A Formal Fragment of Hybrid TLCG

A.1 Syntactic, Semantic, and Prosodic Types

Syntactic types are defined as follows:

(634) Atomic types include (at least) NP, N, PP, and S.

(635) Directional types
- a. An atomic type is a directional type.
- b. If A and B are directional types, then (A/B) is a directional type.
- c. If A and B are directional types, then $(B\backslash A)$ is a directional type.
- d. If A and B are directional types, then $(A \vee B)$ is a directional type.
- e. If A and B are directional types, then $(A \wedge B)$ is a directional type.
- f. Nothing else is a directional type.

(636) Syntactic types
- a. A directional type is a syntactic type.
- b. If A and B are syntactic types, then $(A \uparrow B)$ is a syntactic type.
- c. Nothing else is a syntactic type.

We omit outermost parentheses and parentheses for a sequence of the same type of slash, assuming that $/$ and $\uparrow$ are left associative, and $\backslash$ right associative. Thus, $\mathrm{S}/\mathrm{NP}/\mathrm{NP}$, $\mathrm{NP}\backslash\mathrm{NP}\backslash\mathrm{S}$, and $\mathrm{S}\uparrow\mathrm{NP}\uparrow\mathrm{NP}$ are abbreviations of $((\mathrm{S}/\mathrm{NP})/\mathrm{NP})$, $(\mathrm{NP}\backslash(\mathrm{NP}\backslash\mathrm{S}))$, and $((\mathrm{S}\uparrow\mathrm{NP})\uparrow\mathrm{NP})$, respectively.

Note that the algebra of syntactic types is *not* a free algebra generated over the set of atomic types with the three binary connectives $/$, $\backslash$, and $\uparrow$. Specifically, given the definitions in (634)–(636), in Hybrid TLCG, a vertical slash cannot occur "under" a directional slash. Thus, $\mathrm{S}/(\mathrm{S}\uparrow\mathrm{NP})$ is not a well-formed syntactic type. One way to make sense of this restriction is to think of it as a "filter" on uninterpretable prosodic objects. An expression with syntactic type $\mathrm{X}/(\mathrm{Y}\uparrow\mathrm{Z})$ would have to concatenate a string to the left of a functor of type **st→st**, but it does not make sense (at least if one takes the "meanings" of slashes literally) to "concatenate" a string to the left of a *function* from strings to strings since concatenation is an operation defined only on strings.

The functions Sem and Pros, which map syntactic types to semantic and prosodic types, can be defined as follows:

(637) a. For atomic syntactic types,
$\mathsf{Sem}(\mathrm{NP}) = \mathsf{Sem}(\mathrm{PP}) = e$, $\mathsf{Sem}(\mathrm{S}) = t$, $\mathsf{Sem}(\mathrm{N}) = e \rightarrow t$

b. For complex syntactic types,

$\mathsf{Sem}(A/B) = \mathsf{Sem}(B\backslash A) = \mathsf{Sem}(A\uparrow B) = \mathsf{Sem}(B) \to \mathsf{Sem}(A)$
$\mathsf{Sem}(A \vee B) = \mathsf{Sem}(A) = \mathsf{Sem}(B)$
$\mathsf{Sem}(A \wedge B) = \mathsf{Sem}(A) \times \mathsf{Sem}(B)$

(638) a. For any directional type A, $\mathsf{Pros}(A) = \mathbf{st}$ (with **st** for "strings").
b. For any complex syntactic type $A\uparrow B$ involving the vertical slash $\uparrow$, $\mathsf{Pros}(A\uparrow B) = \mathsf{Pros}(B) \to \mathsf{Pros}(A)$.

Note in particular that, corresponding to the asymmetry between $/$, $\backslash$, and $\uparrow$ in the definition of syntactic types, the two types of slashes behave differently in the mapping from syntactic to prosodic types. Specifically, for the mapping from syntactic types to prosodic types, only the vertical slash $\uparrow$ is effectively interpreted as functional. Thus, for example, $\mathsf{Sem}(\mathrm{S}\uparrow(\mathrm{S}/\mathrm{NP})) = (e \to t) \to t$, whereas $\mathsf{Pros}(\mathrm{S}\uparrow(\mathrm{S}/\mathrm{NP})) = \mathbf{st}\to\mathbf{st}$.

The prosodic calculus is a λ-calculus with constants of type **st** (which includes ordinary string prosodies such as john, walks, and so on, and the empty string $\boldsymbol{\varepsilon}$) and a binary connective $\circ$ for string concatenation. In addition to beta equivalence, the following axioms hold in the prosodic calculus:

(639) a. $\boldsymbol{\varepsilon} \circ a \equiv a$ (left identity)
b. $a \circ \boldsymbol{\varepsilon} \equiv a$ (right identity)
(640) $a \circ (b \circ c) \equiv (a \circ b) \circ c$ (associativity)

In the derivations, we omit parentheses for $\circ$ and implicitly assume associativity (except in the multi-modal fragment in chapter 11).

A.2 Sample Lexicon

(641) a. john; **j**; NP
b. mary; **m**; NP
c. walks; **walk**; $\mathrm{NP}\backslash\mathrm{S}$
d. loves; **love**; $(\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP}$
e. $\lambda\sigma.\sigma(\mathsf{everyone})$; $\mathbf{V}_{\mathbf{person}}$; $\mathrm{S}\uparrow(\mathrm{S}\uparrow\mathrm{NP})$
f. $\lambda\varphi\lambda\sigma.\sigma(\mathsf{every} \circ \varphi)$; $\mathbf{V}$; $\mathrm{S}\uparrow(\mathrm{S}\uparrow\mathrm{NP})\uparrow\mathrm{N}$
g. $\lambda\sigma.\sigma(\mathsf{must})$; $\lambda\mathscr{F}.\Box\mathscr{F}(\mathsf{id}_{et})$; $\mathrm{S}\uparrow(\mathrm{S}\uparrow(\mathrm{VP}/\mathrm{VP}))$ (where $\mathsf{id}_{et} = \lambda P_{et}.P$)

For strings in prosodic representations we use sans-serif. Semantic constants are written in **boldface**. For prosodic variables we use Greek letters $\varphi_1, \varphi_2, \ldots$ (type **st** for string). $\sigma_1, \sigma_2, \ldots$ (type **st→st**, **st→st→st**, and so on); $\tau_1, \tau_2, \ldots$ (type **(st→st)→st**, and so on). For semantic variables we use roman italics ($x, y, z, p, q, \ldots, P, Q, R, \ldots$). Calligraphic letters ($\mathcal{U}, \mathcal{V}, \mathcal{W}, \ldots$) are invariably used for variables with polymorphic types. We use copperplate letters ($\mathscr{P}, \mathscr{Q}, \ldots$) for higher-order variables of fixed types.

A.3 Syntactic Rules

A.3.1 Logical Rules

(642)

Connective	Introduction	Elimination
/	$\dfrac{\begin{array}{c}\vdots\ [\varphi; x; A]^n\ \vdots \\ \vdots\qquad\vdots \\ b\circ\varphi;\mathcal{F};B\end{array}}{b;\lambda x.\mathcal{F};B/A}/\mathrm{I}^n$	$\dfrac{a;\mathcal{F};A/B \quad b;\mathcal{G};B}{a\circ b;\mathcal{F}(\mathcal{G});A}/\mathrm{E}$
\	$\dfrac{\begin{array}{c}\vdots\ [\varphi; x; A]^n\ \vdots \\ \vdots\qquad\vdots \\ \varphi\circ b;\mathcal{F};B\end{array}}{b;\lambda x.\mathcal{F};A\backslash B}\backslash\mathrm{I}^n$	$\dfrac{b;\mathcal{G};B \quad a;\mathcal{F};B\backslash A}{b\circ a;\mathcal{F}(\mathcal{G});A}\backslash\mathrm{E}$
↑	$\dfrac{\begin{array}{c}\vdots\ [\varphi; x; A]^n\ \vdots \\ \vdots\qquad\vdots \\ b;\mathcal{F};B\end{array}}{\lambda\varphi.b;\lambda x.\mathcal{F};B\uparrow A}\uparrow\mathrm{I}^n$	$\dfrac{a;\mathcal{F};A\uparrow B \quad b;\mathcal{G};B}{a(b);\mathcal{F}(\mathcal{G});A}\uparrow\mathrm{E}$

All of the three slashes are linear. That is, the three connectives can bind only one occurrence of a hypothesis at a time. Note also that the prosodic labels in Hybrid TLCG are not proof terms but rather are used for the purpose of narrowing down the set of possible derivations. Specifically, the applicability of the Introduction rule for / (\) is conditioned by the presence of the variable φ at the right (left) periphery in the prosody of the premise.

In addition to the rules for the implication connectives above, we assume the following rules for the meet and join connectives:

(643) a. Left Meet Elimination

$$\frac{a;\mathcal{F};A\wedge B}{a;\pi_1(\mathcal{F});A}\wedge\mathrm{E}_l$$

b. Right Meet Elimination

$$\frac{a;\mathcal{F};A\wedge B}{a;\pi_2(\mathcal{F});B}\wedge\mathrm{E}_r$$

(644) a. Left Join Introduction

$$\frac{a;\mathcal{F};B}{a;\mathcal{F};A\vee B}\vee\mathrm{I}_l$$

b. Right Join Introduction

$$\frac{a;\mathcal{F};A}{a;\mathcal{F};A\vee B}\vee\mathrm{I}_r$$

Note that we assume that meet is semantically potent whereas join is semantically nonpotent (see Bayer [1996] for the distinction between semantically potent and nonpotent variants of meet and join).

A.3.2 Nonlogical Rule

(645) P-Interface rule

$$\frac{\varphi_0;\ \mathcal{F};\ A}{\varphi_1;\ \mathcal{F};\ A}\text{PI}$$

——where φ_0 and φ_1 are equivalent terms in the prosodic calculus

The P-Interface rule is the analogue of the structural rules in other variants of TLCG (Morrill 1994; Moortgat 1997). This rule is used for applying beta-reduction to prosodic terms obtained by function application via the $\uparrow$E rule. For example, a more "pedantic" and technically precise (but cumbersome) version of the quantifier scope derivation in (22) from chapter 2 would contain the following proof steps (note the step-by-step β-reduction in the prosodic component mediated by PI):

(646)

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\begin{array}{l}\lambda\sigma.\sigma(\text{someone});\\ \exists_{\textbf{person}};\ \text{S}\uparrow(\text{S}\uparrow\text{NP})\end{array}}\quad\dfrac{\vdots}{\begin{array}{l}\lambda\varphi_2.\varphi_2\circ\text{talked}\circ\text{to}\circ\varphi_1\circ\text{yesterday};\\ \lambda x_2.\textbf{yest}(\textbf{talked-to}(x_1)(x_2));\ \text{S}\uparrow\text{NP}\end{array}}\uparrow\text{E}}{\begin{array}{l}\lambda\sigma.[\sigma(\text{someone})](\lambda\varphi_2.\varphi_2\circ\text{talked}\circ\text{to}\circ\varphi_1\circ\text{yesterday});\\ \exists_{\textbf{person}}(\lambda x_2.\textbf{yest}(\textbf{talked-to}(x_1)(x_2)));\ \text{S}\end{array}}\text{PI}}{\begin{array}{l}\lambda\varphi_2.[\varphi_2\circ\text{talked}\circ\text{to}\circ\varphi_1\circ\text{yesterday}](\text{someone});\\ \exists_{\textbf{person}}(\lambda x_2.\textbf{yest}(\textbf{talked-to}(x_1)(x_2)));\ \text{S}\end{array}}\text{PI}}{\begin{array}{l}\text{someone}\circ\text{talked}\circ\text{to}\circ\varphi_1\circ\text{yesterday};\\ \exists_{\textbf{person}}(\lambda x_2.\textbf{yest}(\textbf{talked-to}(x_1)(x_2)));\ \text{S}\end{array}}$$

$$\vdots$$

In practice, we omit explicitly writing the application of this rule to avoid cluttering the derivations.

One way to understand the (apparent) asymmetry between the $\uparrow$I rule and the /I and \I rules as formulated above as to the explicit presence of λ-binding in the prosodic component in the former but not in the latter two is to think of the /I and \I rules as abbreviations of the following proof steps:

(647)

$$\dfrac{\dfrac{\dfrac{\vdots\ \underline{[\varphi;\ x;\ A]^n}\ \vdots}{\vdots\quad\vdots\quad\vdots}}{b\circ\varphi;\ \mathcal{F};\ B}/\text{I}^n}{\lambda_r\varphi.[b\circ\varphi](\boldsymbol{\varepsilon});\ \lambda x.\mathcal{F};\ B/A}\text{PI}}{b;\ \lambda x.\mathcal{F};\ B/A}$$

That is, there is actually lambda binding (by "left" and "right" lambdas) in the prosody in the /I and \I rules, but the functional lambda terms obtained are immediately applied to the empty string to "close off" the gap. On this view, the /I and \I rules are not so different from the $\uparrow$I rule after all, but the key difference is that, unlike the $\uparrow$I rule, the /I and \I rules are immediately followed by an obligatory application to the empty string, so that a string prosody rather than a functional prosody is obtained.

A.4 The Status of Syntactic Features

In the present monograph, we have assumed the following syntactic features:

(648) a. S:
- $\{bse, fin, pst, ger, \ldots\}$ (for "vform")
- $\{+, -, \varnothing\}pol$

b. NP:
- $\pm p$ ("pronominal")
- $\pm wh$
- $\{nom, acc, dat\}$ (for case)
- $\{pl, sg\}$ (for number)

c. PP:
- $\{to, from, about, \ldots\}$ (for "pform"),

d. (for all atomic types): $\{1, 2, 3, 4, \ldots\}$ (for clause-level index, written superscript)

These features are to be thought of as playing analogous roles to syntactic features in lexicalist frameworks of syntax such as HPSG and LFG. While it is in principle possible to work out a formal theory of syntactic features and feature instantiation/unification along the lines done, for example, in HPSG, we leave the full development of a theory of syntactic features in categorial grammar for future work and instead adopt a simplistic view (described below) for the time being (for a more sophisticated approach dealing with syntactic features via subtyping within the logic itself making use of dependent types, see Morrill [1994] and Pogodalla and Pompigne [2012]).

Specifically, for the fragment in this book, we take all syntactic categories with different subscript (and superscript) notations to be distinct *atomic* categories. Formally, syntactic category notations in which the relevant features are omitted are all metavariable notations (and thus, not real syntactic categories/types) in the lexicon. The metavariable notations range over all the atomic categories that instantiate the omitted feature(s) in one way or another. For example, taking the clause-level index superscript n to range over $\{1, 2, 3\}$, the case feature to range over $\{nom, acc, dat\}$, and three binary features $\pm wh$, $\pm p$, sg/pl, NP is a metavariable notation for $3\times3\times2\times2\times2 = 72$ distinct atomic categories. This may give the misleading impression that our lexicon/grammar lacks a way to handle lexical generalizations, but this is an artificial consequence of leaving out the development of a proper theory of the lexicon for the time being. In syntactic derivations, whenever we omit feature specifications, it should be understood as an abbreviation for some appropriately specified atomic category.

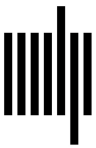
Type-Logical Syntax

By: Yusuke Kubota, Robert D. Levine

Citation:

Type-Logical Syntax
By: Yusuke Kubota, Robert D. Levine
DOI: 10.7551/mitpress/11866.001.0001
ISBN (electronic): 9780262360807
Publisher: The MIT Press
Published: 2020

The open access edition of this book was made possible by generous funding and support from Arcadia – a charitable fund of Lisbet Rausing and Peter Baldwin



The MIT Press

The open access edition of this book was made possible by generous funding from Arcadia—a charitable fund of Lisbet Rausing and Peter Baldwin.

This book was set in Syntax and Times Roman by the authors.

Library of Congress Cataloging-in-Publication Data

Names: Kubota, Yusuke, author. | Levine, Robert, 1947- author.
Title: Type-logical syntax / Yusuke Kubota, Robert D. Levine.
Description: Cambridge, Massachusetts : The MIT Press, [2016] | Includes bibliographical references and index.
Identifiers: LCCN 2020000483 | ISBN 9780262539746 (paperback)
Subjects: LCSH: Categorial grammar. | Grammar, Comparative and general–Syntax.
Classification: LCC P161 .K83 2016 | DDC 415–dc23
LC record available at https://lccn.loc.gov/2020000483

ISBN: 978-0-262-53974-6