Codability and Robustness
in Formal Natural Language Semantics

Kristina Liefke[*]

Ludwig-Maximilians-University Munich[**]
K.Liefke@lmu.de

Abstract According to the received view of type-logical semantics (suggested by Montague and adopted by many of his successors), the correct prediction of entailment relations between lexically complex sentences requires many different types of semantic objects. This paper argues *against* the need for such a rich semantic ontology. In particular, it shows that Partee's temperature puzzle – whose solution is commonly taken to require a basic type for indices or for individual concepts – can be solved in the more parsimonious type system from [11], which only assumes basic individuals and propositions. We generalize this result to show the soundness of the PTQ-fragment in the class of models from [11]. Our findings support the robustness of type-theoretic models w.r.t. their objects' codings.

Keywords: PTQ-fragment, individual concepts, temperature puzzle, entailment-preservation, coding, robustness

In choosing theories, we always invoke a principle of theoretical simplicity or parsimony: given two theories of equal explanatory power, the theory that postulates fewer irreducibly distinct kinds or types of entities is preferable.   [10, p. 51]

1   Introduction

It is a commonplace in ontology engineering that, to model *complex* target systems, we need to assume *many* different types of objects. The semantic ontology of natural language is no exception to this: To interpret a reasonably rich fragment of English, we assume the existence of individuals, propositions, properties of individuals, relations between individuals, situations, events, degrees, times, substances, kinds, and many other types of objects. These objects serve the interpretation of proper names, declarative sentences or complement phrases, common nouns or intransitive verbs, transitive verbs, neutral perception verbs, adverbial or degree modifiers, temporal adjectives, mass terms, bare noun phrases, etc.

Traditional type-logical semantics (esp. [11, 12]) tames this zoo of objects by assuming only a small set of primitive objects and constructing all other types of objects from these primitives via a number of object-forming rules. In this way, Montague reduces the referents of the basic fragment of English from [11] (hereafter, the *EFL-fragment*) to *two* basic types of objects: individuals (type ι) and propositions (analyzed as functions from indices to truth-values, type $\sigma \to t$;[1] abbreviated 'o'). From these objects, properties and binary relations of individuals are constructed as functions from individuals to propositions (type $\iota \to o$), resp. as curried functions from pairs of individuals to propositions (type $\iota \to (\iota \to o)$).

Since Montague's semantics *reduces* the number of basic objects in the linguistic ontology to a small set of primitives, we hereafter refer to this view of formal semantics as the *reduction view*. The latter is characterized below:

Reduction view. *Many (types of) objects in the linguistic ontology* **can be coded** *as constructions out of a few (types of) primitives. The coding relations between objects enable the compositional interpretation of natural language.*

In the last forty years, revisions and extensions of Montague's formal semantics have caused many semanticists to depart from the reduction view. This departure is witnessed by the introduction of a fair number of new basic types (including types for primitive propositions, situations, events, degrees, and times). The introduction of these new types is motivated by the coarse grain of set-theoretic functions (s.t. Montague's semantics sometimes generates wrong predictions about linguistic entailment), and by the need to find semantic values for 'new' kinds of expressions which are not included in Montague's small linguistic fragments. Since many of these new values are either treated as primitives or are identified with constructions out of new primitives, we will hereafter refer to this view of formal semantics as the *ontology engineering view*. This view is captured below:

Ontology engineering view. *Many (types of) objects in the linguistic ontology* **are not coded** *as constructions out of other objects. The compositionality of interpretation is only due to the coding relations between a small subset of objects.*

The ontology engineering view is supported by Montague's semantics from [11] and [12]. The latter interpret the EFL-fragment in models with primitive individuals and propositions, and interpret the fragment of English from [12] (called the *PTQ-fragment*) in models with primitive individuals, indices (i.e. possible worlds, or world-time pairs), and truth-values. Since the PTQ-fragment extends the lexicon of the EFL-fragment via intensional common nouns (e.g. temperature, price) and intensional intransitive verbs (e.g. rise, change), since PTQ-models extend the frames of EFL-models via individual concepts, and since PTQ-models interpret intensional nouns and intransitive verbs as functions *over individual concepts*, it is commonly assumed that any empirically adequate model for the PTQ-fragment requires a basic type for indices, or for individual concepts.

The ontology engineering view is further supported by its practical advantages. In particular, the availability of a larger number of basic types facilitates

---

[1] We follow the computer science-notation for function types. Thus, $\sigma \to t$ corresponds to Montague's type $\langle \sigma, t \rangle$ (or, given Montague's use of the index-type s, to $\langle s, t \rangle$).

work for the empirical linguist: In a rich type system, fewer syntactic expressions are interpreted in a non-basic type.[2] As a result, the compositional translations of many syntactic structures will be *simpler*, and will involve *less* lambda conversions than their 'reductive' counterparts.

However, the proliferation of basic types is not an altogether positive development: Specifically, the interpretation of new kinds of expressions in (constructions out of) *additional* basic types prevents an identification of the relation between the different type-theoretic models. As a result, we cannot check the relative consistency of the different models, transfer the interpretive success of some classes of models to other models, or identify the minimal semantic requirements on models for certain linguistic fragments. These problems are exemplified by the absence of a consequence-preserving translation between the terms of Montague's three-sorted logic IL from [12] (or of Gallin's logic TY_2 from [5]) and the terms of the two-sorted logic underlying [11], by the resulting impossibility of attributing the PTQ-model's solution of Partee's temperature puzzle to EFL-models, and by the related open question whether a primitive type for indices is really *required* for the interpretation of the PTQ-fragment.

This paper defends the reduction view with respect to the interpretation of the PTQ-fragment. In particular, it shows that the PTQ-fragment can be interpreted in EFL-models, which do not have a designated type for indices or individual concepts. This interpretation is sound: the EFL-interpretation of the PTQ-fragment preserves the entailment relation which is imposed on this fragment by its translation into Montague's logic IL. To illustrate this soundness, we show that our EFL-semantics blocks Partee's temperature puzzle from [12].

The plan of the paper is as follows: We first introduce Partee's temperature puzzle for extensional semantics of natural language and present Montague's solution to this puzzle (in Sect. 2). We then identify a strategy for the EFL-coding of indices and truth-values, which allows us to translate the linguistically relevant sublanguage of TY_2 into an EFL-language[3] (in Sect. 3). Finally, we apply this strategy to solve Partee's temperature puzzle (in Sect. 4). The paper closes with a summary of our results and with pointers to future work.

2 Partee's Temperature Puzzle and Montague's Solution

Partee's temperature puzzle [12, pp. 267–268] identifies a problem with extensional semantics for natural language, which regards their validation of the counter-

---

[2] For example, since linguists often assign degree modifiers (e.g. very) the type for degrees δ (rather than the type for properties of individuals, $(\iota \to o) \to o$), gradable adjectives (e.g. tall) are interpreted in the type $\delta \to (\iota \to o)$, rather than in the type $((\iota \to o) \to o) \to (\iota \to o)$.

[3] In [11], Montague uses a *direct interpretation* of natural language into logical models, which does not proceed via the translation of natural language into the language of some logic. As a result, [11] does not identify a logical language with EFL-typed expressions. However, since such a language is easily definable (cf. Def. 7), we hereafter refer to any EFL-typed language as an 'EFL-language'.

intuitive inference from $(\star)$:

$$\frac{\text{The temperature is ninety.}}{\text{The temperature rises.}} \qquad\qquad (\star)$$
$$\text{Ninety rises.}$$

The origin of this problem lies in the different readings of the phrase the tempe-rature in the two premises of $(\star)$, and in the inability of extensional semantics (e.g. [1]) to accommodate one of these readings: In the second premise, the occurrence of the phrase the temperature is intuitively interpreted as a *function* (type $\sigma \to \iota$) from worlds and times (or 'indices', type σ) to the temperature at those worlds at the given times.[4] In the first premise, the occurrence of the temperature is inter-preted as the *value* (type ι) of this function at the actual world at the current ti-me. Since extensional semantics do not have a designated type for indices – such that they also lack a type for index-to-value functions –, they are unable to cap-ture the reading of the phrase the temperature from the second premise.

The inference from the conjunction of the translations of the two premises of $(\star)$ to the translation of the conclusion of $(\star)$ in classical extensional logic is gi-ven below. There, constants and variables are subscripted by their semantic type:

$$\frac{\exists x_\iota \forall y_\iota.(\text{TEMP}_{\iota\to t}(y) \leftrightarrow x = y) \wedge x = ninety_\iota}{\exists x_\iota \forall y_\iota.(\text{TEMP}_{\iota\to t}(y) \leftrightarrow x = y) \wedge \text{RISE}_{\iota\to t}(x)} \qquad (\text{ext-}\star)$$
$$\text{RISE}_{\iota\to t}(ninety_\iota)$$

The asserted identity of the temperature x with the value *ninety* in the first pre-mise of $(\text{ext-}\star)$ justifies the (counterintuitive) substitution of the translation of the temperature from the second premise by the translation of the name ninety.

Montague's semantics from [12] blocks this counterintuitive inference by inte-rpreting intensional common nouns (e.g. temperature) and intransitive verbs (e.g. rise) as (characteristic functions of) sets of *individual concepts* (type $(\boldsymbol{\sigma \to \iota}) \to t$), and by restricting the interpretation of the copula is to a relation between the *extensions* of two individual concepts at the actual world @ at the current time (i.e. to a curried relation between *individuals*, type $\boldsymbol{\iota} \to (\boldsymbol{\iota} \to t)$). Since the first premise of $(\star)$ thus only asserts the identity of the individual 'the temperature **at @**' and the value ninety, it blocks the substitution of the individual concept-denoting phrase the temperature in the second premise of $(\star)$ by the name ninety.

The invalidity of the inference from the conjunction of the two premises of $(\star)$ to the conclusion of $(\star)$ in a streamlined version of Montague's Intensional Logic (cf. [5]) is captured in $(\text{PTQ-}\star)$. There, *ninety* denotes the constant function from indices to the type-ι denotation of the term ninety (s.t. $\forall i_\sigma . ninety(i) = ninety$):

$$\frac{\exists c_{\sigma\to\iota} \forall c_{1\,\sigma\to\iota}.(temp_{(\sigma\to\iota)\to t}(c_1) \leftrightarrow c = c_1) \wedge c(@_\sigma) = ninety_\iota}{\exists c_{\sigma\to\iota} \forall c_{1\,\sigma\to\iota}.(temp_{(\sigma\to\iota)\to t}(c_1) \leftrightarrow c = c_1) \wedge rise_{(\sigma\to\iota)\to t}(c)} \qquad (\text{PTQ-}\star)$$
$$rise_{(\sigma\to\iota)\to t}(ninety_{\sigma\to\iota})$$

_____

[4] As a result, this reading is sometimes called the *function reading* (cf. [8]). The reading of the phrase the temperature from the first premise is called the *value reading*.

Since Montague's EFL-models from [11] only assume basic types for individuals and propositions (s.t. they do not allow the construction of individual concepts), it is commonly assumed that these models are unable to block the inference from $(\star)$. We show below that this assumption is mistaken.

3 Connecting PTQ and EFL

To demonstrate that Montague's models from [11] enable a solution to Partee's temperature puzzle, we first identify a strategy for the EFL-representation of indices and truth-values, which allows us to code every object in the class of models from [12] as an object in the class of models from [11] (in Sect. 3.1). We will see (in Sect. 3.3; cf. Sect. 3.2) that this strategy enables the translation of every linguistically relevant[5] term in a streamlined version of the language from [12] into a term in the language of a logic with basic types ι and o. We then show that our translation avoids the emergence of Partee's temperature puzzle (in Sect. 4).

3.1 Coding PTQ-Objects as EFL-Objects

To enable the translation of Montague's PTQ-translations from [12] into terms of an EFL-typed language, we code indices and truth-values as type-o propositions. This coding is made possible by the interpretation of o as the type for functions from indices to truth-values, such that there are injective maps, $\lambda i_\sigma \lambda j_\sigma.j = i$ and $\lambda \theta_t \lambda i_\sigma.\theta$, from indices and truth-values to propositions. These maps enable the representation of indices via characteristic functions of their singleton sets, and the representation of truth-values via constant functions from indices to these values.

The existence of these maps suggests the replacement of all non-propositional[6] occurrences of the types σ and t and of all occurrences of the type $\sigma \to t$ by the type o. This replacement scheme converts the type for individual concepts into the type[7] $\boldsymbol{o} \to \iota$, and converts the type for (characteristic functions of) sets of individuals into the type $\iota \to \boldsymbol{o}$. The type for sets of individual concepts is then converted into the type $(\boldsymbol{o} \to \iota) \to \boldsymbol{o}$. However, this scheme fails to associate the types of some EFL-expressions from [12] with the ι- and o-based types from [11]. In particular, it associates the PTQ-type of common nouns, $\sigma \to (\iota \to t)$, with the type $\boldsymbol{o} \to (\iota \to \boldsymbol{o})$, rather than with the type of common nouns from [11], $\iota \to \boldsymbol{o}$. But this is undesirable.

---

[5] The latter are logical terms which are associated with PTQ-expressions.

[6] We hereafter use this term for index- and truth-value types which do not occur as a constituent of the propositional type $\sigma \to t$. The need for this distinction is discussed below.

[7] We will see that, since no other syntactic category of the PTQ-fragment receives an interpretation in a construction involving the type $o \to \iota$, semantic types involving $(o \to \iota)$ still motivate the syntactic categories. This contrasts with the type-$(\iota \to t)$ coding of degrees as equivalence classes of individuals (in [2]), which assigns adjectives (originally, type $\delta \to (\iota \to t)$) the type for verbal modifiers, $(\iota \to t) \to (\iota \to t)$.

To ensure the correct conversion to EFL-types, we refine the above replacement scheme into the type-conversion rule from Definition 4. In the definition of this rule, the sets of TY_2 types and of EFL-types[8] (hereafter called 'TY_1 types'), and a TY_2 type's o-normal form are defined as follows:

Definition 1 (TY_2 types). *The set* 2Type *of* TY_2 types *is the smallest set of strings such that* ι, σ, $t \in$ 2Type *and, for all* $\alpha, \beta \in$ 2Type, $(\alpha \to \beta) \in$ 2Type.

Definition 2 (TY_1 types). *The set* 1Type *of* TY_1-types *is the smallest set of strings such that* ι, $o \in$ 1Type *and, for all* $\alpha, \beta \in$ 1Type, $(\alpha \to \beta) \in$ 1Type.

It is clear from the above and from the definition of o as $\sigma \to t$ that all TY_1 types are TY_2 types, but not the other way around. In particular, the TY_2 types σ, t, and constructions out of these types (esp. the types $\sigma \to \iota$, $(\sigma \to \iota) \to t$, and $\sigma \to (\iota \to t)$) are not TY_1 types.

Definition 3 (o-normal form). *An o-normal form, β, of a TY_2 type α is a TY_2 type that has been obtained from α via a unary variant, $^\Diamond$, of the permutation relation* $\spadesuit$ *from* [14, p. 119]. *The former is defined as follows, where* $0 \le n \in \mathbb{N}$:[9]

$$\text{(i)} \qquad \sigma^\Diamond = \sigma; \qquad t^\Diamond = t; \qquad \iota^\Diamond = \iota;$$
$$\text{(ii)} \qquad (\sigma \to (\alpha_1 \to (\ldots \to (\alpha_n \to t))))^\Diamond = \alpha_1^\Diamond \to (\ldots \to (\alpha_n^\Diamond \to (\sigma \to t)));$$
$$\text{(iii)} \quad (\alpha \to \beta)^\Diamond = (\alpha^\Diamond \to \beta^\Diamond), \ \text{if } (\alpha \to \beta) \neq (\sigma \to (\alpha_1 \to (\ldots \to (\alpha_n \to t))))$$

Definition 3 identifies the type for individual-to-proposition functions, $\iota \to (\sigma \to t)$, as the o-normal form of the type for index-to-individual-set functions, $\sigma \to (\iota \to t)$.

The conversion of TY_2 into TY_1 types is defined below:

Definition 4 (Type-conversion). *The relation ξ connects TY_2 types with TY_1 types via the following recursion:*

I. (i) $\quad \xi(\iota) = \iota;$
  (ii) $\quad \xi(\sigma) = \xi(t) = o, \quad$ *when σ resp. t does not occur in $(\sigma \to t)$;*
II. (i) $\xi(\sigma \to t) = o;$
  (ii) $\xi(\alpha \to \beta) = \xi(\gamma \to \delta) = (\xi(\gamma) \to \xi(\delta))$, *where* $(\gamma \to \delta) = (\alpha \to \beta)^\Diamond$
  $\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ *and* $(\alpha \to \beta) \neq (\sigma \to t)$.

Clauses I and II.(i) capture the conversion of basic and propositional TY_2 types. Clause II.(ii) captures the conversion of all other complex TY_2 types. Specifically, the conjunction of this clause with the clauses for the conversion of basic TY_2 types enables the conversion of the type for individual concepts, **$\sigma \to \iota$**, to the type for proposition-to-individual functions, **$o \to \iota$**, and of the type for sets of individuals, $\iota \to \boldsymbol{t}$, to the type for individual-to-proposition functions (i.e. for properties of individuals), $\iota \to \boldsymbol{o}$. The conjunction of clause II.(ii) with the converted TY_2 types for individual concepts and truth-values then enables the conversion of the type for sets of individual concepts, $(\boldsymbol{\sigma} \to \iota) \to \boldsymbol{t}$, to the type for properties of pro-

---

[8] Since n-ary functions can be coded as unary functions (cf. [16]), our definition of TY_1 types neglects n-ary function types, which are assumed in [11].
[9] Following Muskens, we write '$^\Diamond$' in postfix notation, such that '$\alpha^\Diamond$' denotes $^\Diamond(\alpha)$.

position-to-individual functions, $(\boldsymbol{o} \to \iota) \to \boldsymbol{o}$. Since the type $\iota \to (\sigma \to t)$ is the o-normal form of the type $\sigma \to (\iota \to t)$ (cf. clause II.(ii)), the type $\iota \to o$ is the converted type of both parametrized sets of individuals (type $\sigma \to (\iota \to t)$) and of properties of individuals (type $\iota \to (\sigma \to t)$).

Notably, the restriction of clauses I.(ii) and II.(ii) to non-propositional types, resp. to o-normal forms prevents the undesired conversion of the type $\sigma \to t$ into the type for proposition-to-proposition functions $o \to o$, and of the type for parametrized sets of type-α objects, $\boldsymbol{\sigma} \to (\alpha \to \boldsymbol{t})$, to the type for proposition-to-property functions. The converted TY_2 types for all classes of expressions from the PTQ-fragment are listed in Table 1:[10]

CAT'Y	$\alpha \in 2\mathsf{Type}$	$\xi(\alpha)$	CAT'Y	$\alpha \in 2\mathsf{Type}$	$\xi(\alpha) \in 1\mathsf{Type}$
Name	ι	ι	NP	$\sigma \to ((\sigma \to (\iota \to t)) \to t)$	$(\iota \to o) \to o$
S	$\sigma \to t$	o	SCV	$(\sigma \to t) \to (\iota \to t)$	[6] $o \to (\iota \to o)$
C, SAV	$(\sigma \to t) \to t$	$o \to o$	ADV	$(\sigma \to (\iota \to t)) \to (\iota \to t)$	$(\iota \to o) \to (\iota \to o)$
CN, IV	$\sigma \to (\iota \to t)$	$\iota \to o$ [2,3]	CN, IV	$\sigma \to ((\sigma \to \iota) \to t)$	$(o \to \iota) \to o$
TV	$\sigma \to (\iota \to (\iota \to t))$	$\iota \to (\iota \to o)$	ICV	$(\sigma \to (\iota \to t)) \to (\iota \to t)$	$(\iota \to o) \to (\iota \to o)$
TV [5]	$(\sigma \to ((\sigma \to (\iota \to t)) \to t)) \to (\sigma \to (\iota \to t))$				$((\iota \to o) \to o) \to (\iota \to o)$
DET	$(\sigma \to (\iota \to t)) \to ((\sigma \to (\iota \to t)) \to t)$				$(\iota \to o) \to ((\iota \to o) \to o)$
P	[8] $\iota \to ((\sigma \to (\iota \to t)) \to (\iota \to t))$				$\iota \to ((\iota \to o) \to (\iota \to o))$
P	$(\sigma \to (\sigma \to (\iota \to t)) \to t) \to ((\sigma \to (\iota \to t)) \to (\iota \to t))$				
					$((\iota \to o) \to o) \to ((\iota \to o) \to (\iota \to o))$

Table 1. TY_2 and converted TY_2 (i.e. TY_1) types of PTQ-expressions.

This completes our discussion of the TY_1-coding of TY_2 types. To show the possibility of interpreting the PTQ-fragment in TY_1 models, we next describe the class of languages of the logics TY_2 and TY_1 (in Sect. 3.2), and translate all linguistically relevant TY_2 terms into terms of the logic TY_1 (in Sect. 3.3). We then observe that this translation is entailment-preserving.

3.2 The Languages of TY_2 and TY_1

The languages of the logics TY_2 and TY_1 are defined as countable sets $\cup_{\alpha \in 2\mathsf{Type}} L_\alpha$, resp. $\cup_{\beta \in 1\mathsf{Type}} L_\beta$, of uniquely typed non-logical constants. For every TY_2 type α and TY_1 type β, we further assume a countable set $\mathcal{V}_\alpha^2$, resp. $\mathcal{V}_\beta^1$ of uniquely typed variables, with '$\cup_{\alpha \in 2\mathsf{Type}} \mathcal{V}_\alpha^2$' abbreviated as '$\mathcal{V}^2$' and '$\cup_{\beta \in 1\mathsf{Type}} \mathcal{V}_\beta$' abbreviated as '$\mathcal{V}^1$'. From these basic expressions, we form complex terms inductively with the help of functional application, lambda abstraction, and the constants for *falsum*, $\bot$, and logical implication, $\to$.

In the definition of TY_2 terms, the set CoType of conjoinable TY_2 types is defined as follows (cf. [15]):

---

[10] These type-assignments incorporate a finer-grained (type-$\sigma \to t$) interpretation of declarative sentences, the type-ι interpretation of names, and the meaning postulates from [12, pp. 263–264]. The latter are given in square brackets.

Definition 5 (Conjoinable TY_2 types). *The set* CoType *of conjoinable types of the logic* TY_2 *is the smallest set of strings such that, if* $\alpha_1, \ldots, \alpha_n \in$ 2Type, *then* $\alpha_1 \to (\ldots \to (\alpha_n \to t)) \in$ CoType, *where* $0 \leq n \in \mathbb{N}$.

According to the above, a TY_2 term has a conjoinable type if its type is either the truth-value type t or a construction to the type t (via the rule from Def. 1). In these two cases, we say that the term *is conjoinable*.

Definition 6 (TY_2 terms). *Let* $\alpha, \beta \in$ 2Type, *and let* $\epsilon \in$ CoType. *The set* T_α^2 *of* TY_2 *terms of type* α *is then defined as follows:*

(i) $L_\alpha^2, \mathcal{V}_\alpha^2 \subseteq T_\alpha^2$, $\bot \in T_t^2$;
(ii) *If* $B \in T_{\alpha \to \beta}^2$ *and* $A \in T_\alpha^2$, *then* $(B(A)) \in T_\beta^2$;
(iii) *If* $A \in T_\beta^2$ *and* $x \in \mathcal{V}_\alpha^2$, *then* $(\lambda x. A) \in T_{\alpha \to \beta}^2$;
(iv) *If* $B, C \in T_\epsilon$, *then* $(B \Rightarrow C) \in T_t^2$.

Clause (i) identifies all members of L_α^2 and $\mathcal{V}_\alpha^2$ as TY_2 terms. Clauses (ii) and (iii) identify the results of application and abstraction as TY_2 terms. Clause (iv) specifies the formation of complex TY_2 terms. From $\bot$ and $\to$, the familiar TY_2 connectives and quantifiers are standardly obtained (cf. [7]).

We next define the terms of the logic TY_1. Notably, since TY_1 does not have a type for truth-values, the TY_2 constants $\bot$ (type t) and $\to$ (type $\epsilon \to (\epsilon \to t)$) are not available in TY_1. The non-logical constants $\bigcirc$ (type o) and $\dot{\to}$ (type $\varepsilon \to (\varepsilon \to o)$) serve as their single-type stand-ins, where ε is in the proper subset, Prop-Type $= \{\alpha_1 \to (\ldots \to (\alpha_n \to o)) \,|\, \alpha_1, \ldots, \alpha_n \in$ 1Type$\}$, of the set of conjoinable TY_2 types.[11] We hereafter call members of this set *propositional types*.

Definition 7 (TY_1 terms). *Let* $\alpha, \beta \in$ 1Type, *and let* $\varepsilon \in$ PropType. *The set* T_α^1 *of* TY_1 *terms of type* α *is then defined as follows:*

(i) $L_\alpha^1, \mathcal{V}_\alpha^1 \subseteq T_\alpha^1$, $\bigcirc \in T_o^1$;
(ii) *If* $B \in T_{\alpha \to \beta}^1$ *and* $A \in T_\alpha^1$, *then* $(B(A)) \in T_\beta^1$;
(iii) *If* $A \in T_\beta^1$ *and* $x \in \mathcal{V}_\alpha^1$, *then* $(\lambda x. A) \in T_{\alpha \to \beta}^1$;
(iv) *If* $B, C \in T_\varepsilon^1$, *then* $(B \dot{\to} C) \in T_o^1$.

Notably, the typing[12] of $\dot{\to}$, B, and C, suggests that the term $B \dot{\to} C$ from clause (iv) be instead written as '$\dot{\to}(B)(C)$'. Our use of infix notation for $\dot{\to}$ (and similarly, for the TY_1 proxies of all other logical TY_2 constants) is intended to remind the reader of their emulated logical role (cf. Def. 8).

---

[11] [17] and [4] use a similar strategy for the introduction of intensional connectives.

[12] Since we only stipulate that $\varepsilon \in$ PropType, clause (iv) describes $\dot{\to}$ as a non-uniquely typed constant, which applies to pairs of arguments of all propositional TY_1 types. To avoid an extension of the TY_1 type system via polymorphic types, we assume a *schematic* (or *abbreviatory*) *polymorphism* of types. The latter is a syntactic device whereby a metatheoretical symbol is used to abbreviate a range of (monomorphic) types. Thus, in (iv), ε may be instantiated by any of the elements in PropType. The constant $\dot{\to}$ then represents a family, $\{\dot{\to}_{\varepsilon \to (\varepsilon \to o)} \,|\, \varepsilon \in$ PropType$\}$, of *distinct* identical-looking constants, one for each type.

From $\bigcirc\!\!\!\perp$ and $\dot{\rightarrow}$, the TY_1 proxies of the other truth-functional connectives and quantifiers are easily obtained. In particular, the TY_1 proxies for the logical constants $\top, \forall, =, \neg, \wedge$, and $\square$ (i.e. $\dot{\top}, \bigwedge, \dot{=}, \vdash, \wedge$, and $\boxdot$) are obtained by variants of the definitions from [7]. Below, we let A, x and y, X (or B, C), and Y be variables (resp. constants) of the type o, α, $\alpha \to o$, resp. $(\alpha \to o) \to o$, where $\alpha \in$ 1Type:

Notation 1 *We write*

$$
\begin{array}{llll}
\dot{\top} & for & (\bigcirc\!\!\!\perp \dot{\rightarrow} \bigcirc\!\!\!\perp); & \qquad (\bigwedge x.\,A) \quad for \quad ((\lambda x.\dot{\top}) \dot{\rightarrow} (\lambda x.\,A)); \\
B \dot{=} C & for & (\bigwedge Y.\,Y(B) \dot{\rightarrow} Y(C)); & \qquad \vdash B \quad for \quad (\lambda x.\,B(x) \dot{=} \bigcirc\!\!\!\perp); \\
(B \wedge C) & for & (\lambda x.(\lambda X.\,X(B \dot{=} C)) \dot{=} (\lambda X.\,X(\dot{\top}))) \\
(B \vee C) & for & \vdash(\vdash B \wedge \vdash C); & \qquad \boxdot A \quad for \quad (\bigwedge x.x \dot{=} \bigcirc\!\!\!\perp \vee (x \dot{\rightarrow} A))
\end{array}
$$

The TY_1 stand-ins, $\neq, \dot{\leftrightarrow}, \bigvee$, and $\diamondsuit$, of the familiar symbols for inequality, (material) biimplication, the existential quantifier, and the modal diamond operator have their expected definitions.

The behavior of $\bigcirc\!\!\!\perp$, $\dot{\rightarrow}$, and of the defined constants from Notation 1 is governed by the constraints from Definition 8:

Definition 8 (Constraints on L^1-constants). *The interpretations of the TY_1 constants $\bigcirc\!\!\!\perp$ and $\dot{\rightarrow}$ obey the following semantic constraints:*[13]

$$
\text{(C1)} \quad \bigcirc\!\!\!\perp \;=\; (\lambda i_\sigma.\bot); \qquad \text{(C2)} \quad (B \dot{\rightarrow} C) \;=\; (\lambda i_\sigma \forall x.\,B(x)(i) \to C(x)(i))
$$

The constraints (C1) and (C2) define the designated TY_1 constants $\bigcirc\!\!\!\perp$ and $\dot{\rightarrow}$ as the results of lifting the TY_2 connectives $\bot$ and $\to$ to terms of the logic TY_1.[14] In particular, (C1) defines the constant $\bigcirc\!\!\!\perp$ as the designator of the constant function from indices to *falsum*. From (C1), (C2), and Notation 1, the remaining designated TY_1 constants are easily defined. Since the TY_1 constants $\wedge$, $\vee$, and $\vdash$ are η-equivalent to their TY_2 definitions, we hereafter use instead the familiar connectives $\wedge$, $\vee$, and $\neg$.

3.3 Translating $\mathcal{L}^{\mathbf{TY_2}}$ into $\mathcal{L}^{\mathbf{TY_1}}$

To prepare the TY_1 translation of the PTQ-fragment, we next introduce the particular TY_2 and TY_1 languages, $\mathcal{L}^2$ and $\mathcal{L}^1$, whose constants are associated with the lexical elements of the PTQ-fragment. Following a streamlined presentation of Montague's PTQ-to-IL (or TY_2) translation from [12] (cf. [5]), we then identify a relation between $\mathcal{L}^2$- and $\mathcal{L}^1$-terms.

Tables 2 and 3 (next page) contain the non-logical constants of the designated languages $\mathcal{L}^2$, resp. $\mathcal{L}^1$. The small grey tables introduce our notational conventions for variables. In the tables, brackets contain the relevant meaning postulates from [12], resp. the constants' interpretive domains from [11]. We will abbreviate x_1, x_2, and x_3 as 'x', 'y', resp. 'z', abbreviate i_1, i_2, and i_3 as 'i', 'j', resp. 'k', and abbreviate p_1, p_2, and p_3 and c_1, P_1 and Q_1 as 'p', 'q', resp. 'r', and 'c', 'P', resp. 'Q'.

---

[13] These constraints are formulated in the TY_1 metatheory, TY_2 (cf. Sect. 3.3).

[14] This is reminiscent of the translation of dynamic to typed terms from [13, p. 9].

CONSTANT	TY$_2$ TYPE		VAR.	TY$_2$ TYPE
@ $\quad \sigma \quad \mid \quad \theta \quad t \quad \mid \quad$ *ninety*	$(\sigma \to \iota)$		$i_1, \ldots, i_n$	σ
john, mary, bill, ninety	ι		$x_1, \ldots, x_n$	ι
man, woman, park, fish, pen, unicorn	$\sigma \to (\iota \to t)$	[2]	$c_1, \ldots, c_n$	$\sigma \to \iota$
run, walk, talk	$\sigma \to (\iota \to t)$	[3]	$p_1, \ldots, p_n$	$\sigma \to t$
temp, price, rise, change	$\sigma \to ((\sigma \to \iota) \to t)$		$P_1, \ldots, P_n$	$\sigma \to (\iota \to t)$
find, lose, eat, love, date	$\sigma \to (\iota \to (\iota \to t))$	[4]	$\vec{T}$	$\sigma \to ((\sigma \to \iota) \to t)$
believe, assert	$(\sigma \to t) \to (\iota \to t)$	[6]	$\vec{Q}$	$\sigma \to ((\sigma \to (\iota \to t)) \to t)$
seek, conceive	$(\sigma \to ((\sigma \to (\iota \to t)) \to t)) \to (\iota \to t)$	[5]		
rapidly, slowly, ..., allegedly, try, wish	$(\sigma \to (\iota \to t)) \to (\iota \to t)$	[7]		
in	$\iota \to ((\sigma \to (\iota \to t)) \to (\iota \to t))$	[8]		
about	$(\sigma \to (\sigma \to (\iota \to t)) \to t) \to ((\sigma \to (\iota \to t)) \to (\iota \to t))$			
All TY$_1$ constants from Table 3 are members of $\mathcal{L}^2$.				

Table 2. $\mathcal{L}^2$ constants and variables.

CONSTANT	TY$_1$ TYPE		VAR.	TY$_1$ TYPE
$\textcircled{\top}, \textcircled{\bot}, \textbf{@} \quad o \quad \mid \quad \bigwedge, \bigvee$	$(\alpha \to o) \to o$		$x_1, \ldots, x_n \quad \iota \quad \mid \quad \vec{p} \quad o$	
$\Box, \Diamond \quad\quad o \to o \quad \mid \quad \dot{\to}, \dot{=}, \not=, \dot{\leftrightarrow}$	$\alpha \to (\alpha \to o)$		$c_1, \ldots, c_n$	$o \to \iota$
john, mary, bill, ninety	ι	[U_0]	$P_1, \ldots, P_n$	$\iota \to o$
man, *woman*, **park**, *fish*, **pen**, *unicorn*	$\iota \to o$	[U_4]	$T_1, \ldots, T_n$	$(o \to \iota) \to o$
run, *walk*, **talk**	$\iota \to o$	[U_2]	$Q_1, \ldots, Q_n$	$(\iota \to o) \to o$
temp, *price*, **rise**, *change*	$(o \to \iota) \to o$			
find, *lose*, **eat**, *love*, **date**	$\iota \to (\iota \to o)$	[curried U_3]		
believe, *assert*	$o \to (\iota \to o)$	[U_5]		
seek, *conceive*	$((\iota \to o) \to o) \to (\iota \to o)$			
rapidly, *slowly*, ..., **allegedly**, *try*, **wish**	$(\iota \to o) \to (\iota \to o)$	[U_6]		
in	$\iota \to ((\iota \to o) \to (\iota \to o))$	[U_6]		
ninety $\quad (o \to \iota) \quad \mid \quad$ **about**	$((\iota \to o) \to o) \to ((\iota \to o) \to (\iota \to o))$			

Table 3. $\mathcal{L}^1$ constants and variables.

We denote the sets of TY$_2$ and TY$_1$ terms which are obtained from $\mathcal{L}^2$ and $\mathcal{L}^1$ via the operations from Definition 6 and 7 by '$\mathcal{T}^2$', resp. '$\mathcal{T}^1$'.

Note that $\mathcal{L}^1$ adopts the individual constants, *john*, *mary*, *bill*, and *ninety*, of the language $\mathcal{L}^2$. To connect the designated languages of the logics TY$_2$ and TY$_1$, we further assume that each term from $\mathcal{L}^1$ is also a member of $\mathcal{L}^2$ (s.t. $\mathcal{L}^1$ is a sublanguage of $\mathcal{L}^2$), and that the designated TY$_2$ frame $\mathcal{F}^2$ and interpretation function $\mathcal{I}_{\mathcal{F}^2}$ embed the designated frame $\mathcal{F}^1$ and interpretation function $\mathcal{I}_{\mathcal{F}^1}$ of the logic TY$_1$, such that $\mathcal{F}^1 = \mathcal{F}^2{\restriction}_{\text{1Type}}$ and $\mathcal{I}_{\mathcal{F}^1} = \mathcal{I}_{\mathcal{F}^2{\restriction}_{\text{1Type}}}$.

We next give a streamlined presentation of Montague's PTQ-to-IL (or TY$_2$) translation from [12] (cf. [5]):

We identify Logical Form (LF) with the component of syntactic representation which is interpreted in TY$_2$ models. Logical forms are translated into TY$_2$ terms via the process of type-driven translation (cf. [9]). The latter proceeds in two steps, by first defining the translations of lexical elements, and then defining the translations of non-lexical elements compositionally from the translation of their constituents.

Definition 9 (Basic TY_2 translations). *The base rule of type-driven transla-tion translates the lexical PTQ-elements*[15] *into the following TY_2 terms, where $X_1, \ldots, X_n$, R, and R_1 are TY_2 variables of the types $\alpha_1, \ldots, \alpha_n$, resp. $\alpha_1 \to (\ldots \to (\alpha_n \to t))$, and where t_n is the trace of a moved constituent in a logical form that is translated as a free variable:*

John $\rightsquigarrow john$;	man $\rightsquigarrow man$;	walks $\rightsquigarrow walk$;	who $\rightsquigarrow \lambda P.P$;
ninety $\rightsquigarrow ninety$;	temp. $\rightsquigarrow temp$;	rise $\rightsquigarrow rise$;	finds $\rightsquigarrow find$;
seeks $\rightsquigarrow seek$;	tries to $\rightsquigarrow try$;	about $\rightsquigarrow about$;	in $\rightsquigarrow in$;
that $\rightsquigarrow \lambda\theta\lambda i.\theta$;	believes $\rightsquigarrow believe$;	asserts $\rightsquigarrow \lambda p \lambda x.\, assert\,(p)(x) \wedge p$;

rapidly $\rightsquigarrow \lambda P \lambda x.\, rapidly\,(@)(P)(x) \wedge P(x)$; allegedly $\rightsquigarrow allegedly$;

and $\rightsquigarrow \lambda R_1 \lambda R \lambda \vec{X}.R(\vec{X}) \wedge R_1(\vec{X})$; necessarily $\rightsquigarrow \lambda p.\Box p(@)$;

a $\rightsquigarrow \lambda P_2 \lambda P \lambda i\, \exists x.\, P_2(i)(x) \wedge P(i)(x)$; $\quad t_n/\mathsf{he}_n \rightsquigarrow x_n$ *for each* $n \in \mathbb{N}$;

the $\rightsquigarrow \lambda P_2 \lambda P \lambda i\, \exists x \forall y.\, (P_2(i)(y) \leftrightarrow x = y) \wedge P(i)(x)$; $\quad$ is $\rightsquigarrow \lambda y \lambda x.x = y$

On the basis of the above, we define the relation between $\mathcal{T}^2$ and $\mathcal{T}^1$ terms as follows:

Definition 10 (Embedding TY_2 in TY_1). *The relation $^\bullet$ connects the desig-nated terms of the logic TY_2 with terms of the logic TY_1, such that*

I. (i) $john^\bullet = john$; $\quad man^\bullet = \boldsymbol{man}$; $\quad walk^\bullet = \boldsymbol{walk}$; $temp^\bullet = \boldsymbol{temp}$;
$\quad\quad ninety^\bullet = ninety$; $\quad ninety^\bullet = \boldsymbol{ninety}$; $\quad rise^\bullet = \boldsymbol{rise}$; $\quad find^\bullet = \boldsymbol{find}$;
$\quad\quad seek^\bullet = \boldsymbol{seek}$; $\quad allegedly^\bullet = \boldsymbol{allegedly}$; $\quad about^\bullet = \boldsymbol{about}$; $\quad in^\bullet = \boldsymbol{in}$;
$\quad\quad believe^\bullet = \boldsymbol{believe}$; $\quad assert^\bullet = \boldsymbol{assert}$; $\quad rapidly^\bullet = \boldsymbol{rapidly}$; $try^\bullet = \boldsymbol{try}$;

(ii) $\quad x_k^\bullet = x_k$ $\quad$ *for* $1 \le k \in \mathbb{N}$; $\quad\quad\quad\quad$ *(i)* $@^\bullet = \boldsymbol{@}$;
$\quad\quad\quad c_k^\bullet = \boldsymbol{c}_k$ $\quad$ *for* $1 \le k \in \mathbb{N}$; $\quad\quad\quad\quad\quad\quad p_k^\bullet = p_k$ $\quad$ *for* $1 \le k \in \mathbb{N}$;
$\quad\quad\quad P_k^\bullet = \boldsymbol{P}_k$ $\quad$ *for* $1 \le k \in \mathbb{N}$; $\quad\quad\quad\quad\quad\quad i_k^\bullet = p_k$ $\quad$ *for* $1 \le k \in \mathbb{N}$;
$\quad\quad\quad \theta^\bullet = p_1$; $\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad T_k^\bullet = \boldsymbol{T}_k$ $\quad$ *for* $1 \le k \in \mathbb{N}$;

II. (i) $(B_{\sigma\to\beta}(A_\sigma))^\bullet = B^\bullet$ $\quad\quad$ *if* $\beta = t$ *or* $\beta = (\gamma \to t)$, *where* $\gamma \in 2\mathsf{Type}$;
$\quad\quad\quad (B_{\alpha\to\beta}(A_\alpha))^\bullet = (B^\bullet(A^\bullet))$ $\quad$ *otherwise*;

(ii) $\quad (\lambda x_\sigma.A_\beta)^\bullet = (A[x := @])^\bullet$ *if* $\beta = t$ *or* $\beta = (\gamma \to t)$, *where* $\gamma \in 2\mathsf{Type}$;
$\quad\quad\quad (\lambda x_\alpha.A_\beta)^\bullet = (\lambda x^\bullet.A^\bullet)$ $\quad\quad$ *otherwise, if* $(\alpha \to \beta) = (\alpha \to \beta)^\diamond$;
$\quad\quad\quad (\lambda x_\alpha.A_\beta)^\bullet = (\lambda X_\gamma^\bullet(\lambda x.A(X))^\bullet)$ *otherwise, granted* $\beta := (\gamma \to \delta)$, *w.*
$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad X$ *the 1st variable that doesn't occur free in A;*

(iii) $\quad\quad\quad\quad\quad \perp^\bullet = \oslash$; $\quad\quad (B \to C)^\bullet = (B^\bullet \overset{\cdot}{\to} C^\bullet)$

In the first item from II.(ii), '$A[x := @]$' denotes the replacement of all occurren-ces of x in A by '$@$'.

The translation rules from Definition 10 respect the behavior of the type con-verter ξ from Definition 4. Thus, the relation $^\bullet$ translates individual and proposi-tional TY_2 terms (e.g. *ninety*, p_k) into themselves, translates TY_2 terms for indi-vidual concepts (e.g. c_k) into TY_1 terms for proposition-to-individual functions, and translates TY_2 terms for functions from indices to sets of individuals (or of individual concepts; e.g. *man*, *temp*) into TY_1 terms for functions from propositi-

---

[15] For reasons of space, we only translate some representative elements. Expressions of the same (sub-)category receive an analogous translation.

ons to properties of individuals (resp. to properties of proposition-to-individual functions).

The translation rules from clause II ensure the correct translation of complex TY_2 terms: Specifically, the rules for the TY_1 translation of $\bot$ and $\to$ (cf. clause II.(iii)) associate the logical TY_2 constants with their propositional correspondents from TY_1. From the translations in clause I, the rules for application and abstraction (clause II.(i), (ii)) enable the compositional translation of all PTQ-translations from Definition 9. In these rules, the contraints on abstraction block the undesired translation of type-$(\sigma \to t)$ terms as TY_1 terms of the type $o \to o$. The constraints on application enable the translation of the result of applying a type-$(\sigma \to t)$ (or type-$(\sigma \to (\gamma \to t))$) term to a type-$\sigma$ term.

The translations of some example TY_2 terms are given below. In these translations, the TY_1 correlates of logical TY_2 constants other than $\bot$ and $\to$ are obtained from the TY_1 translations of $\bot$ and $\to$ via the definitions of the remaining logical constants from [7]. In particular, the TY_2 constant $\bot$ and terms of the form $\forall x.A$ are translated as follows:

$$
\begin{aligned}
\top^\bullet &= (\bot \to \bot)^\bullet = ((\bot^\bullet) \,\dot\to\, (\bot^\bullet)) \quad \text{(by [7]; II.(iii))} \\
&= (\textcircled{\bot} \,\dot\to\, \textcircled{\bot}) \;=\; \textcircled{\top} \qquad\qquad \text{(by II.(iii); Nota. 1)}
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
(\forall x.A)^\bullet &= ((\lambda x.\top) \to (\lambda x.A))^\bullet \;=\; ((\lambda x.\top)^\bullet \,\dot\to\, (\lambda x.A)^\bullet) \quad \text{(by [7]; II.(iii))} \\
&= ((\lambda x^\bullet.\textcircled{\top}) \,\dot\to\, (\lambda x^\bullet.A^\bullet)) \;=\; (\bigwedge x^\bullet.A^\bullet) \qquad \text{(by II.(ii), (iii); Nota. 1)}
\end{aligned} \tag{2}
$$

The translations of $\exists$, $=$, $\wedge$, and $\leftrightarrow$ are analogously obtained, such that

$$
\begin{aligned}
(\exists x.A)^\bullet &= (\bigvee x^\bullet.A^\bullet); & (B = C)^\bullet &= B^\bullet \,\dot=\, C^\bullet; \\
(B \wedge C)^\bullet &= (B^\bullet \wedge C^\bullet); & (B \leftrightarrow C)^\bullet &= (B^\bullet \,\dot\leftrightarrow\, C^\bullet).
\end{aligned}
$$

From the above translations, the translations of the copula is and of the determiner the are obtained thus:

$$
\text{is} \;\rightsquigarrow\; (\lambda y \lambda x.x = y)^\bullet = (\lambda y^\bullet \lambda x^\bullet.x^\bullet \,\dot=\, y^\bullet) = (\lambda y \lambda x.x \,\dot=\, y) \quad \text{(by II.(ii); I.(ii))} \tag{3}
$$

$$
\begin{aligned}
\text{the} \;\rightsquigarrow\; &(\lambda P_2 \lambda P \lambda i\, \exists x \forall y.(P_2(i)(y) \leftrightarrow x = y) \wedge P(i)(x))^\bullet \\
&= \lambda P_2^\bullet \lambda P^\bullet (\lambda i \exists x \forall y.(P_2(i)(y) \leftrightarrow x = y) \wedge P(i)(x))^\bullet \quad \text{(by II.(ii), double)} \\
&= \lambda P_2^\bullet \lambda P^\bullet (\exists x \forall y.(P_2(@)(y) \leftrightarrow x = y) \wedge P(@)(x))^\bullet \quad \text{(by II.(i))} \\
&= \lambda \boldsymbol{P_2} \lambda \boldsymbol{P} \bigvee x \bigwedge y.(\boldsymbol{P_2}(y) \,\dot\leftrightarrow\, x \,\dot=\, y) \wedge \boldsymbol{P}(x) \quad \text{(by I.(ii), [7]; all of II.)}
\end{aligned} \tag{4}
$$

Since the relation $^\bullet$ respects the structure of each TY_2 term from Definition 9 (cf. [12]), the interpretation of the PTQ-fragment in the class of designated TY_1 models preserves the entailment relation which is imposed on this fragment by its translation into the logic TY_2. This observation is captured below:

Theorem 1 (Soundness of translation). *Let Γ and Δ, and $\Gamma^\bullet := \{\gamma^\bullet \mid \gamma \in \mathcal{T}^2\}$ and $\Delta^\bullet := \{\delta^\bullet \mid \delta \in \mathcal{T}^2\}$ be sets of designated TY_2 formulas and their TY_1 translations. Then,* $\quad \Gamma^\bullet \vdash_{TY_1} \Delta^\bullet \quad iff \quad \Gamma \vdash_{TY_2} \Delta.$

In the above case, we say that the TY_2-to-TY_1 translation is sound.

Proof. The proof relies on the definition of $^\bullet$ and on the proof theories of TY_2 and TY_1.

4 Solving the Temperature Puzzle in EFL

To illustrate Theorem 1, we next show that the TY_1(-via-TY_2) translation of the PTQ-fragment blocks Partee's temperature puzzle. Since we use the strategy of "try[ing] simplest types first" (cf. Tbl. 1–2, Def. 9), the application of the TY_2 (or TY_1) translations of intensional expressions to the translations of other PTQ-expressions needs to be handled through type-shifting. In particular, to apply[16] the TY_2 translations of determiners (e.g. the; type $\underline{(\sigma \to (\iota \to t))} \to ((\sigma \to (\iota \to t)) \to t))$ to the TY_2 translations of intensional common nouns (temperature; type $\sigma \to ((\boldsymbol{\sigma \to \iota}) \to t))$, we introduce the *extensionalization* operator *ext*. This operator sends the designators of functions from indices to sets of *individual concepts* (type $\sigma \to ((\boldsymbol{\sigma \to \iota}) \to t))$ to the designators of functions from indices to sets of *individuals* (type $\sigma \to (\iota \to t)$).

Definition 11 (Extensionalization). *The function* $ext := \lambda T \lambda i \lambda x \exists c. T(i)(c) \wedge x = c(@)$ *sends type-*$(\sigma \to ((\sigma \to \iota) \to t))$ *terms to type-*$(\sigma \to (\iota \to t))$ *terms.*

The operator *ext* enables the 'extensionalization' of the TY_2 translation, *temp*, of the noun temperature to the TY_2 term $\lambda i \lambda x \exists c. temp(i)(c) \wedge x = c(@)$. This term denotes a function from indices to the set of individuals whose members are identical to the result of applying some type-$(\sigma \to \iota)$ witness, c, of the property denoted by *temp* to the current index.

The possibility of interpreting intensional nouns in the type $\sigma \to (\iota \to t)$ enables the TY_1 translation of the first premise from $(\star)$:

1. $[_{\mathrm{NP}}\text{ninety}] \rightsquigarrow ninety^{\bullet} = ninety$ \hfill (5)

2. $[_{\mathrm{TV}}\text{is}] \rightsquigarrow (\lambda y \lambda x. x = y)^{\bullet} = (\lambda y \lambda x. x \doteq y)$

3. $[_{\mathrm{VP}}[\text{is}][\text{ninety}]] \rightsquigarrow (\lambda x. x = ninety)^{\bullet}$
   $= (\lambda y \lambda x. x = y)^{\bullet}[ninety^{\bullet}] = (\lambda y \lambda x. x \doteq y)[ninety] = (\lambda x. x \doteq ninety)$

4. $[_{\mathrm{N}}\text{temperature}] \rightsquigarrow (ext(temp))^{\bullet} = (\lambda i \lambda x \exists c. temp(i)(c) \wedge x = c(@))^{\bullet}$
   $= (\lambda x \exists c. temp(@)(c) \wedge x = c(@))^{\bullet} = (\lambda x^{\bullet} \bigvee c^{\bullet}. temp^{\bullet}(c^{\bullet}) \wedge x^{\bullet} \doteq c^{\bullet}(@^{\bullet}))$
   $= \lambda x \bigvee \boldsymbol{c}. \boldsymbol{temp}(\boldsymbol{c}) \wedge x \doteq \boldsymbol{c}(\boldsymbol{@})$

5. $[_{\mathrm{DET}}\text{the}] \rightsquigarrow (\lambda P_2 \lambda P \lambda i \exists x \forall y. (P_2(i)(y) \leftrightarrow x = y) \wedge P(i)(x))^{\bullet}$
   $= \lambda \boldsymbol{P_2} \lambda \boldsymbol{P} \bigvee x \bigwedge y. (\boldsymbol{P_2}(y) \leftrightarrow x \doteq y) \wedge \boldsymbol{P}(x)$

6. $[_{\mathrm{NP}}[_{\mathrm{DET}}\text{the}][_{\mathrm{N}}\text{temperature}]]$
   $\rightsquigarrow (\lambda P \lambda i \exists x \forall y. ((\exists c. temp(i)(c) \wedge y = c(@)) \leftrightarrow x = y) \wedge P(i)(x))^{\bullet}$
   $= (\lambda \boldsymbol{P_2} \lambda \boldsymbol{P} \bigvee x \bigwedge y. (\boldsymbol{P_2}(y) \leftrightarrow x \doteq y) \wedge \boldsymbol{P}(x))[\lambda z \bigvee \boldsymbol{c}. \boldsymbol{temp}(\boldsymbol{c}) \wedge z \doteq \boldsymbol{c}(\boldsymbol{@})]$
   $= \lambda \boldsymbol{P} \bigvee x \bigwedge y. ((\bigvee \boldsymbol{c}. \boldsymbol{temp}(\boldsymbol{c}) \wedge y \doteq \boldsymbol{c}(\boldsymbol{@})) \leftrightarrow x \doteq y) \wedge \boldsymbol{P}(x)$

7. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\text{the}][_{\mathrm{N}}\text{temperature}]][_{\mathrm{VP}}[\text{is}][\text{ninety}]]]$
   $\rightsquigarrow (\lambda i \exists x \forall y. ((\exists c. temp(i)(c) \wedge y = c(@)) \leftrightarrow x = y) \wedge x = ninety)^{\bullet}$
   $= (\lambda \boldsymbol{P} \bigvee x \bigwedge y. ((\bigvee \boldsymbol{c}. \boldsymbol{temp}(\boldsymbol{c}) \wedge y \doteq \boldsymbol{c}(\boldsymbol{@})) \leftrightarrow x \doteq y) \wedge \boldsymbol{P}(x))[\lambda z. z \doteq ninety]$
   $= \bigvee x \bigwedge y. ((\bigvee \boldsymbol{c}. \boldsymbol{temp}(\boldsymbol{c}) \wedge y \doteq \boldsymbol{c}(\boldsymbol{@})) \leftrightarrow x \doteq y) \wedge x \doteq ninety$

---

[16] Here, the type of the argument is underlined.

Notably, the term from (5.7) does not result from the term in the first premise of ($\textsc{ptq}$-$\star$) by replacing 'c' and 'c_1' by '$\boldsymbol{c}$' and '$\boldsymbol{c_1}$', and by replacing '$temp$' and '$rise$' by '$\boldsymbol{temp}$' resp. '$\boldsymbol{rise}$'. In particular, while the term in the first premise of ($\textsc{ptq}$-$\star$) states the existence of a unique witness of the type-$((\boldsymbol{(\sigma \to \iota)} \to t)$ property of being a temperature, the term from (5.7) only states the existence of a unique witness of the TY_1 correlate of the type-$(\sigma \to (\boldsymbol{\iota} \to t))$ property of being the temperature *at the current index*. Yet, since the occurrence of the temperature in the first premise of ($\star$) receives an extensional interpretation (type ι), this weakening is unproblematic. We will see at the end of this section that (the TY_2 correlate of) our weaker TY_1 term still blocks Partee's temperature puzzle.

To enable an *intensional* (type-$((\sigma \to ((\boldsymbol{\sigma \to} \iota) \to t)) \to t)$) interpretation of the phrase the temperature, we introduce the *intensionalization* operator *int*. This operator sends the designators of individuals to the designators of individual concepts, and sends the designators of functions from properties of *individuals* to generalized quantifiers over *individuals* (type $(\sigma \to (\boldsymbol{\iota} \to t)) \to ((\sigma \to (\boldsymbol{\iota} \to t)) \to t))$ to the designators of functions from properties of *individual concepts* to generalized quantifiers over *individual concepts* (type $(\sigma \to (\boldsymbol{(\sigma \to} \iota) \to t)) \to ((\sigma \to (\boldsymbol{(\sigma \to} \iota) \to t)) \to t)).$[17]

Definition 12 (Intensionalization). *The operator 'int' then works as follows:*

$$int\,(ninety) := \mathsf{ninety}$$

$$int\,(\lambda P_2 \lambda P \lambda i\,\exists x.\,P_2(i)(x) \wedge P(i)(x)) := \lambda T_2 \lambda T \lambda i\,\exists c.\,T_2(i)(c) \wedge T(i)(c)$$

$$int\,(\lambda P_2 \lambda P \lambda i\,\forall x.\,P_2(i)(x) \to P(i)(x)) := \lambda T_2 \lambda T \lambda i\,\exists c.\,T_2(i)(c) \to T(i)(c)$$

$$int\,(\lambda P_2 \lambda P \lambda i\,\exists x \forall y.\,(P_2(i)(y) \leftrightarrow x = y) \wedge P(i)(x))$$
$$:= \lambda T_2 \lambda T \lambda i\,\exists c \forall c_2.\,(T_2(i)(c_2) \leftrightarrow c = c_2) \wedge T(i)(c)$$

The operator *int* is an 'ι-to-$(\sigma \to \iota)$'-restricted variant of the intensionalization operator for extensional TY_2 terms from [6] (cf. [3, Ch. 8.4]). This operator systematically replaces each occurrence of ι in the type of a linguistic expression by the type $\boldsymbol{\sigma \to} \iota$. As a result, the type for generalized quantifiers over individuals, $((\sigma \to (\boldsymbol{\iota} \to t)) \to t)$ will be replaced by the type $((\sigma \to (\boldsymbol{(\sigma \to} \iota) \to t)) \to t)$.

The interpretation of intensional noun phrases in the type $((\sigma \to ((\sigma \to \iota) \to t)) \to t)$ enables the TY_1 translation of the second premise from ($\star$):

1. $[_{\mathrm{IV}}\mathsf{rises}] \rightsquigarrow rise^{\bullet} = \boldsymbol{rise}$ \hfill (6)

2. $[_{\mathrm{N}}\mathsf{temperature}] \rightsquigarrow temp^{\bullet} = \boldsymbol{temp}$

3. $[_{\mathrm{DET}}\mathsf{the}] \rightsquigarrow (int\,(\lambda P_2 \lambda P \lambda i\,\exists x \forall y.\,(P_2(i)(y) \leftrightarrow x = y) \wedge P(i)(x)))^{\bullet}$
   $= (\lambda T_2 \lambda T \lambda i\,\exists c \forall c_2.\,(T_2(i)(c_2) \leftrightarrow c = c_2) \wedge T(i)(c))^{\bullet}$
   $= \lambda T_2^{\bullet} \lambda T^{\bullet} \bigvee c^{\bullet} \bigwedge c_2^{\bullet}.\,(T_2^{\bullet}(c_2^{\bullet}) \leftrightarrow c^{\bullet} \doteq c_2^{\bullet}) \wedge T^{\bullet}(c^{\bullet})$
   $= \lambda \boldsymbol{T_2} \lambda \boldsymbol{T} \bigvee \boldsymbol{c} \bigwedge c_2.\,(\boldsymbol{T_2}(\boldsymbol{c_2}) \mathbin{\dot\leftrightarrow} \boldsymbol{c} \doteq c_2) \wedge \boldsymbol{T}(\boldsymbol{c})$

4. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{temperature}]]$
   $\rightsquigarrow (\lambda T \lambda i\,\exists c \forall c_2.\,(temp(i)(c_2) \leftrightarrow c = c_2) \wedge T(i)(c))^{\bullet}$

---

[17] Note that the *int*-operator also generalizes the denotations of P_2 and P (resp. of T_2 and T) across indices. As a result, this operator achieves a *double* intensionalization.

$$= (\lambda \boldsymbol{T} \bigvee \boldsymbol{c} \bigwedge \boldsymbol{c}_2.(\boldsymbol{temp}\,(\boldsymbol{c}_2) \leftrightarrow \boldsymbol{c} \doteq \boldsymbol{c}_2) \wedge \boldsymbol{T}(\boldsymbol{c}))$$

5. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{temp.}]][_{\mathrm{IV}}\mathsf{rises}]]$

$\qquad \rightsquigarrow (\lambda i \exists c \forall c_2.(temp\,(i)(c_2) \leftrightarrow c = c_2) \wedge rise\,(i)(c))^{\bullet}$

$\qquad = \bigvee \boldsymbol{c} \bigwedge \boldsymbol{c}_2.(\boldsymbol{temp}\,(\boldsymbol{c}_2) \leftrightarrow \boldsymbol{c} \doteq \boldsymbol{c}_2) \wedge \boldsymbol{rise}(\boldsymbol{c})$

The possibility of interpreting proper names in the type for individual concepts enables us to translate the conclusion from $(\star)$ as follows:

1. $[_{\mathrm{NP}}\mathsf{ninety}] \rightsquigarrow (int\,(ninety))^{\bullet} = ninety^{\bullet} = \boldsymbol{ninety}$ \hfill (7)

2. $[_{\mathrm{IV}}\mathsf{rises}] \rightsquigarrow rise^{\bullet} = \boldsymbol{rise}$

3. $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{ninety}][_{\mathrm{IV}}\mathsf{rises}]] \rightsquigarrow (rise\,(ninety))^{\bullet} = \boldsymbol{rise}\,(\boldsymbol{ninety})$

This completes our translation of the 'ingredient sentences' for Partee's temperature puzzle. The invalid inference from the conjunction of (5.7) and (6.5) to (7.3) in the logic TY_1 is captured below:

$$\bigvee x \bigwedge y.((\bigvee \boldsymbol{c}.\,\boldsymbol{temp}\,(\boldsymbol{c}) \wedge y \doteq \boldsymbol{c}\,(\text{\@})) \leftrightarrow x \doteq y) \wedge x \doteq ninety$$

$$\cancel{\frac{\bigvee \boldsymbol{c} \bigwedge \boldsymbol{c}_2.(\boldsymbol{temp}\,(\boldsymbol{c}_2) \leftrightarrow \boldsymbol{c} \doteq \boldsymbol{c}_2) \wedge \boldsymbol{rise}(\boldsymbol{c})}{\boldsymbol{rise}\,(\boldsymbol{ninety})}} \qquad (\textsc{efl-}\star)$$

In particular, while the formula in the second premise attributes the property 'rise' to the type-$(\boldsymbol{o} \rightarrow \iota)$ object which has the property of being a temperature, the formula in the first premise attributes the property 'is ninety' only to the result (type ι) of applying a temperature-object to the EFL-correlate of @. In virtue of this fact – and the resulting invalidity of substituting *ninety* for $\boldsymbol{c}$ in the second premise of $(\textsc{efl-}\star)$ –, the formula in the conclusion does not follow from the conjunction of the two premise-formulas by the rules of the logic TY_1.

5 Conclusion

This paper has shown the possibility of interpreting Montague's PTQ-fragment in the class of EFL-models from [11], which only contain basic individuals and propositions. We have obtained this result by coding the interpretations of the PTQ-expressions from [12] into EFL-objects, and by translating the linguistically relevant sublanguage of a streamlined version, TY_2, of Montague's logic IL into the EFL-typed language TY_1 which respects this coding. Since this translation preserves the relation of logical consequence on the TY_2 translations of PTQ-sentences, it enables a new, *extensional*, solution to Partee's temperature puzzle.

The previously-assumed impossibility of such a solution can be attributed to the various challenges which emerge for any TY_2-to-TY_1 translation. These challenges include the different forms of the linguistically relevant TY_2 and TY_1 types, and the unavailability of truth-functional connectives or quantifiers in the language of TY_1. Our solutions to these challenges build on existing work on the relation between TY_2 and IL types ([14]), and on hyperintensional semantics ([17]).

Our TY_2-to-TY_1 translation enables a transfer of the interpretive success of PTQ-models to EFL-models (esp. of the solvability of Partee's temperature puz-

zle) and a proof of the relative consistency of the two classes of models. At the same time, it identifies the minimal semantic requirements on models for the PTQ-fragment. Contrary to what is suggested by a comparison of [12] and [11], suitable PTQ-models need *not* contain a designated type for indices. We take these results to support the reduction view of formal natural language semantics.

References

[1] A. Church, *A formulation of the Simple Theory of Types*, Journal of Symbolic Logic **5/2** (1940), 56–68.

[2] M.J. Cresswell, *The semantics of degree*, Montague Grammar (B. Partee, ed.), Academic Press, New York, 1976.

[3] J. van Eijck and Ch. Unger, *Computational Semantics with Functional Programming*, Cambridge University Press, Cambridge and New York, 2010.

[4] C. Fox and S. Lappin, *An expressive first-order logic with flexible typing for natural language semantics*, Logic Journal of the IGPL **12/2** (2004), 135–168.

[5] D. Gallin, *Intensional and Higher-Order Modal Logic with Applications to Montague Semantics*, North Holland, Amsterdam, 1975.

[6] P. de Groote and M. Kanazawa, *A note on intensionalization*, Journal of Logic, Language and Information **22/2** (2013), 173–194.

[7] L. Henkin, *Completeness in the theory of types*, J. of Symb. Logic **15** (1950), 81–91.

[8] T.M.V. Janssen, *Individual concepts are useful*, Varieties of Formal Semantics: Proceedings of the 4th Amsterdam Colloquium (F. Landman and F. Veltman, eds.), 1984.

[9] E. Klein and I. Sag, *Type-driven translation*, Linguistics and Philosophy **8** (1985), 163–201.

[10] M.J. Loux, *Metaphysics*: *a contemporary introduction*, Routledge, New York, 2006.

[11] R. Montague, *English as a formal language*, Formal Philosophy: Selected papers of Richard Montague (R.H. Thomason, ed.), Yale University Press, New Haven, 1976.

[12] _____, *The proper treatment of quantification in ordinary English*, Formal Philosophy, 1976.

[13] R. Muskens, *Anaphora and the logic of change*, Logic in AI **478** (1991), 412–427.

[14] _____, *Meaning and Partiality*, CSLI Lecture Notes, FoLLI, Stanford, 1995.

[15] B. Partee and M. Rooth, *Generalized conjunction and type ambiguity*, Meaning, Use and Interpretation of Language (R. Bauerle, C. Schwarz, and A. v. Stechow, eds.), Walter De Gruyter, Berlin, 1983.

[16] M. Schönfinkel, *Über die Bausteine der mathematischen Logik*, Mathematische Annalen **92** (1924), 305–316.

[17] R.H. Thomason, *A model theory for the propositional attitudes*, Linguistics and Philosophy **4** (1980), 47–70.