Ambiguous propositions typed

Tim Fernando
Philosophy Department
University of Texas
Austin, TX 78712-1180, USA
fernando@ims.uni-stuttgart.de*

Abstract

Ambiguous propositions are analyzed in a type system where disambiguation is effected during assembly (i.e. by coercion). Ambiguity is introduced through a layer of types that are underspecified relative to a pre-existing collection of dependent types, construed as unambiguous propositions. A simple system of reasoning directly with such underspecification is described, and shown to be sound and complete for the full range of disambiguations. Beyond erasing types, the system supports constraints on disambiguations, including co-variation.

1 Introduction

A widely held view expressed in (Carbonell and Hayes, 1987) is that "if there were one word to describe why natural language processing is hard, it is ambiguity." For any given natural language utterance, a formal language such as predicate logic typically offers several non-equivalent (well-formed) formulas as possible translations. An obvious approach is to take the disjunction of all alternatives, assuming (for the sake of the argument) that the disjunction is a formula. Even if it were, however, various objections have been raised against this proposal (e.g. (Deemter, 1996)). For the purposes of the present paper, what is interesting about a word, phrase, sentence or discourse that is ambiguous in isolation is how it may get disambiguated when combined with other expressions (or, more generally, when placed in a wider context); the challenge for any theory of ambiguity is to throw light on that process of disambiguation.

*From June to mid-August 1999, I will be visiting IMS, Uni Stuttgart, Azenbergstr 12, 70174 Stuttgart, Germany. Where I might be after that is unclear.

More concretely, suppose $\bullet$ were a binary connective on propositions A and B such that $A \bullet B$ is a proposition ambiguous between A and B. Under the "propositions-as-types" paradigm (e.g. (Girard et al., 1989)) identifying proofs of a proposition with programs of the corresponding type (so that "$t{:}A$" can be read as t is a proof of proposition A, or equivalently, t is a program of type A), disambiguation may take the form of type coercion. An instructive example with Γ as the context

$$x{:}(A \to B) \bullet C, \quad y{:}D \bullet A$$

is

$$\Gamma \;\vdash\; ap(p_\bullet(x), q_\bullet(y)){:}B \qquad (1)$$

where ap is function application (corresponding to modus ponens), while $p_\bullet$ and $q_\bullet$ are the first and second $\bullet$-projections, so that

$$x{:}(A \to B) \bullet C \;\vdash\; p_\bullet(x){:}A \to B$$

and

$$y{:}D \bullet A \;\vdash\; q_\bullet(y){:}A\,.$$

Evidently, there is something conjunctive (never mind disjunctive) about $\bullet$; but beyond the question as to whether the unambiguous propositions constituting the possible readings of an ambiguous proposition form a conjunctive or disjunctive set (whatever that may precisely mean), there is also the matter of the interconnected choices from such sets, mediated by terms such as $p_\bullet(x)$ and $q_\bullet(y)$.

To ground these abstract considerations in natural language processing, a few words about how to think of the terms t and types A are useful. For predicate logic formulas A, the terms t might be intuitionistic natural deduction proofs, related by the Curry-Howard isomorphism to a suitable typed λ-calculus. A notable innovation made in *Intuitionistic Type Theory* (ITT, (Martin-Löf,

1984)) is to allow proofs to enter into judgments of well-formedness (propositionhood). This stands in sharp contrast to ordinary predicate logic (be it intuitionistic or classical), where well-formedness is a trivial matter taken for granted (rather than analyzed) by the Curry-Howard isomorphism. For a natural language, however, it is well-formedness that is addressed by building types A over sentences, nouns, etc (in categorial grammar; e.g. (Morrill, 1994)) or LFG f-structures (in the "glue" approach, (Dalrymple et al., 1993; Dalrymple et al., 1997)). Now, while ITT's rules for propositionhood hardly constitute an account of grammaticality in English, the combination (in ITT) of assertions of well-formedness (A type) and theoremhood ($t{:}A$) re-introduces matters of information content (over and above grammatical form), which have been applied in (Ranta, 1994) (among other places) to discourse semantics (in particular, anaphora). The present paper assumes the machinery of dependent functions and sums in ITT, without choosing between grammatical and semantic applications. In both cases, what ambiguity contributes to the pot is indeterminacy in typing, the intuition being that an expression is ambiguous to the extent that its typing is indeterminate.

That said, let us return to (1) and consider how to capture sequent inferences such as

$$\frac{\Gamma \vdash x{:}(A \to B) \bullet C \qquad \Gamma \vdash y{:}D \bullet A}{\Gamma \vdash ap(p_\bullet(x), q_\bullet(y)){:}B}$$

and more complicated cases from iterated applications of $\bullet$, nested among other type constructs. The idea developed below is to set aside the connective $\bullet$ (as well as notational clutter $p_\bullet$, $q_\bullet$), and to step up from assertions $t : A$ to (roughly) $t :: \mathcal{A}$, where $\mathcal{A}$ is a set of types A (roughly, $t : A : \mathcal{A}$). For instance, a direct transcription of the $\to$-introduction rule into $::$ is

$$\frac{\Gamma, x{::}\mathcal{A} \vdash t{::}\mathcal{B}}{\Gamma \vdash \lambda x.t{::}\mathcal{A} \to \mathcal{B}} \qquad (2)$$

where $\mathcal{A} \to \mathcal{B}$ abbreviates the set

$$\{A \to B \mid A \in \mathcal{A} \text{ and } B \in \mathcal{B}\}.$$

But what exactly could $t{::}\mathcal{A}$ mean? The disjunctive conception

$$t{::}\mathcal{A} \quad \text{iff} \quad t{:}A \text{ for some } A \in \mathcal{A} \qquad (3)$$

would have as a consequence the implication

$$t{::}\mathcal{A} \text{ and } \mathcal{A} \subseteq \mathcal{B} \quad \text{implies} \quad t{::}\mathcal{B}\,.$$

Now, if combinatorial explosion is a problem for ambiguity, then surely we ought to avoid feeding it with cases of spurious ambiguity. A complementary alternative is conjunction,

$$t{::}\mathcal{A} \quad \text{iff} \quad t{:}A \text{ for all } A \in \mathcal{A}\,, \qquad (4)$$

the object this time being to identify the $\subseteq$-largest such set $\mathcal{A}$, as (4) supports

$$t{::}\mathcal{A} \text{ and } \mathcal{B} \subseteq \mathcal{A} \quad \text{implies} \quad t{::}\mathcal{B}\,.$$

But while (4) and (2) will do for $\lambda x.y$ where y is a variable distinct from x, (4) suggests that (2) overgenerates for $\lambda x.x$. Spurious ambiguity may also arise to the left of $\vdash$ (not just to the right), if we are not careful to disambiguate the context. (1) illustrates the point; compare

$$\frac{\Gamma \vdash x{::}\{A \to B, C\} \qquad \Gamma \vdash y{::}\{A, D\}}{\Gamma \vdash ap(x,y){::}\{B\}} \qquad (5)$$

where the context Γ is left untouched, to

$$\frac{\Gamma \vdash x{::}\{A \to B, C\} \qquad \Gamma \vdash y{::}\{A, D\}}{x{::}\{A \to B\}, y{::}\{A\} \vdash ap(x,y){::}\{B\}} \qquad (6)$$

where the context gets trimmed. (5) and (2) yield

$$\vdash \lambda x.\lambda y.ap(x,y){::}\{A \to B, C\} \to (\{A, D\} \to \{B\})$$

whereas (6) and (2) yield

$$\vdash \lambda x.\lambda y.ap(x,y){::}\{A \to B\} \to (\{A\} \to \{B\})\,.$$

To weed out spurious ambiguity, we will

(i) attach variables onto sets $\mathcal{A}$ of types, to form *decorated expressions* α

and

(ii) collect constraints on α's in sets C, hung as subscripts, $\vdash_{\mathsf{C}}$, on $\vdash$.

(3) and (4) are then sharpened by a contextual characterization, semantically interpreting judgments of the form $t{::}\alpha$ and α typ by disambiguations respecting suitable constraints.

2 Two systems

Let us begin with a system of dependent types, confining our attention to three forms of judgments, Γ context, A type and $t : A$. (That is, for simplicity, we leave out equations between types and between terms.) Contexts can be formed from the empty sequence $\langle\rangle$

$$(\langle\rangle\mathrm{c}) \quad \frac{}{\vdash \langle\rangle \text{ context}}$$

$$(\mathrm{tc}) \quad \frac{\Gamma \vdash A \text{ type}}{\vdash \Gamma, x{:}A \text{ context}}\ x \notin \mathsf{Var}(\Gamma)$$

where $\mathsf{Var}(\Gamma)$ is the set of variables occurring in Γ. Assumptions cross $\vdash$

$$(\mathrm{As})\ \frac{\vdash \Gamma, x{:}A\ \mathsf{context}}{\Gamma, x{:}A \vdash x{:}A}$$

and contexts weaken to the right

$$(\mathrm{Weak})\ \frac{\Gamma \vdash \Theta \qquad \vdash \Gamma, \Delta\ \mathsf{context}}{\Gamma, \Delta \vdash \Theta}$$

(where Θ ranges over judgments A type and $t : A$). Next come formation (F), introduction (I) and elimination (E) rules for dependent functions $\prod$ (generalizing non-dependent functions $\rightarrow$)

$$(\textstyle\prod \mathrm{F})\ \frac{\vdash \Gamma, x{:}A\ \mathsf{context} \qquad \Gamma, x{:}A \vdash B\ \mathsf{type}}{\Gamma \vdash (\prod x{:}A)B\ \mathsf{type}}$$

$$(\textstyle\prod \mathrm{I})\ \frac{\Gamma, x{:}A \vdash t{:}B}{\Gamma \vdash \lambda x.t{:}(\prod x{:}A)B}$$

$$(\textstyle\prod \mathrm{E})\ \frac{\Gamma \vdash t{:}(\prod x{:}A)B \qquad \Gamma \vdash u{:}A}{\Gamma \vdash ap(t,u){:}B[x := u]}$$

(where $B[x := u]$ is B with x replaced by u) and for dependent sums $\sum$ (generalizing Cartesian products $\times$)

$$(\textstyle\sum \mathrm{F})\ \frac{\vdash \Gamma, x{:}A\ \mathsf{context} \qquad \Gamma, x{:}A \vdash B\ \mathsf{type}}{\Gamma \vdash (\sum x{:}A)B\ \mathsf{type}}$$

$$(\textstyle\sum \mathrm{I})\ \frac{\Gamma \vdash t{:}A \qquad \Gamma \vdash u{:}B[x := t]}{\Gamma \vdash \langle t,u\rangle{:}(\sum x{:}A)B}$$

$$(\textstyle\sum \mathrm{E}_p)\ \frac{\Gamma \vdash t{:}(\sum x{:}A)B}{\Gamma \vdash p(t){:}A}$$

$$(\textstyle\sum \mathrm{E}_q)\ \frac{\Gamma \vdash t{:}(\sum x{:}A)B}{\Gamma \vdash q(t){:}B[x := p(t)]}\ .$$

Now for the novel part: a second system, with terms t as before, but colons squared, and :-types A, B replaced by *decorated expressions* α, β and *unadorned expressions* $\mathcal{A}$ generated simultaneously according to

$$\begin{aligned} \mathcal{A} &::= a \mid (\textstyle\prod x::\alpha)\mathcal{A} \mid (\sum x::\alpha)\mathcal{A} \\ \alpha, \beta &::= \mathcal{A}_x \mid (\textstyle\prod x::\alpha)\beta \mid (\sum x::\alpha)\beta \mid \\ &\qquad \alpha_\beta\{t\} \mid \alpha^p \mid \alpha^q\{t\} \end{aligned}$$

where a belongs to a fixed countable set $\mathcal{X}$ of variables. The intent (made precise in the next section) is that a u-expression $\mathcal{A}$ describes a set of :-types, while a d-expression α denotes a choice from such a set. D-expressions of the form a_x, α^p, $\alpha^q\{t\}$ and $\alpha_\beta\{t\}$ are said to be *non-dependent*, and are used, in conjunction with *constraints* of the form $\mathrm{fcn}(\alpha,\beta)$, $\mathrm{sum}(\alpha)$ and $\mathrm{eq}(\alpha,\beta)$, to infer sequents relativized to finite sets C of constraints as follows

$$(\textstyle\prod \mathrm{n})\ \frac{\Gamma \vdash_{\mathsf{C}} t::\alpha \qquad \Gamma \vdash_{\mathsf{C}'} u::\beta}{\Gamma \vdash_{\mathsf{C}\cup\mathsf{C}'\cup\{\mathrm{fcn}(\alpha,\beta)\}} ap(t,u)::\alpha_\beta\{u\}}$$

$$(\textstyle\sum \mathrm{n}_p)\ \frac{\Gamma \vdash_{\mathsf{C}} t::\alpha}{\Gamma \vdash_{\mathsf{C}\cup\{\mathrm{sum}(\alpha)\}} p(t)::\alpha^p}$$

$$(\textstyle\sum \mathrm{n}_q)\ \frac{\Gamma \vdash_{\mathsf{C}} t::\alpha}{\Gamma \vdash_{\mathsf{C}\cup\{\mathrm{sum}(\alpha)\}} q(t)::\alpha^q\{p(t)\}}\ ,$$

where each of the three rules have the side condition that α is non-dependent.[1] In addition,

$$(\textstyle\prod \mathrm{E})_{\neq}\ \frac{\Gamma \vdash_{\mathsf{C}} t::(\prod x::\alpha)\beta \qquad \Gamma \vdash_{\mathsf{C}'} u::\gamma}{\Gamma \vdash_{\mathsf{C}\cup\mathsf{C}'\cup\{\mathrm{eq}(\alpha,\gamma)\}} ap(t,u)::\beta[x := u]}$$

with the side condition $\alpha \neq \gamma$. The intuition (formalized in clauses (c2)-(c4) of the next section) is that

- the constraint $\mathrm{eq}(\alpha,\gamma)$ is satisfied by a disambiguation equating α with γ,
- $\mathrm{fcn}(\alpha,\beta)$ is satisfied by a disambiguation of α and β to :-types of the form $(\prod x : A)B$ and A respectively

and

- $\mathrm{sum}(\alpha)$ is satisfied by a disambiguation of α to a :-type of the form $(\sum x{:}A)B)$.

Rules of the previous system translate to

$$(\langle\rangle\mathrm{c})^\circ\ \frac{}{\vdash_\emptyset \langle\rangle\ \mathsf{cxt}}$$

$$(\mathrm{tc})^\circ\ \frac{\Gamma \vdash_{\mathsf{C}} \mathcal{A}\ \mathsf{typ}}{\vdash_{\mathsf{C}} \Gamma, x::\mathcal{A}_x\ \mathsf{cxt}}\ x \notin \mathsf{Var}(\Gamma)$$

$$(\mathrm{As})^\circ\ \frac{\vdash_{\mathsf{C}} \Gamma, x::\alpha\ \mathsf{cxt}}{\Gamma, x::\alpha \vdash_{\mathsf{C}} x::\alpha}$$

$$(\mathrm{Weak})^\circ\ \frac{\Gamma \vdash_{\mathsf{C}} \Theta \qquad \vdash_{\mathsf{C}'} \Gamma, \Delta\ \mathsf{cxt}}{\Gamma, \Delta \vdash_{\mathsf{C}\cup\mathsf{C}'} \Theta}$$

$$(\textstyle\prod \mathrm{F})^\circ\ \frac{\vdash_{\mathsf{C}} \Gamma, x::\alpha\ \mathsf{cxt} \qquad \Gamma, x::\alpha \vdash_{\mathsf{C}'} \mathcal{B}\ \mathsf{typ}}{\Gamma \vdash_{\mathsf{C}\cup\mathsf{C}'} (\prod x::\alpha)\mathcal{B}\ \mathsf{typ}}$$

$$(\textstyle\prod \mathrm{I})^\circ\ \frac{\Gamma, x::\alpha \vdash_{\mathsf{C}} t::\beta}{\Gamma \vdash_{\mathsf{C}} \lambda x.t::(\prod x::\alpha)\beta}$$

$$(\textstyle\prod \mathrm{E})^\circ\ \frac{\Gamma \vdash_{\mathsf{C}} t::(\prod x::\alpha)\beta \qquad \Gamma \vdash_{\mathsf{C}'} u::\alpha}{\Gamma \vdash_{\mathsf{C}\cup\mathsf{C}'} ap(t,u)::\beta[x := u]}$$

$$(\textstyle\sum \mathrm{F})^\circ\ \frac{\vdash_{\mathsf{C}} \Gamma, x::\alpha\ \mathsf{cxt} \qquad \Gamma, x::\alpha \vdash_{\mathsf{C}'} \mathcal{B}\ \mathsf{typ}}{\Gamma \vdash_{\mathsf{C}\cup\mathsf{C}'} (\sum x::\alpha)\mathcal{B}\ \mathsf{typ}}$$

$$(\textstyle\sum \mathrm{I})^\circ\ \frac{\Gamma \vdash_{\mathsf{C}} t::\alpha \qquad \Gamma \vdash_{\mathsf{C}'} u::\beta[x := t]}{\Gamma \vdash_{\mathsf{C}\cup\mathsf{C}'} \langle t,u\rangle::(\sum x::\alpha)\beta}$$

$$(\textstyle\sum \mathrm{E}_p)^\circ\ \frac{\Gamma \vdash_{\mathsf{C}} t::(\sum x::\alpha)\beta}{\Gamma \vdash_{\mathsf{C}} p(t)::\alpha}$$

$$(\textstyle\sum \mathrm{E}_q)^\circ\ \frac{\Gamma \vdash_{\mathsf{C}} t::(\sum x::\alpha)\beta}{\Gamma \vdash_{\mathsf{C}} q(t)::\beta[x := p(t)]}\ .$$

[1] Variations on this side condition are taken up in §5 below.

Further rules provide co-varying choices

$$(::c)\quad \frac{\Gamma \vdash_C t::\alpha}{\vdash_C \Gamma, x::\alpha\ \mathsf{cxt}}\ x \notin \mathsf{Var}(\Gamma)$$

$$(\prod c)\quad \frac{\vdash_C \Gamma, x::\alpha\ \mathsf{cxt} \qquad \Gamma, x::\alpha \vdash_{C'} t::\beta}{\vdash_{C \cup C'} \Gamma, y::(\prod x::\alpha)\beta\ \mathsf{cxt}}$$

$$(\sum c)\quad \frac{\vdash_C \Gamma, x::\alpha\ \mathsf{cxt} \qquad \Gamma, x::\alpha \vdash_{C'} t::\beta}{\vdash_{C \cup C'} \Gamma, y::(\sum x::\alpha)\beta\ \mathsf{cxt}}\ ,$$

where $(\prod c)$ and $(\sum c)$ each have the side condition $y \notin \mathsf{Var}(\Gamma) \cup \{x\}$.

3 Disambiguating ::

Let **Ty** be the collection of :-type expressions A, and for every d-expression α, let

- $\mathcal{X}(\alpha)$ be the set of variables in $\mathcal{X}$ occurring in α
- $\mathbf{D}(\alpha)$ be the set of (sub-)d-expressions β occurring in α (including α)

and

- $\mathbf{U}(\alpha)$ be the set of (sub-)u-expressions $\mathcal{A}$ occurring in α.

Suppressing the tedious inductive definitions of $\mathbf{D}(\alpha)$ and $\mathbf{U}(\alpha)$, let us just note that, for instance, $\mathbf{D}((\prod x::a_x)(\sum y::a'_y)a_z)$ is

$$\{(\prod x::a_z)(\sum y::a'_y)a_z,\ (\sum y::a'_y)a_z,\ a_x,\ a'_y,\ a_z\}$$

and $\mathbf{U}((\prod x::a_x)(\sum y::a'_y)a_z)$ is

$$\{(\prod x::a_x)(\sum y::a'_y)a,\ (\sum y::a'_y)a,\ a,\ a'\}\ .$$

Next, given a d-expression α_0 and a function $\rho : \mathbf{D}(\alpha_0) \to \mathbf{Ty}$, let $\cdot^\rho$ be the function from $\mathbf{U}(\alpha_0)$ to Pow(**Ty**) such that for $a \in \mathcal{X}(\alpha_0)$,

$$a^\rho \ = \ \mathbf{Ty}\ ,$$

for $(\prod x::\alpha)\mathcal{A} \in \mathbf{U}(\alpha_0)$,

$$((\prod x::\alpha)\mathcal{A})^\rho \ = \ \{(\prod x:\rho(\alpha))A \mid A \in \mathcal{A}^\rho\}$$

and for $(\sum x::\alpha)\mathcal{A} \in \mathbf{U}(\alpha_0)$,

$$((\sum x::\alpha)\mathcal{A})^\rho \ = \ \{(\sum x:\rho(\alpha))A \mid A \in \mathcal{A}^\rho\}\ .$$

Now, call ρ a *disambiguation of* α_0 if the following conditions hold:

(i) for every $\mathcal{A}_x \in \mathbf{D}(\alpha_0)$, $\rho(\mathcal{A}_x) \in \mathcal{A}^\rho$

(ii) for every $(\prod x::\alpha)\beta \in \mathbf{D}(\alpha_0)$,
$\rho((\prod x::\alpha)\beta) = (\prod x:\rho(\alpha))\rho(\beta)$

(iii) for every $(\sum x::\alpha)\beta \in \mathbf{D}(\alpha_0)$,
$\rho((\sum x::\alpha)\beta) = (\sum x:\rho(\alpha))\rho(\beta)$

(iv) for every $\alpha_\beta\{t\} \in \mathbf{D}(\alpha_0)$,
$\rho(\alpha) = (\prod x:\rho(\beta))A$
for some x and A with $A[x := t] = \rho(\alpha_\beta\{t\})$

(v) for every $\alpha^p \in \mathbf{D}(\alpha_0)$,
$\rho(\alpha) = (\sum x:\rho(\alpha^p))B$ for some x and B

and

(vi) for every $\alpha^q\{t\} \in \mathbf{D}(\alpha_0)$,
$\rho(\alpha) = (\sum x:A)B$
for some x, A and B with
$B[x := t] = \rho(\alpha^q\{t\})$.

Next, let us pass from a single d-expression α_0 to a fixed set $\mathbf{D}_0$ of d-expressions. A *disambiguation of the set* $\mathbf{D}_0$ of d-expressions is a function ρ from $\bigcup\{\mathbf{D}(\alpha) \mid \alpha \in \mathbf{D}_0\}$ to **Ty** such that for all $\alpha \in \mathbf{D}_0$, ρ restricted to $\mathbf{D}(\alpha)$ is a disambiguation of α.[2] A disambiguation ρ of $\mathbf{D}_0$ *respects* a set C of constraints if there is an extension $\rho^+ \supseteq \rho$ so that

(c1) ρ^+ is a disambiguation of

$$\mathbf{D}_0 \cup \{\alpha \mid \alpha \text{ is mentioned in C}\}$$

(c2) whenever $\mathrm{eq}(\alpha, \beta) \in \mathsf{C}$, $\rho^+(\alpha) = \rho^+(\beta)$

(c3) whenever $\mathrm{fcn}(\alpha, \beta) \in \mathsf{C}$,
$\rho^+(\alpha) = (\prod x:\rho^+(\beta))B$ for some x and B

and

(c4) whenever $\mathrm{sum}(\alpha) \in \mathsf{C}$,
$\rho^+(\alpha) = (\sum x:A)B$ for some x, A and B.

Given a sequence Γ of the form

$$x_1:\alpha_1,\ \cdots, x_n:\alpha_n,$$

let $ima(\Gamma) = \{\alpha_1, \ldots, \alpha_n\}$, and for every disambiguation ρ of a set $\mathbf{D}_0$ containing $ima(\Gamma)$, let

$$\Gamma_\rho \ = \ x_1:\rho(\alpha_1),\ \cdots,\ x_n:\rho(\alpha_n)\ .$$

Let us say that $\vdash_C \Gamma$ cxt *can be disambiguated to* $\vdash \Gamma'$ context if there is a disambiguation ρ of $ima(\Gamma)$ respecting C such that $\Gamma' = \Gamma_\rho$. Similarly, $\Gamma \vdash_C \alpha$ typ $(t :: \alpha)$ *can be disambiguated to* $\Gamma' \vdash A$ type $(t : A)$ if there is a disambiguation ρ of $ima(\Gamma) \cup \{\alpha\}$ respecting C such that $\Gamma' = \Gamma_\rho$ and $A = \rho(\alpha)$.

[2] It is crucial for this formulation that the set $\mathsf{Var}(\Gamma)$ mentioned in side conditions for various rules in the previous section include all variables in Γ, whether they occur freely or bound.

4 Relating the derivations

Observe that to derive a sequent other than $\vdash \langle\rangle$ context in the first system, or $\vdash_\emptyset \langle\rangle$ cxt in the second, we need to assume a non-empty set $\mathcal{T}$ of sequents. Let us agree to write $\Gamma \vdash^{\mathcal{T}} \Theta$ to mean that the sequent $\Gamma \vdash \Theta$ is derivable from $\mathcal{T}$, and $\vdash^{\mathcal{T}} \Gamma$ context to mean that $\vdash \Gamma$ context is derivable from $\mathcal{T}$. Similarly, for the second system (with $\vdash$ replaced by $\vdash_{\mathsf{C}}$, context by cxt, etc). As every rule (R) for the first system has a counterpart (R)° in the second system, it is tempting to seek a natural translation $\cdot^\circ$ from the first system to the second system validating the following

Claim: $\Gamma \vdash^{\mathcal{T}} \Theta$ implies $\Gamma^\circ \vdash_\emptyset^{\mathcal{T}^\circ} \Theta^\circ$.

For example, if $\mathcal{T}$ consists of the sequent $\vdash A$ type, Γ is empty, and Θ is $\lambda x.x : (\prod x:A)A$, then $\mathcal{T}^\circ$ is $\{\vdash_\emptyset a\ \mathsf{typ}\}$, Γ° is empty, and Θ° is $\lambda x.x :: (\prod x :: a_x)a_x$. Replacing Γ by $y:A$, and Θ by $\lambda x.y : (\prod x:A)A$, we get $y :: a_y$ for Γ° and $\lambda x.y :: (\prod x :: a_x)a_y$ for Θ°.

To pin down a systematic definition of $\cdot^\circ$, it is easy enough to fix a 1-1 mapping $X \mapsto a^X$ of atomic :-types X to variables a^X in $\mathcal{X}$, and set

$$X^\circ = a^X \quad (7)$$

$$((\prod x:A)B)^\circ = (\prod x :: A^\circ{}_x)B^\circ \quad (8)$$

$$((\sum x:A)B)^\circ = (\sum x :: A^\circ{}_x)B^\circ \quad (9)$$

$$(A\ \mathsf{type})^\circ = A^\circ\ \mathsf{typ} \quad (10)$$

$$(x:A)^\circ = x :: A^\circ{}_x\ . \quad (11)$$

While (11) induces a translation Γ° of a context Γ, what about $(t:A)^\circ$, where t is not just, as in (11), a variable x? Before revising the definition of d-expressions α to accommodate subscripts t on A°, let us explore what we can do with (7)-(11). Define a *simple type base* $\mathcal{T}$ to be a set of sequents of the form $\Gamma \vdash A$ type. Given a simple type base $\mathcal{T}$, let $\mathcal{T}^\circ$ be its translation into :: according to equations (11) and (10). By induction on derivations from $\mathcal{T}$, we can prove a reformulation of the claim above, where Γ° and Θ° are replaced by disambiguations.

Proposition 1. *Let $\mathcal{T}$ be a simple type base.*

(a) $\vdash^{\mathcal{T}} \Gamma$ context *implies* $\vdash_\emptyset^{\mathcal{T}^\circ} \Gamma'$ cxt *for some* Γ' *such that* $\vdash_\emptyset \Gamma'$ cxt *can be disambiguated to* $\vdash \Gamma$ context.

(b) $\Gamma \vdash^{\mathcal{T}} A$ type *implies* $\Gamma' \vdash_\emptyset^{\mathcal{T}^\circ} \alpha$ typ *for some* Γ' *and* α *such that* $\Gamma' \vdash_\emptyset \alpha$ typ *can be disambiguated to* $\Gamma \vdash A$ type.

(c) $\Gamma \vdash^{\mathcal{T}} t:A$ *implies* $\Gamma' \vdash_\emptyset^{\mathcal{T}^\circ} t :: \alpha$ *for some* Γ' *and* α *such that* $\Gamma' \vdash_\emptyset t :: \alpha$ *can be disambiguated to* $\Gamma \vdash t:A$.

Moreover, as the rules $(\prod \mathrm{n})$, $(\sum \mathrm{n}_p)$ and $(\sum \mathrm{n}_q)$ can, for disambiguations that meet the appropriate constraints, be replaced by $(\prod \mathrm{E})$, $(\sum \mathrm{E}_p)$ and $(\sum \mathrm{E}_q)$, it follows that

Proposition 2. *Let $\mathcal{T}$ be a simple type base.*

(a) If $\vdash_{\mathsf{C}}^{\mathcal{T}^\circ} \Gamma$ cxt *and* $\vdash_{\mathsf{C}} \Gamma$ cxt *can be disambiguated to* $\vdash \Gamma'$ context, *then* $\vdash^{\mathcal{T}} \Gamma'$ context.

(b) If $\Gamma \vdash_{\mathsf{C}}^{\mathcal{T}^\circ} \alpha$ typ *and* $\Gamma \vdash_{\mathsf{C}} \alpha$ typ *can be disambiguated to* $\Gamma' \vdash A$ type, *then* $\Gamma' \vdash^{\mathcal{T}} A$ type.

(c) If $\Gamma \vdash_{\mathsf{C}}^{\mathcal{T}^\circ} t :: \alpha$ *and* $\Gamma \vdash_{\mathsf{C}} t :: \alpha$ *can be disambiguated to* $\Gamma' \vdash t:A$, *then* $\Gamma' \vdash^{\mathcal{T}} t:A$.

Conversely, going from $(\prod \mathrm{E})^\circ$, $(\sum \mathrm{E}_p)^\circ$ and $(\sum \mathrm{E}_q)^\circ$ to $(\prod \mathrm{n})$, $(\sum \mathrm{n}_p)$ and $(\sum \mathrm{n}_q)$, we have

Proposition 3. *Let $\mathcal{T}$ be a simple type base.*

(a) If $\vdash^{\mathcal{T}} \Gamma'$ context *and* $\vdash_{\mathsf{C}} \Gamma$ cxt *can be disambiguated to* $\vdash \Gamma'$ context, *then* $\vdash_{\mathsf{C}}^{\mathcal{T}^\circ} \Gamma$ cxt.

(b) If $\Gamma' \vdash^{\mathcal{T}} A$ type *and* $\Gamma \vdash_{\mathsf{C}} \alpha$ typ *can be disambiguated to* $\Gamma' \vdash A$ type, *then* $\Gamma \vdash_{\mathsf{C}}^{\mathcal{T}^\circ} \alpha$ typ.

(c) If $\Gamma' \vdash^{\mathcal{T}} t:A$ *and* $\Gamma \vdash_{\mathsf{C}} t :: \alpha$ *can be disambiguated to* $\Gamma' \vdash t:A$, *then* $\Gamma \vdash_{\mathsf{C}}^{\mathcal{T}^\circ} t :: \alpha$.

Proposition 3(c) is roughly $\Leftarrow$ of (3), while Proposition 2(c) approximates $\Rightarrow$ of (4). If Proposition 2 says that the system for :: above is sound, Proposition 3 says it is complete.[3] To tie together Propositions 2 and 3 in an equivalence, it is useful to define a set C of constraints to be *satisfiable* if $\emptyset$ is a disambiguation (of $\emptyset$) respecting C. Note that sequents $\vdash_{\mathsf{C}} \Gamma$ and $\Gamma \vdash_{\mathsf{C}} \Theta$ have disambiguations exactly when C is satisfiable. Consequently, Propositions 2 and 3 yield (focussing on ::)

Corollary 4. *Given a simple type base $\mathcal{T}$ and a satisfiable set* C *of constraints, the following are equivalent.*

(i) $\Gamma \vdash_{\mathsf{C}}^{\mathcal{T}} t :: \alpha$

(ii) $\Gamma' \vdash^{\mathcal{T}} t : A$, *for every sequent* $\Gamma' \vdash t : A$ *to which* $\Gamma \vdash_{\mathsf{C}} t :: \alpha$ *can be disambiguated*

(iii) $\Gamma' \vdash^{\mathcal{T}} t : A$, *for some sequent* $\Gamma' \vdash t : A$ *to which* $\Gamma \vdash_{\mathsf{C}} t :: \alpha$ *can be disambiguated.*

[3] As for how this relates to soundness and completeness in say, classical predicate logic, please see the discussion of translation versus entailment in the concluding paragraph below.

The formulation above of Corollary 4 depends on the possibility of deriving sequents $\Gamma \vdash_{\mathsf{C}} \Theta$ where C is not satisfiable. We could have, of course, added side conditions to $(\prod \mathrm{n})$, $(\sum \mathrm{n}_p)$ and $(\sum \mathrm{n}_q)$ checking that the constraints are satisfiable. By electing not to do so, we have exposed a certain separability of inference from constraint satisfaction, which we will explore in the next section.

For now, turning to the general case of a set $\mathcal{T}$ of :-sequents, observe that if $\mathcal{T}$ is to be compatible with the first system, then

(i) whenever $\Gamma \vdash \lambda x.t{:}C$ belongs to $\mathcal{T}$, C must have the form $(\prod x{:}A)B$ with $\Gamma, x{:}A \vdash^{\mathcal{T}} t{:}B$

(ii) whenever $\Gamma \vdash \langle t, u\rangle{:}C$ belongs to $\mathcal{T}$, C must have the form $(\sum x{:}A)B$ with $\Gamma \vdash^{\mathcal{T}} t{:}A$ and $\Gamma \vdash^{\mathcal{T}} u{:}B[x := t]$

(iii) whenever $\Gamma \vdash ap(t,u){:}B$ belongs to $\mathcal{T}$, $\Gamma \vdash^{\mathcal{T}} t{:}(\prod x:A)B$ for some x and A such that $\Gamma \vdash^{\mathcal{T}} u{:}A$

(iv) whenever $\Gamma \vdash p(t){:}A$ belongs to $\mathcal{T}$, $\Gamma \vdash^{\mathcal{T}} t{:}(\sum x{:}A)B$ for some x and B

(v) whenever $\Gamma \vdash q(t){:}B$ belongs to $\mathcal{T}$, $\Gamma \vdash^{\mathcal{T}} t{:}(\sum x{:}A)B$ for some x and A

(vi) whenever $\Gamma \vdash \Theta$ belongs to $\mathcal{T}$, $\vdash^{\mathcal{T}} \Gamma$ context

(vii) whenever $\vdash \Gamma, x : A$ context or $\Gamma \vdash t : A$ belongs to $\mathcal{T}$, $\Gamma \vdash^{\mathcal{T}} A$ type

and

(viii) whenever $\Gamma \vdash (\prod x{:}A)B$ type or $\Gamma \vdash (\sum x{:}A)B$ type belongs to $\mathcal{T}$, $\Gamma \vdash^{\mathcal{T}} A$ type and $\Gamma, x{:}A \vdash^{\mathcal{T}} B$ type.

Thus, a base set $\mathcal{T}$ compatible with the first system can be assumed without loss of generality to consist of sequents of two forms: $\Gamma \vdash A$ type and $\Gamma \vdash t{:}B$, where A and t are atomic (i.e. indecomposable by $\prod, \sum$ and $\lambda, \langle,\rangle, ap, p, q$ respectively). By clause (vii) above, it follows that for every sequent $\Gamma \vdash t : B$ in $\mathcal{T}$, there is some $\mathcal{T}_0 \subseteq \mathcal{T}$ such that $\Gamma \vdash^{\mathcal{T}_0} B$ type. So starting with simple type bases $\mathcal{T}_0$, we can take (for B) the D-expression β which Proposition 1(b) returns, given $\Gamma \vdash^{\mathcal{T}_0} B$ type. We can then define $\mathcal{T}^\circ$ by translating $\Gamma \vdash t{:}B$ as $\Gamma^\circ \vdash t :: \beta$. Alternatively, we might make do with simple type bases by reformulating t as a variable x_t, and smuggling x_t into enriched contexts Γ' for which a $\mathcal{T}$-derivation of $\Gamma' \vdash \Theta'$ is sought (with Θ' adjusted for x_t, rather than t). That is, instead of injecting t on top of $\vdash$ (within some superscript $\mathcal{T}$), we might add it (along with the context it depends on) to the left of $\vdash$.

5 Variations and refinements

The sequent rules for :: chosen above lie between two extremes. The first is obtained by dropping the side conditions of $(\prod \mathrm{n})$, $(\sum \mathrm{n}_p)$ and $(\sum \mathrm{n}_q)$, rendering the four rules $(\prod \mathrm{E})^\circ$, $(\sum \mathrm{E}_p)^\circ$, $(\sum \mathrm{n}_q)^\circ$ and $(\prod \mathrm{E})_{\neq}$ redundant. The idea is to put off constraint satisfaction to the very end. Alternatively, the side conditions of $(\prod \mathrm{n})$, $(\sum \mathrm{n}_p)$, $(\sum \mathrm{n}_q)$ and $(\prod \mathrm{E})_{\neq}$ might be strengthened to check that the constraints are satisfiable (adding to $(\prod \mathrm{n})$, for example, the requirement that $\mathrm{sum}(\alpha) \notin \mathsf{C} \cup \mathsf{C}'$ and $\mathrm{eq}(\alpha, \beta') \notin \mathsf{C} \cup \mathsf{C}'$ for all $\beta' \in \mathbf{D}(\beta)$). Assuming that we did, we might as well rewrite the relevant d-expressions, and dispense with the subscript C. (For example, with the appropriate side conditions, $(\prod \mathrm{n})$ might be revised to

$$\frac{\Gamma \vdash t :: \alpha \qquad \Gamma \vdash u :: \beta}{\Gamma[\alpha := (\prod x :: \beta)\alpha] \vdash ap(t,u) :: \alpha[x := u]}$$

where $\Gamma[\alpha := (\prod x :: \beta)\alpha]$ is Γ with α replaced by $(\prod x :: \beta)\alpha$.) An increase in complexity of the side conditions is a price that we may well be willing to pay to get rid of subscripts C. Or perhaps not.

Among the considerations relevant to the interplay between inference and constraint satisfaction are:

(I) the difficulty/ease of applying/abusing inference rules

(D) the difficulty of disambiguating (i.e. of verifying the assumption in Corollary 4 of a "satisfiable set C")

(W) wasted effort on spurious readings (i.e. sequents $\Gamma \vdash_{\mathsf{C}} \Theta$ with non-satisfiable C).

Designing sequent rules balancing (I), (D) and (W) is a delicate language engineering problem, about which it is probably best to keep an open mind. Consider again the binary connective $\bullet$ mentioned in the introduction (which we set aside to concentrate instead on certain underspecified representations). It is easy enough to refine the notion of a disambiguation to an ϵ-disambiguation, where ϵ is a function encoding the readings specified by $\bullet$. In particular, example (1) can be re-conceptualized in terms of

(i) the instance

$$\frac{\Gamma \vdash_{\emptyset} x :: \alpha \qquad \Gamma \vdash_{\emptyset} y :: \beta}{\Gamma \vdash_{\{\mathrm{fcn}(\alpha,\beta)\}} ap(x,y) :: \alpha\beta\{y\}}$$

of the rule $(\prod \mathrm{n})$ where Γ is the context $x :: \alpha, y :: \beta$, and say, α is a_x and β is a'_y (against the base set of sequents $\vdash_{\emptyset} a$ typ and $\vdash_{\emptyset} a'$ typ)

and

(ii) an ϵ-disambiguation of $\alpha_\beta\{y\}$, where $\epsilon(\alpha) = \{A \to B, C\}$ and $\epsilon(\beta) = \{A, D\}$.

Given a (partial) function ϵ from some set $\mathbf{D}_0$ of d-expressions to $\mathrm{Pow}(\mathbf{Ty}) - \{\emptyset\}$, an *$\epsilon$-disambiguation* of $\mathbf{D}_0$ is a disambiguation ρ of $\mathbf{D}_0$ such that for every α in the domain of ϵ, $\rho(\alpha) \in \epsilon(\alpha)$.[4] Now, there are at least two ways to incorporate ϵ-disambiguations into Corollary 4. The first is to leave the sequent rules for :: as before, but to relativize the notion of a satisfiable set C of constraints to ϵ (adding to the definition of "ρ respects C" the requirement that the extension ρ^+ be an ϵ-disambiguation). A more interesting approach is to bring ϵ into the sequent rules by forming constraints to guarantee that disambiguations are ϵ-disambiguations (the general point being that all kinds of information might be encoded within the subscripts C on $\vdash$). For starters, we might change the rule $(\langle\rangle \mathrm{c})^\circ$ to

$$(\langle\rangle \mathrm{c})^\circ \quad \frac{}{\vdash_{\emptyset,\epsilon} \langle\rangle\ \mathrm{cxt}}$$

where the subscript $\emptyset, \epsilon$ denotes a constraint set requiring that for every α in the domain of ϵ, α can only be disambiguated into an element of $\epsilon(\alpha)$. The rules $(\prod \mathrm{n})$, $(\sum \mathrm{n}_p)$, $(\sum \mathrm{n}_q)$ and $(\prod \mathrm{E})_{\neq}$ might then be modified to trim the sets $\epsilon(\alpha)$ so that in example (1), for instance, the application of $(\prod \mathrm{n})$ reduces $\epsilon(\alpha) = \{A \to B, C\}$ to $\epsilon'(\alpha) = \{A \to B\}$. More specifically, let $(\prod \mathrm{n})$ be

$$(\prod \mathrm{n}) \quad \frac{\Gamma \vdash_{\mathsf{C},\epsilon} x :: \alpha \qquad \Gamma \vdash_{\mathsf{C}',\epsilon'} y :: \beta}{\Gamma \vdash_{\mathsf{C}'',\epsilon''} ap(x,y) :: \alpha_\beta\{y\}}$$

with the side condition that

> α is non-dependent, and ϵ is consistent with ϵ' (i.e. for every α in the domain of both ϵ and ϵ', $\epsilon(\alpha) \cap \epsilon'(\alpha) \neq \emptyset$)

and where C'' is $\mathsf{C} \cup \mathsf{C}' \cup \{\mathrm{fcn}(\alpha, \beta)\}$ and ϵ'' combines ϵ and ϵ' in the obvious way (e.g. mapping every α in the domain of both ϵ and ϵ' to $\epsilon(\alpha) \cap \epsilon'(\alpha)$). (Subscripts C, ϵ may, as in the case of $\emptyset, \epsilon$, be construed as single constraint sets, which are convenient for certain purposes to decompose into pairs C, ϵ.)

We could put a bit more work into $(\prod \mathrm{n})$ as follows. Given an integer $k \geq 0$, let $\mathbf{D}_k(\beta)$ be the subset of the set $\mathbf{D}(\beta)$ of sub-d-expressions of β, from which β can be constructed with $\leq k$ applications of d-expression formation rules. (For example, $\mathbf{D}_1((\sum x :: \alpha)(\prod y :: \beta)\gamma)$ is

$$\{(\textstyle\sum x :: \alpha)(\prod y :: \beta)\gamma,\ (\prod y :: \beta)\gamma, \alpha\}$$

with β and γ buried too deeply to be included.) Now, for a fixed k, add to the side condition of $(\prod \mathrm{n})$ the requirement that $\mathrm{sum}(\alpha) \notin \mathsf{C} \cup \mathsf{C}'$ and $\mathrm{eq}(\alpha, \beta') \notin \mathsf{C} \cup \mathsf{C}'$ for all $\beta' \in \mathbf{D}_k(\beta)$; and choose ϵ'' to also rule out the possibility that α is β' for some $\beta' \in \mathbf{D}_k(\beta)$. Clearly, the larger k is, the stronger the rule becomes. But so long as a satisfiability check is made after inference (as suggested by Corollary 4), it is not necessary that the constraint set C in a sequent $\Gamma \vdash_{\mathsf{C}} \Theta$ that has been derived be reduced (to make all its consequences explicit) any more than it is necessary to require that C be satisfiable. (Concerning the latter, notice also that spurious sequents may drop out as further inferences are made, eliminating the need there to ever disambiguate.)

To establish (the analog of) Corollary 4, a crucial property for a sequent rule

$$(*) \quad \frac{\Gamma_1 \vdash_{\mathsf{C}_1} \Theta_1 \quad \cdots \quad \Gamma_n \vdash_{\mathsf{C}_n} \Theta_n}{\Gamma \vdash_{\mathsf{C}} \Theta}$$

to have is ***monotonicity***: for every disambiguation ρ respecting C, ρ respects C_i for $1 \leq i \leq n$.[5] (This is a generalization of $\mathsf{C}_i \subseteq \mathsf{C}$, suggested by the encoding above of ϵ-disambiguations/$\bullet$ in terms of constraints.) To weed out spurious readings (consideration (W) above), side conditions might be imposed on (*), which ought (according to (I)) to be as simple as possible. The trick in designing C (and (*)) is to make inference $\vdash$ just complicated enough so as, (D), not to put an undue burden on disambiguating at the end. The whole idea is to distribute the work between inferring sequents and (subsequently) checking satisfiability. The claim is that the middle ground between the two extremes mentioned at the beginning of this section (i.e. between lax side conditions that leave the bulk of the work to disambiguation at the end, and strict side conditions that essentially reduce :: to :) is fertile.

6 Discussion

More than one reader (of a previous draft of this paper) has asked about linguistic examples. The

[4] We can also introduce $\bullet$ as a binary connective on u-expressions and/or on d-expressions, although this would require a bit more work and would run against the spirit of underspecified representations, insofar as $\bullet$ spells out possible disambiguations.

[5] Compare to (Alshawi and Crouch, 1992). Monotonicity is used above for soundness, Proposition 2. Completeness, Proposition 3, follows from having enough such rules (*) (or equivalently, making the side conditions for (*) comprehensive enough).

short, easy answer is that the sort of ambiguity addressed here can be syntactic (with types A ranging over grammatical categories) or semantic (with types drawn, say, from a higher-order predicate logic). Clearly, more must be said — for example, to properly motivate the rules ($::c$), ($\prod c$) and ($\sum c$) mentioned at the end of §2. Detailed case studies are bound to push $::$ in various directions; and no doubt, after applying enough pressure, the system above will break.

Be that as it may, I hope that case studies will be carried out (by others and/or by myself), testing, by stretching, the basic idea above. I close with a few words on that idea, and, begging the reader's indulgence, on the theoretical background out of which, in my experience, it grew. From examining the binary connective $\bullet$ in (Fernando, 1997), I concluded that $\bullet$ is unlike any ordinary logical connective related to entailment because the force of $\bullet$ is best understood relative not to entailment, but to translation. Underlying the distinction between entailment and translation is that between well-formed formulas and possibly ambiguous expressions (corresponding, in the present work, to :-types, on the one hand, and d- and u-expressions, on the other). An abstract picture relating the processes of translation and entailment is framed in (Fernando, in press), which I have attempted to flesh out here for the case of ITT, with a view to extending ITT's applications beyond anaphora to underspecification. The obvious step is to drop all types, and construe the terms as belonging to a type-free λ-calculus. The twist above is that ambiguous expressions *are* typed by d-expressions α, distinct from u-expressions $\mathcal{A}$. The construction is, in fact, quite general, and can be applied to linear derivations as well. The essential point is to break free from a generative straitjacket, relaxing the inference rules for derivations by collecting constraints that are enforced at various points of the derivation, including the end.

References

H. Alshawi and R. Crouch. 1992. Monotonic semantic interpretation. In *Proc. 30th Annual Meeting of the Association for Computational Linguistics.*

J. Carbonell and P. Hayes. 1987. Natural language understanding. In S. Shapiro, D. Eckroth, and G. Vallasi, editors, *Encyclopedia of Artificial Intelligence.* Wiley and Sons, New York.

M. Dalrymple, J. Lamping, F.C.N. Pereira, and V. Saraswat. 1993. LFG semantics via constraints. In *Proc. Sixth European ACL.* University of Utrecht.

M. Dalrymple, V. Gupta, J. Lamping, and V. Saraswat. 1997. Relating resource-based semantics to categorial semantics. Mathematics of Language 5, Saarbrücken.

Kees van Deemter. 1996. Towards a logic of ambiguous expressions. In K. van Deemter and S. Peters, editors, *Semantic Ambiguity and Underspecification.* CSLI Lecture Notes Number 55, Stanford.

Tim Fernando. 1997. Ambiguity under changing contexts. *Linguistics and Philosophy,* 20(6).

Tim Fernando. In press. A modal logic for nondeterministic discourse processing. *Journal of Logic, Language and Information.*

Jean-Yves Girard, Yves Lafont, and Paul Taylor. 1989. *Proofs and Types.* Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press.

Per Martin-Löf. 1984. *Intuitionistic Type Theory.* Bibliopolis, Napoli. Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980.

Glyn V. Morrill. 1994. *Type Logical Grammar.* Kluwer Academic Publishers, Dordrecht.

Aarne Ranta. 1994. *Type-Theoretical Grammar.* Oxford University Press, Oxford.