize.

As far as we know this requirement of incrementality was absent in *all* the previous work in the area. This requirement has caused various technical choices which are quite innovative with respect to the state of the art. Let us discuss them in detail, together with the similarities, considering in turn the language, the semantics and the calculus of SND.

The language of SND is exactly the same as ND's, with the same alphabet and formation rules. Sorts are added by giving a selected subset of predicate symbols the status of sort, and by associating a sort to variables and constants. There is no notion of wellsortedness, and of wellsortedness being a prerequisite to wellformedness, contrarily to what is the case in all the work done in the past (see for instance [BHP+92; SS89; Wal87; Coh87]), with the notable exception of the work described in [Wei91; Wei93b]. (In [Fri91], Frisch does not have a notion of wellsorted formula, but uses a similar – but weaker – notion of well-sorted substitution.) [2] There is again a difference in the fact that sorts are in the language, contrarily to what happens in a lot of the previous work (see for instance [Wal87; Fri91; SS89]), but similarly to what happens in [Wei91; Wei93a; BHP+92; Coh87].

From the point of view of the semantics, incrementality really means that adding sorts is simply another way (with respect to axioms) of restricting the set of first order structures. Furthermore, we want to reason within a sort as if the elements of that sort were the only objects we have, and without giving up the usual treatment for (sorted) variables. In particular we want to consider only interpretations where $\exists x.\alpha(x)$ is true if $\forall x.\alpha(x)$ is. But for this to be true, it must be the case that the extension of S is not empty, where S is the sort of x. This is again a quite non standard feature: for instance, [Coh92; BHP+92; Fri91; Wei93a] allow for empty sorts (with slightly different technical solutions). The motivation for their choice is to extend the scope of the (efficient) sorted inference machinery. However, there is

[2] SND has been motivated by some work in mathematical logic done in the fifties (see e.g. [Wan52]) which seems to have been largely unnoticed in the previous work in Artificial Intelligence. The main motivation for the definition of SND comes from the ideas described in [Kle52]. The work described in this paper can be seen as extending to ND the ideas embodied in the calculus S_2 (see [Kle52], page 420). Besides the work by Kleene, Hailperin's $\mathcal{QR}$ is the system closest in spirit to SND, being the only system we are aware of, where the reduction from sorted to unsorted formulas is achieved *"within the system (rather than by a syntactic rule of translation from one system to another)... [Hai57a]"*. On the other hand, the results in [Hai57a] do not allow free occurrences of restricted variables, and the whole work in [Hai57b] is a (partially successful) attempt to give a satisfactory treatment of restricted free variables.

a fee to be paid: restricted quanfications are no longer interpreted with the same semantic of (unsorted) first order quantification, and $\forall x.\alpha \supset \exists x.\alpha$ is in general no longer valid. Furthermore, additional complication (e.g. ad hoc inference rules) is required in order to deal with empty sorts (see e.g. [Coh92; BHP$^+$92]).

From the point of the deductive machinery, the calculus of SND is incremental with respect to ND [Pra65]. The propositional rules are the same as ND's, while the quantification rules behave inside each sort exactly as the original ND rules, and allow for the introduction of the (appropriate) sort literals when and if needed (see section 4.3). Sorts are really meant as syntactic sugar, i.e. as a convenient way to code (a particular kind of) information otherwise expressible in the language. Therefore, we require that every formula containing sorted variables can be proved equivalent, *within* SND, to a corresponding formula not involving sorted variables. This property, called eliminability, is not usually required in general (see e.g. [Coh86; Fri91; Wei93a]). As far as we know, the only system with this feature can be found in the mathematical logic literature (see footnote 2) We also require that the information encoded by sorts is clearly identifiable and can be reduced to simple first order axioms. The correspondence between SND and (unsorted) first order logic allows us to reduce sorted problems to unsorted problems, and provides a formal basis for the use of (unsorted) deciders during interactive theorem proving.

3 The plan

The rest of the paper basically develops all the technical work needed to show that we have actually achieved what we have announced in section 2. In section 4 we present SND, by describing its language, its (first-order) semantics and its calculus. In this section we prove the eliminability of sorted variables within SND, and the correctness and completeness of the calculus with respect to the semantics. In section 5 we give a precise characterization of the representational power of SND by reducing it to unsorted logic: first, we show the semantic correspondence of SND with first order logic (section 5.1); then, we define the unsorted formal system ND_Σ, we prove that SND is a conservative extension of ND_Σ, and, finally, we show the correspondence between (sorted) derivations in SND and (unsorted) derivations in ND_Σ (section 5.2). This kind of analysis, which takes into account the structure of proofs rather than simple derivability, was advocated by Cohn [Coh86] as relevant to

the understanding of the advantages of sorts in automated deduction. In section 6 we present the solution of a case study in the mechanization of SND in FOL. In section 7 we draw some conclusions. The proofs of the theorems are collected in the appendix of this paper.

The sort information which can be expressed in SND includes the definition of a sort hierarchy, and the association of sorts to individual constants, variables and function symbols. This is done in the obvious and trivial way, and it is described in section 4. Much more complicated sort manipulations can be defined (see for instance [SS89; Wei93a]) and included in SND; this development has not been reported here as not novel and out of the main focus of the paper.

4 SND

Let us consider in turn, the language, the semantics and the calculus of SND.

4.1 Language

As previously said, we require that the language for SND is a standard first order language. In the following $\mathcal{L}$ is a first order language with equality, c is an individual constant, $t, t_1 \ldots t_n$ are terms, A, B, C are formulas, Γ, Φ, Ψ are sets of formulas, $x, x_1, \ldots, x_n, y, u, u_1, \ldots, u_n$ are variables. $frees(A)$ [$frees(\Gamma)$] is the set of variables occurring free in A [in every formula of Γ]. The notation of substitution is as follows. Let $x_1, \ldots, x_n$ be pairwise distinct variables, then $t^{t_1}_{x_1} \ldots {}^{t_n}_{x_n}$ is the term obtained by simultaneously substituting every occurrence of x_i with t_i in t. If each t_i is free for x_i in A, then $A^{t_1}_{x_1} \ldots {}^{t_n}_{x_n}$ is the formula obtained by simultaneously substituting every free occurrence of x_i in A with t_i.

We introduce sorts by associating the *sort information* to $\mathcal{L}$. Sorts are unary predicate symbols which are syntactically "tagged". They denote a non-empty subset of the domain of interpretation. For instance, $\mathcal{L}$ may contain the unary predicates *Integer*, *Even*, *Odd* and *Prime*, where only the first three are sorts. Sorts are partially ordered according to a generality relation; for instance, *Even* and *Odd* can be less general than *Integer*. This ordering has a top element, called "the most general sort", which holds of all the terms in $\mathcal{L}$. Individual constants and variables have an

associated sort, and are intended to denote an individual of the corresponding class, and to range over the corresponding class of objects, respectively. For instance, we may have the constant 1, of sort *Odd*, and the variable n of sort *Integer*. Function symbols come with tuples of sorts, called fmaps. For instance, the binary function symbol + may come with the fmap $\langle$*Integer, Integer, Integer*$\rangle$, the intuitive meaning being that the term $(t_1 + t_2)$ denotes an integer if t_1 and t_2 denote integers. Functions are overloaded; for instance, + may also come with the fmap $\langle$*Odd, Odd, Even*$\rangle$, specifying that the sum of two odds is an even. The sort information Σ contains all the information related to sorts, and is formally defined as follows:

Definition 4.1 (Sort information) *The sort information of $\mathcal{L}$ is a triple $\Sigma = \langle \mathcal{S}, \preceq, sort(.) \rangle$, where:*

- *the set of sort symbols $\mathcal{S}$ is a set of unary predicates including U;*
- *the less general relation $\preceq$ is a binary transitive relation, on $\mathcal{S}$, with top element U (the most general sort);*
- *the sorting function $sort(.)$ maps every individual variable and constant onto a sort, and every function symbol of arity n onto a set of $n+1$-tuples of sorts, called fmaps.*

For brevity, from now on we will sometimes write "sorts" meaning "sort symbols". $S, S_1, S_2, \ldots S_n$ and T are sort symbols. An individual variable or constant i is of sort S if $sort(i) = S$. We call unsorted a variable of sort U, and sorted a variable of a different sort. In the following, if unspecified otherwise, $u, u_1, \ldots, u_n$ are unsorted variables, x is a variable of sort S, $x_1, \ldots, x_n$ are variables of sort $S_1, \ldots, S_n$, y is a variable of sort T.

The following notion formalizes the relation between terms and sorts.

Definition 4.2 ($t \leqslant S$) *The individual (constant or variable) $i \leqslant T$ (i* belongs to *T) iff $sort(i) \preceq T$. The term $f(t_1, \ldots, t_n) \leqslant T$ iff f has an fmap $\langle S_1, \ldots, S_n, S \rangle$ such that $S \preceq T$ and, for every i, $t_i \leqslant S_i$.*

Notice that, because of the partial order on the sorts and of the overloading of function symbols, a term can belong to different sorts. For instance, according to the sort information defined above, $(1 + 1) \leqslant$ *Integer* and also $(1 + 1) \leqslant$ *Even*.

4.2 Semantics

We do not modify the usual semantic notions for first order logic. In the following, $\mathcal{U}$ is a first order structure, $\|\mathcal{U}\|$ is the domain of $\mathcal{U}$, $\mathcal{U}(c)$ is the element of $\|\mathcal{U}\|$ corresponding to the constant c, $\mathcal{U}(f)$ is the function from $\|\mathcal{U}\|^n$ to $\|\mathcal{U}\|$ corresponding to the n-ary function symbol f, $\mathcal{U}(P)$ is the relation on $\|\mathcal{U}\|^n$ corresponding to the n-ary predicate symbol P. Given a structure $\mathcal{U}$, an assignment β over $\mathcal{U}$ is a function mapping variables in $\mathcal{L}$ into elements of $\|\mathcal{U}\|$. An interpretation $\mathcal{I}$ is a pair $\langle \mathcal{U}, \beta \rangle$. Intuitively, the semantics is obtained by considering structures, assignments and interpretations which agree with the sort information Σ.

Definition 4.3 (Σ-structure) *$\mathcal{U}$ is a Σ-structure iff:*

(i) $\mathcal{U}(U) = \|\mathcal{U}\|$, *where U is the most general sort*

(ii) *for every sort S, $\mathcal{U}(S)$ is not empty*

(iii) *for every sort S and T such that $S \preceq T$, $\mathcal{U}(S) \subseteq \mathcal{U}(T)$*

(iv) *for every individual constant c of sort S, $\mathcal{U}(c) \in \mathcal{U}(S)$*

(v) *for every fmap $\langle S_1, \ldots, S_n, S \rangle$ of the function symbol f, if, for all i, $d_i \in \mathcal{U}(S_i)$, then $\mathcal{U}(f)(d_1, \ldots, d_n) \in \mathcal{U}(S)$*

β is a Σ-assignment iff for every variable x of sort S, $\beta(x) \in \mathcal{U}(S)$. $\mathcal{I} = \langle \mathcal{U}, \beta \rangle$ is a Σ-interpretation iff $\mathcal{U}$ is a Σ-structure and β is a Σ-assignment. The interpretation of a term, $\mathcal{I}(t)$, is defined as usual. Let β be a Σ-assignment on the Σ-structure $\mathcal{U}$, $x_1, \ldots, x_n$ pairwise distinct variables, and, for each $i = 1, \ldots, n$, $d_i \in \mathcal{U}(S_i)$; then $\beta^{d_1 \ldots d_n}_{x_1 \ldots x_n}$ is the Σ-assignment which maps x_i to d_i, and every other variable x to $\beta(x)$. Let $\mathcal{I}$ be the Σ-interpretation $\langle \mathcal{U}, \beta \rangle$; then $\mathcal{I}^{d_1 \ldots d_n}_{x_1 \ldots x_n}$ is the Σ-interpretation $\langle \mathcal{U}, \beta^{d_1 \ldots d_n}_{x_1 \ldots x_n} \rangle$.

Definition 4.4 ($\mathcal{I} \models_\Sigma A$) *Let $\mathcal{I}$ be the Σ-interpretation $\langle \mathcal{U}, \beta \rangle$. $\mathcal{I}$ Σ-satisfies A ($\mathcal{I}$ is a Σ-model of A, written $\mathcal{I} \models_\Sigma A$), is inductively defined as follows:*

(i) $\mathcal{I} \models_\Sigma P(t_1, \ldots, t_n)$ *iff* $\langle \mathcal{I}(t_1), \ldots, \mathcal{I}(t_n) \rangle \in \mathcal{U}(P)$

(ii) $\mathcal{I} \models_\Sigma A \wedge B$ *iff* $\mathcal{I} \models_\Sigma A$ *and* $\mathcal{I} \models_\Sigma B$ *(analogously for $\neg A, A \vee B, A \supset B, A \leftrightarrow B$)*

(iii) $\mathcal{I} \models_\Sigma \forall x.A$ *iff for all $d \in \mathcal{U}(sort(x))$* $\mathcal{I}^d_x \models_\Sigma A$

(iv) $\mathcal{I} \models_\Sigma \exists x.A$ *iff there is* $d \in \mathcal{U}(sort(x))$ $\mathcal{I}_x^d \models_\Sigma A$

A is a Σ-*logical consequence* of the set Γ ($\Gamma \models_\Sigma A$) iff every Σ-model of Γ is a Σ-model of A. A is Σ-*valid* ($\models_\Sigma A$) iff every Σ-interpretation is a Σ-model of A. A is Σ-*satisfiable* iff it has a Σ-model.

The correspondence between Σ-interpretations and the sort information is made clear by the following lemma.

Lemma 4.1 *For all* t, S, $t \leqslant S$ *iff* $\models_\Sigma S(t)$.

In section 2 we have pointed out that we wanted sorts to be only syntactic sugar and that we wanted sorted variables to be eliminable within SND. We show below that every formula A is logically equivalent to its relativization A^+ containing only unsorted variables. We start by defining a relativization function $(.)^+$, mapping expressions of $\mathcal{L}$ to expressions not containing sorted variables (in the following we call $\mathcal{L}'$ the subset of $\mathcal{L}$ not containing sorted variables). The notion of relativization function is a well known construction, which maps restricted quantifications in terms of unrestricted quantifications (see for instance [Kle52; Ros53; End72; Hai57a; Wal87; Coh87; Wei91]). In the relativization function defined below, quantifications of sorted variables are expanded in the standard way, while unsorted quantifications are renamed to preserve injectivity. $(.)^+$ associates to every variable in $\mathcal{L}$ an unsorted variable, with the understanding that different variables are mapped to different variables.

Definition 4.5 (Relativization) *The relativization function* $(.)^+$ *is defined as follows:*

$$
\begin{array}{rcl}
(A)^+ & = & A^{x_1^+ \ldots x_n^+}_{x_1 \ldots x_n} \quad A \text{ atomic, } frees(A) = \{x_1, \ldots, x_n\} \\
(\neg A)^+ & = & \neg(A^+) \\
(A \circ B)^+ & = & A^+ \circ B^+, \quad \text{with} \quad \circ \in \{\wedge, \vee, \supset, \leftrightarrow\}, \\
(\forall u.A)^+ & = & \forall u^+.(A^+) \\
(\exists u.A)^+ & = & \exists u^+.(A^+) \\
(\forall x.A)^+ & = & \forall x^+.(S(x^+) \supset A^+) \\
(\exists x.A)^+ & = & \exists x^+.(S(x^+) \wedge A^+)
\end{array}
$$

By t^+ we mean the term of $\mathcal{L}'$ $t^{x_1^+}_{x_1} \ldots {}^{x_n^+}_{x_n}$, where $x_1, \ldots, x_n$ are the variables occurring in t. We write x instead of x^+ whenever no ambiguity arises. We write indifferently $(A^+)^{t^+}_{x^+}$ and $(A^t_x)^+$, as they denote the same expression.

The logical equivalence between a formula and its relativization is formally stated as follows.

Theorem 4.1 (Eliminability) *Let $MapVars(A)$ be the conjunction of the equalities $x = x^+$, for every $x \in frees(A)$. Then, for all A, for all $\mathcal{I}$ such that $\mathcal{I} \models_\Sigma MapVars(A)$,*

$$\mathcal{I} \models_\Sigma A \leftrightarrow A^+$$

$MapVars(.)$ take into account the correspondence between x and x^+, when they occur free. When A is closed, the theorem states the validity of $A \leftrightarrow A^+$. When A contains free variables, it states the validity of $MapVars(A) \supset (A \leftrightarrow A^+)$.

This result is general in two respects. First, although it may look similar to the correspondence between a sorted formula with its relativization (see for instance [Wal87; Coh86]), it is much stronger, as it requires truth *in the same models*, rather than preservation of satisfiability. Therefore, eliminability is a mapping within a formal system rather than a metatheoretic correspondence between formulas of different formal systems. Second, our result generalizes the investigation of this correspondence carried out by Hailperin [Hai57b] (only for closed formulas) to the case of formulas with free variables.

4.3 Calculus

The calculus of SND is the triple $\langle \mathcal{L}, \Sigma, \mathcal{D} \rangle$, where the deductive machinery $\mathcal{D}$ is composed of sort axioms and inference rules.

A sort axiom is a formula of shape $\forall x.S(x)$, where x is of sort S. Sort axioms make explicit the information implicit in the quantification of sorted variables. They state that the quantification of a variable x of sort S ranges over (the class denoted by) S.

The inference rules are the propositional natural deduction inference rules [Pra65], the reflexivity and substitution rules for equality, and the quantifier rules of figure 1. The quantifier rules are different from Prawitz' in two respects. First, they

Let S be $sort(x)$, and T be $sort(y)$.

$$(\forall E) \quad \begin{cases} \dfrac{\forall x.A}{A^t_x} & \text{if } t{\leqslant}S \\[2ex] \dfrac{\forall x.A}{S(t) \supset A^t_x} & \text{otherwise} \end{cases}$$

$$(\forall I) \quad \frac{A}{\forall x.A^x_y} \quad \text{applicable only if } y \text{ does not occur free in any undischarged assumption of } A,\ x \text{ does not occur free in } A \text{ and } x{\leqslant}T$$

$$(\exists I) \quad \begin{cases} \dfrac{A^t_x}{\exists x.A} & \text{if } t{\leqslant}S \\[2ex] \dfrac{A^t_x}{S(t) \supset \exists x.A} & \text{otherwise} \end{cases}$$

$$(\exists E) \quad \frac{\exists x.A \quad \begin{matrix}[A^y_x] \\ B\end{matrix}}{B} \quad \text{applicable only if } x \text{ and } y \text{ do not occur free in the undischarged assumptions of } B \text{ other than } A^y_x,\ y \text{ does not occur free in } B \text{ nor in } \exists x.A, \text{ and } x{\leqslant}T$$

Figure 1: Sorted Quantifier Inference Rules

have extra conditions which make sure that quantification works correctly. This is necessary as SND allows variables to occur both free and bound (the reason for this choice is that introducing a parameter any time a quantifier is eliminated would make the interaction with `FOL` much less natural). Second, by checking whether a term belongs to a sort, they take into account the fact that SND is sorted.

Let $\vdash_{\text{SND}}$ be the derivability relation of SND. If $t{\leqslant}S$ then, by $\forall$ elimination from axiom $\forall x.S(x)$, $\vdash_{\text{SND}} S(t)$. From that, $\vdash_{\text{SND}} \exists x.S(x)$ follows by $\exists I$. These two facts are the syntactic counterpart of the semantic conditions of lemma 4.1 and of interpreting a sort as non-empty subset of the domain. Furthermore, for any formula α and any variable x of sort S, we have that $\forall x.\alpha \vdash_{\text{SND}} \exists x.\alpha$.

Notice that the rules of $\forall E$ and $\exists I$ are applicable also when the term t does not belong to the "right" sort (i.e. S), which causes the appearance of the sort literal $S(t)$. This is due to the fact that in SND the sort hierarchy is interpreted order-theoretically rather than lattice-theoretically (this terminology is taken from [HS90]),

i.e., the greatest lower bound of two sorts in the sort hierarchy is interpreted as a subset of the two sorts rather than as their intersection (in this respect SND is similar to LILOG [BHP+92] and Walther's logic [Wal87], and different from LLAMA [Coh87]). This choice, motivated by our interest in the quality of user interaction, allows for an increase of expressiveness, as explained in [BHP+92].Although it may seem that $\forall E$ and $\exists I$ are not restrictive enough to rule out instantiations which may be of little interest, this does not imply a loss of efficiency: in interactive theorem proving it is possible to specify control strategies (tactics) restricting inference according to the information available in Σ. For instance, the instantiation of a quantified variable (i.e. an application of $\forall E$) can be limited to the (more promising) cases when the term t belongs to the sort of the quantified variable x, which yields a simpler consequence.

The calculus of SND is correct and complete with respect to the semantic defined in previous section. The relations between derivability in SND and Σ-logical consequence are formally stated by the following theorems.

Theorem 4.2 (Correctness of SND**)** *For every* A, Γ, $\Gamma \vdash_{\text{SND}} A \quad \Longrightarrow \quad \Gamma \models_{\Sigma} A$

Theorem 4.3 (Completeness of SND**)** *For every* A, Γ, $\Gamma \models_{\Sigma} A \quad \Longrightarrow \quad \Gamma \vdash_{\text{SND}} A$

5 From SND to unsorted logic

In this section we start by describing the representational power of SND via a reduction of its semantics to first order (unsorted) semantics. In particular, we make explicit in terms of first order formulas what is the information encoded in (the sorts of) SND. Then, we define an unsorted formal system ND_{Σ}, of which SND is a conservative extension. Finally, we show an algorithmic correspondence between the derivations in SND and those in ND_{Σ}. This analysis clearly shows to what extent the introduction of sorts allows for a more compact encoding of information, and for more effective inference.

5.1 Model-theoretic equivalence

The information encoded in sorted quantification is made explicit in the relativization function (see theorem 4.1). However, the relativization function does not take into account the sort information for free variables (see the transformation for atomic formulas). The information expressed by restrictions on free variables could not be taken into account by the relativization function, as the scope of free variables is not limited to the formula where they occur. Therefore this information needs to be made explicit by the notion of sort assumptions, $SA(.)$, which takes into account the sort of free variables.

Definition 5.1 ($SA(.)$) *$SA(A)$, the* sort assumptions *of A, are the formulas of $\mathcal{L}'$ of form $S(x^+)$, for every $x \in frees(A)$. The sort assumptions of Γ, $SA(\Gamma)$, are the sort assumptions of all the formulas in Γ. $SA(\mathcal{L})$ are the sort assumptions of all the formulas in the language $\mathcal{L}$.*

Furthermore, Σ contains information about the sort of symbols of the signature, e.g. individual constants, function symbols. The following formulas encode precisely the sort information about symbols of the signature.

Definition 5.2 ($Uns(\Sigma)$) *$Uns(\Sigma)$ contains the following formulas:*

(i) $\forall u.U(u)$;

(ii) *for every sort S,* $\exists u.S(u)$;

(iii) *for every sort S and T such that $(S \preceq T)$,* $\forall u.(S(u) \supset T(u))$;

(iv) *for every individual constant c of sort S,* $S(c)$;

(v) *for every fmap $\langle S_1, \ldots, S_n, S\rangle$ of f,*

$$\forall u_1 \ldots u_n.(S_1(u_1) \wedge \ldots \wedge S_n(u_n) \supset S(f(u_1, \ldots, u_n)))$$

The following theorem makes explicit the notion of Σ-logical consequence in terms of first order logical consequence.

Theorem 5.1 (Model equivalence) *For all A, Γ in $\mathcal{L}$,*

$$Uns(\Sigma), SA(\Gamma, A), \Gamma^+ \models A^+ \quad \Longleftrightarrow \quad \Gamma \models_\Sigma A$$

Theorem 5.1 makes it clear that there are precisely three different forms of information implicitly encoded in the sorts of SND. First, the information contained in Σ about the symbols of the signature (constants, functions and sort predicates), made explicit in the axioms of $Uns(\Sigma)$; second, the sort information for quantified restricted variables, made explicit by the relativization of formulas; third, the sort information for free variables, made explicit by the corresponding sort assumptions.

Theorem 5.1 also shows the way sorted free variables are dealt with in our logic. Informally, *assuming* $A(x)$, with x of sort S, corresponds in a unsorted framework to $S(x) \wedge A(x)^+$. On the other hand, *trying to prove* $A(x)$, with x of sort S, corresponds in a unsorted framework to $S(x) \supset A(x)^+$. Consider now what happens in our logic. $A(x) \models_\Sigma B$ amounts to $\{A(x)^+, S(x)\} \models B^+$, i.e. $A(x)$ amounts to $S(x) \wedge A(x)^+$ in the unsorted framework; on the other hand, if $\models_\Sigma A(x)$, then $S(x) \models A(x)^+$, and, therefore, $\models S(x) \supset A(x)^+$. That is, in SND free variables are given the very same intuitive meaning which Rosser claims to be difficult to formalize (see footnote 1). This feature is extremely important from the point of view of interactive theorem proving; indeed, it makes the (interactive) search for a proof very close to the intuitive argument.

The proof of theorem 5.1, reported in full detail in the Appendix, gives interesting insights on the semantics. Basically, it shows that the Σ-models for Γ and A are exactly the same first order models as those for Γ^+, A^+, $Uns(\Sigma)$ and $SA(\Gamma, A)$ (modulo renaming of variables). The proof is based on the bijection $(.)^+$ between Σ-interpretations and the interpretations for $\mathcal{L}'$ which satisfy $Uns(\Sigma)$ and $SA(\mathcal{L})$. Given this correspondence, it is proven that A is true in $\mathcal{I}$ if and only if A^+ is true in $\mathcal{I}^+$.

5.2 Proof-theoretic equivalence

Consider the formal system ND_Σ, defined as follows.

Definition 5.3 (ND_Σ) *Let* $\mathrm{ND}_\Sigma = \langle \mathcal{L}', \mathcal{D}' \rangle$, *where* $\mathcal{L}'$ *is the language and* $\mathcal{D}'$ *is the deductive machinery of* ND_Σ. $\mathcal{D}'$ *is composed of the natural deduction inference rules, the reflexivity and substitution rules for equality, and the set of axioms* $Uns(\Sigma)$.

As a simple consequence of previous theorems, we have (see figure 2):

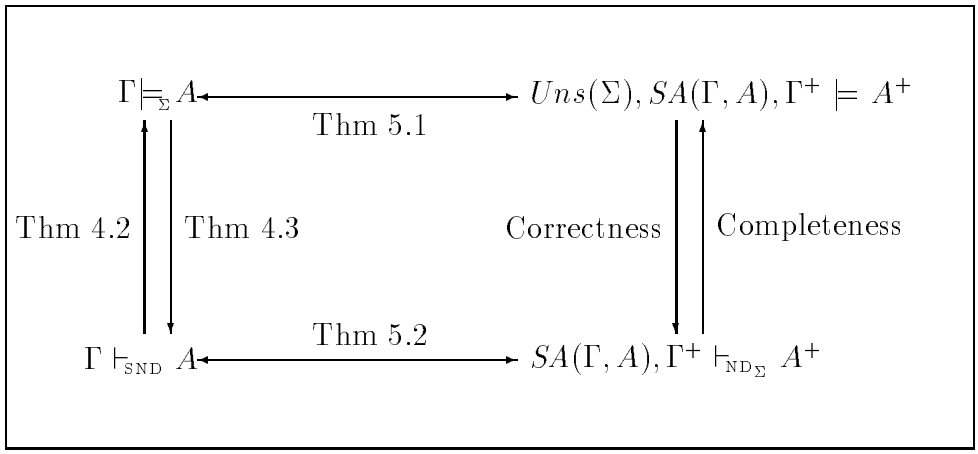

Figure 2: An overview of the theoretical results

Theorem 5.2 (Derivation Equivalence) *For all A, Γ in $\mathcal{L}$,*

$$SA(\Gamma, A), \Gamma^+ \vdash_{\mathrm{ND}_\Sigma} A^+ \quad \Longleftrightarrow \quad \Gamma \vdash_{\mathrm{SND}} A$$

Another consequence of theorem 5.1 is that SND is a conservative extension of ND_Σ.

Theorem 5.3 (Conservativeness) *For all A, Γ in $\mathcal{L}'$,*

$$\Gamma \vdash_{\mathrm{SND}} A \quad \Longrightarrow \quad \Gamma \vdash_{\mathrm{ND}_\Sigma} A$$

More interestingly, let us now analyze the structural relation between proofs in SND and in ND_Σ. We describe an effective procedure mapping an SND derivation Π of A from Γ to a ND_Σ derivation of A^+ from Γ^+ and $SA(A, \Gamma)$. The translation is performed in two steps. First, we build a derivation Π^+ of A^+ from Γ^+ and $SA(\mathcal{L})$; then we discharge the sort assumptions about variables which are not in $frees(A, \Gamma)$.

By structural induction on the derivation Π of A from Γ, we define Π^+ and we show that it is a ND_Σ derivation of A^+ from Γ^+ and $SA(\mathcal{L})$. If Π is the assumption A, then Π^+ is simply A^+. If Π is the sort axiom $\forall x.S(x)$ then Π^+ is a (trivial) derivation of $\forall x.(S(x) \supset S(x))$. If Π ends with the application of a propositional inference rule ρ to $\Pi_1, \ldots, \Pi_n$, then Π^+ is obtained by applying ρ to $\Pi_1^+, \ldots, \Pi_n^+$. For instance, in the case of $\supset$-elimination we have:

$$\dfrac{\begin{array}{cc}\Pi_1 & \Pi_2 \\ A \supset B & A\end{array}}{B} \supset E \quad \Longrightarrow \quad \dfrac{\begin{array}{cc}\Pi_1^+ & \Pi_2^+ \\ (A \supset B)^+ & A^+\end{array}}{B^+} \supset E$$

Π^+ is a derivation of ND_Σ because the Π_i^+'s are such by the induction hypothesis, and ρ can be applied because $(.)^+$ distributes over the propositional connectives.

If Π ends with the application of a quantifier inference rule, the structure of Π^+ has to be modified in order to make explicit the sort information which is (implicitly) taken into account in SND. We show below the construction of Π^+ only for the most interesting cases (the definition of Π_S is given below). Consider $\forall E$, where $t \Leftarrow S$:

$$\dfrac{\dfrac{\Pi_1}{\forall x.A}}{A_x^t}\,\forall E \quad\Longrightarrow\quad \dfrac{\dfrac{\Pi_1^+}{\forall x.(S(x) \supset A^+)}\,\forall E \qquad \dfrac{\Pi_S}{S(t)^+}}{(A_x^t)^+}\,\supset E$$

Consider $\forall I$ (from the applicability of the rule in SND it follows that $S \preceq T$):

$$\dfrac{\dfrac{\Pi_1}{A}}{\forall x.(A_y^x)}\,\forall I \quad\Longrightarrow\quad \dfrac{\dfrac{\dfrac{\dfrac{[T(y)]_{(1)}}{\dfrac{\Pi_1^+}{A^+}}}{T(y) \supset A^+}\,\supset I_{(1)} \qquad \dfrac{\dfrac{\forall u.(S(u) \supset T(u))}{S(y) \supset T(y)}\,\forall E \qquad [S(y)]_{(2)}}{T(y)}\,\supset E}{A^+}\,\supset E}{S(y) \supset A^+}\,\supset I_{(2)}}{\forall x.(S(x) \supset (A_y^x)^+)}\,\forall I$$

Consider $\exists I$. If $t \Leftarrow S$ we have:

$$\dfrac{\dfrac{\Pi_1}{A_x^t}}{\exists x.A}\,\exists I \quad\Longrightarrow\quad \dfrac{\dfrac{\Pi_S}{S(t)^+} \quad \dfrac{\Pi_1^+}{(A_x^t)^+}}{S(t)^+ \wedge (A^+)_x^t}\,\wedge I}{\exists x.(S(x) \wedge A^+)}\,\exists I$$

Finally, consider the case of $\exists E$ with free u:

$$\dfrac{\dfrac{\Pi_1}{\exists x.A} \quad \dfrac{[A_x^u]_{(1)}}{\dfrac{\Pi_2}{B}}}{B}\,\exists E_{(1)} \quad\Longrightarrow\quad \dfrac{\dfrac{\Pi_1^+}{\exists x.(S(x) \wedge A^+)} \quad \dfrac{\dfrac{[(S(x) \wedge A^+)_x^u]_{(1)}}{(A^+)_x^u}\,\wedge E}{\dfrac{\Pi_2^+}{B^+}}}{B^+}\,\exists E_{(1)}$$

In the derivations above, Π_S is a derivation of $S(t)^+$ from $SA(\mathcal{L})$, where $t \Leftarrow S$. Π_S can be built by induction on the structure of t. If t is the constant c of sort T, then $T(c)$ is an axiom of $Uns(\Sigma)$; if t is the variable x of sort T, $T(x)^+ \in SA(\mathcal{L})$. In both cases $T \preceq S$ has to hold, and therefore $\forall y.(T(y) \supset S(y))$ is an axiom. Π_S is built

by applying $\forall E$ (instantiating y with t^+) and $\supset E$. If t is $f(t_1, \ldots, t_n)$, Π^+ is the following derivation (where $\wedge I^*$ and $\forall E^*$ indicate repeated application of $\wedge I$ and $\forall E$, respectively):

$$\dfrac{\dfrac{\forall x.(T(x) \supset S(x))}{T(t)^+ \supset S(t)^+}\,\forall E \quad \dfrac{\dfrac{\begin{array}{ccc}\Pi^+_{S1} & \ldots & \Pi^+_{Sn} \\ T_1(t_1)^+ & & T_n(t_n)^+\end{array}}{T_1(t_1)^+ \wedge \ldots \wedge T_n(t_n)^+}\,\wedge I^* \quad \dfrac{\forall x_1 \ldots x_n.(T_1(x_1) \wedge \ldots \wedge T_n(x_n) \supset T(f(x_1, \ldots, x_n)))}{T_1(t_1)^+ \wedge \ldots \wedge T_n(t_n)^+ \supset T(f(t_1, \ldots, t_n))^+}\,\forall E^*}{T(f(t_1, \ldots, t_n))^+}\,\supset E}{S(f(t_1, \ldots, t_n))^+}\,\supset E \qquad (\dagger)$$

We are now left with the problem of getting rid of the sort assumptions for variables not in $frees(\Gamma, A)$. For the finiteness of proofs, there is only a finite number of such sort assumptions. Therefore, the desired derivation is obtained by the repeated application of $\exists E$ applied to the formula $\exists x.S(x)$ of $Uns(\Sigma)$:

$$\dfrac{\begin{array}{c}\Gamma^+, SA(\Gamma, A), \ldots, [S_i(x)]_{(i)} \\ \Pi^+ \\ A^+\end{array} \qquad \exists x_i.S(x_i)}{A^+}\,\exists E_{(i)} \qquad (\ddagger)$$

The translation defined above shows that reasoning in SND is less complex than the corresponding reasoning in ND_Σ. First, inference in SND involves simpler formulas, α^+ is in general more complex than α. Furthermore, the information encoded in Σ, exploited by the quantifier inference rules in SND to test whether $t \leqslant S$, has to be explicitly manipulated in ND_Σ. For instance, a derivation of $S(t)^+$, as shown in ($\dagger$), has to be repeated every time. Finally, derivations in SND are simpler because of the use of restricted free variables: in ND_Σ these have to be explicitly made (as in $\dagger$), and discharged with the construction ($\ddagger$).

6 A Case Study

In this section we show a solution of the following problem (described in [Ros53]) in the current implementation of `FOL` ([Wey77; Wey80; GW91]):

Let δ and ϵ denote positive real numbers, x,y and z denote points, and α and β denote point sets. Given the definitions: x is a limit point for α iff for every ϵ there is a point of α different from x whose distance from x is less than ϵ; α is a closed set iff all limit points of α are in α; β is the derived set of α iff β consists of the limit points of α; prove that the derived set of a point set is closed.

Consider the beginning of the FOL session (where FOL:: is the system prompt). If we ask the FOL to show the available sorts,

```
FOL:: show sort;
The only symbol declared to be a sort is UNIVERSAL
```

the answer is that there are no sorts other than the most general sort, UNIVERSAL, which corresponds to U in SND. In other words, if no sorts are explicitly introduced by the user, FOL implements a calculus behaving like Prawitz' (unsorted) natural deduction calculus.

The first step for mechanizing in FOL the case study is to declare the language. We represent as sorts the basic mathematical concepts used in the example, namely real numbers, positive real numbers, points and point sets. We also state that Real is more general than Pos_Real.

```
FOL:: declare sort Real Pos_Real Point Point_Set;
FOL:: moregeneral Real < Pos_Real >;
```

Then we introduce suitable variables (the sort being specified in square brackets). Notice the similarity between these declarations and the introduction of variables in the informal statement of the problem.

```
FOL:: declare indvar eps del [Pos_Real];
FOL:: declare indvar x y z [Point];
FOL:: declare indpar alpha beta [Point_Set];
```

Here we declare the basic mathematical functions of the example. The command setfmap associates an fmap to a function symbol. The function - represents both the difference between real numbers and the oriented distance between points. It is an example of the overloading of functions in FOL.

```
FOL:: declare funconst + 2 [INF];
FOL:: setfmap + ( Real  Real )  = Real;
FOL:: declare funconst - 2 [INF];
FOL:: setfmap - ( Point Point ) = Real;
FOL:: setfmap - ( Real  Real )  = Real;
FOL:: declare funconst / 2 [INF];
FOL:: setfmap / ( Pos_Real  Pos_Real ) =  Pos_Real;
FOL:: declare funconst abs 1;
FOL:: setfmap abs (Real) = Real;
```

After declaring the (infix) binary predicates `<`, `>` and `in` (standing for membership), we can formalize the notions of limit point, closed set and derived set. The command `define` declares a new symbol and introduces a corresponding definitional axiom. The similarities between the informal definitions and the definitional formulas arise from the restricted quantification mechanism.

```
FOL:: define predconst Limit_Point_of 2 [INF] by:
         (x Limit_Point_of alpha) iff
         forall eps. exists y.
               ((not(y=x)) and (y in alpha) and (abs(x - y) < eps));
FOL:: define predconst Closed_Set 1 by:
         Closed_Set(alpha) iff
         forall x.(x Limit_Point_of alpha imp x in alpha);
FOL:: define predconst Derived_Set_of 2 [INF] by:
         (beta Derived_Set_of alpha) iff
         forall x.(x Limit_Point_of alpha iff x in beta);
```

The informal solution to the problem is described in [Ros53] as follows:

> *Let x be a limit point of β. As $\epsilon/2 > 0$ there exists a point y in β different from x whose distance from x is less than $\epsilon/2$. Since $x \neq y$, $|x-y| > 0$; since y is in β it is a limit point of α. Therefore there exists a point z of α, different from y, whose distance from y is less than $|x-y|$. Since $|y-z| < |x-y|$, $z \neq x$ and $|y-z| < \epsilon/2$. From this $|x-z| = |(x-y)+(y-z)| \leq |x-y|+|y-z| < \epsilon/2+\epsilon/2 = \epsilon$. So there exists a z different form x such that $|x-z| < \epsilon$. Therefore x is a limit point of α and so belongs to β.*

In the following we outline the mechanized solution in FOL. We first assume that (the set) beta is the derived set of (the set) alpha (which is the hypothesis of the theorem to prove, step (1)), and that (the set) x is a limit point of beta (step (2)).

```
FOL:: assume beta Derived_Set_of alpha;
    1   beta Derived_Set_of alpha     (1)
FOL:: assume x Limit_Point_of beta;
    2   x Limit_Point_of beta     (2)
```

We can avoid making explicit that alpha, beta and x are sets because we have declared them to be of sort Set. Then, we unfold the definition of limit point (step 3), and we instantiate the universal quantifier (step 4). Now we let y denote the object satisfying the existential quantifier: this is performed by the command existe, which instantiates the existential quantifier by substituting the bound variable y0 with the free variable y, and generates the suitable assumption (step 5).

```
FOL:: unfold 2 DEF1;
    3   forall eps0. exists y0. (((not (y0 = x)) and (y0 in beta)) and
        (abs(x - y0) < eps0))     (2)
FOL:: alle 3 eps/2;
    4   exists y0. (((not (y0 = x)) and (y0 in beta)) and
        (abs(x - y0) < (eps / 2)))     (2)
FOL:: existe 4 y;
    5   ((not (y = x)) and (y in beta)) and (abs(x - y) < (eps / 2))     (5)
```

Notice that y is intended to satisfy the explicitly stated properties (i.e. being different from x, being in beta and being at a certain distance from x), but also the ones that are implicitly understood (i.e. being a set). The steps from 6 to 17 show that there exists another limit point between x and y (step 17). We instantiate the existential with z, yielding 18:

```
...
    17   exists y2. (((not (y2 = y)) and (y2 in alpha)) and
         (abs(y - y2) < abs(x - y)))     (1 5)
FOL:: existe ^1 z;
    18   (not (z = y) and (z in alpha)) and (abs(y - z) < abs(x - y))     (18)
```

Starting from derivation of step 39, we introduce an existential quantifier by applying the existi command. The existential elimination is mechanized in FOL so that the assumptions resulting from existe (5 and 18 in this case) are automatically discharged in the following inference steps and replaced by the dependencies of the existential formulas, as soon as the suitable conditions are satisfied. Step 40 depends on 1 and 2 because the discharging mechanism takes into account all previous existe, i.e. steps 5 and 18. The thesis (step 50) is then obtained by folding the definitions and discharging assumptions 1 and 2.

```
...
    39   ((not (z = x)) and (z in alpha)) and (abs(x - z) < eps)     (5 18)
FOL:: existi 39 z;
    40   exists z.
         (not (z = x) and (z in alpha)) and (abs(x - z) < eps)     (1 2)
...
    50   (beta Derived_Set_of alpha) imp Closed_Set(beta)
```

7 Conclusion

In this paper we have motivated and presented SND, a sorted logic extending ND and particularly suitable for interactive theorem proving. We have started by motivating the major design decisions underlying the definition of SND and, then, we have carried out the technical work needed to show that we have actually achieved our goals. SND turns out to have various features which are quite innovative with respect to the calculi defined in the past. All these features are motivated by the shift of interest from automated to interactive deduction.

References

[AG93] A. Armando and E. Giunchiglia. Embedding Complex Decision Procedures inside an Interactive Theorem Prover. *Annals of Mathematics and Artificial Intelligence*, 8(3–4):475–502, 1993.

[BHP+92] C. Beierle, U. Hedtstück, U. Pletat, P. H. Schmitt, and J. Siekmann. An Order-Sorted logic for Knowledge Representation Systems. *Artificial Intelligence*, 55:149–191, 1992.

[BHR90] K.H. Bläsius, U. Hedtstück, and C.-R. Rollinger, editors. *Sorts and Types in Arificial Intelligence*. Springer-Verlag, 1990.

[CAB+86] R.L. Constable, S.F. Allen, H.M. Bromley, et al. *Implementing Mathematics with the NuPRL Proof Development System*. Prentice Hall, 1986.

[Coh86] A.G. Cohn. Many Sorted Logic = Unsorted Logic + ControlΓ In *Expert Systems 86*, pages 184–194. Cambridge University Press, 1986.

[Coh87] A.G. Cohn. A More Expressive Formulation of Many Sorted Logic. *Journal of Automated Reasoning*, 3:113–200, 1987.

[Coh92] A. G. Cohn. A Many Sorted Logic with Possibly Empty Sorts. In Kapur [Kap92], pages 633–647. Published as Springer LNAI, number 607.

[EFT84] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Spinger-Verlag, 1984.

[End72] H.B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[Fri91] A. M. Frisch. The Substitutional Framework for Sorted Deduction: Fundamental Results on Hybrid Reasonig. *Artificial Intelligence*, 49:161–198, 1991.

[GAP95] E. Giunchiglia, A. Armando, and P. Pecchiari. Structured Proof Procedures. *Annals of Mathematics and Artificial Intelligence*, 15(I), 1995. Extended version of Structured Proof Procedures, Third Bar-Ilan Symposium on the Foundations of Artificial Intelligence. Also DIST-Technical Report 93-0015, Genova, Italy, 1993.

[Gen69] G. Gentzen. Investigations into logical deduction. In M.E. Szabo, editor, *The collected papers of Gerhard Gentzen*, pages 68–128. North Holland Publishing Company, 1969.

[Giu92] F. Giunchiglia. The GETFOL Manual - GETFOL version 1. Technical Report 92-0010, DIST - University of Genova, Genoa, Italy, 1992.

[GMW79] M.J. Gordon, A.J. Milner, and C.P. Wadsworth. *Edinburgh LCF - A mechanized logic of computation*, volume 78 of *Lecture Notes in Computer Science*. Springer Verlag, 1979.

[GPT94] F. Giunchiglia, P. Pecchiari, and C. Talcott. Reasoning Theories: Towards an Architecture for Open Mechanized Reasoning Systems. Technical Report 9409-15, IRST, Trento, Italy, 1994. Also DIST Technical Report 94-0021, DIST, University of Genova, Italy. Also published as Stanford Computer Science Department Technical note number STAN-CS-TN-94-15, Stanford University.

[GW91] F. Giunchiglia and R.W. Weyhrauch. FOL User Manual - FOL version 2. Manual 9109-08, IRST, Trento, Italy, 1991. Also DIST Technical Report 91-0006, DIST, University of Genova.

[Hai57a] T. Hailperin. A Theory of Restricted Quantification I. *Journal of Symbolic Logic*, 22(1):19–35, 1957.

[Hai57b] T. Hailperin. A Theory of Restricted Quantification II. *Journal of Symbolic Logic*, 22(2):112–129, 1957.

[HS90] U. Hedtstück and P.H. Schmitt. A Calculus for Order-Sorted Predicate Logic with Sort Literals. In *Sorts and Types in Artificial Intelligence*. Springer-Verlag, 1990.

[Kap92] D. Kapur, editor. *Proc. of the 11th Conference on Automated Deduction*, Saratoga Springs, NY, USA, June 1992. Published as Springer LNAI, number 607.

[Kle52] S.C. Kleene. *Introduction to Metamathematics*. North Holland, 1952.

[ORSvH95] S. Owre, J. Rushby, N. Shankar, and F. von Henke. Formal verification for fault-tolerant architectures: Prolegomena to the design of PVS. *IEEE Transactions on Software Engineering*, 20(2):107–125, February 1995.

[Pau79] L. Paulson. Tactics and Tacticals in Cambridge LCF. Technical Report 39, Computer Laboratory, University of Cambridge, 1979.

[Pra65] D. Prawitz. *Natural Deduction - A proof theoretical study.* Almquist and Wiksell, Stockholm, 1965.

[RC89] D. Randell and A.G. Cohn. Modelling Topological and Metrical Properties in Physical Processes. In R.J. Brachman, H. Levesque, and R. Reiter, editors, *Proc. 1st Int. Conference on Knowledge Representation*, pages 357–368. Morgan Kaufmann, 1989.

[Ros53] J.B. Rosser. *Logic for Mathematicians.* McGraw-Hill, 1953.

[SS89] M. Schmidt-Schauss. *Computational Aspects of an Order-Sorted Logic with Term Declarations.* Springer-Verlag, 1989.

[SW90] P.H. Schmitt and W. Wernecke. Tableaux Calculus for Order-Sorted Logic. In Bläsius et al. [BHR90].

[Wal87] C. Walther. *A Many Sorted Calculus Based on Resolution and Paramodulation.* Pitman and Kaufmann Publishers, 1987.

[Wan52] H. Wang. Logic of Many-Sorted Theories. *Journal of Symbolic Logic*, 17(2):105–116, 1952.

[Wei91] C. Weidenbach. A Sorted Logic using Dynamic Sorts. Technical Report MPI-Report MPI-I-91-218, Max-Plank-Institut für Informatik, Saarbrücken, 1991.

[Wei93a] C. Weidenbach. Extending the Resolution Method with Sorts. In *Proc. of the 13th International Joint Conference on Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., 1993.

[Wei93b] C. Weidenbach. Unification in sort theories and its applications. Technical Report MPI-Report MPI-I-93-211, Max-Plank-Institut für Informatik, Saarbrücken, 1993.

[Wey77] R.W. Weyhrauch. A Users Manual for `FOL`. Technical Report STAN-CS-77-432, Computer Science Department, Stanford University, 1977.

[Wey80] R.W. Weyhrauch. Prolegomena to a Theory of Mechanized Formal Reasoning. *Artificial Intelligence*, 13(1):133–176, 1980.

A Proof of theorem 4.1

Theorem 4.1 (Eliminability) *Let $MapVars(A)$ be the conjunction of the equalities $x = x^+$, for every $x \in frees(A)$. Then, for all A, for all $\mathcal{I}$ such that $\mathcal{I} \models_\Sigma MapVars(A)$,*

$$\mathcal{I} \models_\Sigma A \leftrightarrow A^+$$

We prove theorem 4.1 by proving $MapVars(A) \vdash_{\text{SND}} A \leftrightarrow A^+$, and relying on the completeness and correctness theorems proved in next sections[3]. The proof is given by structural induction on A. We give below only the cases with formulas with sorted quantification.

$\boxed{\forall x.B, sort(x) = S}$ Let A be $\forall x.B$. Let u be x^+. The thesis is $MapVars(\forall x.B) \vdash_{\text{SND}} \forall x.B \leftrightarrow \forall u.(S(u) \supset B^+)$. The inductive hypothesis is the sequent (if $x \in frees(B)$) $u = x, MapVars(\forall x.B) \vdash_{\text{SND}} B \leftrightarrow B^+$. We first prove $MapVars(\forall x.B) \vdash_{\text{SND}} \forall x.B \supset \forall u.(S(u) \supset B^+)$ (in the application of the derived inference labeled with *Ind.Hp.* we only show the assumptions in $MapVars$ which are involved in the remaining derivation, e.g. discharged):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\forall x.B]_{(1)}}{B}\,\forall E \quad [u = x]_{(2)}}{B^+}\,Ind.Hp. \quad \dfrac{\dfrac{\dfrac{u = u}{S(u) \supset \exists x.u = x}\,\exists I \quad [S(u)]_{(3)}}{\exists x.u = x}\,\supset E}{B^+}\,\exists E_{(2)}}{S(u) \supset B^+}\,\supset I_{(3)}}{\forall u.(S(u) \supset B^+)}\,\forall I}{\forall x.B \supset \forall u.(S(u) \supset B^+)}\,\supset I_{(1)}$$

Then we prove the converse, i.e $MapVars(\forall x.B) \vdash_{\text{SND}} \forall u.(S(u) \supset B^+) \supset \forall x.B$:

$$\dfrac{\dfrac{\dfrac{\dfrac{[\forall u.(S(u) \supset B^+)]_{(1)}}{S(u) \supset B^+}\,\forall E \quad \dfrac{[S(u) \wedge u = x]_{(2)}}{S(u)}\,\wedge E}{B^+}\,\supset E \quad \dfrac{[S(u) \wedge u = x]_{(2)}}{u = x}\,\wedge E}{B}\,Ind.Hp. \quad \dfrac{\dfrac{\dfrac{\forall x.S(x)}{S(x)}\,\forall E \quad x = x}{S(x) \wedge x = x}\,\wedge I}{\exists u.(S(u) \wedge u = x)}\,\exists I}{B}\,\exists E_{(2)}}{\dfrac{B}{\forall x.B}\,\forall I}{\forall u.(S(u) \supset B^+) \supset \forall x.B}\,\supset I_{(1)}$$

[3]For the sake of clarity of presentation, the theorems have been presented in the body of the paper in an order which is different from the order of dependency of the proofs.

$\boxed{\exists x.B,\ sort(x) = S}$ Let A be $\exists x.B$. Let u be x^+. The thesis is $MapVars(\exists x.B) \vdash_{\text{SND}} \exists x.B \leftrightarrow \exists u.S(u) \wedge B^+$. The inductive hypothesis is $MapVars(\exists x.B), u = x \vdash_{\text{SND}} B \leftrightarrow B^+$. We first prove $MapVars(\exists x.B) \vdash_{\text{SND}} \exists x.B \supset \exists u.S(u) \wedge B^+$ as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{[B]_{(3)} \quad \dfrac{[x = u \wedge S(u)]_{(2)}}{u = x} \quad \dfrac{[x = u \wedge S(u)]_{(2)}}{S(u)}}{S(u) \wedge B^+}\ Ind.Hp.}{\exists u.(S(u) \wedge B^+)}\ \exists I \qquad \dfrac{\dfrac{x = x \quad \dfrac{\forall x.S(x)}{S(x)}\ \forall E}{x = x \wedge S(x)}\ \wedge I}{\exists u.(x = u \wedge S(u))}\ \exists I}{\exists u.(S(u) \supset B^+)}\ \exists E_{(2)} \qquad [\exists x.B]_{(1)}}{\exists u.(S(u) \supset B^+)}\ \exists E_{(3)}}{\exists x.B \supset \exists u.(S(u) \wedge B^+)}\ \supset I_{(1)}$$

Then we prove the converse, i.e. $MapVars(\exists x.B) \vdash_{\text{SND}} \exists u.(S(u) \wedge B^+) \supset \exists x.B$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[S(u) \wedge B^+(u)]_{(2)}}{S(u)} \quad \dfrac{[S(u) \wedge B^+(u)]_{(2)}}{B^+} \quad [u = x]_{(3)}}{B}\ Ind.Hp.}{\exists x.B}\ \exists I \qquad \dfrac{\dfrac{[S(u) \wedge B^+(u)]_{(2)}}{S(u)} \quad \dfrac{u = u}{S(u) \supset \exists x.u = x}\ \exists I}{\exists x.u = x}\ \supset E}{\exists x.B}\ \exists E_{(3)} \qquad [\exists u.(S(u) \wedge B^+)]_{(1)}}{\exists x.B}\ \exists E_{(2)}}{\exists u.(S(u) \wedge B^+) \supset \exists x.B}\ \supset I_{(1)}$$

B Proof of theorem 4.2

We preliminarily state the sorted version of coincidence and substitution lemmas, where the interpretation of terms is restricted to the "right" sort. The proofs are routinary, and are not reported here.

Lemma B.1 (Coincidence) *Let x be a variable of sort S, and $\mathcal{I}$ be a Σ-interpretation. Then*

(i) *for every t not containing x, for every $d \in \mathcal{I}(S)$,*

$$\mathcal{I}(t) = \mathcal{I}^d_x(t)$$

(ii) *for every A not containing free occurrences of x, for every $d \in \mathcal{U}(S)$,*

$$\mathcal{I} \models_\Sigma A \quad \Longleftrightarrow \quad \mathcal{I}^d_x \models_\Sigma A$$

Lemma B.2 (Substitution) *For every Σ-interpretation $\mathcal{I}$, variable x_i of sort S_i, term t_i such that $\mathcal{I}(t_i) \in \mathcal{I}(S_i)$,*

$$\mathcal{I}(t^{t_1}_{x_1} \ldots {}^{t_n}_{x_n}) = \mathcal{I}^{\mathcal{I}(t_1)}_{x_1} \ldots {}^{\mathcal{I}(t_n)}_{x_n}(t)$$

and

$$\mathcal{I} \models_\Sigma (A^{t_1}_{x_1} \ldots {}^{t_n}_{x_n}) \quad \Longleftrightarrow \quad \mathcal{I}^{\mathcal{I}(t_1)}_{x_1} \ldots {}^{\mathcal{I}(t_n)}_{x_n} \models_\Sigma A$$

The following lemma ensures that a term belonging to a certain sort is properly interpreted in every Σ-interpretation.

Lemma 4.1 *For all t, S, $t{\leqslant}S$ iff $\models_\Sigma S(t)$.*

The proof of this lemma is left to the reader, as routine. $t{\leqslant}S$ implies $\models_\Sigma S(t)$ is directly proved by structural induction on t. $\models_\Sigma S(t)$ implies $t{\leqslant}S$ is proved by showing that, if $t{\not\leqslant}S$, then there is a Σ-interpretation $\mathcal{I}$ such that $\mathcal{I} \not\models_\Sigma S(t)$.

Theorem 4.2 (Correctness of SND) *For every A, Γ, $\Gamma \vdash_{\mathrm{SND}} A \quad \Longrightarrow \quad \Gamma \models_\Sigma A$*

Theorem 4.2 is proved by induction on the structure of derivation in SND. In the base case we prove the validity of the sort axioms. $\models_\Sigma \forall x.S(x)$ with x of sort S follows from the definition of satisfaction and validity for Σ-interpretations. In the step case, the correctness of the inference rules for the propositional connectives and equality is independent of the sort information. Below we consider only $\forall I$ and $\exists I$. The cases of $\forall E$ and $\exists E$ are similar. x and y are distinct variables, with x of sort S and y of sort T.

$\boxed{\forall I}$ We prove $\Gamma \models_\Sigma \forall x.A^x_y$ from the inductive hypothesis $\Gamma \models_\Sigma A$ and the applicability conditions of $\forall I$, i.e. $x \notin frees(A)$, $y \notin frees(\Gamma)$, and $x{\leqslant}T$. Let $\mathcal{I} \models_\Sigma \Gamma$. From the coincidence lemma, for all $d \in \mathcal{I}(S)$, $\mathcal{I}^d_y \models_\Sigma \Gamma$, as $y \notin frees(\Gamma)$ and $\mathcal{I}(S) \subseteq \mathcal{I}(T)$. From the inductive hypothesis, $\mathcal{I}^d_y \models_\Sigma A$. From the coincidence lemma, $\mathcal{I}^{dd}_{yx} \models_\Sigma A$, as $x \notin frees(A)$. $\mathcal{I}^{d\mathcal{I}^d_x(x)}_{xy} \models_\Sigma A$, as $\mathcal{I}^d_x(x) = d$. From the substitution lemma, $\mathcal{I}^d_x \models_\Sigma A^x_y$. Then $\mathcal{I} \models_\Sigma \forall x.A^x_y$, from which the thesis.

$\boxed{\exists I}$ We consider two cases. First we prove $\Gamma \models_\Sigma \exists x.A$ from the inductive hypothesis $\Gamma \models_\Sigma A^t_x$ and the applicability condition that $t{\leqslant}S$. Let $\mathcal{I} \models_\Sigma \Gamma$. From the inductive hypothesis, $\mathcal{I} \models_\Sigma A^t_x$. From the substitution lemma, $\mathcal{I}^{\mathcal{I}(t)}_x \models_\Sigma A$, as $t{\leqslant}S$. Let $d = \mathcal{I}(t)$, with $d \in \mathcal{I}(S)$ from the sort lemma. Then there is $d \in \mathcal{I}(S)$ such that $\mathcal{I}^d_x \models_\Sigma A$, i.e. $\mathcal{I} \models_\Sigma \exists x.A$, from which the thesis.
Then we prove $\Gamma \models_\Sigma S(t) \supset \exists x.A$ from the inductive hypothesis $\Gamma \models_\Sigma A^t_x$. Let

$\mathcal{I} \models_\Sigma \Gamma$. If $\mathcal{I}(t) \notin \mathcal{I}(S)$, then $\mathcal{I} \not\models_\Sigma S(t)$, from which $\mathcal{I} \models_\Sigma S(t) \supset \exists x.A$. Let $\mathcal{I}(t) \in \mathcal{I}(S)$. From the inductive hypothesis, $\mathcal{I} \models_\Sigma A^t_x$. From the substitution lemma, $\mathcal{I}^{\mathcal{I}(t)}_x \models_\Sigma A$. Then there is $d = \mathcal{I}(t) \in \mathcal{I}(S)$ such that $\mathcal{I}^d_x \models_\Sigma A$, from which $\mathcal{I} \models_\Sigma \exists x.A$ and $\mathcal{I} \models_\Sigma S(t) \supset \exists x.A$.

C Proof of theorem 4.3

Theorem 4.3 (Completeness of SND**)** *For every* A, Γ, $\Gamma \models_\Sigma A \quad \Longrightarrow \quad \Gamma \vdash_{\text{SND}} A$

The completeness theorem is a direct consequence of lemma C.1.

Lemma C.1 *If* Γ *is consistent, then it is* Σ*-satisfiable.*

The proof of lemma C.1 is as follows. First we introduce maximally consistent sets containing Σ-witnesses (def. C.1, lemma C.2). Second, we define the term interpretation associated with a maximally consistent set (def. C.2) and we show that it is a Σ-interpretation (lemma C.3). Then, we prove a sorted version of Henkin's lemma (lemma C.4): a formula is in a maximally consistent set containing Σ-witnesses iff it is satisfied by the corresponding term interpretation. Lemma C.1 follows directly from lemma C.6 (a consistent set can be extended to a consistent set with Σ-witnesses) and lemma C.7 (the resulting set can be extended to a maximally consistent set). The corresponding term interpretation is the model we want.

Definition C.1 (Maximally consistent set containing Σ-witnesses)

(i) Γ *is said to be maximally consistent iff it is consistent and every formula* A *such that* $\Gamma \cup \{A\}$ *is consistent belongs to* Γ.

(ii) Γ *contains* Σ*-witnesses iff for every formula of the form* $\exists x.A$*, there exists a term* t *such that* $S(t) \in \Gamma$ *and* $(\exists x.A \supset A^t_x) \in \Gamma$

Lemma C.2 (Properties of maximally consistent set containing Σ-witnesses) *Let* Γ *be a maximally consistent set containing* Σ*-witnesses. Then for every* A *and* B*:*

(i) *if* $\Gamma \vdash_{\text{SND}} A$*, then* $A \in \Gamma$.

(ii) *either* $A \in \Gamma$ *or* $\neg A \in \Gamma$

(iii) $(A \vee B) \in \Gamma$ *iff* $A \in \Gamma$ *or* $B \in \Gamma$

(iv) *if* $(A \supset B) \in \Gamma$ *and* $A \in \Gamma$, *then* $B \in \Gamma$

(v) $\exists x.A \in \Gamma$ *iff there is* t *such that* $S(t) \in \Gamma$ *and* $A^t_x \in \Gamma$.

The proof of cases from (i) to (iv) is routinary (see [EFT84]). Case (v,$\Rightarrow$) follows from (iv) applied to $\exists x.A \supset A^t_x$. Case (v,$\Leftarrow$) follows from (i) ($\exists I$ applied to A^t_x); $S(t) \in \Gamma$ can be used to apply (iv) to the conclusion $S(t) \supset \exists x.A$ when $t \not\in S$.

Definition C.2 (Term interpretation) *Let* Γ *be a maximally consistent set containing* Σ*-witnesses. Let* $\sim$ *be the equivalence relation defined as* $t_1 \sim t_2$ *iff* $t_1 = t_2 \in \Gamma$. *Let* $\overline{t}$ *denote the (equivalence) class of terms that are* $\sim$*-related to* t. *The* term interpretation *associated with* Γ, *is the interpretation* $\mathcal{I}_\Gamma = \langle \mathcal{U}_\Gamma, \beta_\Gamma \rangle$ *such that:*

(i) $\|\mathcal{U}_\Gamma\|$ *is the set of equivalence classes of* $\sim$;

(ii) *for every individual constant* c, $\mathcal{U}_\Gamma(c) = \overline{c}$;

(iii) *for every variable* x, $\beta_\Gamma(x) = \overline{x}$

(iv) *for every function symbol* f, $\mathcal{U}_\Gamma(f)(\overline{t}_1, \ldots, \overline{t}_n) = \overline{f(t_1, \ldots, t_n)}$;

(v) *for every predicate symbol* R, $\langle \overline{t_1}, \ldots, \overline{t_n} \rangle \in \mathcal{U}_\Gamma(R)$ *iff* $R(t_1, \ldots, t_n) \in \Gamma$;

Lemma C.3 (Term interpretation is Σ-interpretation) *Let* Γ *be a maximally consistent set containing* Σ*-witnesses. The term interpretation* $\mathcal{I}_\Gamma$ *is a* Σ*-interpretation.*

Cases from (i) to (v) prove that $\mathcal{U}_\Gamma$ is a Σ-structure (see the corresponding conditions of definition 4.3); case (vi) shows that β_Γ is a Σ-assignment.

(i) $\vdash_{\text{SND}} U(t)$ holds for every t (by $\forall E$ from the sort axiom $\forall u.U(u)$). Therefore $U(t) \in \Gamma$, and $\overline{t} \in \mathcal{U}_\Gamma(U)$ for all $\overline{t}$.

(ii) $\vdash_{\text{SND}} S(x)$ holds for every x of sort S (by $\forall E$ from the sort axiom $\forall x.S(x)$). Therefore $S(x)$ is in Γ, and $\overline{x} \in \mathcal{U}_\Gamma(S) \neq \emptyset$.

(iii) For all t, $\vdash_{\text{SND}} S(t) \supset T(t)$ and $S(t) \supset T(t) \in \Gamma$. Then for all $\overline{t}$, if $\overline{t} \in \mathcal{U}_\Gamma(S)$, then $\overline{t} \in \mathcal{U}_\Gamma(T)$, i.e. $\mathcal{U}_\Gamma(S) \subseteq \mathcal{U}_\Gamma(T)$.

(iv) $\vdash_{\text{SND}} S(c)$ by $\forall E$ from the sort axiom $\forall x.S(x)$. Then $S(c) \in \Gamma$, and $\overline{c} = \mathcal{U}_\Gamma(c) \in \mathcal{U}(S)$.

(v) For all $t_1, \ldots, t_n$, $\vdash_{\mathrm{SND}} S_1(t_1) \supset \ldots \supset S_n(t_n) \supset S(f(t_1, \ldots, t_n))$. Then, for all $\overline{t_1} \in \mathcal{U}_\Gamma(S_1), \ldots, \overline{t_n} \in \mathcal{U}_\Gamma(S_n)$ we have $\overline{f(t_1, \ldots, t_n)} = \mathcal{U}_\Gamma(f)(\overline{t_1}, \ldots, \overline{t_n}) \in \mathcal{U}_\Gamma(S)$.

(vi) For every x of sort S, $\vdash_{\mathrm{SND}} S(x)$. Therefore $S(x)$ is in Γ, and $\overline{x} = \beta_\Gamma(x) \in \mathcal{U}_\Gamma(S)$.

Lemma C.4 (Sorted Henkin's lemma) *Let Γ be a maximally consistent set containing Σ-witnesses. Then for every A,*

$$\mathcal{I}_\Gamma \models_\Sigma A \quad \Longleftrightarrow \quad A \in \Gamma$$

The proof is performed by induction on the rank of formulas. The base cases are taken into account by lemma C.5.

Lemma C.5 *Let Γ be a maximally consistent set containing Σ-witnesses.*

(i) *for every t, $\mathcal{I}_\Gamma(t) = \overline{t}$*

(ii) *for every atomic A, $\mathcal{I}_\Gamma \models_\Sigma A$ iff $A \in \Gamma$*

The proof of lemma C.5 is routinary (see [EFT84]), as it does not involve the sort information. For lemma C.4, we only show the case for existentially quantified formulas (the other cases are similar). $\mathcal{I}_\Gamma \models_\Sigma \exists x.A$ iff there is $\overline{t} \in \mathcal{I}_\Gamma(S)$ such that ${\mathcal{I}_\Gamma}_x^{\overline{t}} \models_\Sigma A$, or, for lemma C.5, ${\mathcal{I}_\Gamma}_x^{\mathcal{I}_\Gamma(t)} \models_\Sigma A$. Equivalently, for the substitution lemma, there is $\overline{t} \in \mathcal{I}_\Gamma(S)$: $\mathcal{I}_\Gamma \models_\Sigma A_x^t$ iff there is t such that $S(t) \in \Gamma$ and $A_x^t \in \Gamma$ iff $\exists x.A \in \Gamma$.

Lemma C.6 *Let Γ be a consistent set. Then there is a consistent set Ψ such that $\Gamma \subseteq \Psi$ and Ψ contains Σ-witnesses.*

We prove lemma C.6 under the hypothesis that $frees(\Gamma)$ is finite; the generalization is routinary (see [EFT84]). Let $\exists x_0.A_0$, $\exists x_1.A_1, \ldots$ be a list of all the existentially quantified formulas in $\mathcal{L}$. For every n, we define the witness formula B_n as $\exists x_n.A_n \supset {A_n}_{x_n}^{y_n}$, where y_n is a variable of sort $S_n = sort(x_n)$, such that $y_n \notin frees(\Gamma, \{B_m | m < n\}, \exists x_n.A_n)$. $\Psi = \Gamma \cup \{B_0, B_1, \ldots\}$ clearly contains witnesses. In order to prove that Ψ is consistent, it is enough to prove that $\Gamma_n = \Gamma \cup \{B_m | m < n\}$ is consistent for every n. The proof is by induction on n. Γ_0 is Γ, which is consistent by hypothesis. We assume that Γ_n is consistent for the

induction hypothesis. Suppose, for a contradiction, that $\Gamma_{n+1} = \Gamma_n \cup \{B_n\}$ is inconsistent. Then, for every A there exists $\Theta \subseteq \Gamma_n$ such that $\Theta, B_n \vdash_{\mathrm{SND}} A$, i.e. there is a proof of A from $\Theta \cup (\neg\exists x_n.A_n \vee (A_n)^{y_n}_{x_n})$. Without loss of generality, suppose that A is a sentence. Then we can build the following proof of A from Θ:

$$\dfrac{(2)\vee(4) \qquad \dfrac{\dfrac{\dfrac{[(\neg\exists x_n.B_n)\vee(B_n)^{y_n}_{x_n}]_{(1)}}{\begin{array}{c}\Pi\\ A\end{array}}}{((\neg\exists x_n.B_n)\vee(B_n)^{y_n}_{x_n})\supset A}\supset I_{(1)} \qquad \dfrac{[\neg\exists x_n.B_n]_{(2)}}{(\neg\exists x_n.B_n)\vee(B_n)^{y_n}_{x_n}}\vee I}{A}\supset E \qquad \vdots \qquad \dfrac{\dfrac{\dfrac{[(B_n)^{y_n}_{x_n}]_{(3)}}{(\neg\exists x_n.B_n)\vee(B_n)^{y_n}_{x_n}}\vee I \qquad [\exists x_n.B_n]_{(4)}}{(\neg\exists x_n.B_n)\vee(B_n)^{y_n}_{x_n}}\exists E_{(3)}}{A}\supset E}{A}\vee E_{(2,4)}$$

where $(2) \vee (4)$ is the theorem $(\exists x_n.B_n) \vee (\neg\exists x_n.B_n)$, and $\vdots$ stands for the proof of $((\neg\exists x_n.B_n)\vee(B_n)^{y_n}_{x_n}) \supset A$ (the hypotheses that $y_n \leqslant S_n$ and $y_n \notin frees(\Gamma, \{B_m | m < n\}, \exists x_n.A_n)$ guarantee the correctness of the $\exists E_{(3)}$ step). Hence we have that for any sentence $\Gamma \vdash_{\mathrm{SND}} A$. We can see that Γ_n is inconsistent, by choosing A to be $\exists u.u = u$ and $\neg\exists u.u = u$, which yields a contradiction.

Lemma C.7 *Let Ψ be consistent. Then there is a maximally consistent set Θ with $\Psi \subseteq \Theta$*

The proof of lemma C.7 is routinary (see [EFT84]), as it does not involve the sort information.

D Proofs of theorems 5.1, 5.2, 5.3

Theorem 5.1 (Model equivalence) *For all A, Γ in $\mathcal{L}$,*

$$Uns(\Sigma), SA(\Gamma, A), \Gamma^+ \models A^+ \quad \Longleftrightarrow \quad \Gamma \models_\Sigma A$$

Theorem 5.2 (Derivation Equivalence) *For all A, Γ in $\mathcal{L}$,*

$$SA(\Gamma, A), \Gamma^+ \vdash_{\mathrm{ND}_\Sigma} A^+ \quad \Longleftrightarrow \quad \Gamma \vdash_{\mathrm{SND}} A$$

Theorem 5.3 (Conservativeness) *For all A, Γ in $\mathcal{L}'$,*

$$\Gamma \vdash_{\text{SND}} A \quad \Longrightarrow \quad \Gamma \vdash_{\text{ND}_\Sigma} A$$

The proofs of theorems 5.3 and 5.2 are stated after the proof of theorem 5.1, from which they follow. The proof of theorem 5.1 goes as follows. We first extend the mapping $(.)^+$ from Σ-interpretations to interpretations for $\mathcal{L}'$ (definition D.1), and we prove that for every Σ-interpretation $\mathcal{I}$, the $\mathcal{L}'$-interpretation $\mathcal{I}^+$ satisfies $Uns(\Sigma)$ and $SA(\mathcal{L})$ (lemma D.1). Then, we define the $(.)^-$ mapping, from $\mathcal{L}'$-interpretations to $\mathcal{L}$-interpretations (definition D.2), and we prove that for every $\mathcal{L}'$-interpretation $\mathcal{I}$ satisfying $Uns(\Sigma)$ and $SA(\mathcal{L})$ $\mathcal{I}^-$ is a Σ-interpretation (lemma D.2); furthermore, we prove that $(.)^+$ and $(.)^-$ are inverse functions (lemma D.3). Then we prove that Σ-satisfiability in $\mathcal{I}$ is equivalent to satisfiability in $\mathcal{I}^+$ (lemma D.4). We prove the two directions of theorem 5.1 (lemma D.5 and lemma D.6) with $SA(\mathcal{L})$ instead of $SA(\Gamma, A)$. Finally, we show that the result holds for $SA(\Gamma, A)$ (lemma D.7).

Definition D.1 $((.)^+)$ *Let $\mathcal{I} = \langle \mathcal{U}, \beta \rangle$ be a Σ-interpretation. Then $\mathcal{I}^+$ is the $\mathcal{L}'$-interpretation $\langle \mathcal{U}, \beta^+ \rangle$, where $\beta^+(x^+) = \beta(x)$.*

Lemma D.1 *For every Σ-interpretation $\mathcal{I}$, (i) $\mathcal{I}^+ \models Uns(\Sigma)$, and (ii) $\mathcal{I}^+ \models SA(\mathcal{L})$.*

(i) follows directly from the definitions of Σ-structure and $Uns(\Sigma)$. For (ii), we have $\mathcal{I}^+(x^+) = \beta^+(x^+) = \beta(x) \in \mathcal{I}(S) = \mathcal{I}^+(S)$

Definition D.2 $((.)^-)$ *Let $\mathcal{I} = \langle \mathcal{U}, \beta \rangle$ be a $\mathcal{L}'$-interpretation. We define $\mathcal{I}^-$ as the $\mathcal{L}$-interpretation $\langle \mathcal{U}, \beta^- \rangle$, where $\beta^-(x) = \beta(x^+)$.*

Lemma D.2 *For every $\mathcal{L}'$-interpretation $\mathcal{I}$ such that $\mathcal{I} \models Uns(\Sigma), SA(\mathcal{L})$, $\mathcal{I}^-$ is a Σ-interpretation.*

Let $\mathcal{I}^- = \langle \mathcal{U}, \beta^- \rangle$. $\mathcal{U}$ is a Σ-structure, from the definitions of $Uns(\Sigma)$ and Σ-structure. β^- is a Σ-assignment: for every variable x of sort S $\mathcal{I}^-(x) = \beta^-(x) = \mathcal{I}(x^+) \in \mathcal{I}(S) = \mathcal{I}^-(S)$

Lemma D.3 *For every $\mathcal{L}$-interpretation $\mathcal{I}$, $(\mathcal{I}^+)^- = \mathcal{I}$. For every $\mathcal{L}'$-interpretation $\mathcal{I}$, $(\mathcal{I}^-)^+ = \mathcal{I}$.*

Lemma D.3 follows directly from the definitions (consider that $(.)^+$ and $(.)^-$ are injective mappings).

Lemma D.4 *For every Σ-interpretation $\mathcal{I}$ and for every A in $\mathcal{L}$*

$$\mathcal{I} \models_\Sigma A \quad \Longleftrightarrow \quad \mathcal{I}^+ \models A^+$$

The lemma is proved by induction. We use the fact that for every term t, $\mathcal{I}(t) = \mathcal{I}^+(t^+)$. The cases of propositional connectives are not reported here, as routinary. The cases of unsorted quantifications are similar to the cases of sorted quantifications reported below.

Awff Let A be $P(t_1, \ldots, t_n)$. A^+ is $P(t_1^+, \ldots, t_n^+)$. $\mathcal{I} \models_\Sigma A$ iff $\langle \mathcal{I}(t_1), \ldots, \mathcal{I}(t_n) \rangle \in \mathcal{I}(P)$ iff $\langle \mathcal{I}^+(t_1^+), \ldots, \mathcal{I}^+(t_n^+) \rangle \in \mathcal{I}^+(P)$ iff $\mathcal{I}^+ \models P(t_1^+, \ldots, t_n^+)$.

$\forall x, sort(x) = S$ Let A be $\forall x.B$. Then A^+ is $\forall x^+.(S(x^+) \supset B^+)$. $\mathcal{I} \models_\Sigma \forall x.B$ iff for all $d \in \mathcal{I}(S)$, $\mathcal{I}_x^d \models_\Sigma B$. By induction hypothesis, $\mathcal{I}_x^d \models_\Sigma B$ iff $(\mathcal{I}_x^d)^+ \models B^+$. For all d, either $d \notin \mathcal{I}(S)$ or $(\mathcal{I}_x^d)^+ \models_\Sigma B^+$, i.e. for all d, $(\mathcal{I}^+)_{x^+}^d \models (S(x^+) \supset B^+)$. Equivalently, $\mathcal{I}^+ \models \forall x^+.(S(x^+) \supset B^+)$

$\exists x, sort(x) = S$ Let A be $\exists x.B$. Then A^+ is $\exists x^+.(S(x^+) \wedge B^+)$. $\mathcal{I} \models_\Sigma \exists x.B$ iff there is $d \in \mathcal{I}(S)$ such that $\mathcal{I}_x^d \models_\Sigma B$. By induction hypothesis, $\mathcal{I}_x^d \models_\Sigma B$ iff $(\mathcal{I}_x^d)^+ \models B^+$. Equivalently, there is d such that ${\mathcal{I}^+}_{x^+}^d \models (S(x^+) \wedge B^+)$, i.e. $\mathcal{I}^+ \models \exists x^+.(S(x^+) \wedge B^+)$.

Lemma D.5 *For all A, Γ:*

$$\Gamma \models_\Sigma A \quad \Longrightarrow \quad Uns(\Sigma), SA(\mathcal{L}), \Gamma^+ \models A^+$$

Let $\mathcal{I} \models Uns(\Sigma), SA(\mathcal{L}), \Gamma^+$. For lemma D.2 $\mathcal{I}^-$ is a Σ-interpretation. For lemma D.4 $\mathcal{I}^- \models_\Sigma \Gamma$; then $\mathcal{I}^- \models_\Sigma A$ by hypothesis. For lemma D.4, $(\mathcal{I}^-)^+ \models A^+$. For lemma D.3, $(\mathcal{I}^-)^+ = \mathcal{I}$, from which the thesis.

Lemma D.6 *For all A, Γ:*

$$Uns(\Sigma), SA(\mathcal{L}), \Gamma^+ \models A^+ \quad \Longrightarrow \quad \Gamma \models_\Sigma A$$

Let $\mathcal{I} \models_\Sigma \Gamma$. For lemma D.4 $\mathcal{I}^+ \models \Gamma^+$, and $\mathcal{I}^+ \models A^+$ from the hypothesis. For lemma D.1, $\mathcal{I}^+ \models Uns(\Sigma), SA(\mathcal{L})$. Then $(\mathcal{I}^+)^-$ is a Σ-structure, $(\mathcal{I}^+)^- = \mathcal{I}$, and $\mathcal{I} \models_\Sigma A$, from which the thesis.

Lemma D.7 *For all A^+, Γ^+ in $\mathcal{L}'$:*

$$Uns(\Sigma), SA(\mathcal{L}), \Gamma^+ \models A^+ \quad \Longleftrightarrow \quad Uns(\Sigma), SA(\Gamma, A), \Gamma^+ \models A^+$$

Direction ($\Leftarrow$) by $SA(A, \Gamma) \subseteq SA(\mathcal{L})$. Direction ($\Rightarrow$) is proved as follows. Let $\mathcal{I} = \langle \mathcal{U}, \beta \rangle \models Uns(\Sigma)$, $SA(\Gamma, A), \Gamma^+$. Let β' be an assignment that coincides with β for all variables in $frees(A, \Gamma)$, and such that $\langle \mathcal{U}, \beta' \rangle \models SA(\mathcal{L})$. For the coincidence lemma, $\langle \mathcal{U}, \beta' \rangle \models Uns(\Sigma), SA(\mathcal{L}), \Gamma^+$. From the hypothesis, $\langle \mathcal{U}, \beta' \rangle \models A^+$. For the coincidence lemma, $\langle \mathcal{U}, \beta \rangle \models A^+$.

Theorem 5.2 is a consequence of correctness and completeness results for ND_Σ and SND, and of theorem 5.1 (see fig. 2). Theorem 5.3 is simply proved as follows. As A and Γ do not contain sorted variables, $SA(A, \Gamma) = \emptyset$, and it is possible to choose a suitable map for the variables so that $A^+ = A$ and $\Gamma^+ = \Gamma$. The thesis follows from theorem 5.2.