Henkin semantics for reasoning with natural language

Michael Hahn[1] *and Frank Richter*[2]
[1] Eberhard Karls Universität Tübingen, Tübingen, Germany
[2] Goethe Universität Frankfurt, Frankfurt a.M., Germany

ABSTRACT

The frequency of intensional and non-first-order definable operators in natural languages constitutes a challenge for automated reasoning with the kind of logical translations that are deemed adequate by formal semanticists. Whereas linguists employ expressive higher-order logics in their theories of meaning, the most successful logical reasoning strategies with natural language to date rely on sophisticated first-order theorem provers and model builders. In order to bridge the fundamental mathematical gap between linguistic theory and computational practice, we present a general translation from a higher-order logic frequently employed in the linguistics literature, two-sorted Type Theory, to first-order logic under Henkin semantics. We investigate alternative formulations of the translation, discuss their properties, and evaluate the availability of linguistically relevant inferences with standard theorem provers in a test suite of inference problems stated in English. The results of the experiment indicate that translation from higher-order logic to first-order logic under Henkin semantics is a promising strategy for automated reasoning with natural languages.

Keywords: Henkin semantics, reasoning, reducing higher-order reasoning to first-order reasoning

1 INTRODUCTION

One of the big challenges for applying automated inference to natural language input comes from a stark discrepancy in the preferred logical languages in theoretical semantics on the one hand and in computational semantics on the other. Theoretically-minded linguists custom-

Journal of Language Modelling Vol 3, No 2 (2015), pp. 513–568

arily employ expressive higher-order logics in their theories of meaning in order to elegantly account for important and intricate features of the human language such as intensionality and generalized quantifiers. In contrast to these established linguistic theories, the most successful logical reasoning strategies with natural language rely on theorem provers and model builders for first-order logic. Advanced and sophisticated theories of meaning thus seem entirely out of reach for applications of automated reasoning, and any hope for adequate logic-based reasoning with language may seem doomed even before we start to consider additional challenges such as the necessary integration of world knowledge and discourse pragmatics.

To cope with the discrepancy, previous work by Bos and Markert (2006) on applying first-order inference tools to natural language in part approximated intensions and higher-order quantification in first-order logic, and in part ignored their role in language. Of course, this strategy restricts the fragment of natural language that can be treated, and it forces computational semanticists to recast the theories of formal linguists in a different logical language. Analyses of intensional contexts in terms of possible worlds can be simulated in first-order logic by adding worlds to the first-order structure (Lewis 1968), but some generalized quantifiers such as ‘most’, when given a plausible formal definition of their meaning, can be shown not to be expressible in first-order logic (Barwise and Cooper 1981).

However, on second glance, all hope is not lost for wielding the higher-order descriptions of formal semanticists in computational environments in a more direct, systematic and comprehensive fashion. A standard approach to automated inference with higher-order logic outside of linguistics exploits a reduction to first-order logic that is complete for *Henkin semantics*, a semantics for higher-order logic that is weaker than the standard semantics, but for which complete proof systems exist. In this paper, we explore the application of this idea to natural language input, starting our inferencing toolchain with logical representations for natural language sentences couched in two-sorted Type Theory (Ty2, Gallin 1975), one of the standard higher-order logics favored by formal semanticists.

The inferencing architecture we will introduce thus avoids an error-prone and ad-hoc case-by-case approximation of higher-order phenomena that requires a separate verification of the adequacy of each hand-encoded solution. Instead reasoning starts with the original higher-order representations of linguists, which are reduced to first-order logic by a systematic translation with well-understood properties. Rather than being tailored to specific linguistic applications, the present proposal provides a general translation of full higher-order logic, and the fine-grained semantic representations of the formal semantics literature are accepted as input without any modification. This means that higher-order representations of challenging natural language facts can be developed independently of implementations in formalisms familiar to semanticists without having to worry about a possible manual reduction to first-order logic, and the original representations may then serve as input to automated reasoning. Since the semantics of higher-order logic is preserved in the translation process (in a sense to be elucidated shortly), the first-order translation is guaranteed to be adequate for any input.

We begin by defining a Henkin semantics for Ty2 and illustrating how it differs from its standard semantics, arguing that Henkin semantics is not only formally interesting but also adequate for reasoning with natural language. After defining two translations of different logical strength from higher-order logic to first-order logic and describing their mathematical properties, we assess the practical value of the general strategy outlined above with a test suite of natural-language inference problems that focuses on phenomena that have figured prominently in linguistics. Test items that encode reasoning problems in natural language are translated into Ty2 under standard semantic analyses derived from Montague's seminal PTQ fragment of English (Montague 1973), and from there into first-order logic by our Henkin-complete translation function. We then apply standard first-order inference tools to evaluate the feasibility of automated reasoning on the resulting first-order translations.

In Section 2, we introduce the formal definition of Henkin semantics and argue that it provides much of the proof-theoretic strength needed to formalize linguistically relevant natural language inferences. Section 3 defines a systematic translation from Ty2 to first-order logic and its properties. Section 4 is devoted to the evaluation of the approach, presenting a grammar fragment with meaning postulates, the test suite, and the results of our experiments. In the remaining sections we discuss related work and future perspectives. The appendix contains axioms for the translations, essential proofs of their properties, meaning postulates for lexical items in our grammar, and the test suite.

2 HENKIN SEMANTICS FOR TY2

Validity in first-order logic is semi-decidable, while validity in higher-order logic is not. It follows that there can be no computable translation from higher-order logic to first-order logic that is both sound and complete for standard semantics. However, Henkin (1950) showed that higher-order logic can be given a natural semantics such that the valid formulae are exactly the theorems of a certain formal calculus. As all semi-decidable problems are reducible to first-order theorem proving, the task of proving higher-order theorems for *Henkin semantics* can in principle be reduced to first-order theorem proving.

While not every higher-order tautology is valid for Henkin semantics, we will demonstrate that certain linguistically interesting higher-order theorems that are not expressible in first-order logic indeed are. This observation subsumes, for instance, many natural inferences licensed by the quantifier 'most', which is undefinable in first-order logic.

2.1 *Ty2*

We assume two-sorted Type Theory (Ty2), a standard language for formalizing semantic analyses for natural language (see, e.g., Groenendijk and Stokhof 1982), as representation language.

Classical type theory as formulated by Church (1940) has only two basic types, *e* for entities and *t* for truth values (ι and o in Church's notation). Ty2 has two basic types apart from *t*, namely *e* for entities and *s* for possible worlds. Since classical type theory consists of those expressions of Ty2 in which types containing s do not occur, our translation below can also be applied to analyses in classical one-sorted higher-order logic.

Let us first define the syntax of Ty2, and Henkin semantics. The presentation essentially follows Gallin (1975).

Definition 1. *Types is the smallest set such that:*

- $s, e, t \in$ *Types,*
- *if* $\sigma, \tau \in$ *Types, then* $\langle \sigma\tau \rangle \in$ *Types.*

t is the type of the truth values *true* and *false*, s is the type of possible worlds, and e the type of entities. $\langle\sigma\tau\rangle$ is the type of functions mapping objects of type σ to objects of type τ. We let c^n be an enumeration of the words over some finite alphabet.

Definition 2 (Syntax of Ty2). *The set $\mathscr{L}_{\mathrm{Ty2}}$ of Ty2 terms is the smallest set such that:*

- *for every type τ and every $n \in \mathbb{N}$, $x^n_\tau \in \mathscr{L}_{\mathrm{Ty2}}$ (variables),*
- *for every type τ and every $n \in \mathbb{N}$, $c^n_\tau \in \mathscr{L}_{\mathrm{Ty2}}$ (constants),*
- *if $\alpha_{\langle\sigma\tau\rangle}$ and β_σ are in $\mathscr{L}_{\mathrm{Ty2}}$, then $(\alpha\beta)_\tau$ is in $\mathscr{L}_{\mathrm{Ty2}}$ (function application),*
- *if α_τ is in $\mathscr{L}_{\mathrm{Ty2}}$, then $(\lambda x^n_\sigma \alpha_\tau)_{\langle\sigma\tau\rangle}$ is in $\mathscr{L}_{\mathrm{Ty2}}$ for every $n \in \mathbb{N}$ (lambda abstraction).*

For every type σ, the constants $\dot{\forall}^\sigma_{\langle\langle\sigma t\rangle t\rangle}$ (universal quantifier over objects of type σ), $\dot{\exists}^\sigma_{\langle\langle\sigma t\rangle t\rangle}$ (existential quantifier), $\iota^\sigma_{\langle\langle\sigma t\rangle\sigma\rangle}$ (choice operator), and $\dot{\equiv}^\sigma_{\langle\sigma\langle\sigma t\rangle\rangle}$ (equality) are constants of $\mathscr{L}_{\mathrm{Ty2}}$.[1] Moreover, $\dot{\neg}_{\langle tt\rangle}$, $\dot{\wedge}_{\langle t\langle tt\rangle\rangle}$, $\dot{\vee}_{\langle t\langle tt\rangle\rangle}$ and $\dot{\rightarrow}_{\langle t\langle tt\rangle\rangle}$ are constants of $\mathscr{L}_{\mathrm{Ty2}}$. The dots are intended to prevent confusion with the corresponding logical symbols of first-order logic. Furthermore, for all types σ, τ, ρ, we assume the combinator symbols $\mathbf{I}^\sigma_{\langle\sigma\sigma\rangle}$, $\mathbf{K}^{\sigma,\tau}_{\langle\sigma\langle\tau\sigma\rangle\rangle}$, and $\mathbf{S}^{\rho,\sigma,\tau}_{\langle\langle\rho\langle\sigma\tau\rangle\rangle\langle\langle\rho\sigma\rangle\langle\rho\tau\rangle\rangle\rangle}$. These are all *logical constants*. In addition, there is a countably infinite supply of *non-logical constants* for every type.

As the definition indicates, we write $\alpha, \beta, \ldots$ for meta-variables for terms, c for meta-variables for constants, and σ, τ for meta-variables for types. Terms of the form $(\lambda x\alpha)$ are called lambda abstracts.

We will also need weaker versions of Ty2 that contain fewer terms:

Definition 3 (Restrictions of Ty2). *Let $\mathscr{C}$ be a non-empty set of variables, constants and lambda abstracts. The language $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$, the restriction of Ty2 to $\mathscr{C}$, is the smallest set such that:*

- *$\mathscr{C} \subseteq \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$,*
- *whenever $\alpha \in \mathscr{C}$, then every sub-term of α is also in $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$,*
- *if $\alpha_{\langle\sigma\tau\rangle}$ and β_σ are in $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$, then $(\alpha\beta)_\tau$ is in $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$ (function application).*

[1] The superscript types are regarded part of the constants' names. Generally, we formalize polymorphic constants as families of constants whose names contain the type parameters.

Following standard practice in formal semantics, we employ some abbreviations and conventions that make Ty2 terms look more similar to first-order formulae: Types are omitted where redundant. $\forall x_\sigma$ and $\exists x_\sigma$ denote $\dot{\forall}^\sigma \lambda x_\sigma$ and $\dot{\exists}^\sigma \lambda x_\sigma$, respectively. Variables are often represented by letters other than x, and without a number superscript. In particular, variables of type s are often denoted by w, and variables over propositions, properties and other higher-order objects by P or Q. Functional application is written in an 'uncurried' functional notation: $P(x, y(z))$ stands for $((Px)(yz))$. Logical constants such as $\dot{\rightarrow}$ and $\dot{\equiv}$ are rendered in infix notation.

2.2 *Henkin semantics*

Henkin semantics was originally introduced by Henkin (1950) for one-sorted type theory, but the generalization to Ty2 is straightforward (Gallin 1975, p. 59). We will use a more general variant in which the universes for higher types may be empty, which will be needed for a weak version of our translation.

Definition 4. *A frame D is a collection of mutually disjoint sets $\{D_\sigma\}_{\sigma \in Types}$ such that:*

1. *D_e, $D_s \neq \emptyset$,*
2. *$D_t \subseteq \{T, F\}$,*
3. *for $\langle \sigma \tau \rangle \in Types$, $D_{\langle \sigma \tau \rangle}$ is a (possibly empty) set of functions from D_σ to D_τ.*

An $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-assignment v with respect to a frame D is a mapping from $\mathscr{V}^{\mathscr{C}}$, the variables of $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$, into the domain of D such that variables of type σ are mapped to elements of D_σ.[2] *An $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-interpretation $\mathscr{I}$ is a mapping from the constants of $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$ to D such that constants of type σ are mapped to elements of D_σ, and, for every type σ, the following conditions hold:*

1. *If $\dot{\forall}^\sigma \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$, then $\mathscr{I}(\dot{\forall}^\sigma)(\mathrm{x}) = T$ if and only if $\mathrm{x}(\mathrm{y}) = T$ for every $\mathrm{y} \in D_\sigma$.*
2. *If $\dot{\equiv}^\sigma \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$, then $\mathscr{I}(\dot{\equiv}^\sigma)(\mathrm{x})(\mathrm{y}) = T$ if and only if $\mathrm{x} = \mathrm{y}$ $(\mathrm{x}, \mathrm{y} \in D_\sigma)$.*
3. *If $\dot{\neg} \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$, then $\mathscr{I}(\dot{\neg})(\mathrm{x}) = T$ if $\mathrm{x} = F$ and F if $\mathrm{x} = T$.*
4. *If $\dot{\wedge} \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$, then $\mathscr{I}(\dot{\wedge})(\mathrm{x})(\mathrm{y}) = T$ if and only if $\mathrm{x} = \mathrm{y} = T$.*

[2] Note that $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-assignments w.r.t. D only exist when $D_\tau \neq \emptyset$ for all $x^n_\tau \in \mathscr{C}$; *mutatis mutandis*, the same holds for interpretations.

5. *Analogous definitions are assumed for other connectives and* $\dot{\exists}^{\sigma}$*.*
6. *If* $\iota^{\sigma} \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*, and* $\mathrm{g} \in D_{\langle \sigma t \rangle}$ *is such that* $\mathrm{g}(\mathrm{x}) = T$ *for some* $\mathrm{x} \in D_{\sigma}$*, then* $\mathrm{g}(\mathscr{I}(\iota^{\sigma})(\mathrm{g})) = T$*. Otherwise,* $\mathrm{g}(\mathscr{I}(\iota^{\sigma})(\mathrm{g})) = F$*.*
   Informally, ι *selects an element out of every non-empty set. Because of this property,* ι *is called a choice operator.*
7. *Appropriate equations are assumed for the combinators.*[3]

Given a frame D*, an* $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*-interpretation* $\mathscr{I}$*, and an assignment* v*, the interpretation function* $[\![\cdot]\!]^{v}_{D,\mathscr{I}}$ *is defined on the terms of* $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$ *by induction as follows:*

1. *if* $x^{n}_{\sigma} \in \mathscr{V}^{\mathscr{C}}$*, then* $[\![x^{n}_{\sigma}]\!]^{v}_{D,\mathscr{I}} := v(x^{n}_{\sigma})$*,*
2. *if* $c^{n}_{\sigma} \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$ *is a constant, then* $[\![c^{n}_{\sigma}]\!]^{v}_{D,\mathscr{I}} := \mathscr{I}(c^{n}_{\sigma})$*,*
3. *if* $(\alpha_{\langle \sigma\tau \rangle}\beta_{\sigma})_{\tau} \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*, then* $[\![\alpha\beta]\!]^{v}_{D,\mathscr{I}} := [\![\alpha]\!]^{v}_{D,\mathscr{I}}\left([\![\beta]\!]^{v}_{D,\mathscr{I}}\right)$*,*
4. *if* $(\lambda x^{n}_{\sigma}\alpha_{\tau})_{\sigma\tau} \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*, then* $[\![(\lambda x^{n}_{\sigma}\alpha_{\tau})_{\sigma\tau}]\!]^{v}_{D,\mathscr{I}} :=$ *the function* $f : D_{\sigma} \to D_{\tau}$ *such that* $f(\mathrm{x}) := [\![\alpha_{\tau}]\!]^{v[x^{n}_{\sigma} \mapsto \mathrm{x}]}_{D,\mathscr{I}}$*,*[4]

where the third and the fourth clause result in an undefined value if $[\![\beta]\!]^{v}_{D,\mathscr{I}} \notin D_{\sigma}$*, or if* $[\![\alpha]\!]^{v}_{D,\mathscr{I}}$ *is not defined.*

Definition 5. *A frame* D *with an* $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*-interpretation* $\mathscr{I}$ *is called a general* $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*-model if, for every* $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*-assignment* v *and every term* $\alpha_{\sigma} \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*,* $[\![\alpha_{\sigma}]\!]^{v}_{D,\mathscr{I}}$ *is a well-defined element of* D_{σ}*. A term* α_{t} *is called a* $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*-Henkin tautology iff* $[\![\alpha_{t}]\!]^{v}_{D,\mathscr{I}} = T$ *for all general* $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*-models* D *and all* $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*-assignments* v*.*

If $\mathscr{C}$ *contains all logical constants, we refer to* $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*-Henkin tautologies simply as Henkin tautologies, and to general* $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*-models as general* $\mathscr{L}_{\mathrm{Ty2}}$*-models.*

A general $\mathscr{L}_{\mathrm{Ty2}}$*-model* $\langle D, \mathscr{I} \rangle$ *is called a full model if, for every* $\langle \sigma\tau \rangle \in$ *Types,* $D_{\langle \sigma\tau \rangle}$ *contains all functions from* D_{σ} *to* D_{τ}*. A term* α_{t} *is called a tautology in the standard sense if* $[\![\alpha_{t}]\!]^{v}_{D,\mathscr{I}} = T$ *for all full models* $\langle D, \mathscr{I} \rangle$ *and all assignments* v*.*

[3] When $\mathscr{C}$ does not contain all lambda abstracts, the presence of the (finitely many) combinators still yields the full strength of Henkin semantics as it is usually defined. However, as we will mostly be concerned with weaker versions of Henkin semantics, the combinators play no role for our immediate purposes, and we omit the equations for reasons of space. See Hindley and Seldin 2008, p. 110, for the necessary equations.

[4] Note that $v[x^{n}_{\sigma} \mapsto \mathrm{x}]$ is a $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$*-assignment*, as $x^{n}_{\sigma} \in \mathscr{C}$.

Informally, there are two types of models for higher-order logic. *Full models* are defined by the requirement that they contain any higher-order function over their domain that in principle exists. *General $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-models* are only required to provide *some* interpretation for every term of $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$. The notion of 'general model' is in turn graded by the set $\mathscr{C}$: having more elements in $\mathscr{C}$ results in a stronger semantics, i.e., a semantics that allows fewer models.

Every full model is also a general $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-model for every $\mathscr{C}$, but the converse does not hold: some general $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-models are not full models. Similarly, if $\mathscr{C} \subseteq \mathscr{D}$, then every general $\mathscr{L}^{\mathscr{D}}_{\text{Ty2}}$-model is also a general $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-model. There is an inverse relationship between the sets of tautologies for the various notions of semantics. Since every full model is a general model, all $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-Henkin tautologies hold in every full model and are therefore tautologies in the standard sense. However, there are tautologies in the standard sense that are not $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-Henkin tautologies for any $\mathscr{C}$. Analogously, if $\mathscr{C} \subseteq \mathscr{D}$, then all $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-tautologies are also $\mathscr{L}^{\mathscr{D}}_{\text{Ty2}}$-tautologies. It is in this sense that the notion of 'general model' yields a weaker semantics than the standard semantics of higher-order logic, and that increasing $\mathscr{C}$ results in a stronger semantics. A semantics based on full models is called a *standard semantics*, and a semantics based on general models is a *Henkin semantics*. When $\mathscr{C}$ contains all logical constants, our definitions of 'general models' and 'Henkin tautologies' coincide with the usual definition of Henkin semantics, since all lambda abstracts can be defined with the combinators (Hindley and Seldin 2008, p. 110). The significance of Henkin semantics for our application rests on the following theorem of Henkin (1950):

Theorem 6 (Henkin's Completeness Theorem). *There is a (finitary) calculus that generates exactly the set of Henkin tautologies of $\mathscr{L}_{\text{Ty2}}$.*

Because the set of tautologies in the standard sense is not recursively enumerable, no such theorem is available in the standard case.

Example

Let us assume that $\mathscr{C} = \{woman_{\langle et\rangle}, dance_{\langle et\rangle}\} \cup \{x^n_\tau : n \in \mathbb{N}, \tau \in \{e, t, \langle ee\rangle, \langle et\rangle, \langle tt\rangle, \langle t\langle tt\rangle\rangle, \langle\langle et\rangle t\rangle, \langle\langle et\rangle\langle\langle et\rangle t\rangle\rangle\}\}$. Consider the frame characterized by the following sets:

- $D_e := \{a, b, c, d, e\}$,
- $D_{\langle ee\rangle} := \{\{a \mapsto a, b \mapsto a, c \mapsto a, d \mapsto a, e \mapsto a\}\}$,

- $D_{\langle et\rangle}, D_{\langle tt\rangle}, D_{\langle t\langle tt\rangle\rangle}, D_{\langle\langle et\rangle t\rangle}, D_{\langle\langle et\rangle\langle\langle et\rangle t\rangle\rangle}$ are the full sets of functions with the respective domains and ranges,
- for other types τ, we set $D_\tau := \emptyset$.

For this frame, we define an interpretation $\mathscr{I}$ given by $\mathscr{I}(woman) = \chi_{D_e}$ and $\mathscr{I}(dance) = \chi_{\{a,b\}}$. Since D_τ is empty for most τ, the frame does not constitute a full model. However, the frame together with $\mathscr{I}$ is a general $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-model.

We now add a constant $most_{\langle\langle et\rangle\langle\langle et\rangle t\rangle\rangle}$, representing the natural-language quantifier 'most'. 'Most' is often assumed to express that more than half of the elements in the restrictor are also in the nuclear scope (e.g., Gamut 1991, p. 252, Westerståhl 2011). In $most(woman, dance)$, *woman* is the restrictor and *dance* is the nuclear scope, and the term is true iff more than half of the women are also dancers. Barwise and Cooper (1981, C13) show that, under this interpretation, the meaning of 'most' cannot be expressed in first-order logic.[5] However, it is definable in $\mathscr{L}_{Ty2}$. Generalized to sets of any cardinality, MOST(P,Q) is true if and only if the cardinality of $P \cap Q$ is strictly greater than that of $P\backslash Q$. Equivalently, MOST(P,Q) is true if and only if $P \cap Q \neq \emptyset$ and there is no surjective mapping from $P\backslash Q$ to $P \cap Q$. If we identify subsets of D_e with their characteristic functions, i.e., the functions of type $\langle et\rangle$, we can express this definition in $\mathscr{L}_{Ty2}$ as follows:

(1) $\forall P_{\langle et\rangle} \forall Q_{\langle et\rangle} : most(P,Q) \leftrightarrow [\exists x_e(P(x) \wedge Q(x)) \wedge$
$\forall f_{\langle ee\rangle} : (\forall y_e : (P(y) \wedge \neg Q(y)) \rightarrow (P(f(y)) \wedge Q(f(y))))$
$\rightarrow \exists x_e(P(x) \wedge Q(x) \wedge \forall z_e : (P(z) \wedge \neg Q(z)) \rightarrow f(z) \neq x)]$
'MOST(P,Q) holds if and only if $P \cap Q \neq \emptyset$, and
for every mapping f from $P\backslash Q$ to $P \cap Q$
there is an $x \in P \cap Q$ that is not in the image of $P\backslash Q$ under f'
(i.e., f is not surjective)

[5] The intuition is that, when describing the cardinality of a set using a formula of first-order logic, one can only count up to some fixed finite number which depends on the formula, not being able to distinguish sets of greater cardinality. Choosing for each first-order formula (1) a sufficiently large universe of a model M and (2) sets U and V such that $U, V, M\backslash U, M\backslash V$, and the relative difference of U and V are each sufficiently large, we see that MOST(U, V) cannot be first-order definable. For the version generalized to infinite sets that we will consider, the undefinability follows more easily from the compactness theorem.

In the sense of the standard semantics of higher-order logic, this definition indeed ensures that MOST has the desired model-theoretic interpretation: in a full model that satisfies (1), $\mathscr{I}(most)(\chi_P, \chi_Q)$ holds if and only if the cardinality of $P \cap Q$ is strictly greater than that of $P \backslash Q$.

This is not always true for general models. To see this, consider the general $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-model we constructed. Since $D_{\langle\langle et\rangle\langle\langle et\rangle t\rangle\rangle}$ contains all possible functions, it includes in particular a function such that (1) becomes true when $\mathscr{I}(most)$ is set to this function. Under this interpretation, the term $((most\ woman)\ dance)$ is true in the model: the single function $f : D_e \rightarrow D_e$ included in the frame maps $\mathscr{I}(woman) \backslash \mathscr{I}(dance)$ to $\{a\} \subsetneq \mathscr{I}(woman) \cap \mathscr{I}(dance)$. In other words, there is no surjective function in this particular general model from $P \backslash Q$ to $P \cap Q$, as required for *most* according to (1). However, intuitively the statement 'most women dance' is not satisfied in the model: $\mathscr{I}(woman)$ contains five elements, while $\mathscr{I}(dance)$ only contains two elements. Thus, even when the definition of 'most' is satisfied in a general model, it need not actually have the intended model-theoretic interpretation. The problem with definitions like (1) is that, in a general model, the domain of the quantifier $\forall^{\langle ee\rangle}$ is not the set of functions from D_e to D_e. Instead it is $D_{\langle ee\rangle}$, which need not contain all functions from D_e to D_e.

Henkin semantics comes closer to standard semantics when $\mathscr{C}$ contains more terms – in particular, if $\mathscr{C}$ contains all logical constants of $\mathscr{L}_{\mathrm{Ty2}}$, then every general $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-model for which D_e and D_s are finite is a full model. The reason is that every function between two finite sets D_σ and D_τ is definable with the choice operator. On the other hand, if D_e or D_s is infinite, then by the Löwenheim-Skolem theorem there will always be general $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-models which are not full models, and in which the interpretation of 'most' differs from the one intended.

2.3 *Henkin semantics for natural language*

The fact that MOST cannot be defined in a model-theoretically adequate way in Henkin semantics might be taken as evidence that it is too weak to express the concept 'most' meaningfully. But this is not the case. Many interesting facts about MOST are logical consequences of (1) under Henkin semantics. A case in point is monotonicity, one of the properties of generalized quantifiers that have received significant attention in linguistics (Barwise and Cooper 1981; Westerståhl 2011). MOST is upward monotonic in the second argument:

Proposition 7. *MOST is upward monotonic in the second argument:*
In $\mathscr{L}_{\mathrm{Ty2}}$*:* $\forall P_{et} Q_{et} B_{et} : (most(P, Q) \wedge (\forall x_e : Q(x) \to B(x))) \to most(P, B)$
Informally: If $\mathrm{MOST}(P, Q)$ *and* $B \supseteq Q$ *hold, then* $\mathrm{MOST}(P, B)$ *holds.*

The upward monotonicity of MOST in its second argument corresponds to the linguistic observation that the inferences in (2) are valid. In fact, different quantifiers exhibit different inference patterns, showing that these monotonicity properties are both interesting and non-trivial. For instance, 'few' does not license the parallel pattern (3):

(2) a. Most children are playing in the street. ⊢ Most children are playing.

b. Most men sing and dance. ⊢ Most men dance.

(3) a. Few children are playing in the street. ⊬ Few children are playing.

b. Few men sing and dance. ⊬ Few men dance.

A system for automated reasoning from natural language should account for these facts. Proposition 7 (and, by extension, formalizations of the inferences in (2)) are consequences of (1) under Henkin semantics. To see this, consider the following elementary argument:

Proof. Let $\langle D, \mathscr{I} \rangle$ be a general $\mathscr{L}_{Ty2}$-model in which $\mathrm{MOST}(P, Q)$ and $B \supseteq Q$ hold. Clearly, $P \cap B \supseteq P \cap Q \neq \emptyset$. Let $f \in D_{\langle ee \rangle}$ with $\{f(x) : x \in P \backslash B\} \subseteq P \cap B$. For all $x \in D_e$, set $\pi(x)$ to be x if $x \in P \cap Q$ and an arbitrary element of $P \cap Q$ otherwise. Define $f' : D_e \to P \cap Q$ by $f'(x) := \pi(f(x))$. As a suitable π can be defined in $\mathscr{L}_{\mathrm{Ty2}}$ with the choice operator ι,[6] $f' \in D_{\langle ee \rangle}$. By the assumption $\mathrm{MOST}(P, Q)$, we know that $f'|_{P \backslash Q} : P \backslash Q \to P \cap Q$ is not surjective. Thus, $f|_{P \backslash B} : P \backslash B \to P \cap B$ cannot be surjective. As f was arbitrary, $\mathrm{MOST}(P, B)$ holds. □

As we only assumed that $\langle D, \mathscr{I} \rangle$ is a general $\mathscr{L}_{\mathrm{Ty2}}$-model, Proposition 7 is a Henkin tautology. It should also be noted that the proof crucially relies on the choice operator, and the proposition does indeed not hold in all general $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-models if $\mathscr{C}$ is too small.

[6] Informally, we use lambda abstraction to define $A_x := \{x\} \cap P \cap Q$ if $x \in P \cap Q$ and $A_x := P \cap Q$ otherwise. Then we set $\pi(x) := \iota(A_x)$. Formally, $\pi := [\![\lambda x^1_e . \iota^e \lambda x^2_e \left(x^3(x^2) \wedge x^4(x^2) \wedge ((x^3(x^1) \wedge x^4(x^1)) \to x^1 = x^2)\right)]\!]^{v[x^3 \mapsto \chi_P,\ x^4 \mapsto \chi_Q]}_{D, \mathscr{I}}$.

A similar argument shows that Henkin semantics is also strong enough to prove that, if there are at least four objects of type e, MOST is not downward monotonic in either argument, and not upward monotonic in the first argument. It is also possible to formalize more specific numerical inferences, such as 'If exactly four out of five members of P are also in Q, then most members of P are in Q.' Three of the four semantic postulates for 'most' given by Barwise and Cooper (1981, p. 208)[7] are provable from our definition of 'most' under Henkin semantics as well.

It may seem surprising that Henkin semantics is strong enough to prove non-trivial facts about MOST, even though it cannot define it in a model-theoretically adequate way. The point is that many consequences of (1) do not depend on the existence of functions that are not definable by lambda abstraction, and are, for that reason, true in every general $\mathscr{L}_{\mathrm{Ty2}}$-model.[8] This is an instance of a general phenomenon. Although concrete mathematical theorems can be constructed that are true in all full models but not valid for Henkin semantics, we are not aware of any known theorem of this kind that is not of a meta-mathematical nature and is interesting to mathematicians working outside of logic. Given this it seems plausible that Henkin semantics provides all the proof-theoretical strength that is needed for typical natural language inferences.

3 TRANSLATING TY2 TO FIRST-ORDER LOGIC

The crucial step for leveraging the power of first-order reasoning engines when coming from semantic representations in higher-order logic is in the formulation of an appropriate translation from higher-

[7] In our notation, they are the following: (1) MOST(A,A) always holds, (2) upward monotonicity in the second argument, (3) if $A \neq \emptyset$, then MOST(A,X) is true for some but not all sets X, (4) if $A \neq \emptyset$, then MOST(A,X) and MOST(A,Y) together imply $X \cap Y \neq \emptyset$. (4) follows under the standard semantics, the other three postulates also follow under Henkin semantics. To be precise, our definition proves MOST(A,A) only under the assumption $A \neq \emptyset$, as MOST($\emptyset$,$\emptyset$) is false according to our axiom. Depending on whether one views it as being intuitively true or false, our axiom could be modified to evaluate MOST($\emptyset$,$\emptyset$) to true.

[8] As opposed to consequences that do depend on the existence of such functions and, for that reason, are only guaranteed to hold in full models; they may be false in some general models.

order logic to first-order logic. In this section, we will show how the translation from Ty2 to first-order logic can be effected in such a way that Henkin tautologies are translated into first-order tautologies. The guiding idea is that the translation of terms of Ty2 into terms and formulae of first-order logic preserves the term structure as faithfully as possible and aims at exploiting the strengths of first-order provers by translating terms representing connectives and quantifiers into the corresponding symbols of first-order logic. Moreover, two groups of first-order axioms are added in the process that encode the typing and the intended behavior of the translations of Ty2 terms. Given these axioms the translations of Henkin tautologies are theorems of first-order logic – the translation is *complete* for Henkin semantics. It is also *sound*: if the translation of a term is a first-order tautology, then the term itself must be a Henkin tautology.

3.1 *Translation*

The translation consists of three parts: a type translation T_{ty} (translating types into first-order terms), a term translation T_{term} (translating terms of Ty2 into terms of first-order logic), and a formula translation T_f (translating Ty2 terms of type t into first-order formulae). The overall translation T of a term of type t is obtained as its formula translation with the addition of two groups of axioms. The components of the translation will be described next.

Types are represented by first-order terms. The basic types t, s, e are directly represented by first-order constants. Higher types are represented by terms of first-order logic by replacing $\langle \cdot\cdot \rangle$ by $g(\cdot,\cdot)$ as follows:

(4) a. $T_{ty}(\tau) := \tau$ if $\tau = e, s, t$
  b. $T_{ty}(\langle \sigma\tau \rangle) := g(T_{ty}(\sigma), T_{ty}(\tau))$

The idea behind the translation of Ty2 terms is that terms of type t are translated into first-order formulae, and other terms into first-order terms. We first define a *term translation* T_{term}, translating every Ty2 term into a first-order term. Polymorphic constants are represented by first-order functions whose arguments represent their type arguments. For instance, ι^e is translated as $\iota(e)$, where ι is a one-place first-order function symbol. Other constants and variables are translated as themselves: $T_{term}(c_\tau) := c$, where c is a first-order constant, and $T_{term}(x_\tau) := x$. Note that while variables and constants are translated as themselves, the type information attached to the terms is not directly accessible to the first-order language and will be encoded in additional axioms.

Functional application is translated recursively as a two-place function symbol:

(5) $T_{term}(\alpha\beta) := f(T_{term}(\alpha), T_{term}(\beta))$

Lambda abstracts are translated by introducing a function symbol whose arguments represent the free variables of the lambda abstract. Formally, assume that we are given a term $\lambda x.\alpha$ with free variables $\{(v_1)_{\sigma^1}, ..., (v_n)_{\sigma^n}\}$. Then $T_{term}(\lambda x_\tau.\alpha_\sigma) := g_{\lambda x.\alpha}(v_1, ..., v_n)$, where $g_{\lambda x.\alpha}$ is a fresh n-place function symbol which by itself does not carry any meaning. Its intended behavior will be encoded in an additional axiom.

Our third translation function, *formula translation* (T_f), is only applied to terms of type t; it translates them into first-order formulae. Propositional connectives, quantifiers, and the equality operator are translated into the corresponding logical symbols of first-order logic whenever possible. The remaining terms of type t are converted to first-order formulae using the predicate symbol *isTrue*:

(6) a. $T_f((\dot\circ\alpha_t)\beta_t) := T_f(\alpha) \circ T_f(\beta)$
if $\circ$ is a binary propositional connective

b. $T_f(\dot\neg\alpha_t) := \neg T_f(\alpha)$

c. $T_f(\dot\forall^\tau(\lambda x_\tau.\alpha_t)) := \forall x : \mathit{hasType}(x, T_{ty}(\tau)) \rightarrow T_f(\alpha)$ (similarly for $\exists$)

d. $T_f(\dot\forall^\tau \alpha_{\langle\tau t\rangle}) := \forall x : \mathit{hasType}(x, T_{ty}(\tau)) \rightarrow \mathit{isTrue}(f(T_{term}(\alpha), x))$
if α is not a lambda abstract (similarly for $\exists$)

e. $T_f((\dot\equiv^t\alpha)\beta) := (T_f(\alpha) \leftrightarrow T_f(\beta))$

f. $T_f((\dot\equiv^\tau\alpha)\beta) := (T_{term}(\alpha) = T_{term}(\beta))$ if $\tau \neq t$

g. $T_f(\alpha_t) := \mathit{isTrue}(T_{term}(\alpha))$ when no other case applies

If α_t is a term of type t, the overall translation $T(\alpha)$ is defined to be the formula translation $T_f(\alpha)$.

To illustrate the definitions, let us consider the term in (7a). This term straightforwardly encodes an (extensional) translation of the sentence 'Most men sing and dance.' The actual Ty2 term behind the simplified notation in (7a) is (7b). By the preceding definitions, its term translation is (7c). Thus, the overall (formula) translation, as given by (6g), is the first-order formula (7d).

(7) a. $\mathit{most}(\mathit{man}, \lambda x_e.\mathit{sing}(x) \wedge \mathit{dance}(x))$

b. $((\mathit{most}\ \mathit{man})\ \lambda x_e.\phi)$ with $\phi = (\dot{\wedge}\ (\mathit{sing}\ x))\ (\mathit{dance}\ x)$

c. $T_{\mathit{term}}((\mathit{most}\ \mathit{man})\ \lambda x_e.\phi)$
$= f(f(\mathit{most}(e), \mathit{man}), T_{\mathit{term}}(\lambda x_e.\phi))$ (by 5)
$= f(f(\mathit{most}(e), \mathit{man}), g_{\lambda x_e.\phi})$

d. $\mathit{isTrue}(f(f(\mathit{most}(e), \mathit{man}), g_{\lambda x_e.\phi})$

Note that the term translation of $\mathit{most}_{\langle\langle et\rangle\langle\langle et\rangle t\rangle\rangle}$ as $\mathit{most}(e)$ follows from assuming that *most* is a polymorphic constant $\mathit{most}^{\sigma}_{\langle\langle \sigma t\rangle\langle\langle \sigma t\rangle t\rangle\rangle}$ with type argument $\sigma = e$ in our example.

Axioms

To ensure that translations of Henkin tautologies are in fact provable, axioms need to be added that encode the meaning and the intended behavior of the function and predicate symbols. They are stated in the first-order language of the translation.

Type information is not encoded in the first-order translation of Ty2 terms. A first group of axioms guarantees the correct typing of all objects. For instance the following axiom states that the result of applying a functor of type $\langle \sigma\tau\rangle$ to an argument of type σ has type τ:

(8) $\forall x_0 \forall x_1 \forall x_2 : [\exists x_3(\mathit{hasType}(x_0, g(x_3, x_2)) \ \wedge\ \mathit{hasType}(x_1, x_3))]$
$\rightarrow \mathit{hasType}(f(x_0, x_1), x_2)$

A second group of axioms encodes postulates of Henkin's system, defining the types and the intended behavior of constants and functions. For instance, given a type τ, the next axiom states that the translation $\iota(T_{ty}(\tau))$ of the iota operator ι^{τ} selects an element from every non-empty set of objects of type τ:

(9) $\forall y : \mathit{hasType}(y, g(T_{ty}(\tau), t)) \rightarrow$
$\big[(\exists\ z\ \mathit{isTrue}(f(y,z))) \rightarrow \mathit{isTrue}(f(y, f(\iota(T_{ty}(\tau)), y)))\big]$
'For every object y of type $\langle \tau, t\rangle$ such that $y(z) = T$ for some z, $y(\iota^{\tau}(y)) = T$ also holds.'

For every function symbol $g_{\lambda x.\alpha}$, there is an axiom which states that, given arguments of the appropriate types, the function has the value defined by the lambda abstract. More formally, assuming that the free variables of α are $\{v^1_{\sigma_1}, ..., v^n_{\sigma_n}\}$, the axiom takes the form (10a) if α is of type t, and (10b) otherwise:

(10) a. $\forall v_1, ..., v_n : (hasType(v_1, T_{ty}(\sigma^1)) \wedge \ ... \ \wedge hasType(v_n, T_{ty}(\sigma^n)))$
$\rightarrow [hasType(g_{\lambda x.\alpha}(v_1, ..., v_n), g(T_{ty}(\tau), t))$
$\wedge \ \forall x : hasType(x, T_{ty}(\tau))$
$\rightarrow [isTrue(f(g_{\lambda x.\alpha}(v_1, ..., v_n), x)) \longleftrightarrow T_f(\alpha)]]$

b. $\forall v_1, ...v_n : [hasType(v_1, T_{ty}(\sigma^1)) \wedge ... \wedge hasType(v_n, T_{ty}(\sigma^n))] \rightarrow$
$[hasType(g_{\lambda x.\alpha}(v_1, ..., v_n), g(T_{ty}(\tau), T_{ty}(\sigma)))$
$\wedge \ \forall x : hasType(x, T_{ty}(\tau)) \rightarrow f(g_{\lambda x.\alpha}(v_1, ..., v_n), x) = T_{term}(\alpha)]$

In the case of example (7c), the defining axiom of $g_{\lambda x_e.\phi}$ is (11).

(11) $hasType(g_{\lambda x_e.\phi}, g(e, t)) \wedge \forall x : hasType(x, e) \ \rightarrow$
$\big[isTrue(f(g_{\lambda x_e.\phi}, x)) \longleftrightarrow (isTrue(f(sing, x)) \wedge isTrue(f(dance, x)))\big]$

Given $\mathscr{C} \subset \mathscr{L}_{\text{Ty2}}$, we define the $\mathscr{C}$-axiomatization, $\mathscr{A}^{\mathscr{C}}$, as the set containing the first group of axioms (for typing) and the defining axioms for all constants, variables, and lambda abstracts in $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$. The intention is that $\mathscr{A}^{\mathscr{C}}$ provides the necessary information to prove $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-Henkin tautologies.[9] We can show that this is indeed the case:[10]

Theorem 8. *Let $\mathscr{C} \subset \mathscr{L}_{\text{Ty2}}$ and $\alpha_t \in \mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$. Then α is an $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-Henkin tautology if and only if $\mathscr{A}^{\mathscr{C}} \vdash T(\alpha)$.*

In this sense our translation is sound and complete for Henkin semantics.

3.2 *Restricting the axiomatization*

The strength of Henkin semantics and, in consequence, the usefulness of the first-order translation of higher-order meaning characterizations of natural language expressions depends on the choice of $\mathscr{C}$. On the one hand, we have seen that choosing $\mathscr{C}$ to be too

[9] The full set of axioms can be found in Appendix A.

[10] A proof is given in Appendix B.1.

small may result in linguistically relevant inferences not being covered. On the other hand, when automated reasoning techniques come into play, a surplus of axioms may easily distract the algorithms, making automated inference inefficient to the point of being practically infeasible.

In our experiments to be described in the next section we will use two axiomatizations. For a term α_t, the *strong axiomatization* $\mathscr{A}^s(\alpha)$ is constructed from the set $\mathscr{C}$ containing all constants, variables, and lambda abstracts in α and, furthermore, all logical constants of Ty2. If $\mathscr{C}$ contains instances of a polymorphic constant, such as ι^τ for some type τ, a single axiom is used for all types, like (12), replacing the infinitely many axioms in (9). This choice keeps $\mathscr{A}^s(\alpha)$ finite. Due to the combinators, the strong axiomatization has the full strength of Henkin semantics.

(12) $$\forall x \forall y : \mathit{hasType}(y, g(x,t)) \rightarrow \\ [(\exists z\ \mathit{isTrue}(f(y,z))) \rightarrow \mathit{isTrue}(f(y, f(\iota(x), y)))]$$

The *weak axiomatization* $\mathscr{A}^w(\alpha)$ is constructed from the set $\mathscr{C}(\alpha)$ that contains only the lambda abstracts, variables, and constants occurring in α. If α contains instances of a polymorphic constant, only axioms for those specific instances occurring in α are used. We can go even further and leave out constants and lambda abstracts when they are eliminated by the formula translation. More precisely, we add logical constants to $\mathscr{C}(\alpha)$ only when they are translated into corresponding first-order constants rather than into first-order connectives or quantifiers. Similarly, lambda abstracts enter $\mathscr{C}(\alpha)$ only when they do not exclusively occur as arguments of constants which represent first-order quantifiers.

Unlike the strong axiomatization, the weak axiomatization lacks the full power of Henkin semantics, but it also has considerable advantages. As it introduces fewer axioms than the translation with strong axiomatization, it might remove an unnecessary burden from the theorem provers. Where they fail for the strong axiomatization, they might still be able to prove theorems of weak translations of semantic representations of natural language. More importantly, under certain conditions the weak translation has finite models, which is highly relevant and desirable in the context of automated reasoning, since finite models can be constructed automatically.[11] We obtain the following theorem:[12]

Theorem 9. *Assume $\alpha_t \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$ is true in a general $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-model for which D_e and D_s are finite. Then $\bigwedge_{\phi \in \mathscr{A}^w(\alpha)} \phi \wedge T(\alpha)$ is true in some finite (first-order) model.*

However, note that satisfiability of the weak translation of α_t only ensures satisfiability in a $\mathscr{L}^{\mathscr{C}(\alpha)}_{\mathrm{Ty2}}$-model, not necessarily satisfiability in a $\mathscr{L}_{\mathrm{Ty2}}$-model. In the context of our application, this means that weak translations of Henkin validities might not be provable, and that models for weak translations may not correspond to $\mathscr{L}_{\mathrm{Ty2}}$-models.

3.3 *Relationship to previous translations*

The translation we outlined in this section is similar to previous translations from higher-order logic to first-order logic, in particular to the 'lambda lifting' translation of Meng and Paulson (2008) and to the translation of Hurd (2002), who also encodes types as first-order terms. Compared to approaches which represent types by means of first-order predicate symbols (Kerber 1992), typing by terms offers the advantage that it can be expressed with finitely many axioms. This is of course crucial for the application of automated reasoning tools. The main difference between our formulation and Meng and Paulson (2008) resides in the special treatment of connectives and quantifiers in the formula translation, T_f. Treating the logical constants separately makes it possible to exploit the strengths of first-order theorem provers at the inferencing step. Unlike Hurd (2002) and Meng and Paulson (2008), we provide a formal proof of soundness and completeness (Appendix B.1).

4 TESTING AND EVALUATION

We have defined a translation from Ty2 to first-order logic and made precise in which sense it preserves the semantics of Ty2. To assess the feasibility of automated inference on the resulting first-order formulae in a linguistic context, we now apply our translation and standard

[11] The possibility of finite models may be surprising at first, as even $\mathscr{A}^w(\alpha)$ seems to model an infinite set of types. However, note that $\mathscr{A}^w(\alpha)$ does not contain an axiom that demands that e, s, and t and the higher types be distinct.

[12] A proof is given in Appendix B.2.

first-order reasoning engines to a set of natural language reasoning problems of the type commonly considered in linguistics. Our selection focuses on sentences with lexical elements whose semantic analysis involves intensionality and generalized quantifiers, because these are typically cited as the main motivation for higher-order logic rather than first-order logic in the semantic characterization of natural language expressions. A sizable part of our test suite is derived from the FraCaS test suite (Cooper *et al.* 1996), which was created precisely for evaluating the semantic competence of natural language processing systems.

We will first introduce a fragment of English with Montague-style semantic representations and standard meaning postulates for (classes of) lexical elements from the literature. We then describe the contents and structure of our test suite and proceed to assess the performance of first-order inference engines (comprising theorem provers and model builders) on the task of classifying valid and invalid inferences that are handed to them in the form of first-order translations of the higher-order logical representations which our grammar assigns to the test items. The inference engines of course also draw on the meaning postulates as additional axioms. Throughout our evaluation, we will disregard complications arising from possible ambiguities and only consider pre-determined intended readings of our items.

The experiments will show that the weak translation performs significantly better than the strong translation, confirming or refuting 87.7% of those items in the test suite where a proof or refutation exists in principle. While every item that is challenging due to intensionality is correctly recognized, items involving generalized quantifiers are considerably harder for automated reasoning.

4.1 *Fragment*

Our fragment is derived from the English textbook grammar of Blackburn and Bos (2005), who construct semantic representations directly in first-order logic. Their grammar architecture is well-suited for our purposes because its modular design easily supports alternative semantic representation languages by simply plugging in other lexical semantic specifications and adding syntactic rules where needed. Moreover, Blackburn and Bos' grammar is already equipped with an interface to different reasoning engines that we can exploit for evaluating the performance of inference engines on our first-order translations.

The semantic analyses are inspired by Montague's PTQ fragment (Montague 1973), with two major changes: As laid out in the previous sections, we use Ty2 rather than Intensional Logic (IL). Ty2 offers technical advantages (Friedman and Warren 1980, p. 323), and, as a version of typed lambda calculus, its formal properties are well-understood. Montague's representations can be translated straightforwardly into Ty2, since IL can be regarded a sublanguage of Ty2 (Gallin 1975). Second, we follow Bennett (1974) and Dowty *et al.* (1981, p. 188) in representing the arguments of extensional predicates as individuals rather than individual concepts. For instance, *walk* is translated into a term of type $\langle s\langle et\rangle\rangle$, while Montague chose the more elaborate $\langle s\langle\langle se\rangle t\rangle\rangle$.

Representative lexical entries are shown in Figure 1. They are mostly standard. 'Believe' and 'know' take as their arguments a possible world, a proposition, and an entity that represents the agent (Montague 1973). Adverbs attaching to VPs map properties to properties. We translate the definite article by means of the ι operator, i.e., a choice function (von Heusinger 1997). We opt for a uniform treatment of all adjectives as functions mapping properties to properties, following Montague (1970). Generalized quantifiers are rendered as functions of type $\langle\langle et\rangle\langle\langle et\rangle t\rangle\rangle$, i.e., as relations between sets of objects of type e. The fragment licenses both singular and plural noun phrases, but no special plural semantics is assumed; the occurrence of plurals is restricted to NPs with quantifiers such as 'most' and 'many'.

The context-free grammar rules of the fragment stipulate how the semantic representations of daughter constituents are combined to derive the semantic representation of their mother node. The typical mode of composition is functional application. The phrase structure rules of our grammar needed for the test suite are shown in Figure 2 together with their semantic composition rules. The fragment generates one translation per syntactic analysis and does not account for scope ambiguities. This is not a substantial restriction since ambiguities could be captured by adopting one of Blackburn and Bos' alternatives of semantic composition with a more sophisticated underspecified semantics by means of dominance constraints. Our choice here is

Cat.	Words	Translation
V$_{intr}$	dance	$dance_{\langle s\langle et\rangle\rangle}$
V$_{tr}$	see	$\lambda P_{\langle\langle s\langle et\rangle\rangle t\rangle}\lambda w_s\lambda x_e.P(\lambda w_s\lambda y_e\ see_{\langle s\langle e\langle et\rangle\rangle\rangle}(w_s, x_e, y_e))$
V$_{i\text{-}tr}$	seek	$\lambda P_{\langle s\langle\langle s\langle et\rangle\rangle t\rangle\rangle}\lambda w_s\lambda x_e.seek_{\langle s\langle\langle s\langle\langle s\langle et\rangle\rangle t\rangle\rangle\langle et\rangle\rangle\rangle}(w, P, x)$
V$_{cop}$	be	$\lambda P_{\langle\langle s\langle et\rangle\rangle t\rangle}\lambda w_s\lambda x_e.P(\lambda w_s\lambda y_e\ (x = y))$
V$_s$	know	$\lambda P_{\langle st\rangle}\lambda w_s\lambda x_e.know_{\langle s\langle\langle st\rangle\langle et\rangle\rangle\rangle}(w, P, x)$
V$_{aux}$	does not	$\lambda P_{\langle s\langle et\rangle\rangle}\lambda w_s\lambda x_e.\neg P(w, x)$
Adv	possibly	$\lambda P_{\langle s\langle et\rangle\rangle}\lambda w^1_s\lambda x_e.possibly_{\langle s\langle\langle st\rangle t\rangle\rangle}(w^1, \lambda w^2_s P(w^2, x))$
Adj	tall	$\lambda P_{\langle s\langle et\rangle\rangle}\lambda w_s\lambda x_e.tall_{\langle s\langle\langle s\langle et\rangle\rangle\langle et\rangle\rangle\rangle}(w, P, x)$
Det	most	$\lambda P_{\langle s\langle et\rangle\rangle}\lambda Q_{\langle s\langle et\rangle\rangle}.most_{\langle\langle et\rangle\langle\langle et\rangle t\rangle\rangle}(P(w_s), Q(w_s))$
	every	$\lambda P_{\langle s\langle et\rangle\rangle}\lambda Q_{\langle s\langle et\rangle\rangle}.\forall x_e(P(w_s, x) \rightarrow Q(w_s, x))$
	the	$\lambda P_{\langle s\langle et\rangle\rangle}\lambda Q_{\langle s\langle et\rangle\rangle}.Q(w_s, \iota^e(P(w_s)))$
PN	John	$\lambda P_{\langle s\langle et\rangle\rangle}.P(w_s, john_e)$
N	unicorn	$unicorn_{\langle s\langle et\rangle\rangle}$
P	in	$in_{\langle\langle\langle s\langle et\rangle\rangle t\rangle\langle\langle s\langle et\rangle\rangle\langle s\langle et\rangle\rangle\rangle\rangle}$
Conj	and	$\lambda P_{\langle s\langle et\rangle\rangle}\lambda Q_{\langle s\langle et\rangle\rangle}\lambda w_s\lambda x_e.(P(w, x) \wedge Q(w, x))$

Figure 1: Lexical Entries. For every category, an example word is given

S:$\alpha\beta \rightarrow$ NP:α VP:β
VP:$\alpha \rightarrow$ V$_{intr}$:α
VP:$\alpha\beta \rightarrow$ V$_{tr}$:α NP:β
VP:$\alpha(\lambda w_s.\beta) \rightarrow$ V$_{i\text{-}tr}$:α NP:β
VP:$\alpha(\lambda w_s.\beta) \rightarrow$ V$_s$:α S:β
VP:$\lambda x \exists P_{\langle s\langle e\rangle\rangle}\alpha(P, w, x) \rightarrow$ V$_{cop}$ Adj:α
VP:$\alpha\beta \rightarrow$ V$_{cop}$:α NP:β
VP:$\beta(\alpha, \gamma) \rightarrow$ VP:α Conj:β VP:γ
VP:$\alpha\beta \rightarrow$ V$_{aux}$:α VP:β

VP:$\alpha\beta \rightarrow$ Adv:α VP:β
VP:$\alpha\beta \rightarrow$ VP:β PP:α
PP:$\alpha\beta \rightarrow$ P:α NP:β
NP:$\alpha\beta \rightarrow$ Det:α N:β
NP:$\alpha \rightarrow$ PN:α
N:$\alpha\beta \rightarrow$ Adj:α N:β

Figure 2: Phrase Structure Rules

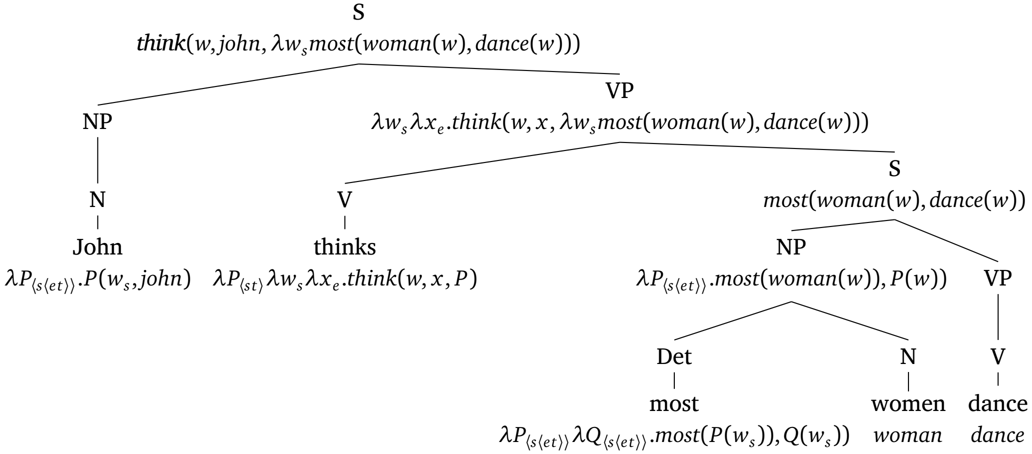
S
$think(w, john, \lambda w_s most(woman(w), dance(w)))$
VP
$\lambda w_s \lambda x_e.think(w, x, \lambda w_s most(woman(w), dance(w)))$
NP
N
John
$\lambda P_{\langle s\langle et\rangle\rangle}.P(w_s, john)$
V
thinks
$\lambda P_{\langle st\rangle} \lambda w_s \lambda x_e.think(w, x, P)$
S
$most(woman(w), dance(w))$
NP
$\lambda P_{\langle s\langle et\rangle\rangle}.most(woman(w)), P(w))$
VP
Det
most
$\lambda P_{\langle s\langle et\rangle\rangle} \lambda Q_{\langle s\langle et\rangle\rangle}.most(P(w_s)), Q(w_s))$
N
women
$woman$
V
dance
$dance$

Figure 3: An analysis in our fragment

motivated by simplicity and the compatibility of the easiest choice of composition mechanism with our main objectives.

The analysis of the sentence ‘John thinks most women dance’ is shown in Figure 3. The following translations illustrate the coverage of the semantic fragment:[13]

(13) Mia possibly dances.
$possibly(w^1_s, \lambda w^2_s.dance(w^2_s, mia_e))$

(14) Mia thinks that John dances.
$think(w^1_s, mia_e, \lambda w^2_s.dance(w^2_s, john_e))$

(15) Most men sing and dance.
$most(man(w_s), \lambda x_e(sing(w_s, x_e) \land dance(w_s, x_e)))$

(16) The blond man dances.
$dance(w_s, \iota^e_{\langle\langle et\rangle e\rangle}(blond(w_s, man)))$

(17) John seeks a unicorn.
$seek(w_s, john_e, \lambda w_s \lambda P_{\langle s\langle et\rangle\rangle} \exists x_e \ (unicorn(w_s, x_e) \land P(w_s, x_e)))$

[13] With the notation of arguments we follow the linguistic convention of putting subjects before objects to enhance readability. The relationship to a strict Ty2 representation should be transparent.

4.2 Meaning postulates

The semantic representations we obtain from the grammar are insufficient for drawing inferences that go beyond simple first-order tautologies expressed in natural language. A substantive portion of the semantic import of words such as 'most' and 'believe' is hidden behind inconspicuous Ty2 constants such as *most* and *believe*. Since these constants by themselves are atomic expressions with arbitrary meaning, further information about their actual meaning must be made available for exploitation in reasoning.

There are two ways to add the relevant information: either by stating meaning postulates in $\mathscr{L}_{\mathrm{Ty2}}$ and adding them as axioms, or by restricting the class of models to those where the interpretations of the constants satisfy certain restrictions. A prominent example of the first option is Montague (1973);[14] the second option was chosen in the semantic postulates of Barwise and Cooper (1981) and in the treatment of generalized quantifiers in Discourse Representation Theory (Kamp and Reyle 1993, Def. 4.24). In the present context, an axiomatic solution is to be preferred as it makes it possible to enlist our translation functions to also translate potentially higher-order meaning postulates to first-order logic. In effect, the first-order translations of the postulates may simply be added to the axiomatization $\mathscr{A}$ of the first-order translation. The situation is more complicated if the information is supplied model-theoretically, as Henkin's completeness theorem need not remain true if the class of permissible models is constrained. Therefore, we opt for the first solution and supply information about constants such as *believe* and *most* by meaning postulates in $\mathscr{L}_{\mathrm{Ty2}}$.

To see the impact of meaning postulates on reasoning and to appreciate the relevance of the translation functions for them, we discuss a selection of postulates for representative constants in our fragment.[15] The examples will also indicate the sorts of difficult semantic questions which have to be addressed in formulating appropriate axioms.

[14] Montague actually understood meaning postulates as constraints on the interpretations, but the completeness theorem for Henkin semantics guarantees that this is equivalent to treating them as axioms.

[15] The full set of meaning postulates is given in Appendix C.

Verbs: belief and knowledge

There is a considerable amount of work on the logic of knowledge and belief from a philosophical point of view (cf. Hintikka 1962; Rescher 2005 for an overview). It has often been argued that belief and knowledge should be closed under logical inferences (Rescher 2005). We assume a principle of logical omniscience which states that if an agent knows (thinks) something, she knows everything which follows from it logically (see (18)). Such a postulate is not without problems as no actual person could be aware of every logical truth, but we accept it as a general consequence of the standard possible-worlds analysis of propositional attitudes. We also assume that only true propositions can be known (Rescher 2005), as formalized in (19):

(18) Deductivity Axiom

a. $\forall x_e \forall P_{\langle st \rangle} \forall Q_{\langle st \rangle} \forall w^1_s : \mathit{think}(w^1, P, x) \rightarrow$
$\big(\forall w^2_s (P(w^2) \rightarrow Q(w^2))\big) \rightarrow \mathit{think}(w^1, Q, x)$

b. 'If x knows/believes P in world w^1 and $P \rightarrow Q$ holds necessarily, then x knows/believes Q in world w^1.'

(19) Veridicality Axiom

a. $\forall x_e \forall P_{\langle st \rangle} \forall w_s (\mathit{know}(w, P, x) \rightarrow P(w))$

b. 'If x knows P in world w, then P is true in world w.'

Adjectives

Adjectives are commonly classified based on the inference patterns they license (Kamp and Partee 1995, Partee 1995). As examples like (20a) show, adjectives like 'blond' are *intersective* (see (20b)). This property is formalized by the meaning postulate (20c) (Partee 1995, p. 324).

(20) a. Mia is a blond woman. Mia is a robber. ⊢ Mia is a woman and Mia is a blond robber.

b. $[\![\mathit{blond}\ \mathrm{N}]\!] = [\![\mathit{blond}]\!] \cap [\![\mathrm{N}]\!]$

c. For each intersective adjective meaning ADJ:
$\exists P_{\langle s \langle et \rangle \rangle} \forall w_s \forall Q_{\langle s \langle et \rangle \rangle} \forall x_e : \mathrm{ADJ}(w, Q, x) \longleftrightarrow [P(w, x) \wedge Q(w, x)]$
where P, which is uniquely defined by the axiom, represents the set $[\![\mathrm{blond}]\!]$ in (20b).

Similar meaning postulates account for other *subsective* and for *privative* adjectives (Partee 1995).[16]

Other adjectives, such as 'alleged', 'potential', and 'arguable', are neither subsective nor privative. They do not allow any inference on whether the property denoted by the noun holds: an 'alleged robber' may or may not be a robber. For the modal modifier 'alleged' we adopt the following postulate, adapted from Jespersen and Primiero (2013, p. 104):

(21) a. $\forall P_{\langle s\langle et\rangle\rangle} \forall x^1_e \forall w^1_s : \mathit{alleged}(w^1, P, x^1)$
$\leftrightarrow \exists x^2_e\, \mathit{allege}(w^1, x^2, \lambda w^2(P(w^2, x^1)))$

b. 'Somebody alleges that x is a P if and only if x is an alleged P.'

Adverbs

It is often assumed that 'necessarily' can be modeled via universal quantification over possible worlds (Montague 1973). This is formalized by the following postulate (Gamut 1991, p. 201, MP7):

(22) a. $\forall w^1_s \forall P_{\langle st\rangle} \big(\mathit{necessarily}(w^1, P) \leftrightarrow \forall w^2_s P(w^2)\big)$

b. 'P is necessarily true if and only if it is true in all worlds.'

'Possibly' is characterized by replacing the universal quantifier by an existential quantifier. Note that the world argument w^1_s plays no role and is only needed because we assume a uniform analysis of all adverbs, and the extension of many adverbs does depend on the world.

Generalized quantifiers

In (1) we saw how 'most' can be defined in Ty2. Certain quantifiers have straightforward definitions in first-order logic. Besides 'all' and 'some', these include, for instance, 'exactly two', 'at most two', and 'only'. (23a) provides a (simplistic) definition of 'only' as in 'Only men danced':

(23) a. $\forall P_{\langle et\rangle} \forall Q_{\langle et\rangle} (\mathit{only}(P, Q) \leftrightarrow \forall x_e (Q(x) \rightarrow P(x)))$

b. 'ONLY(P,Q) holds if and only if $Q \subseteq P$.'

[16] Subsective adjectives are a superclass of intersective adjectives. They license the inference that the property denoted by the noun holds: a skillful writer is a writer, but need not be a skillful violinist even if she is known to be a violinist. Privative adjectives such as 'fake' license the inference that the property denoted by the noun does not hold: a fake diamond is not a diamond.

There are other quantifiers whose meaning is less straightforward to capture, including 'few' and 'many'. However, we can indirectly characterize these quantifiers by postulating rules concerning properties such as monotonicity (Barwise and Cooper 1981, p. 209). While our two examples are upward and downward monotonic, respectively, in the second argument (Barwise and Cooper 1981, p. 185, SP 2), it is less clear whether they are also monotonic in the first argument (Barwise and Cooper 1981, p. 185). We assume that they are (see (24a)). We also state that 'few' and 'many' are incompatible (see (24b)), and postulate that 'few' holds if the intersection of its two arguments is empty (see (24c)). These axioms are somewhat weaker than the optional axiom SP4 (NOT MANY ⇔ FEW) of Barwise and Cooper (1981, p. 209), which seems unnatural to us.

(24) a. i. FEW is downward-monotonic in both arguments.
ii. MANY is upward-monotonic in both arguments.
b. $\neg(\text{FEW}(P, Q) \wedge \text{MANY}(P, Q))$
c. $P \cap Q = \emptyset \rightarrow \text{FEW}(P, Q)$

The axioms, whose rendering in $\mathscr{L}_{Ty2}$ is similar to what we saw for 'most' in (1), suffice to prove important facts about these quantifiers and to make relevant predictions on the validity of natural language inferences. For instance, they entail that MANY is true if the extension of the second argument is 'large', while FEW is true if it is 'small', and that MANY is *conservative*, i.e., inferences like 'Many men dance' ⇒ 'Some men dance' are valid, which corresponds to Axiom SP6 in Barwise and Cooper (1981, p. 209). Conversely, they predict that 'No men dance' is incompatible with the statement 'Many men dance'.

Considering the gradual and context-dependent nature of these two quantifiers, not many more inferences seem possible without appealing to a notion of discourse context.

Example

As an illustration of the interaction of the grammar fragment, the meaning postulates, and the translations from Ty2 to first-order logic, consider (25), a variant of one of the examples in (2). The Ty2 translations generated by our fragment for the premise and the conclusion are given in (26). The respective first-order translations are shown in (27).

(25) Most women sing and dance. ⊢ Most women dance.

(26) a. $most(woman(w), \lambda x(sing(w,x) \wedge dance(w,x))$

b. $most(woman(w), dance(w))$

(27) a. $isTrue(f(f(most, f(woman, w)), g_\phi(w)))$
where g_ϕ is defined by
$\forall w : hasType(w,s) \rightarrow [hasType(g_\phi(w), g(e,t))$
$\wedge\ \forall x : (hasType(x,e) \rightarrow ([isTrue(f(f(sing,w),x))$
$\wedge\ isTrue(f(f(dance,w),x))] \longleftrightarrow isTrue(f(g_\phi(w),x))))]$

b. $isTrue(f(f(most, f(woman, w)), f(dance, w)))$

To show that the inference in (25) is valid, we need to prove that, given the axioms and the first-order translations of the meaning postulates, (27a) logically entails (27b). We need the meaning postulate for *most*, whose first-order translation is too complex to be easily readable, but whose meaning is essentially captured by the informal explanation in (1). At this point the problem of proving the entailment in (25) has been reduced to proving a first-order formula which consists of the elements of $\mathscr{A}$, the translation of (1) and (27a) as its premises, and of (27b) as its conclusion.

For the proof, one may first remodel the higher-order proof of Proposition 7 in the first-order translation. The defining axiom of g_ϕ can then be exploited to prove that $isTrue(f(g_\phi(w),x))$ entails $isTrue(f(f(dance,w),x))$, which corresponds to the fact that ‘x sings and dances’ logically entails ‘x dances’. By the first-order version of (1), the claim (27b) follows.

4.3 *Test suite*

We created a small test suite for natural language inference which requires solving inference problems that have figured prominently in formal semantics research. Inferences relying on world knowledge typical for prominent tasks such as the Recognizing Textual Entailment challenges (Dagan *et al.* 2009) are not addressed with our test suite, because we are interested in the feasibility and quality of reasoning with first-order translations of higher-order meaning specifications of natural language rather than in the bigger (and even more intricate) question of modeling typical human reasoning by means of other types of knowledge resources. The test suite contains 117 items divided into six sections which focus on *modality*, *knowledge and belief*, *generalized quantifiers*, *adjectives*, *de dicto readings*, and *first-order inferences*, respectively.

Each item consists of a set of *premises*, a *conjecture*, and a symbol connecting those two. The items are grouped in three classes: If the premises entail the conjecture, the inference is *valid*. If the premises are incompatible with the conjecture, we call the inference *contradictory*. Items for which the correctness of the inference is not determined by their form or by meaning postulates are *contingent*; the inferences that they represent might be supported by some models but not by others.[17] In each item the conjecture is separated from the premises by a symbol that indicates the class to which the item belongs. If the inference is valid, the separator is '⊢'; '⊢ NON' designates contradictory items, and '⋉' appears in contingent items. Consider the following example:

(28) Mia is a woman. Mia dances. ⊢ A woman dances.

The first two sentences are the premises, 'A woman dances' is the conjecture. The conjecture is entailed by the premise, as indicated by the symbol '⊢'.[18]

52 items are valid (44.4%), 12 are contradictory (10.3%), and 53 are contingent (45.3%). These judgments are based on whether the Ty2 representations of the sentences that are provided by our grammar fragment support the inference under standard higher-order semantics or not, assuming the meaning postulates as axioms.[19] For every one of our items, its membership in the three inference classes coincides for standard semantics and Henkin semantics. One item, (3.24), which tests for monotonicity properties of 'most', is special in that it requires the strong axiomatization. For all others the weak axiomatization is sufficient.

[17] Logically speaking, this means that the inferences in the last class are also invalid (like those in the second class).

[18] Our *conjecture* corresponds to what the literature on Textual Entailment calls *hypothesis* (Dagan *et al.* 2009). The subtle difference in terminology is meant to stress that there is a deeper difference in the conception of what exactly constitutes reasoning with natural language.

[19] To put it differently, the inference patterns follow the linguistic theory our fragment of English implements.

29 items in the sections on adjectives and generalized quantifiers are derived from the FraCaS test suite. The original FraCaS test suite contains question-answer pairs, but it was later converted by MacCartney (2009) to the format employed in our test suite. The following example from FraCaS together with MacCartney's conversion illustrates the difference:

(29) Original FraCaS format (item 197)

a. Premise: John has a genuine diamond.

b. Question: Does John have a diamond?

c. Answer: Yes

(30) Converted: John has a genuine diamond. $\vdash$ John has a diamond.

For our experiments, we draw on those parts of FraCaS that are covered by our fragment. It captures 18% (14 out of 80 items) of the FraCaS section on generalized quantifiers and 65% (15 out of 23 items) of the section on adjectives. Other items would require nontrivial additions to syntax or lexical items for which there is no standard analysis in the semantics literature.

In three instances the predictions implied by our grammar and meaning postulates do not align with those assumed in FraCaS: in two cases additional information about the expression 'on time' would be needed to infer that finishing on time implies finishing. In the third case the reason for the deviation is due to different assumptions about the properties of certain adjectives in FraCaS compared to what our meaning postulates assume. It is important to note that these differences between the predictions of our fragment and the FraCaS annotation result from differences in linguistic modeling, not from a weakness of Henkin semantics – given our meaning postulates, the predictions are the same for Henkin semantics and standard semantics.

4.4 *Experiment*

The goal of the experiment was to assess to what extent the translation supports efficient automated inference on reasoning problems that are typically encountered in formal semantics research. To this end we applied first-order reasoners to the natural language inference problems in the test suite. The transformation pipeline from the test suite to the application of inference engines is straightforward: The inference problems encoded in the test suite were translated to Ty2 by parsing the natural language sentences in the test items according to our grammar fragment. This step resulted in a syntactic analysis coupled with higher-order logical representations. The latter served as input to the translation to first-order logic introduced in Section 3. The first-order formulae were then ready to be processed by freely available first-order reasoners, following the implementation developed by Bos (2004).

Following Bos, two types of reasoning tools were employed: *theorem provers* and *finite model builders*. The theorem provers try to find a *proof* for each first-order formula, while the model builders try to construct a finite *model*. A complete theorem prover will find a proof for every valid formula, and it will find a proof for the negation of every contradictory formula. Thus, ideally, a proof or refutation can be found for the first-order translation of every valid or contradictory inference. However, by complexity and undecidability results of first-order logic, proving even a small formula may take a very long time, and there is no general algorithm determining whether or not a formula has a proof. In particular, there is no general procedure for showing that a formula is contingent. Finite model builders provide a partial solution to this problem: If a formula is contingent, there exist models for both the formula and its negation. If they are finite, these models can be found by a model builder. Since statements made in natural language often concern situations involving finitely many objects, it may be expected that the restriction to finite models is not critical and that for many inference problems either a proof or a counter-model is found in a reasonable amount of time. If this is the case, an automated decision of many natural-language inferences is possible.

Let us now take a closer look at the technicalities involved in putting this idea to work. As indicated earlier, the premises and the conjecture of each test item were translated to Ty2 representations according to the specifications of the grammar fragment. The *premise term* of a test item is the term $p := [\alpha_1 \wedge ... \wedge \alpha_n \wedge \beta_1 \wedge ... \wedge \beta_k]$, where $\alpha_1, ..., \alpha_n$ are the meaning postulates, and $\beta_1, ..., \beta_k$ are the Ty2 translations of the premises 1 to k. Let γ be the Ty2 translation of the conjecture. Using the taxonomy introduced in the previous section, an inference is valid if and only if $p \rightarrow \gamma$ is a tautology. It is contradictory if and only if $p \rightarrow \neg\gamma$ is a tautology. If neither of these cases holds, the inference pattern is contingent.

The inference engine constructed from theorem provers and model builders was tasked to determine if $\mathscr{A} \vdash T(p \rightarrow \gamma)$ or $\mathscr{A} \vdash T(p \rightarrow \neg\gamma)$ holds, with $\mathscr{A}$ either the strong or weak axiomatization. Table 1 summarizes the questions to the inference engine and their possible answers. Obtaining an answer is of course constrained by the general undecidability of the questions in first-order logic, entailing the risk of non-terminating searches. The theorem provers try to find a proof for either $T(p \rightarrow \gamma)$ or $T(p \rightarrow \neg\gamma)$ given the axioms $\mathscr{A}$. If the inference pattern is contingent, no proof will be found. The model builder tries to find a finite model for either $\mathscr{A} \cup \{T(p \wedge \gamma)\}$ (the inference is not contradictory) or $\mathscr{A} \cup \{T(p \wedge \neg\gamma)\}$ (the inference is not valid). If translations of natural language expressions are well-behaved for our purposes, a proof or refutation is found whenever an inference is valid or contradictory, and both a model and a counter-model are found whenever an inference pattern is contingent. Under these circumstances it is possible to determine if an item is valid, contingent, or contradictory.

	Valid	Contingent	Contradictory
$\mathscr{A} \vdash T(p \rightarrow \gamma)$	proof	–	–
$\mathscr{A} \vdash T(p \rightarrow \neg\gamma)$	–	–	proof
$\mathscr{A} \cup \{T(p \wedge \gamma)\}$	model	model	–
$\mathscr{A} \cup \{T(p \wedge \neg\gamma)\}$	–	model	model

Table 1:
Maximal possible output for valid, contingent, and contradictory inference patterns

With our experiments we are interested in determining how well our first-order translation of typical natural language reasoning problems behaves in supporting these decisions with currently available standard first-order reasoning tools. The implementation was based on the theorem provers Spass (Weidenbach 2001), E (Schulz 2004), and Prover9 (McCune 2005–2010), and on the model builder Mace4 (McCune 2005–2010). The provers and the model builder were assigned a maximum of 30 seconds to work on each problem, and were terminated if this was insufficient to find a result.

Two experiments were conducted, one with the strong axiomatization $\mathscr{A}^s(p \rightarrow \gamma)$ and one with the weak axiomatization $\mathscr{A}^w(p \rightarrow \gamma)$. Since finite model building techniques are restricted to finite models and the strong axiomatization has only infinite models, the experiment with the strong axiomatization was run with the theorem provers alone. Only those meaning postulates α_i were included in the premise term p which belonged to constants occurring in the translation of the input. It is to be expected that meaning postulates unrelated to the input will usually not be relevant, at the same time, their presence would likely slow down the search more than they would help.

Since a term that is a $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-tautology for some $\mathscr{C}$ is also a Henkin tautology and a tautology in the standard sense, finding a proof with the strong or the weak translation guarantees that an inference is valid. Model building behaves differently: finding a model for the weak translation only guarantees satisfiability in a weak notion of semantics, but not satisfiability in a general $\mathscr{L}_{\mathrm{Ty2}}$-model, much less satisfiability in a full model. Applied to weak translations, our inference engine may therefore deem invalid an inference that would in fact be valid under standard semantics. As mentioned above, our test suite contains only one item whose treatment requires the strong axiomatization. For all other items, the predictions are the same for the semantics underlying the weak translation and for standard semantics.

4.5 *Results*

The overall success rates of the provers on valid and contradictory items in the test suite are summarized in Table 2. The figure in the column 'Some' expresses the percentage of items for which at least one prover found a proof or refutation. The 'strong' row shows results for the Henkin-complete axiomatization $\mathscr{A}^s$, the 'weak' row for the translation with the weakened axiomatization $\mathscr{A}^w$.

Table 3 shows the percentage of proofs found within a certain time interval, ranging from 0.1 up to 5 seconds.

There was no test item for which a proof was found under the strong axiomatization but not under the weak axiomatization. In other words, for our test items no proofs were lost by weakening the axiomatization. The performance of the model builder is summarized in Table 4.

Table 2:
Success rates of provers

	Spass	Prover9	E	Some
strong	24.6%	47.7%	23.1%	50.8%
weak	53.8%	76.9%	38.5%	87.7%

	≤ 0.1 s	≤ 1 s	≤ 2 s	≤ 5 s
strong	24.2%	79%	83.9%	91.9%
weak	45.5%	74.5%	82.7%	94.5%

Table 3:
Time required by provers

Recall (models found where expected)	78.7%
Accuracy (models expected where found)	98.5%
Determined items (among contingent ones)	56.6%
% of models found within 0.1 seconds	91.1%
% of models found within a second	96.3%
% of models found within two seconds	96.3%
% of models found within five seconds	97.8%

Table 4:
Performance of model builder

		Total	Sp	P9	E	Mace4	Success
first-order	valid	4	2	4	4	3/0/0	4
	contradictory	1	1	1	1	0/1/0	1
	contingent	1	0	0	0	1/1/1	1
modality	valid	8	7	8	7	7/0/0	8
	contingent	7	0	0	0	7/4/4	4
knowledge/ belief	valid	6	5	5	2	6/0/0	6
	contingent	4	0	0	0	4/3/3	3
quantifiers	valid	24	16	13	1	18/1/1	18
	contradictory	4	1	1	0	1/3/1	2
	contingent	31	0	0	0	27/16/16	16
adjectives	valid	10	3	10	4	10/0/0	10
	contradictory	7	0	7	5	0/5/0	7
	contingent	8	0	0	0	6/8/6	6
de dicto	valid	1	0	1	1	1/0/0	1
	contingent	1	0	0	0	1/1/1	1
(total)		117	35	50	25	92/43/33	88

Table 5:
Results for the weak translation. For the theorem provers, the figures are the numbers of items proven. For Mace4, the figures are the number of items such that a model was found (a) for $\mathscr{A} \cup \{T(p \wedge \gamma)\}$, (b) for $\mathscr{A} \cup \{T(p \wedge \neg\gamma)\}$, and (c) for both problems

Combining the information obtained from the model builder with the results from the provers, all components of our inference engine together provide enough information to determine whether an item is valid, contingent, or contradictory in 75.2% of the cases. The success rates of each theorem prover and the model builder are summarized in Table 5, organized by the semantic phenomena that structure the test suite (as depicted in detail in Appendix D).

4.6 *Discussion*

The lower success rates for the strong axiomatization indicate that the additional axioms make automated inference harder when they are not relevant to proving. This effect has often been observed when automated deduction is applied to large axiom sets (e.g., Hoder and Voronkov 2011). The additional strength does not provide an advantage in the context of our test suite, as only one item depends on it, and for that item a proof is not found even with the strong axiomatization. The predictions with respect to $\mathscr{A}^w$ and the Henkin-complete $\mathscr{A}^s$ agree on all other items. With this general result in mind, we will focus our discussion on the results obtained with the weak axiomatization.

The difficulty of the test items for reasoning varied with the linguistic phenomena. The combined performance of the reasoning engines on items dealing with first-order tautologies, modality, knowledge and belief, and adjectives is satisfactory, with a success rate of 89.3%. It is unclear why Spass and E perform rather poorly on the section on adjectives, while Prover9 proves all items.

The section on generalized quantifiers was clearly much harder for the systems and reveals the limitations of the current approach. Among the validities and contradictions in that section, eight items (28.6%) remain undetermined. Two of the undetermined items are statements about monotonicity. While it is not clear why item (3.23) (monotonicity of 'at least three') is not proved, items (3.15) ('Most women dance. ⊢ Some women dance.') and (3.22) ('Most men dance and play air guitar. ⊢ Most men dance.') express properties of 'most' that are probably too hard to prove automatically on the basis of (1). It is not clear why the provers did not succeed on the items derived from FraCaS.

Two items, (3.2) and (3.14), are wrongly classified as contingent because the proof requires the meaning postulate (24c) for 'few',[20] which is not included in the input, as 'few' does not occur in the items in question. It is noticeable that among the contingent items, models

[20] Both of these items require the inference that MANY(P, Q) cannot hold when $P \cap Q = \emptyset$. In the context of our meaning postulates, this follows from the mutual exclusiveness of FEW and MANY (see (24b)) and the monotonicity of MANY (see (24a-ii)) when also considering that FEW(P, Q) holds whenever $P \cap Q = \emptyset$ (see (24c)).

verifying the conjecture are found more often than models falsifying it, resulting in only 56.6% of them being determined. The reason might be that models falsifying these inferences are often necessarily larger than the smallest models verifying it. For instance, a model verifying the implication 'Few blond men dance' ⇒ 'Few men dance' need only contain a single element of type e, but to show that 'Few blond men dance' does not generally entail 'Few men dance' (item (3.36)), one needs at least two objects and also a function $f \in D_{ee}$.[21]

With respect to our axiomatization, the quantifiers and determiners fall into three classes: those for which we have given a direct definition (*most, at most two, at least three*), those which are indirectly characterized in terms of their properties (*few, many, several*), and the definite article, which is directly translated as the ι operator. As the indirect characterizations involve direct statements about monotonicity and conservativity, which are targeted by most test items, it is not surprising that the inference engines perform better on *few, many*, and *several* than on *most*. The first-order-definable quantifiers *at most two* and *at least three* show a success rate comparable to the other quantifiers. This difference is not surprising, either: the definition of *most* is more complex than the definitions of the numerical quantifiers, and the test items on 'most' require that the provers make inferences that are equivalent to proving statements such as Proposition 7 (upward monotonicity in the second argument). Our observations on the success rates of the provers simply emphasize the point that the way in which meaning postulates are stated may have a significant influence on the feasibility of automated inference. Nonetheless, the success of the model builder Mace4 on item (3.29) ('Most men dance. ⋉ Most men dance and play air guitar.') demonstrates that the direct definition of *most* can in principle be useful for automated reasoning.

5 RELATED WORK

Higher-order reasoning with natural language is not a lively research area, but there are a number of related fields. In this section we discuss an alternative recent system for reasoning with quantifiers, and earlier work with higher-order provers and higher-order model building.

[21] Since $[\![man(w)]\!] \supseteq [\![blond(w, man)]\!]$, $[\![man(w)]\!]$ cannot be equal to $[\![blond(w, man)]\!]$ when the implication is false.

The Natlog system (MacCartney 2009) is in some respects closest to some of the targets of our reasoning architecture, and it was also evaluated on the basis of the FraCaS test suite. By aligning premises with conjectures at the word level, computing entailment relations between them, and deriving the entailment relation between the sentences by projecting the individual entailment relations using a syntactic dependency analysis, Natlog is able to reason with quantifiers and negation. In contrast to our grammar fragment, Natlog has wide coverage, but it cannot handle problems with more than one premise (MacCartney 2009, p. 142), which, as we saw, are unproblematic for approaches based on theorem proving such as ours. On the single-premise problems, which constitute 53% of the test suite, MacCartney reports an accuracy of 70.5%, with 89.3% precision and 65.7% recall on the binary task of recognizing valid inferences. On the section on adjectives, our experiments show an accuracy of 66.7%, with 86.6% precision and 75% recall (relative to the original FraCaS annotation). These results are comparable to MacCartney's figures for the same section: 71.4% accuracy, 83.3% precision, and 80% recall. However, both our system and Natlog only covered 15 items from this section; they only intersect on 11 items. The situation is different in the section on generalized quantifiers. The decision rate of our system is 57%, whereas Natlog, by virtue of its architecture, makes some decision on every sentence. With undetermined items taken as 'wrongly classified', our system achieves 28.6% accuracy, 66.7% precision, and 0% recall, since only the model builder had some success on the FraCaS data on generalized quantifiers. These figures are far lower than those of the Natlog system, which are 95.2% accuracy, 100% precision, and 97.7% recall, respectively.

Unfortunately, a quantitative comparison between MacCartney's and our results is not very meaningful overall with the data we have so far, considering that (1) our system as well as Natlog only tested portions of the FraCaS test suite, (2) the intersection between the tested items was even smaller, and (3) the development of Natlog was guided by the FraCaS data whereas the predictions of our model partly deviate from the FraCaS annotation. Although comparing the raw numbers produced by Natlog with our system's performance clearly indicates that there is much room for improvement in the generalized quantifiers section, our results also suggest that in principle natural language problems of the type encoded in the FraCaS test suite can be solved by theorem proving, provided that a system is given access to appropriate meaning postulates for (classes of) lexical items.

A very exciting competitor to reasoning with a translation to first-order logic arises from automated reasoning tools that work directly on higher-order logic. Ramsay (1995) presents a special automatic proof system for an intensional logic, based on the property theory of Turner (1987). Kohlhase and Konrad (1998) apply the higher-order theorem prover HOT to corrections in natural language, using the higher-order unification analysis proposed by Dalrymple *et al.* (1991).[22] The difference in the application domain and the considerable advances in automated theorem proving in the last 15 years makes this older work hard to compare to our present study. In order to see what higher-order reasoning can achieve and how it compares to translations under Henkin semantics, it would be interesting to observe the performance of more recent higher-order theorem provers such as LEO II (Benzmüller *et al.* 2007) or Satallax (Brown 2013) with Ty2 representations that result from parsing natural language. We leave such comparison for future work.

Another interesting perspective on higher-order reasoning is provided by investigating the potential of model builders for natural language. Konrad (2004) presents the higher-order model builder *Kimba* and puts it to the test with linguistic data. In particular he uses it to determine the referent of definites within a discourse and to find the valid readings of sentences involving reciprocals. Konrad develops a model builder for a fragment of higher-order logic whose design is guided by typical properties of representations for natural language. In our reasoning architecture, first-order model generation comes after the weak version of the translation from Ty2 representations, and our approach to model-building targets full Ty2, which of course comprises a large class of expressions which are irrelevant for natural language semantics. A further notable difference concerns our treatment of generalized quantifiers such as ‘most’, which turns out to be complementary to what Konrad did. Recall that we define MOST by a meaning postulate within the representation language. Konrad de-

[22] They use HOT to prove that, for instance, ‘No, PETER likes Mary’ is a valid response to ‘Jon likes Mary’, while ‘No, PETER likes Sarah’ is not.

fines it by means of MORE, whose interpretation is directly fixed by a model-theoretic constraint. It seems plausible that defining functions such as MOST model-theoretically rather than by meaning postulates can make model generation vastly more efficient. In particular, numerical quantifiers like ‘two’ and ‘three’ have complex definitions in $\mathscr{L}_{\text{Ty2}}$, which soon become completely intractable as their size grows with the number that they encode. Such quantifiers are far more naturally defined model-theoretically. While model-theoretic definitions have their limitations in the context of proof systems (as we argued in Section 4.2), they fit very naturally into systems for finite model generation: a reasonable model-theoretic definition will in general be decidable on finite structures. Conversely, it would be interesting to explore whether extending our fragment could lead to a successful application of this type of architecture to Konrad’s data. Of particular interest would be a treatment of plurals, reciprocals, and definites. For the latter we assumed a simplistic analysis with a choice operator, and plural did not receive any treatment at all, although it is clearly highly relevant for a more realistic and comprehensive coverage of naturally occurring data.

6 CONCLUSION

We defined and discussed translations under Henkin semantics from Ty2 to first-order logic for automated reasoning with natural language, and investigated the performance of a reasoning architecture with several first-order theorem provers and a model generator on a test suite targeting typical reasoning tasks of theoretical semantics. Unlike previous work on automated reasoning with natural language, we took as input formulae in higher-order logic as proposed by formal semanticists. The architecture was evaluated on a set of 117 natural language inference problems, partly derived from the classical FraCaS test suite originally compiled for such purposes. The inference tasks were expressed in a small fragment of English; they focused on modality, propositional attitudes, generalized quantifiers, and adjectives, and relied on a set of associated meaning postulates commonly assumed in semantics.

The results are promising: Despite the general undecidability of first-order logic, 75.2% of the test items could be determined, a great majority in less than a second. The success rate of the combined inference engines suggests that theorem proving with higher-order representations for natural-language expressions can indeed be reduced to first-order proving by adopting Henkin semantics. At the same time, the system's poor performance on generalized quantifiers confirms expectations that the syntactic form of meaning postulates plays a significant role in the efficiency and ultimate success (or failure) of automated inference. Fine-tuning of meaning postulates and finding a good balance in exploiting the complementary strengths of theorem provers and model builders will be necessary to improve performance.

Our experiments enlisted model builders only in combination with a weak translation, which makes model generation unsound relative to stronger versions of Henkin semantics. Unsoundness did not affect any items in our particular test suite, but we need a better understanding of which classes of terms occurring in logical translations of natural language may cause trouble. In addition, the models created by the model builder suffer from being virtually incomprehensible to human readers due to their compact encoding of functions and types. Readability and usefulness of the models could be greatly enhanced by disentangling these structures automatically for human exploration.

Determining the precise advantages and disadvantages of translations of different strength remains a general desideratum. First-order generation for stronger translations is not generally impossible, but it must be prepared to cope with the fact that general models of higher-order logic are always infinite because the set of types is infinite. Advanced techniques for generating and representing infinite models, such as the ones introduced by Caferra *et al.* (2004), might offer viable solutions. At the other end of the scale, it is also interesting to explore which weakenings of the translation are best suited to natural language applications, and to exploit the advantages of smaller structures. This strategy has to take into account the dangers of potentially unsound translations. Our results suggest that the strength of our weak axiomatization is a promising choice as long as the meaning postulates are chosen carefully.

The setting in which we tested the feasibility of first-order translations of Ty2 in automated reasoning was restricted to a toy grammar and to test cases that belong to the theoretical toolbox of formal semanticists. Opening up its application to broad coverage and to accounting for effects of discourse pragmatics and world knowledge would of course expose the usual weaknesses of deductive reasoning when confronted with potentially incomplete knowledge on the one hand and an overwhelming amount of relevant facts on the other hand (see the discussion in Ovchinnikova 2012, pp. 73–92). However, despite the considerable challenges ahead, Bos (2006) and Bos and Markert (2006) report that automated inference on first-order representations of natural language can succeed in real-world applications. If this is correct, future work should investigate to what extent our translations can successfully widen the empirical scope of Bos' work to encompass semantic effects of intensionality and generalized quantifiers, replacing hand-encoded first-order approximations with well-studied higher-order analyses. The DRT-based wide-coverage Boxer system (Bos 2008) seems a promising starting point to extending the linguistic coverage.

ACKNOWLEDGMENTS

We are indebted to the three reviewers for their valuable comments, and to Johannes Dellert for extensive discussion and comments that had a guiding influence on this work. Janina Radó mastered the challenge of proofreading the paper and suggested numerous improvements.

APPENDICES

A AXIOMS

All axioms with type parameters are shown in a form with type constants as described for the weak axiomatization. The versions with quantification over types introduced in Section 3.2 (for the strong axiomatization) are derived straightforwardly.

First group:

Axioms for Typing and the Axioms of Extensionality

(31) Typing

a. $\forall x_0 : \mathit{isTrue}(x_0) \rightarrow \mathit{hasType}(x_0, t)$

b. $\forall x_0 \forall x_1 \forall x_2 : [\exists x_3 \mathit{hasType}(x_0, g(x_3, x_2)) \wedge \mathit{hasType}(x_1, x_3)]$ $\rightarrow \mathit{hasType}(f(x_0, x_1), x_2))$

c. $\exists x : \mathit{hasType}(x, e) \wedge \exists y : \mathit{hasType}(y, s)$

(32) Axioms of Extensionality

a. $\forall x_0 \forall x_1 \forall x_2 \forall x_3 : [\mathit{hasType}(x_0, g(x_2, x_3)) \wedge$
$\mathit{hasType}(x_1, g(x_2, x_3))]$
$\rightarrow [\forall x_4 \mathit{hasType}(x_4, x_2) \rightarrow f(x_0, x_4) = f(x_1, x_4)] \rightarrow x_0 = x_1$

b. $\forall x_0 \forall x_1 : [[\mathit{hasType}(x_0, t) \wedge \mathit{hasType}(x_1, t)]$
$\wedge [\mathit{isTrue}(x_0) \longleftrightarrow \mathit{isTrue}(x_1)]] \rightarrow x_0 = x_1$

Second group:

Defining Axioms for Ty2 Constants instantiated for all types ρ, σ, τ:

(33) For every constant c^n_τ:
$\mathit{hasType}(T_{term}(c^n_\tau), T_{ty}(\tau))$

(34) For every variable x^n_σ:
$\mathit{hasType}(x^n, T_{ty}(\sigma))$

(35) Combinators

a. $\forall x_0 : \mathit{hasType}(x_0, T_{ty}(\sigma)) \rightarrow f(\mathbf{I}(\sigma), x_0) = x_0$

b. $\forall x_0 \forall x_1 : [\mathit{hasType}(x_0, T_{ty}(\sigma)) \wedge \mathit{hasType}(x_1, T_{ty}(\tau))]$
$\rightarrow f(f(\mathbf{K}(T_{ty}(\sigma), T_{ty}(\tau)), x_0), x_1) = x_0$

c. $\forall x_0 \forall x_1 \forall x_2 : [\mathit{hasType}(x_0, g(T_{ty}(\tau), g(T_{ty}(\sigma), T_{ty}(\rho))))$
$\wedge \mathit{hasType}(x_1, g(T_{ty}(\tau), T_{ty}(\sigma))) \wedge \mathit{hasType}(x_2, T_{ty}(\tau))]$
$\rightarrow f(f(f(\mathbf{S}(T_{ty}(\tau), T_{ty}(\sigma), T_{ty}(\rho)), x_0), x_1), x_2)$
$= f(f(x_0, x_2), f(x_1, x_2))$

(36) Equality

a. $\forall x_1 \forall x_2 : \mathit{isTrue}(f(f(\dot{\equiv}(T_{ty}(\sigma)), x_1), x_2))$
$\longleftrightarrow (\mathit{hasType}(x_1, T_{ty}(\sigma)) \wedge x_1 = x_2)$

(37) Choice

a. $\forall x_1 : \mathit{hasType}(x_1, g(T_{ty}(\sigma), t)) \rightarrow [\exists x_2 \mathit{isTrue}(f(x_1, x_2)))$
$\rightarrow \mathit{isTrue}(f(x_1, f(\iota(T_{ty}(\sigma)), x_1)))]$

(38) Existential Quantifier

a. $\forall x_1 \mathit{hasType}(x_1, T_{ty}(\sigma)) \rightarrow [\forall x_2 : \mathit{isTrue}(f(\dot{\exists}(T_{ty}(\sigma)), x_2))$
$\longleftrightarrow \exists x_3(\mathit{hasType}(x_3, T_{ty}(\sigma)) \wedge \mathit{isTrue}(f(x_2, x_3))]$

(39) Universal Quantifier

a. $\forall x_1 \mathit{hasType}(x_1, T_{ty}(\sigma)) \rightarrow [\forall x_2 : \mathit{isTrue}(f(\dot{\forall}(T_{ty}(\sigma)), x_2))$
$\longleftrightarrow \forall x_3(\mathit{hasType}(x_3, T_{ty}(\sigma)) \rightarrow \mathit{isTrue}(f(x_2, x_3)))]$

(40) Propositional Connectives

a. $\forall x_0 : \mathit{hasType}(x_0, t) \rightarrow$
$(\mathit{isTrue}(f(\dot{\neg}, x_0)) \longleftrightarrow not(\mathit{isTrue}(x_0))))$

b. $\forall x_0 \forall x_1 : [\mathit{hasType}(x_0, t) \wedge \mathit{hasType}(x_1, t)]$
$\rightarrow [\mathit{isTrue}(f(f(\dot{\wedge}, x_0), x_1)) \longleftrightarrow (\mathit{isTrue}(x_0) \wedge \mathit{isTrue}(x_1))]$

c. $\forall x_0 \forall x_1 : [\mathit{hasType}(x_0, t) \wedge \mathit{hasType}(x_1, t)]$
$\rightarrow [\mathit{isTrue}(f(f(\dot{\rightarrow}, x_0), x_1)) \longleftrightarrow (\mathit{isTrue}(x_0) \rightarrow \mathit{isTrue}(x_1))]$

d. $\forall x_0 \forall x_1 : [\mathit{hasType}(x_0, t) \wedge \mathit{hasType}(x_1, t)]$
$\rightarrow (\mathit{isTrue}(f(f(\dot{\vee}, x_0), x_1)) \longleftrightarrow (\mathit{isTrue}(x_0) \vee \mathit{isTrue}(x_1)))$

(41) Function symbols for lambda abstracts: see (10)

B PROOFS

B.1 *Soundness and completeness*

In this section, we prove Theorem 8. We adapt familiar proofs of Henkin's completeness theorem based on first-order translations and the first-order completeness theorem (van Benthem and Doets 1983, pp. 276–283, Leivant 1994, sections 5.4–5.5). First we embed $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$ in a multi-sorted first-order language, $\mathscr{F}^{\mathscr{C}}$:

Definition 10. *The sorts of $\mathscr{F}^{\mathscr{C}}$ are the types of $\mathscr{L}_{\mathrm{Ty2}}$. The terms of $\mathscr{F}^{\mathscr{C}}$ are the terms of $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$ plus the variables of $\mathscr{L}_{\mathrm{Ty2}}$, and the sort of α_σ as a term of $\mathscr{F}^{\mathscr{C}}$ is σ. The variables of $\mathscr{F}^{\mathscr{C}}$ are the variables of $\mathscr{L}_{\mathrm{Ty2}}$. A lambda abstract $\lambda x.\alpha$ with n free variables $x^1, ..., x^n$ is understood as an n-place function symbol applied to $x^1, ..., x^n$. The language $\mathscr{F}^{\mathscr{C}}$ of $\mathscr{C}$-formulae is the smallest set such that*

- *$\mathit{isTrue}(\alpha_t) \in \mathscr{F}^{\mathscr{C}}$ for every term $\alpha_t \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$ of sort t. We will write α_t for $\mathit{isTrue}(\alpha_t)$.*
- *If α, $\beta \in \mathscr{F}^{\mathscr{C}}$, then $(\alpha \hat{\circ} \beta) \in \mathscr{F}^{\mathscr{C}}$ for all propositional connectives $\circ$, similarly for negation*
- *If $\alpha \in \mathscr{F}^{\mathscr{C}}$ and x_σ is a variable, then $(\hat{\forall}^\sigma x_\sigma\ \alpha) \in \mathscr{F}^{\mathscr{C}}$ and $(\hat{\exists}^\sigma x_\sigma\ \alpha) \in \mathscr{F}^{\mathscr{C}}$*
- *If $\alpha_\sigma, \beta_\sigma \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$, then $(\alpha_\sigma \hat{\equiv}^\sigma \beta_\sigma) \in \mathscr{F}^{\mathscr{C}}$*

We interpret $\mathscr{F}^{\mathscr{C}}$ over structures in which the universes may be empty for some sorts and where therefore the interpretation of a term or formula *with free variables may be undefined. An $\mathcal{F}^{\mathcal{C}}$-structure $\mathfrak{M}$ has as its universe a family $\{D_\tau : \tau \in Types\}$ of mutually disjoint sets, where $D_s, D_e \neq \emptyset$, and for all other types τ, D_τ is non-empty (at least) if there is a term of type τ in $\mathcal{L}^{\mathcal{C}}_{\mathrm{Ty2}}$, interpreting constants of sort σ by elements of D_σ. $\mathcal{F}^{\mathcal{C}}$-structures interpret $\equiv^\sigma$ as equality between objects of sort σ, and provide an interpretation for the predicate symbol $isTrue(\cdot_t)$, the function symbol $(\cdot_{\langle\tau\sigma\rangle}, \cdot_\tau)_\sigma$ for each pair of sorts τ, $\sigma \in Types$, and all function symbols representing lambda abstracts $\lambda x.\alpha \in \mathcal{C}$. An $\mathfrak{M}$-assignment v is a partial function from the variables of $\mathcal{L}_{\mathrm{Ty2}}$ such that $v(x_\sigma) \in D_\sigma$ for every x_σ in the domain of v. Evaluation is defined as follows:*

- $[\![\alpha]\!]^v_{\mathfrak{M}}$ *is undefined if v is undefined for some variable occurring free in α.*
  Otherwise, we have:
- $[\![x]\!]^v_{\mathfrak{M}} = v(x)$
- $[\![(\alpha_{\langle\tau\sigma\rangle}\beta_\tau)]\!]^v_{\mathfrak{M}} = [\![(\cdot_{\langle\tau\sigma\rangle}, \cdot_\tau)_\sigma]\!]_{\mathfrak{M}}([\![\alpha]\!]^v_{\mathfrak{M}}, [\![\beta]\!]^v_{\mathfrak{M}})$
- $[\![(\hat{\forall}^\sigma x_\sigma\ \alpha)]\!]^v_{\mathfrak{M}} = 1$ *iff* $[\![\alpha]\!]^{v'}_{\mathfrak{M}} = 1$ *for every $\mathfrak{M}$-assignment v' that has x_σ in its domain and that agrees with v on $Domain(v) \backslash \{x_\sigma\}$, and 0 otherwise*
- $[\![(\hat{\exists}^\sigma x_\sigma\ \alpha)]\!]^v_{\mathfrak{M}} = 1$ *iff* $[\![\alpha]\!]^{v'}_{\mathfrak{M}} = 1$ *for some $\mathfrak{M}$-assignment v' that has x_σ in its domain and that agrees with v on $Domain(v) \backslash \{x_\sigma\}$, and 0 otherwise*

with straightforward clauses for atoms and propositional connectives. A structure $\mathfrak{M}$ verifies a formula $\phi \in \mathcal{F}^{\mathcal{C}}$ iff $[\![\phi]\!]^v_{\mathfrak{M}} = 1$ for all $\mathfrak{M}$-assignments v that have all free variables of ϕ in their domain.

There is a canonical translation from $\mathcal{F}^{\mathcal{C}}$ to $\mathcal{L}_{\mathrm{Ty2}}$, but $\mathcal{F}^{\mathcal{C}}$ is in general more expressive than $\mathcal{L}^{\mathcal{C}}_{\mathrm{Ty2}}$. The formula translation T_f can be extended canonically to $\mathcal{F}^{\mathcal{C}}$. Axioms from $\mathcal{A}^{\mathcal{C}}$ that do not contain quantification over types can be understood as formulae of $\mathcal{F}^{\mathcal{C}}$. For instance, the first-order formula representing the Axiom of Extensionality for objects of type t, (32b), can be identified with the $\mathcal{C}$-formula

(42) $\hat{\forall}^t x_t \hat{\forall}^t y_t : (x_t \hat{\leftrightarrow} y_t) \hat{\to} (x_t \hat{\equiv}^t y_t)$.

Furthermore, every axiom that does not contain positive occurrences of variables representing types can be associated with a (possibly infinite) family of $\mathcal{C}$-formulae. For instance the first-order formula of the Axiom of Extensionality for higher types, (32a), can be associated with the set

(43) $\{\hat{\forall}^{\sigma} x_{\sigma}(((\phi_{\sigma\tau} x_{\sigma}) \hat{\equiv}^{\tau} (\psi_{\sigma\tau} x_{\sigma}))) \hat{\rightarrow} ((\phi \hat{\equiv}^{\sigma\tau} \psi) : \sigma, \tau \in \mathit{Types}\}.$

The only axioms from $\mathscr{A}^{\mathscr{C}}$ containing positive occurrences of variables representing types are the typing axioms (31, 33, 34). As these axioms only fix the types of the constant symbols, they are already implicit in the syntax of $\mathscr{F}^{\mathscr{C}}$. Replacing the others by $\mathscr{C}$-formulae in this manner, we obtain a set $\mathscr{B}^{\mathscr{C}} \subset \mathscr{F}^{\mathscr{C}}$ of $\mathscr{C}$-formulae representing all axioms in $\mathscr{A}^{\mathscr{C}}$ apart from the typing axioms.

The notion of an $\mathscr{L}^{\mathscr{C}}_{Ty2}$-interpretation is straightforwardly extended to $\mathscr{F}^{\mathscr{C}}$, and $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-models can therefore be canonically viewed as $\mathscr{F}^{\mathscr{C}}$-structures. We obtain the following characterization:

Proposition 11. *The class of $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-models is (via this identification) equal to the class of $\mathscr{F}^{\mathscr{C}}$-structures that satisfy the formulae in $\mathscr{B}^{\mathscr{C}}$.*

Proof. Immediate from Definition 4. □

We now proceed to the proof of completeness (Lemma 12) and soundness (Lemma 13).

Lemma 12. *If $\phi \in \mathscr{F}^{\mathscr{C}}$ is true in all $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-models, then $\mathscr{A}^{\mathscr{C}} \vdash T(\phi)$.*

Proof. We show that if $T(\phi)$ is satisfiable in a model of $\mathscr{A}^{\mathscr{C}}$, then ϕ is true in some $\mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$-model. The claim then follows from the completeness theorem for first-order logic.

Let $\mathfrak{M}$ be a model of $\mathscr{A}^{\mathscr{C}}$ that verifies $T(\phi)$. For every $\alpha \in \mathfrak{M}$ and every type σ such that $[\![\mathit{hasType}]\!]_{\mathfrak{M}}(\alpha, [\![T_{ty}(\sigma)]\!]_{\mathfrak{M}}) = T$, take a fresh object $\overline{\alpha}_{\sigma}$. Set $\overline{\alpha}_{\sigma\tau}(\overline{\beta}_{\sigma})$ to be $\overline{[\![f]\!]_{\mathfrak{M}}(\alpha, \beta)}_{\tau}$, which exists by the second typing axiom. By the Axiom of Extensionality, there can be at most two objects of type t. Identify the one that verifies *isTrue*, if it exists, with T, and the other one, if it exists, with F. We thus obtain a frame $\{D_{\sigma} : \sigma \in \mathit{Types}\}$, from which we build an $\mathscr{F}^{\mathscr{C}}$-structure $\mathfrak{M}'$ by setting $[\![c]\!]_{\mathfrak{M}'}$ to $[\![c]\!]_{\mathfrak{M}}$ for every constant $c \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$. In the case that c is a polymorphic logical constant $c^{\tau_1,...,\tau_n}$, i.e., a quantifier or a combinator, we set $[\![c^{\tau_1,...,\tau_n}]\!]_{\mathfrak{M}'}$ to $[\![c]\!]_{\mathfrak{M}}([\![T(\tau_1)]\!]_{\mathfrak{M}}, ..., [\![T(\tau_n)]\!]_{\mathfrak{M}})$ for all types $\tau_1, ..., \tau_n$.

We then need to show $\mathfrak{M} \models T(\psi) \Rightarrow \mathfrak{M}' \models \psi$ for all $\psi \in \mathscr{F}^{\mathscr{C}}$. First, by induction, $[\![t]\!]^{v}_{\mathfrak{M}'} = [\![T_{term}(t)]\!]^{w}_{\mathfrak{M}}$ for every term $t \in \mathscr{L}^{\mathscr{C}}_{\mathrm{Ty2}}$ and all assignments $v \colon Var \cap \mathscr{L}^{\mathscr{C}}_{Ty2} \to \mathfrak{M}'$, $w \colon Var \to \mathfrak{M}$ such that $w \supset v$, where *Var* is the set of Ty2 variables. The claim follows by induction over formula structure. The structure $\mathfrak{M}$ verifies the first-order translation of every element of $\mathscr{B}^{\mathscr{C}}$. Therefore, by Proposition 11, $\mathfrak{M}'$ is a $\mathscr{L}^{\mathscr{C}}_{Ty2}$-model and, in particular, ϕ has a $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-model. □

We have shown that every model of $\mathscr{A}^{\mathscr{C}}$ encodes a $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-model, preserving the truth of $\mathscr{F}^{\mathscr{C}}$-formulae. This shows that a finite model $\mathfrak{M}$ of a first-order translation generated by a model builder can be viewed as encoding a (possibly infinite) $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-model such that the value of every $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-term in this model can be computed from $\mathfrak{M}$.

Lemma 13. *Let $\phi \in \mathscr{F}^{\mathscr{C}}$. If $\mathscr{A}^{\mathscr{C}} \vdash T(\phi)$, then ϕ holds in all $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-models.*

Proof. Assume $\neg\phi$ holds in some $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-model $\mathfrak{M}$. As above, we interpret $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-models as multi-sorted first-order structures. We extend them by adding every $\tau \in Types$ to the universe, giving them a separate sort, and defining predicates *isTrue* and $\equiv^{\sigma}$ straightforwardly. Then we can interpret the first-order language of the translation in these structures. Obtain a first-order structure $\mathfrak{M}'$ in this manner. We show that $\mathfrak{M} \models \psi \Rightarrow \mathfrak{M}' \models T(\psi)$ for all $\psi \in \mathscr{F}^{\mathscr{C}}$. Thus $\mathfrak{M}' \models T(\neg\phi)$. The typing axioms are evidently true in $\mathfrak{M}'$. The other axioms are true in $\mathfrak{M}'$ by Proposition 11. Thus $\mathscr{A}^{\mathscr{C}} \not\models T(\phi)$, and by soundness of first-order deduction, $\mathscr{A}^{\mathscr{C}} \nvdash T(\phi)$. □

This concludes the proof of Theorem 8.

B.2 *Model building*

In this section, we prove Theorem 9.

Proof. Set $\mathscr{C} := \{\alpha\}$. Let $\langle D, \mathscr{I} \rangle$ be a general $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-model such that D_e and D_s are finite, and α_t is true in $\langle D, \mathscr{I} \rangle$. We say that e, t, s have rank 1, and the rank of $g(\sigma, \tau)$ is one plus the maximum of the ranks of σ and τ. Let n be twice the maximum rank of all the types of sub-terms occurring in α. Obtain a finite $\mathscr{L}^{\mathscr{C}}_{\text{Ty2}}$-model $\mathfrak{N}$ by setting $D_{\tau} := \emptyset$ for all types τ of rank $> n$. As in the proof of Lemma 13, we can view $\mathfrak{N}$ as a first-order-structure that verifies $\mathscr{A}^{\mathscr{C}} \wedge T(\alpha)$. □

C MEANING POSTULATES

For every postulate, the set of triggering constant symbols is given. Where it occurs, α stands for the triggering constant symbol.

1. Intersective adjectives: *blond, Scandinavian, Irish, British, female, male*
   $\exists P^1_{\langle s\langle et\rangle\rangle}\forall w_s\forall P^2_{\langle s\langle et\rangle\rangle}\forall x_e(\alpha(w,P^2,x)\leftrightarrow(P^1(w,x)\wedge P^2(w,x)))$
2. Subsective, non-intersective adjectives: *genuine, skillful, successful, interesting, large, small, fat, tall, blue*
   $\forall P_{\langle s\langle et\rangle\rangle}\forall x_e\forall w_s(\alpha(w,P,x)\rightarrow P(w,x))$
3. Privative adjectives: *fake, former*
   $\forall P_{\langle s\langle et\rangle\rangle}\forall x_e\forall w_s(\alpha(w,P,x)\rightarrow\neg P(w,x))$
4. *alleged*
   $\forall P_{\langle s\langle et\rangle\rangle}\forall x_e\forall w^1_s(\textit{alleged}(w^1,P,x)\leftrightarrow\textit{allegedly}(w^1,(\lambda w^2P(w^2,x))))$
   Note that this axiom is slightly different from the one given in the text (see (21)), but the version here is sufficient for the relevant test items.
5. Mutual exclusiveness of *small, large*
   $\forall w_s\forall x_e\forall P_{\langle s\langle et\rangle\rangle}(\textit{small}(w,P,x)\rightarrow\neg\textit{large}(w,P,x))$
6. *necessarily*
   $\forall w^1_s\forall P_{\langle st\rangle}(\textit{necessarily}(w^1,P)\leftrightarrow\forall w^2_sP(w^2))$
7. *possibly*
   $\forall w^1_s\forall P_{\langle st\rangle}(\textit{possibly}(w^1,P)\leftrightarrow\exists w^2_sP(w^2))$
8. *two*
   $\forall P^1_{\langle et\rangle}\forall P^2_{\langle et\rangle}(two^e(P^1,P^2)\leftrightarrow\exists x^1_e\exists x^2_e(x^1\not\equiv x^2\wedge(P^1(x^1)\wedge P^1(x^2))))$
9. *at-most-two*
   $\forall P^1_{\langle et\rangle}\forall P^2_{\langle et\rangle}(\textit{at-most-two}^e(P^1,P^2)\leftrightarrow\exists x^1_e\exists x^2_e\forall x^3_e(x^3\not\equiv x^1\rightarrow(x^3\not\equiv x^2\rightarrow\neg(P^1(x^3)\wedge P^2(x^3)))))$
10. *at-least-three*
    $\forall P^1_{\langle et\rangle}\forall P^2_{\langle et\rangle}(\textit{at-least-three}^e(P^1,P^2)\leftrightarrow\exists x^1_e\exists x^2_e\exists x^3_e((((((((x^1\not\equiv x^2\wedge x^1\not\equiv x^3)\wedge x^2\not\equiv x^3)\wedge P^1(x^1))\wedge P^2(x^1))\wedge P^1(x^2))\wedge P^2(x^2))\wedge P^1(x^3))\wedge P^2(x^3)))$
11. *most*
    $\forall P^1_{\langle et\rangle}\forall P^2_{\langle et\rangle}(most^e(P^1,P^2)\leftrightarrow\forall f_{\langle ee\rangle}(\forall x^1_e((P^1(x^1)\wedge\neg P^2(x^1))\rightarrow(P^1(f(x^1))\wedge P^2(f(x^1))))\rightarrow\exists x^2_e((P^1(x^2)\wedge P^2(x^2))\wedge\forall x^3_e((P^1(x^3)\wedge\neg P^2(x^3))\rightarrow f(x^3)\not\equiv x^2))))$

12. *only*
    $\forall P^1_{\langle et\rangle}\forall P^2_{\langle et\rangle}(\mathit{only}^e(P^1,P^2)\leftrightarrow\forall x_e(P^2(x)\rightarrow P^1(x)))$
13. Conservativity of SEVERAL
    $\forall P^1_{\langle et\rangle}\forall P^2_{\langle et\rangle}(\alpha(P^1,P^2)\rightarrow\exists x_e(P^1(x)\wedge P^2(x)))$
14. Monotonicity: upwards on first argument: SEVERAL, MANY
    $\forall P^1_{\langle et\rangle}\forall P^2_{\langle et\rangle}\forall P^3_{\langle et\rangle}(\alpha(P^1,P^2)\rightarrow(\forall x_e(P^1(x)\rightarrow P^3(x))\rightarrow\alpha(P^3,P^2)))$
15. Monotonicity: upwards on second argument: SEVERAL, MANY
    $\forall P^1_{\langle et\rangle}\forall P^2_{\langle et\rangle}\forall P^3_{\langle et\rangle}(\alpha(P^1,P^2)\rightarrow(\forall x_e(P^2(x)\rightarrow P^3(x))\rightarrow\alpha(P^1,P^3)))$
16. Monotonicity: downwards on first argument: FEW
    $\forall P^1_{\langle et\rangle}\forall P^2_{\langle et\rangle}\forall P^3_{\langle et\rangle}(\alpha(P^1,P^2)\rightarrow(\forall x_e(P^3(x)\rightarrow P^1(x))\rightarrow\alpha(P^3,P^2)))$
17. Monotonicity: downwards on second argument: FEW
    $\forall P^1_{\langle et\rangle}\forall P^2_{\langle et\rangle}\forall P^3_{\langle et\rangle}(\alpha(P^1,P^2)\rightarrow(\forall x_e(P^3(x)\rightarrow P^2(x))\rightarrow\alpha(P^1,P^3)))$
18. Non-empty extension: FEW
    $\forall P^1_{\langle et\rangle}\forall P^2_{\langle et\rangle}(\forall x_e\neg(P^1(x)\wedge P^2(x))\rightarrow \mathit{few}^e(P^1,P^2))$
19. Mutual exclusiveness of MANY and FEW
    $\forall P^1_{\langle et\rangle}\forall P^2_{\langle et\rangle}\neg(\mathit{many}^e(P^1,P^2)\wedge few^e(P^1,P^2))$
20. Deductivity: *think, know*
    $\forall x_e\forall P^1_{\langle st\rangle}\forall P^2_{\langle st\rangle}\forall w^1_s((\alpha(w^1,P^1,x)\wedge\forall w^2_s(P^1(w^2)\rightarrow P^2(w^2)))$
    $\rightarrow\alpha(w^1,P^2,x))$
21. Veridicality: *know*
    $\forall x_e\forall P_{\langle st\rangle}\forall w_s(\mathit{know}(w,P,x)\rightarrow P(w))$

D TEST SUITE

Each test item consists of premises and a conjecture, which are separated by a symbol which is $\vdash$ for valid items, $\vdash$ NON for contradictory ones, and $\ltimes$ for contingent items. Our notational conventions and terminology are explained in Section 4.3.

First-order inferences

(0.0) $\vdash$ NON Mia dances and does not dance.

(0.1) Mia dances and does not dance. $\vdash$ Mia dances.

(0.2) $\vdash$ Every man dances or does not dance.

(0.3) Mia is a woman. Mia dances. $\vdash$ A woman dances.

(0.4) Mia is a robber. $\ltimes$ Mia is a man.

(0.5) Mia is a woman. Every woman dances. $\vdash$ Mia dances.

Modality

(1.0) Mia dances. ⊢ Mia possibly dances.

(1.1) Mia dances. ⋉ Mia necessarily dances.

(1.2) Mia is a robber. ⋉ Mia allegedly is a robber.

(1.3) Mia necessarily dances. ⊢ Mia dances.

(1.4) Mia possibly dances. ⋉ Mia dances.

(1.5) Mia allegedly is a robber. ⋉ Mia is a robber.

(1.6) Mia necessarily dances. ⊢ Mia possibly dances.

(1.7) Mia possibly dances. ⋉ Mia necessarily dances.

(1.8) Mia does not possibly dance. ⊢ Mia necessarily does not dance.

(1.9) Mia does not dance. ⊢ Mia does not necessarily dance.

(1.10) Mia does not possibly dance. ⊢ Mia does not dance.

(1.11) ⊢ Mia dances or does not necessarily dance.

(1.12) Mia is an alleged robber. ⊢ Mia allegedly is a robber.

(1.13) Mia necessarily is a robber. ⋉ Mia allegedly is a robber.

(1.14) Mia allegedly is a robber. ⋉ Mia possibly is a robber.

Propositional attitudes

(2.0) Mia thinks that John necessarily dances. ⊢ Mia thinks that John dances.

(2.1) John thinks that Mia knows that Vincent dances. ⊢ John thinks that Vincent dances.

(2.2) John thinks that Mia eats several burgers. ⊢ John thinks that Mia eats a burger.

(2.3) John thinks that Mia is a blond woman. ⊢ John thinks that Mia is a woman.

(2.4) John thinks that Mia is an alleged robber. ⋉ John thinks that Mia is a robber.

(2.5) John knows that Mia dances. ⊢ Mia dances.

(2.6) John thinks that Mia dances. ⋉ Mia dances.

(2.7) Mia necessarily knows that Mia dances. ⊢ Mia necessarily dances.

(2.8) Mia knows that John dances. John is the chairman. ⋉ Mia knows that the chairman dances.

(2.9) Mia knows that John saw a unicorn. ⊢ Some unicorn is a unicorn.

(2.10) Mia thinks that John saw a unicorn. ⋉ Some unicorn is a unicorn.

Generalized quantifiers

(3.0) Few women dance. ⊢ NON Many women dance.

(3.1) No women dance. ⊢ Few women dance.

(3.2) No women dance. ⊢ NON Many women dance.

(3.3) Few women dance. ⋉ No women dance.

(3.4) Many women dance. ⋉ All women dance.

(3.5) All women dance. ⋉ Many women dance.

(3.6) Mia eats every burger. Mia eats a burger. ⊢ Mia eats most burgers.

(3.7) Every man dances. A man dances. ⊢ Most men dance.

(3.8) Mia eats a burger. ⋉ Mia eats most burgers.

(3.9) Mia eats most burgers. ⋉ Mia eats all burgers.

(3.10) Only men dance. No woman is a man. ⊢ No woman dances.

(3.11) John is a man. Every man dances. ⊢ The man dances.

(3.12) At least three women dance. ⊢ NON At most two women dance.

(3.13) The man dances. John is a man. ⊢ A man dances.

(3.14) Many women dance. ⊢ Some women dance.

(3.15) Few women dance. ⋉ Some women dance.

(3.16) Several women dance. ⊢ Some women dance.

(3.17) Most women dance. ⊢ Some women dance.

(3.18) At least three women dance. ⊢ Some women dance.

(3.19) At most two women dance. ⋉ Some women dance.

(3.20) The man dances and plays air guitar. ⊢ The man dances.

(3.21) Many men dance and play air guitar. ⊢ Many men dance.

(3.22) Few men dance and play air guitar. ⋉ Few men dance.

(3.23) Several men dance and play air guitar. ⊢ Several men dance.

(3.24) Most men dance and play air guitar. ⊢ Most men dance.

(3.25) At least three men dance and play air guitar. ⊢ At least three men dance.

(3.26) At most two men dance and play air guitar. ⋉ At most two men dance.

(3.27) The man dances. ⋉ The man dances and plays air guitar.

(3.28) Many men dance. ⋉ Many men dance and play air guitar.

(3.29) Few men dance. ⊢ Few men dance and play air guitar.

(3.30) Several men dance. ⋉ Several men dance and play air guitar.

(3.31) Most men dance. ⋉ Most men dance and play air guitar.

(3.32) At least three men dance. ⋉ At least three men dance and play air guitar.

(3.33) At most two men dance. ⊢ At most two men dance and play air guitar.

(3.34) The blond man dances. ⋉ The man dances.

(3.35) Many blond men dance. ⊢ Many men dance.

(3.36) Few blond men dance. ⋉ Few men dance.

(3.37) Several blond men dance. ⊢ Several men dance.

(3.38) Most blond men dance. ⋉ Most men dance.

(3.39) At least three blond men dance. ⊢ At least three men dance.

(3.40) At most two blond men dance. ⋉ At most two men dance.

(3.41) The man dances. ⋉ The blond man dances.

(3.42) Many men dance. ⋉ Many blond men dance.

(3.43) Few men dance. ⊢ Few blond men dance.

(3.44) Several men dance. ⋉ Several blond men dance.

(3.45) Most men dance. ⋉ Most blond men dance.

(3.46) At least three men dance. ⋉ At least three blond men dance.

(3.47) At most three men dance. ⊢ At most three blond men dance.

Generalized quantifiers (from FraCaS)

(F22) No delegate finished the report on time. ⋉ No delegate finished the report.

(F23) Some delegates finished the survey on time. ⋉ Some delegates finished the survey.

(F24) Many delegates obtained interesting results from the survey. ⊢ Many delegates obtained results from the survey.

(F38) No delegate finished the report. ⋉ Some delegate finished the report on time.

(F39) Some delegates finished the survey. ⋉ Some delegates finished the survey on time.

(F40) Many delegates obtained results from the survey. ⋉ Many delegates obtained interesting results from the survey.

(F54) No Scandinavian delegate finished the report on time. ⋉ Some delegate finished the report on time.

(F55) Some Irish delegates finished the survey on time. ⊢ Some delegates finished the survey on time.

(F56) Many British delegates obtained interesting results from the survey. ⋉ Many delegates obtained interesting results from the survey.

(F63) At least three female commissioners spend time at home. ⊢ At least three commissioners spend time at home.

(F70) No delegate finished the report on time. ⊢ NON Some Scandinavian delegate finished the report on time.

(F71) Some delegates finished the survey on time. ⋉ Some Irish delegates finished the survey on time.

(F72) Many delegates obtained interesting results from the survey. ⋉ Many British delegates obtained interesting results from the survey.

(F79) At least three commissioners spend time at home. ⋉ At least three male commissioners spend time at home.

Adjectives

(4.0) Mia is a blond woman. ⊢ Mia is a woman.

(4.1) Mia is blond. Mia is a woman. ⊢ Mia is a blond woman.

(4.2) Mia is a blond woman. Mia is a robber. ⊢ Mia is a blond robber.

(4.3) Mia has a genuine diamond. ⊢ Mia has a diamond.

(4.4) Mia is a skillful robber. Mia is a boxer. ⋉ Mia is a skillful boxer.

(4.5) Excalibur is a fake sword. ⊢ NON Excalibur is a sword.

(4.6) ⊢ No fake sword is a sword.

(4.7) Excalibur is a weapon. Excalibur is a fake sword. ⊢ NON Excalibur is a fake weapon.

(4.8) Mia is an alleged robber. ⋉ Mia is a robber.

(4.9) Mia is an alleged robber. Mia is a boxer. ⋉ Mia is an alleged boxer.

Adjectives (from FraCaS)

(F197) John has a genuine diamond. ⊢ John has a diamond.

(F198) John is a former university student. ⊢ NON John is a university student.

(F199) John is a successful former university student. ⊢ John is successful.

(F200) John is a former successful university student. ⋉ John is successful.

(F201) John is a former successful university student. ⋉ John is a university student.

(F204) Mickey is a small animal. ⊢ NON Mickey is a large animal.

(F205) Dumbo is a large animal. ⊢ NON Dumbo is a small animal.

(F206) Fido is not a small animal. ⋉ Fido is a large animal.

(F207) Fido is not a large animal. ⋉ Fido is a small animal.

(F210) All mice are small animals. Mickey is a large mouse. ⊢ NON Mickey is a large animal.

(F211) All elephants are large animals. Dumbo is a small elephant. ⊢ NON Dumbo is a small animal.

(F214) All legal authorities are law lecturers. All law lecturers are legal authorities. ⊢ All fat legal authorities are fat law lecturers.

(F215) All legal authorities are law lecturers. All law lecturers are legal authorities. ⋉ All competent legal authorities are competent law lecturers.

(F218) Kim is a clever person. ⊢ Kim is clever.

(F219) Kim is a clever politician. ⊢ Kim is clever.

De dicto

(5.0) Mia seeks a unicorn. ⋉ Some unicorn is a unicorn.

(5.1) Mia sees a unicorn. ⊢ Some unicorn is a unicorn.

REFERENCES

Jon BARWISE and Robin COOPER (1981), Generalized quantifiers and natural language, *Linguistics and Philosophy*, 4(2):159–219.

Michael BENNETT (1974), *Some Extensions of a Montague Fragment*, Ph.D. thesis, UCLA.

Christoph BENZMÜLLER, Larry PAULSON, Frank THEISS, and Arnaud FIETZKE (2007), Progress Report on Leo-II, an Automatic Theorem Prover for Higher-Order Logic, in Klaus SCHNEIDER and Jens BRANDT, editors, *Theorem Proving in Higher Order Logics 2007 – Emerging Trends*, pp. 33–48.

Patrick BLACKBURN and Johan BOS (2005), *Representation and Inference for Natural Language*, CSLI Publications, Stanford.

Johan BOS (2004), Computational semantics in discourse: underspecification, resolution, and inference, *Journal of Logic, Language and Information*, 13:139–157.

Johan BOS (2006), Three stories on automated reasoning for natural language understanding, in *Proceedings of ESCoR (IJCAR Workshop): Empirically Successful Computerized Reasoning*, pp. 81–91.

Johan BOS (2008), Wide-coverage semantic analysis with Boxer, in Johan BOS and Rodolfo DELMONTE, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics, pp. 277–286, College Publications.

Johan BOS and Katja MARKERT (2006), When logical inference helps determining textual entailment (and when it doesn't), in Bernardo MAGNINI and Ido DAGAN, editors, *The Second PASCAL Recognising Textual Entailment Challenge. Proceedings of the Challenges Workshop*, pp. 98–103, Venice, Italy.

Chad E. BROWN (2013), Reducing higher-order theorem proving to a sequence of SAT problems, *Journal of Automated Reasoning*, 51(1):57–77.

Ricardo CAFERRA, Alexander LEITSCH, and Nicolas PELTIER (2004), *Automated Model Building*, Kluwer.

Alonzo CHURCH (1940), A formulation of the simple theory of types, *The Journal of Symbolic Logic*, 5(2):56–68.

Robin COOPER, Dick CROUCH, Jan VAN EIJCK, Chris FOX, Josef VAN GENABITH, Jan JASPARS, Hans KAMP, David MILWARD, Manfred PINKAL, Massimo POESIO, Steve PULMAN, Ted BRISCOE, Holger MAIER, and Karsten KONRAD (1996), Using the Framework, Technical report, FraCaS Consortium, University of Edinburgh, Deliverable D16 — Final Draft.

Ido DAGAN, Bill DOLAN, Bernardo MAGNINI, and Dan ROTH (2009), Recognizing textual entailment: Rational, evaluation and approaches, *Natural Language Engineering*, 15(4):i–xvii.

Mary DALRYMPLE, Stuart M. SHIEBER, and Fernando C. N. PEREIRA (1991), Ellipsis and higher-order unification, *Linguistics and Philosophy*, 14(4):399–452.

David R. DOWTY, Robert E. WALL, and Stanley PETERS (1981), *Introduction to Montague Semantics*, Reidel, Dordrecht.

Joyce FRIEDMAN and David S. WARREN (1980), λ-normal forms in an intensional model for English, *Studia Logica*, 39:311–324.

Daniel GALLIN (1975), *Intensional and Higher-Order Modal Logic*, North-Holland, Amsterdam.

L.T.F. GAMUT (1991), *Logic, Language and Meaning, Volume II: Intensional Logic and Logical Grammar*, University of Chicago Press.

Jeroen GROENENDIJK and Martin STOKHOF (1982), Semantic analysis of wh-complements, *Linguistics and Philosophy*, 5(2):175–233.

Leon HENKIN (1950), Completeness in the theory of types, *The Journal of Symbolic Logic*, 15(2):81–91.

Klaus VON HEUSINGER (1997), Definite descriptions and choice functions, in Seiki AKAMA, editor, *Logic, Language and Computation*, volume 5 of *Applied Logic Series*, pp. 61–91, Springer.

J. Roger HINDLEY and Jonathan P. SELDIN (2008), *Lambda-Calculus and Combinators, an Introduction*, Cambridge University Press.

Jaako HINTIKKA (1962), *Knowledge and Belief. An Introduction to the Logic of the Two Notions*, Cornell University Press.

Kryštof HODER and Andrei VORONKOV (2011), Sine qua non for large theory reasoning, in Nikolaj BJØRNER and Viorica SOFRONIE-STOKKERMANS, editors, *Automated Deduction – CADE-23*, volume 6803 of *Lecture Notes in Computer Science*, pp. 299–314, Springer.

Joe HURD (2002), An LCF-style interface between HOL and first-order logic, in Andrei VORONKOV, editor, *Automated Deduction – CADE-18*, volume 2392 of *Lecture Notes in Computer Science*, pp. 134–138, Springer.

Bjørn JESPERSEN and Giuseppe PRIMIERO (2013), Alleged assassins: realist and constructivist semantics for modal modification, in Guram BEZHANISHVILI, Sebastian LÖBNER, Vincenzo MARRA, and Frank RICHTER, editors, *Logic, Language, and Computation. 9th International Tbilisi Symposium*, number 7758 in Lecture Notes in Computer Science, pp. 94–114, Springer.

Hans KAMP and Barbara PARTEE (1995), Prototype theory and compositionality, *Cognition*, 57(2):129–91.

Hans KAMP and Uwe REYLE (1993), *From Discourse to Logic*, Kluwer, Dordrecht.

Manfred KERBER (1992), *On the Representation of Mathematical Concepts and their Translation into First Order Logic*, Ph.D. thesis, Universität Kaiserslautern.

Michael KOHLHASE and Karsten KONRAD (1998), Higher-Order Automated Theorem Proving for Natural Language Semantics, unpublished manuscript. Web. Sept. 17, 2015.
`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.4186`.

Karsten KONRAD (2004), *Model Generation for Natural Language Interpretation and Analysis*, number 2953 in Lecture Notes in Artificial Intelligence, Springer.

Daniel LEIVANT (1994), Higher order logic, in *Handbook of Logic in Artificial Intelligence and Logic Programming (2)*, pp. 229–322.

David K. LEWIS (1968), Counterpart theory and quantifed modal logic, *Journal of Philosophy*, 65(5):113–126.

Bill MACCARTNEY (2009), *Natural Language Inference*, Ph.D. thesis, Stanford University.

William MCCUNE (2005–2010), Prover9 and Mace4, Web. Sept. 17, 2015.
`http://www.cs.unm.edu/~mccune/prover9/`.

Jia MENG and Lawrence C. PAULSON (2008), Translating higher-order clauses to first-order clauses, *Journal of Automated Reasoning*, 40(1):35–60.

Richard MONTAGUE (1970), English as a formal language, in Bruno VISENTI, editor, *Linguaggi nella Società e nella Tecnica*, pp. 189–224, Edizioni di Comunità, Milan.

Richard MONTAGUE (1973), The proper treatment of quantification in ordinary English, in K. J. J. HINTIKKA, J. M. E. MORAVCSIK, and P. SUPPES, editors, *Approaches to Natural Language*, pp. 221–242, Reidel, Dordrecht.

Ekaterina OVCHINNIKOVA (2012), *Integration of World Knowledge for Natural Language Understanding*, volume 3 of *Atlantis Thinking Machines*, Atlantis Press.

Barbara H. PARTEE (1995), Lexical semantics and compositionality, in Lila R. GLEITMAN and Mark LIBERMAN, editors, *An Invitation to Cognitive Science. Language*, volume 1, pp. 311–360, MIT Press.

Allan RAMSAY (1995), Theorem proving for intensional logic, *Journal of Automated Reasoning*, 14:237–255.

Nicholas RESCHER (2005), *Epistemic Logic: A Survey of the Logic of Knowledge*, University of Pittsburgh Press.

Stephan SCHULZ (2004), System Description: E 0.81, in David BASIN and Michaël RUSINOWITCH, editors, *Automated Reasoning. Second International Joint Conference, IJCAR 2004*, volume 3097 of *Lecture Notes in Computer Science*, pp. 223–228, Springer.

Ray TURNER (1987), A theory of properties, *Journal of Symbolic Logic*, 52(2):455–472.

Johan VAN BENTHEM and Kees DOETS (1983), Higher-order logic, in *Handbook of philosophical logic*, pp. 275–329, Springer.

Christoph WEIDENBACH (2001), Combining superposition, sorts and splitting, in Alan ROBINSON and Andrei VORONKOV, editors, *Handbook of Automated Reasoning*, volume II, chapter 27, pp. 1965–2013, Elsevier Science.

Dag WESTERSTÅHL (2011), Generalized quantifiers, in Edward N. ZALTA, editor, *The Stanford Encyclopedia of Philosophy*, summer 2011 edition.