The *variorum*$^{\psi}$ Model: Foundations

Version 1.0 · July 2026

Giorgos Altanis

bilberry

ORCID: 0009-0003-5373-2985

Abstract

This document introduces the *variorum*$^{\psi}$ model, a formal framework for the representation of structured conceptual systems. Its purpose is twofold.

On the one hand, it is a specification document: it defines the model's ontology, state constraints, and a transition system that is proved to preserve state validity.

On the other hand, it proposes an axiomatic system of relational semantics that characterizes meaning through represented structure, locality, reciprocity, and equivariance, while lexical content remains strictly outside the model's semantics.

DOI: 10.5281/zenodo.21331591

Contents

Preliminaries and Conventions

I. Undefinedness

$\perp$ is a distinguished symbol denoting that a value is undefined with respect to a given context (including both non-applicability and unavailability). It is not a member of any model set.

Unless explicitly stated otherwise, functions propagate undefinedness:

$$f(\perp) = \perp$$

Convention (totalization of partial functions). Unless explicitly stated otherwise, every partial function is understood to be extended to a total function by returning $\perp$ outside its intended domain.

II. Sequences

Let angle brackets '$\langle\rangle$' denote finite sequences; the empty sequence is denoted by $\langle\rangle$.

If Ω is a set, let $\mathrm{Seq}(\Omega)$ denote the set of all finite sequences $\langle\omega_1, \ldots, \omega_k\rangle$, with $k \geq 0$ and $\omega_i \in \Omega$ for all $i = 1, \ldots, k$. $k = 0$ corresponds to the empty sequence $\langle\rangle$, so that $\langle\rangle \in \mathrm{Seq}(\Omega)$.

Definition (s^-) Let $\langle\omega_1, \ldots, \omega_k\rangle \in \mathrm{Seq}(\Omega)$. Define the function

$$\mathrm{s}^- : \mathrm{Seq}(\Omega) \longrightarrow \mathrm{Seq}(\Omega) \cup \{\perp\}$$

by:

$$s^-(\langle\omega_1, \ldots, \omega_k\rangle) = \langle\omega_1, \ldots, \omega_{k-1}\rangle, \quad k \geq 2,$$
$$s^-(\langle\omega\rangle) = \langle\rangle,$$
$$s^-(\langle\rangle) = \perp$$

A. Literals

Let L denote the possibly infinite set of literals. No further structure, classification into data types, or operations on L are specified.

B. Model elements

Let V be a finite set of model elements, disjoint from L:

$$V \cap L = \varnothing$$

Let

$$t : V \to \mathcal{T}$$

be an element-type function, where

$$\mathcal{T} = \{R, C, K\}.$$

For convenience, define the subsets:

$$V_{\mathrm{r}} = \{v \in V \mid t(v) = R\}$$
$$V_{\mathrm{c}} = \{v \in V \mid t(v) = C\}$$
$$V_{\kappa} = \{v \in V \mid t(v) = K\}$$

Convention (addition to typed element subsets). For convenience, "add x to V_{τ}" means add x to V and assign to x the corresponding type under t. In particular, adding x to V_{r}, V_{c}, or V_{κ} means setting $t(x) = R$, $t(x) = C$, or $t(x) = K$, respectively.

C. Relations between elements

Let

$$\mathcal{T}' = \{\mathrm{pu}, \mathrm{cb}, \mathrm{df}\},$$

with

$$\mathcal{T} \cap \mathcal{T}' = \varnothing.$$

Let $E \subseteq V \times \mathcal{T}' \times V$ be the finite set of typed model relations.

For each element $e = (v_1, \tau, v_2) \in E$, define:

$$\mathrm{source}(e) = v_1,$$
$$t'(e) = \tau,$$
$$\mathrm{target}(e) = v_2.$$

For every $e \in E$:

$$\mathrm{source}(e) \neq \mathrm{target}(e)$$

For convenience, define the corresponding binary relation sets:

$$E_{\mathrm{pu}} = \{(u, v) \mid (u, \mathrm{pu}, v) \in E\},$$
$$E_{\mathrm{cb}} = \{(u, v) \mid (u, \mathrm{cb}, v) \in E\},$$
$$E_{\mathrm{df}} = \{(u, v) \mid (u, \mathrm{df}, v) \in E\},$$

and extend the functions source and target to elements of $E_{\mathrm{pu}} \cup E_{\mathrm{cb}} \cup E_{\mathrm{df}}$. For every $e = (u, v) \in E_{\mathrm{pu}} \cup E_{\mathrm{cb}} \cup E_{\mathrm{df}}$:

$$\mathrm{source}(e) = u,$$
$$\mathrm{target}(e) = v$$

Notation. For $v_1, v_2 \in V$ and $\tau \in \mathcal{T}'$, write

$$v_1 \xrightarrow{\tau} v_2$$

if and only if

$$(v_1, \tau, v_2) \in E$$

Convention (addition to typed relation subsets). For convenience, "add (u, v) to E_τ" means add the typed relation (u, τ, v) to E. In particular, adding (u, v) to E_{pu}, E_{cb}, or E_{df} means adding (u, pu, v), (u, cb, v), or (u, df, v) to E, respectively.

D. Arguments and properties

Definition (arguments) For each element $v \in V_{\text{c}}$, define a possibly empty finite set of the element's arguments, denoted by $\text{args}(v)$.
The cardinality of this set is called the arity of v:

$$\text{arity}(v) = |\text{args}(v)|.$$

The arguments of an element v are represented as ordered pairs $<v, i>$, consecutively indexed from 1 to $\text{arity}(v)$:

$$\text{args}(v) = \{<v, i> \mid 1 \leq i \leq \text{arity}(v)\}.$$

Definition (argument tokens) Let

$$A = \bigcup_{v \in V_{\text{c}}} \text{args}(v)$$

be the set of all argument tokens.
Argument tokens are distinct from model elements and literals:

$$A \cap (V \cup L) = \varnothing.$$

Definition (argOwner / argPos) For an argument token $a = <v, i> \in A$, define

$$\text{argOwner}(a) = v$$

and

$$\text{argPos}(a) = i.$$

Definition (properties) For each element $v \in V_{\text{c}}$, define a (possibly empty) finite set of internal property assignments, denoted by $\text{props}(v)$.
If there exists a unique df-relation $v \xrightarrow{\text{df}} w$, then

$$\text{props}(v) \subseteq \text{args}(w) \times (V_{\text{c}} \cup V_{\kappa} \cup L),$$

and property assignment is functional in its first coordinate:

$$\forall <a_1, \alpha_1>, <a_2, \alpha_2> \in \text{props}(v) : \quad a_1 = a_2 \implies \alpha_1 = \alpha_2.$$

Otherwise,

$$\text{props}(v) = \varnothing.$$

Definition (argument-correspondence map) Let $v \in V_c$. The argument-correspondence data carried by v determine a map $\mathrm{argMap}(v)$.
If a unique df-relation $v \xrightarrow{\mathrm{df}} w$ exists, then

$$\mathrm{argMap}(v) : \{1, \ldots, \mathrm{arity}(v)\} \longrightarrow \{1, \ldots, \mathrm{arity}(w)\} \cup \{\perp\}.$$

Otherwise,

$$\mathrm{argMap}(v) : \{1, \ldots, \mathrm{arity}(v)\} \longrightarrow \{\perp\}.$$

Remark By rule E11 of section E below, in every valid state there can be at most one df-relation e having $\mathrm{source}(e) = v$. Thus, in every valid state, the unique-relation cases in the definitions of $\mathrm{props}(v)$ and $\mathrm{argMap}(v)$ are unambiguous.

Principle (formation and immutability of argument and property data) Let $v \in V_c$. Its arity, arguments, properties, and argument-correspondence data are established only when v is created and only in one of the following ways:

1. **Creation of a concept having no outgoing** df**-relation.** Arity of v is specified at creation. Its arguments are then determined by

   $$\mathrm{args}(v) = \{<v, i> \mid 1 \le i \le \mathrm{arity}(v)\},$$

   while

   $$\mathrm{props}(v) = \varnothing$$

   and

   $$\mathrm{argMap}(v)(j) = \perp \quad \text{for every } j \in \{1, \ldots, \mathrm{arity}(v)\}.$$

2. **Creation of a concept by way of the** bind **operation (D1).** The arity, arguments, properties, and argument-correspondence data of v are established as specified in section D1 below.

3. **Creation of a concept by way of the** addArgs **operation (D2).** The arity, arguments, properties, and argument-correspondence data of v are established as specified in section D2 below.

No other operation creates or modifies $\mathrm{arity}(v)$, $\mathrm{args}(v)$, $\mathrm{props}(v)$, or $\mathrm{argMap}(v)$.

D1. Argument binding

Let $v \in V_c$, $B = \{n_1, \ldots, n_k\} \subseteq \{1, \ldots, \mathrm{arity}(v)\}$ be a (possibly empty) set of k argument positions, and $\alpha_1, \ldots, \alpha_k \in V_c \cup V_\kappa \cup L$.

The binding operation

$$v' = \mathrm{bind}(v; n_1 = \alpha_1, \ldots, n_k = \alpha_k)$$

produces an element v', distinct from all existing elements in V, such that:

- **New element:** v' is added to V.
- **Type inheritance:**

$$t(v') = t(v).$$

- **Derivation:** A new df-edge $v' \xrightarrow{\text{df}} v$ is added to E.
- **Internal properties:**

$$\text{props}(v') = \{<<v, n_i>, \alpha_i> \mid i = 1, \ldots, k\}.$$

- **Arguments (canonical reindexing and correspondence):**

Let $U = \{1, \ldots, \text{arity}(v)\} \setminus B$, and let $u_1 < \cdots < u_m$ be the elements of U in increasing order, where $m = |U|$. Then:

$$\text{arity}(v') = m$$

and

$$\text{argMap}(v')(j) = u_j, \quad j = 1, \ldots, m.$$

D2. Argument addition

Let $v \in V_c$, and let $k \in \mathbb{N}$. The argument-addition operation

$$v' = \text{addArgs}(v; k)$$

produces an element v', distinct from all existing elements in V, such that:

- **New element:** v' is added to V.
- **Type inheritance:**

$$t(v') = t(v).$$

- **Derivation:** A new df-edge $v' \xrightarrow{\text{df}} v$ is added to E.
- **Internal properties:**

$$\text{props}(v') = \varnothing.$$

- **Arguments (arity and correspondence):**

$$\begin{aligned} \text{arity}(v') &= \text{arity}(v) + k. \\ \text{argMap}(v')(j) &= j, \quad \text{for } j = 1, \ldots, \text{arity}(v), \\ \text{argMap}(v')(j) &= \perp, \quad \text{for } j = \text{arity}(v) + 1, \ldots, \text{arity}(v'). \end{aligned}$$

E. Variorum states and state validity rules

Definition (variorum state) A variorum state is a pair

$$\Sigma = (V, E),$$

where V is a finite set of structured model elements carrying the type, argument, argument-correspondence, and property data defined in sections B–D, and E is the finite set of typed relations defined in section C.
The functions t, args, arity, props, argOwner, argPos, and argMap, and the sets V_{r}, V_{c}, V_{κ}, and A, are induced by V.

Definition (valid variorum state) A variorum state is valid if and only if it satisfies the following rules E1-E14.

Placement and placement paths

Rule E1. Sovereign frameworks are defined by categories (members of V_{κ}), and only by categories

There exists a category with no placement successor:

$$\exists \kappa \in V_{\kappa} \text{ such that } \nexists e \in E_{\mathrm{pu}} \text{ with } \mathrm{source}(e) = \kappa$$

Such a category is called a root category.

Corollary V has at least one member, a root category.

Rule E2. Concepts (members of V_{c}) must be situated

Every concept has exactly one placement successor.

$$\forall c \in V_{\mathrm{c}}\ \exists! e \in E_{\mathrm{pu}} \text{ with } \mathrm{source}(e) = c$$

Rule E3. Subcategories are allowed

Every category that is not a root category has exactly one placement successor.

$$\forall \kappa \in V_{\kappa} \text{ such that } \kappa \text{ is not a root category, } \exists! e \in E_{\mathrm{pu}} \text{ such that } \mathrm{source}(e) = \kappa$$

Such a category is called a subcategory.

Corollary (functionality of placement) By E2 and E3, the placement relation E_{pu} is functional in its source: every concept and every subcategory has exactly one pu-successor.

Rule E4. Only concepts and categories can participate in placement

$$e \in E_{\mathrm{pu}} \Longrightarrow \mathrm{source}(e) \in V_{\mathrm{c}} \cup V_{\kappa} \text{ and } \mathrm{target}(e) \in V_{\mathrm{c}} \cup V_{\kappa}$$

Rule E5. Concepts may contain only concepts

$$e \in E_{\mathrm{pu}} \wedge \mathrm{target}(e) \in V_{\mathrm{c}} \Longrightarrow \mathrm{source}(e) \in V_{\mathrm{c}}$$

Definition (root and situated elements) Define the sets of root and situated elements by:

$$V_{\text{root}} = \{v \in V_\kappa \mid \nexists e \in E_{\text{pu}} \text{ with } \text{source}(e) = v\}$$
$$V_{\text{situated}} = \{v \in V_c \cup V_\kappa \mid \exists e \in E_{\text{pu}} \text{ with } \text{source}(e) = v\}$$

Define

$$(v \text{ is root}) := v \in V_{\text{root}}$$

and

$$(v \text{ is situated}) := v \in V_{\text{situated}}$$

Definition (placement path) Let $v_1, v_k \in V$. A placement path from v_1 to v_k is a finite sequence of elements $\langle v_1, \ldots, v_k \rangle \in \text{Seq}(V_c \cup V_\kappa)$ with $k \geq 2$ such that $(v_i, v_{i+1}) \in E_{\text{pu}}$ for all $i = 1, \ldots, k-1$.

Definition (forward-maximal placement path) A placement path $\langle v_1, \ldots, v_k \rangle$ is forward-maximal iff there is no $w \in V$ with $(v_k, w) \in E_{\text{pu}}$.

Definition (parent) For all $x \in V_{\text{situated}}$, define parent$(x)$ by:

$$\text{parent}(x) = y \text{ where } (x, y) \in E_{\text{pu}}$$

Rule E6. Placement paths are acyclic
In other words, there is no pu-path from an element $v \in V$ to itself.

Corollary (root reachability) Every concept and every subcategory has a pu-path to some root category.

Definition (backward pu-reachability) For convenience, define the function

$$\text{pu\_path} : V \times V \longrightarrow \text{Seq}(V)$$

by:

$$\text{pu\_path}(v, w) = \langle v, \ldots, w \rangle, \text{ the unique placement path from } v \text{ to } w, \text{ if such a path exists,}$$
$$\text{pu\_path}(v, w) = \langle\rangle, \quad \text{if no placement path from } v \text{ to } w \text{ exists}$$

For $v \in V$, define the backward pu-reachable set rooted at v as the set of all elements $w \in V$, different from v, for which there exists a placement path from w to v. Denote it by

$$R_{\text{pu}}(v) := \{w \in V \mid \text{pu\_path}(w, v) \neq \langle\rangle, \text{ with } w \neq v\}$$

Corollary A category may contain both categories and concepts:

$$e \in E_{\text{pu}} \wedge \text{target}(e) \in V_\kappa \Longrightarrow \text{source}(e) \in V_c \cup V_\kappa$$

Contextualization

Rule E7. Referents (members of V_r) must be contextualized at least once

$$\forall r \in V_{\mathrm{r}}\ \exists c \in V_{\mathrm{c}} \text{ such that } r \xrightarrow{\mathrm{cb}} c$$

Rule E8. Only referents can be contextualized, and only by concepts

$$\forall e \in E : t'(e) = \mathrm{cb} \implies \mathrm{source}(e) \in V_{\mathrm{r}}, \mathrm{target}(e) \in V_{\mathrm{c}}$$

Rule E9. A concept can contextualize at most one referent

$$\forall e \in E_{\mathrm{cb}}\ \nexists e' \in E_{\mathrm{cb}} \text{ such that } \mathrm{target}(e') = \mathrm{target}(e) \wedge \mathrm{source}(e') \neq \mathrm{source}(e)$$

Definition (referent)

$$\forall (r, x) \in E_{\mathrm{cb}}, \mathrm{referent}(x) = r,$$
$$\forall x \in V : \nexists r \in V_{\mathrm{r}} \text{ such that } (r, x) \in E_{\mathrm{cb}}, \mathrm{referent}(x) = \perp.$$

Derivation and derivation paths

Rule E10. Only concepts can be derived, and only from concepts

$$\forall e \in E : t'(e) = \mathrm{df} \implies \mathrm{source}(e) \in V_{\mathrm{c}}, \mathrm{target}(e) \in V_{\mathrm{c}}$$

Definition (origin and derived elements) Define the sets of origin and derived elements by:

$$V_{\mathrm{origin}} = \{v \in V_{\mathrm{c}} \mid \exists e \in E_{\mathrm{df}} \text{ with } \mathrm{target}(e) = v \wedge \nexists e \in E_{\mathrm{df}} \text{ with } \mathrm{source}(e) = v\}$$
$$V_{\mathrm{derived}} = \{v \in V_{\mathrm{c}} \mid \exists e \in E_{\mathrm{df}} \text{ with } \mathrm{source}(e) = v\}$$

Define

$$(v \text{ is origin}) := v \in V_{\mathrm{origin}}$$

and

$$(v \text{ is derived}) := v \in V_{\mathrm{derived}}$$

Definition (derivation path) Let $c_1, c_k \in V_{\mathrm{c}}$. A derivation path from c_1 to c_k is a finite sequence of concepts $\langle c_1, \ldots, c_k \rangle \in \mathrm{Seq}(V_{\mathrm{c}})$ with $k \geq 2$ such that $(c_i, c_{i+1}) \in E_{\mathrm{df}}$ for all $i = 1, \ldots, k-1$.

Definition (forward-maximal derivation path) A derivation path $\langle c_1, \ldots, c_k \rangle$ is forward-maximal iff there is no $w \in V_{\mathrm{c}}$ with $(c_k, w) \in E_{\mathrm{df}}$.

Definition (forward df-reachability set) For $c \in V_{\mathrm{derived}}$, let $\langle c, c_1, \ldots, c_k \rangle$, with $k \geq 1$, be the (unique) forward-maximal derivation path starting at c. Define:

$$A_{\mathrm{df}}(c) = \{c_1, c_2, \ldots, c_k\}$$

to be the forward df-reachability set starting at c.
For $c \notin V_{\mathrm{derived}}$,

$$A_{\mathrm{df}}(c) = \varnothing$$

Rule E11. A concept can derive from at most one concept

$$\forall e \in E_{\mathrm{df}}\ \nexists e' \in E_{\mathrm{df}} \text{ such that } \mathrm{source}(e') = \mathrm{source}(e) \wedge \mathrm{target}(e') \neq \mathrm{target}(e)$$

Definition (predecessor) For $x \in V$, define predecessor(x) by:

$$\mathrm{predecessor}(x) = y, \quad \text{if } x \in V_{\mathrm{derived}} \text{ and } (x, y) \in E_{\mathrm{df}},$$
$$\mathrm{predecessor}(x) = \bot, \quad \text{if } x \notin V_{\mathrm{derived}}.$$

Rule E12. Derivation paths are acyclic

There is no df-path from an element $v \in V$ to itself.

Corollary (origin reachability) Every derived concept has a df-path to some origin concept.

Definition (backward df-reachability) For convenience, define the function

$$\mathrm{df\_path} : V \times V \longrightarrow \mathrm{Seq}(V)$$

by:

$$\mathrm{df\_path}(v, w) = \langle v, \ldots, w \rangle, \text{ the unique derivation path from } v \text{ to } w, \text{ if such a path exists,}$$
$$\mathrm{df\_path}(v, w) = \langle\rangle, \quad \text{if no derivation path from } v \text{ to } w \text{ exists}$$

For $v \in V$, define the backward df-reachable set originating at v as the set of all elements $w \in V$ for which a df-path from w to v exists, and denote it by $R_{\mathrm{df}}(v) := \{w \in V \mid \mathrm{df\_path}(w, v) \neq \langle\rangle\}$.

Rule E13. Direct contextualization and derivation are mutually exclusive

- $\forall e \in E_{\mathrm{df}}\ \nexists e' \in E_{\mathrm{cb}}$ such that $\mathrm{target}(e') = \mathrm{source}(e)$.
- $\forall e \in E_{\mathrm{cb}}\ \nexists e' \in E_{\mathrm{df}}$ such that $\mathrm{source}(e') = \mathrm{target}(e)$.

Rule E14. Concepts ultimately stem from contextualizations

$$\forall c \in V_{\mathrm{c}},\ \exists! r \in V_{\mathrm{r}} \text{ such that}$$
$$\text{either } r \xrightarrow{\mathrm{cb}} c$$
$$\text{or there exists a forward-maximal df-path } \langle c, \ldots, c_k \rangle \text{ with } r \xrightarrow{\mathrm{cb}} c_k$$

Corollary Each forward-maximal df-path $\langle c_1, \ldots, c_k \rangle$ concludes with a concept c_k that directly contextualizes a referent. To see this, assume otherwise. Since there is no $r \in V_{\mathrm{r}}$ such that $r \xrightarrow{\mathrm{cb}} c_k$, by E14 there exists a forward-maximal df-path $\langle c_k, \ldots, c_{k+m} \rangle$ for some $m > 0$ and an $r \in V_{\mathrm{r}}$ with $r \xrightarrow{\mathrm{cb}} c_{k+m}$. But then, $\langle c_1, \ldots, c_k, \ldots, c_{k+m} \rangle$ is a forward-maximal df-path, which contradicts the hypothesis that $\langle c_1, \ldots, c_k \rangle$ is a forward-maximal df-path.

LEMMA Sufficient conditions for state validity
(validity lemma)

Let $\Sigma = (V, E)$ be a variorum state. Suppose:

1. $(V_c \cup V_\kappa, E_{pu})$ is a finite nonempty forest of directed trees whose edges point toward their roots, and whose roots are exactly the members of V_{root}. Moreover, whenever $(u, v) \in E_{pu}$ and $u \in V_\kappa$, then $v \in V_\kappa$.

2. (V_c, E_{df}) is a finite forest of directed trees whose edges point toward their roots, and whose roots are exactly
$$V_o = V_c \setminus V_{derived}.$$

3. There is a surjective function $\rho : V_o \longrightarrow V_r$ such that
$$E_{cb} = \{(\rho(o), o) \mid o \in V_o\}.$$

Then Σ is a valid variorum state.
Proof. See Appendix A.

F. State transitions

The transition system begins with the initial state

$$\Sigma_0 = (V_0, E_0) = (\{\kappa_0\}, \varnothing), \qquad \kappa_0 \in V_\kappa.$$

The following operations define the candidate transitions from a state $\Sigma = (V, E)$ to a successor state Σ'.

Transition F1. `addRoot`

- κ = addRoot().
- **Preconditions:** none.
- Returns a new root category κ and adds it to V_κ.

Transition F2. `createConceptWithNewReferent`

- $[r, c]$ = createConceptWithNewReferent$(p; k)$, with $p \in V_c \cup V_\kappa$, and $k \in \mathbb{N}$ the arity of c.
- Returns a new referent r and a new concept c, created according to the first case of the formation principle of section D, with arity$(c) = k$. That is, args$(c) = \{<c, i> \mid i \in \{1, \ldots, k\}\}$, props$(c) = \varnothing$, and argMap$(c)(j) = \perp$ for every $j \in \{1, \ldots, k\}$.
- **Preconditions:** none.
- Adds r to V_r and c to V_c.
- Adds $r \xrightarrow{cb} c$ and $c \xrightarrow{pu} p$ to E.

Transition F3. `recontextualizeReferent`

- c = recontextualizeReferent$(r, p; k)$, with $r \in V_r$, $p \in V_c \cup V_\kappa$, and $k \in \mathbb{N}$ the arity of c.

- Returns a new concept c, created according to the first case of the formation principle of section D, with $\text{arity}(c) = k$. That is, $\text{args}(c) = \{<c, i> \mid i \in \{1, \ldots, k\}\}$, $\text{props}(c) = \varnothing$, and $\text{argMap}(c)(j) = \perp$ for every $j \in \{1, \ldots, k\}$.
- **Preconditions:** none.
- Adds c to V_c.
- Adds $r \xrightarrow{\text{cb}} c$ and $c \xrightarrow{\text{pu}} p$ to E.

Transition F4. `placeSubcategoryUnder`

- $\kappa = \text{placeSubcategoryUnder}(p)$, with $p \in V_\kappa$.
- Returns a new category κ.
- **Preconditions:** none.
- Adds κ to V_κ and $\kappa \xrightarrow{\text{pu}} p$ to E.

Transition F5. `bindAndPlace`

- $c = \text{bindAndPlace}(v, p; n_1 = x_1, \ldots, n_k = x_k)$, with $v \in V_c$, $p \in V_c \cup V_\kappa$, $k \in \mathbb{N}^*$, $\{<v, n_i> \mid 1 \le i \le k\} \subseteq \text{args}(v)$, $x_i \in V_c \cup V_\kappa \cup L$, and $|\{n_1, \ldots, n_k\}| = k$.
- Returns a new concept
$$c = \text{bind}(v; n_1 = x_1, \ldots, n_k = x_k),$$
with its arity, arguments, properties, and argument-correspondence data as prescribed by section D1.
- **Preconditions:** none.
- Adds $c \xrightarrow{\text{pu}} p$ to E.
- As a result of the bind operation, c is added to V_c, and $c \xrightarrow{\text{df}} v$ is added to E.

Transition F6. `reuseAndPlace`

- $c = \text{reuseAndPlace}(v, p; k)$, with $v \in V_c$, $p \in V_c \cup V_\kappa$, $k \in \mathbb{N}$.
- Returns a new concept
$$c = \text{addArgs}(v; k),$$
with its arity, arguments, properties, and argument-correspondence data as prescribed by section D2.
- **Preconditions:** none.
- Adds $c \xrightarrow{\text{pu}} p$ to E.
- As a result of the addArgs operation, c is added to V_c, and $c \xrightarrow{\text{df}} v$ is added to E.

Transition F7. `detachSubcategory`

- $\text{detachSubcategory}(\kappa)$, with $\kappa \in V_\kappa$, $\kappa \notin V_{\text{root}}$.
- Deletes the pu-relation $\kappa \xrightarrow{\text{pu}} \text{parent}(\kappa)$ from E.
- Thereby, κ becomes a root category.
- **Preconditions:** none.

Transition F8. `relocateElement`

- relocateElement(v, p), with $v \in V_c \cup V_\kappa$, $p \in V_c \cup V_\kappa$.
- Moves v, together with its pu-descendants, under p.
- **Preconditions:**
  - $p \neq v$, $p \notin R_{pu}(v)$ (the move will not create a cycle).
  - If $t(v) = K$, then $t(p) = K$ (so that the next state does not violate (E5)).
- If there exists a (unique) $e \in E$ with $t'(e) = \text{pu}$ and $\text{source}(e) = v$, e is removed from E.
- Adds $v \xrightarrow{\text{pu}} p$ to E.

Transition F9. `removeElement`

- removeElement(v), with $v \in V_c \cup V_\kappa$.
- Removes v from V:
  - If $\exists (r, v) \in E_{cb}$ (that is, $\text{referent}(v) \neq \perp$):
    - If there is no $e' \in E_{cb}$ with $\text{source}(e') = r$ and $\text{target}(e') \neq v$, then r has no other outgoing cb-relations besides (r, v), and r is removed from V.
    - (r, cb, v) is removed from E.
  - If $\exists e \in E$ with $t'(e) = \text{df}$ and $\text{source}(e) = v$, e is removed from E.
  - If $\exists e \in E$ with $t'(e) = \text{pu}$ and $\text{source}(e) = v$, e is removed from E.
  - v is removed from V.
- **Preconditions:**
  - $\nexists e \in E_{pu} \cup E_{df} : \text{target}(e) = v$; that is, v is neither a parent nor a predecessor of any other element.
  - $\nexists w \in V_c,\ a \in A : <a, v> \in \text{props}(w)$; that is, v is not used as a property value.
  - $V_{root} \setminus \{v\} \neq \varnothing$.

THEOREM Preservation of state validity
(preservation theorem)

Let

$$\Sigma_0 = (V_0, E_0) = (\{\kappa_0\}, \varnothing), \qquad \kappa_0 \in V_\kappa,$$

be the initial variorum state. Every state produced from Σ_0 by a finite sequence of applicable transitions is a valid variorum state.

Proof. See Appendix A.

G. Semantics

G1. Meaning bearers and meaning functions

Let $M \subseteq (V \cup A)$ be a finite set of meaning bearers.

Remark Admissibility conditions for M will be given in G2.

Let $\mathcal{D}(M)$ be the set of all finite structural expressions, called dependency terms, built from members of M, the distinguished symbol $\perp$, and structural notation.

Remark Providing a formal grammar of the admissible dependency terms is outside the scope of this work.

Definition (meaning function) A meaning function on M is a deterministic function

$$\mu : M \longrightarrow \mathcal{D}(M).$$

A defining equation for $\mu(x)$ specifies the dependency term $\mu(x) \in \mathcal{D}(M)$. If its right-hand side contains previously defined operations, rather than or in addition to dependency-term notation, these are evaluated first.

Remark Variorum's relational structure contains several commitments in latent form, such as:

- concepts and subcategories are situated within category-rooted frameworks;
- referents are mediated by situated concepts;
- concepts and arguments are tracked through derivational history;
- binding creates new structure rather than modifying old structure;
- literal values do not participate in the model's relational structure;
- state transitions preserve the structural grammar of the model;
- relational involvement is reciprocal, though not necessarily role-symmetric.

In the rest of this section, we formulate an axiomatic system that makes these commitments explicit at the semantic level.

Reference conventions

- Any occurrence of an element y inside $\mu(x)$ denotes a reference to y, not to $\mu(y)$.
- Any argument $a = <u, i>$ that occurs in $\mu(x)$ is an occurrence of a, not an occurrence of u and i. In other words, a is treated as atomic.

Convention on $\perp$

$\perp$ does not bear meaning: $\perp \notin M$. For any $x \in M$, the presence of $\perp$ in $\mu(x)$ introduces no semantic dependence.

Definition (atoms) For a dependency term $d \in \mathcal{D}(M)$, atoms(d) is the set of all members of M that occur in d.

Remark The symbol $\perp$ may occur in d, but is not counted as an atom.

Definition (μ-affects) For $x, y \in M$, y μ-affects x if and only if y occurs in the dependency term $\mu(x)$:

$$y \ \mu\text{-affects } x \iff y \in \text{atoms}(\mu(x)).$$

When μ is clear from the context, we may write "y affects x" instead of "y μ-affects x".

G2. Primary dependency graph

Definition (dependency labels) Let

$$\mathcal{T}^d = \{\mathrm{pu}^d, \mathrm{cb}^{-d}, \mathrm{df}^d, \mathrm{ao}^d, \mathrm{ap}^d, \mathrm{b}^d, \mathrm{f}^d, \mathrm{r}^d\}$$

be the set of potentially semantic dependency labels.

Definition (argument predecessor) Define the function

$$\mathrm{argPred} : A \longrightarrow A \cup \{\perp\}$$

as follows:
Let $a = <v, j> \in A$, so that $\mathrm{argOwner}(a) = v$ and $\mathrm{argPos}(a) = j$.
If there exists a df-edge $v \xrightarrow{\mathrm{df}} w$ and $\mathrm{argMap}(v)(j) = i \neq \perp$, then

$$\mathrm{argPred}(a) = <w, i> \in \mathrm{args}(w) \subseteq A.$$

Otherwise, that is, if no such edge exists, or if $\mathrm{argMap}(v)(j) = \perp$,

$$\mathrm{argPred}(a) = \perp.$$

Let $\Sigma = (V, E)$ be a valid variorum state and let

$$\begin{aligned}
\widehat{E}^d_{\mathrm{pu}} &= \{(x, \mathrm{pu}^d, y) \mid (x, y) \in E_{\mathrm{pu}}\},\\
\widehat{E}^d_{\mathrm{cb}^-} &= \{(x, \mathrm{cb}^{-d}, y) \mid (y, x) \in E_{\mathrm{cb}}\},\\
\widehat{E}^d_{\mathrm{df}} &= \{(x, \mathrm{df}^d, y) \mid (x, y) \in E_{\mathrm{df}}\},\\
\widehat{E}^d_{\mathrm{ao}} &= \{(a, \mathrm{ao}^d, v) \mid a \in A,\ v \in V_{\mathrm{c}},\ \mathrm{argOwner}(a) = v\},\\
\widehat{E}^d_{\mathrm{ap}} &= \{(a, \mathrm{ap}^d, b) \mid a, b \in A,\ \mathrm{argPred}(a) = b\},\\
\widehat{E}^d_{\mathrm{b}} &= \{(v, \mathrm{b}^d, a) \mid v \in V_{\mathrm{c}},\ a \in A,\ \exists \alpha \in V_{\mathrm{c}} \cup V_{\kappa} \cup L : <a, \alpha> \in \mathrm{props}(v)\},\\
\widehat{E}^d_{\mathrm{f}} &= \{(v, \mathrm{f}^d, u) \mid v \in V_{\mathrm{c}},\ u \in V_{\mathrm{c}} \cup V_{\kappa},\ \exists a \in A : <a, u> \in \mathrm{props}(v)\},\\
\widehat{E}^d_{\mathrm{r}} &= \{(u, \mathrm{r}^d, a) \mid u \in V_{\mathrm{c}} \cup V_{\kappa},\ a \in A,\ \exists v \in V_{\mathrm{c}} : <a, u> \in \mathrm{props}(v)\}
\end{aligned}$$

be sets of the dependencies structurally available in Σ.

Conventionally, we will treat the above sets as being indexed by $\tau \in \mathcal{T}^d$.

Definition (admissible semantic basis) Let $\Sigma = (V, E)$ be a valid variorum state. A pair (M, Ω^d), where M is a meaning-bearer set as defined in G1 and $\Omega^d \subseteq \mathcal{T}^d$ is a **semantic activation scheme**, is an admissible semantic basis for Σ if and only if:

$$\begin{aligned}
&M \cap V \neq \varnothing,\\
&\mathrm{pu}^d \in \Omega^d,\\
&\mathrm{ap}^d \in \Omega^d \Longrightarrow \mathrm{df}^d \in \Omega^d \wedge \mathrm{ao}^d \in \Omega^d,\\
&\mathrm{b}^d \in \Omega^d \Longrightarrow \mathrm{ap}^d \in \Omega^d,\\
&\mathrm{b}^d \in \Omega^d \Longleftrightarrow \mathrm{f}^d \in \Omega^d \Longleftrightarrow \mathrm{r}^d \in \Omega^d,
\end{aligned}$$

$$M \cap V_{\mathrm{r}} \neq \varnothing \Longleftrightarrow \mathrm{cb}^{-d} \in \Omega^d,$$
$$M \cap A \neq \varnothing \Longleftrightarrow \mathrm{ao}^d \in \Omega^d,$$
$$\forall \tau \in \Omega^d \; \forall x, y \in V \cup A, \quad (x, \tau, y) \in \widehat{E}^d_\tau \Longrightarrow x, y \in M.$$

Definition (primary dependency graph $\mathcal{G}^d$) Let $\Sigma = (V, E)$ be a valid variorum state, and let (M, Ω^d) be an admissible semantic basis for Σ. For every $\tau \in \mathcal{T}^d$, define

$$E^d_\tau = \widehat{E}^d_\tau \quad \text{if } \tau \in \Omega^d, \qquad E^d_\tau = \varnothing \quad \text{if } \tau \notin \Omega^d.$$

The primary dependency graph is the directed, typed graph

$$\mathcal{G}^d = (M, E^d),$$

where

$$E^d = E^d_{\mathrm{pu}} \cup E^d_{\mathrm{cb}^-} \cup E^d_{\mathrm{df}} \cup E^d_{\mathrm{ao}} \cup E^d_{\mathrm{ap}} \cup E^d_{\mathrm{b}} \cup E^d_{\mathrm{f}} \cup E^d_{\mathrm{r}} \subseteq M \times \mathcal{T}^d \times M.$$

Definition (dependency-type function) Let $\mathcal{G}^d = (M, E^d)$ be a primary dependency graph. Define the dependency-type function:

$$t^d : E^d \longrightarrow \mathcal{T}^d$$

by:

$$t^d((x, \tau, y)) = \tau.$$

Definition (dependency-graph isomorphism and automorphism) Let $\mathcal{G}^d = (M, E^d)$ and $\mathcal{G}'^d = (M', E'^d)$ be directed typed dependency graphs, such as primary dependency graphs or their restrictions, where $E^d \subseteq M \times \mathcal{T}^d \times M$ and $E'^d \subseteq M' \times \mathcal{T}^d \times M'$.
A dependency-graph isomorphism

$$\varphi : \mathcal{G}^d \longrightarrow \mathcal{G}'^d$$

is a bijection $\varphi : M \longrightarrow M'$ such that, for all $u, v \in M$ and every $\tau \in \mathcal{T}^d$,

$$(u, \tau, v) \in E^d \Longleftrightarrow (\varphi(u), \tau, \varphi(v)) \in E'^d.$$

When $\mathcal{G}^d = \mathcal{G}'^d$, a dependency-graph isomorphism is called a dependency-graph automorphism.

G3. Semantic signature

Definition (semantic signature) Let $\mathcal{G}^d = (M, E^d)$ be a primary dependency graph with dependency-type function $t^d : E^d \longrightarrow \mathcal{T}^d$. For $x \in M$, define the semantic signature as:

$$\mathcal{S}^d(x) = \{<\mathrm{out}, \tau, y> \mid y \in M, (x, \tau, y) \in E^d\} \cup \{<\mathrm{in}, \tau, y> \mid y \in M, (y, \tau, x) \in E^d\}.$$

Definition (semantic configuration) A semantic configuration is a triple

$$C = (\Sigma, M, \Omega^d),$$

where Σ is a valid variorum state, and (M, Ω^d) is an admissible semantic basis for Σ.

Convention (configuration-indexed objects). Let $C = (\Sigma, M, \Omega^d)$ be a semantic configuration. Write

$$\Sigma_C = \Sigma, \qquad M_C = M, \qquad \Omega^d_C = \Omega^d,$$

and write

$$\mu_C : M \longrightarrow \mathcal{D}(M)$$

for a meaning function on the meaning-bearer set of C. Also write $\mathcal{G}^d_C$ for its primary dependency graph and $\mathcal{S}^d_C$ for its semantic-signature function.

When C is fixed or clear from context, the index $_C$ may be omitted.

G4. Semantic reachability

Definition (forward semantic path) Let $x_1, x_k \in M$. A forward semantic path from x_1 to x_k is a finite sequence $\langle x_1, \ldots, x_k \rangle$ of $k \geq 2$ members of M, such that for all $i = 1, \ldots, k-1$ there exists $\tau \in \mathcal{T}^d$ with

$$(x_i, \tau, x_{i+1}) \in E^d.$$

Let $\mathcal{RG}^d$ denote the set of all forward semantic paths over $\mathcal{G}^d$, and let

$$\mathcal{R}^+\mathcal{G}^d(x) = \{y \mid \exists \langle x, \ldots, y \rangle \in \mathcal{RG}^d\},$$

$$\mathcal{R}^-\mathcal{G}^d(x) = \{y \mid \exists \langle y, \ldots, x \rangle \in \mathcal{RG}^d\}$$

denote the sets of all meaning bearers that can be semantically reached from x and from which x can be semantically reached, respectively.

Definition (semantic dependency neighborhood) For $x \in M$, let

$$\mathcal{NG}^d(x) = \{x\} \cup \mathcal{R}^+\mathcal{G}^d(x) \cup \mathcal{R}^-\mathcal{G}^d(x).$$

Definition (restricted dependency graph) Let $\mathcal{G}^d[x]$ be the restriction of $\mathcal{G}^d$ to $\mathcal{NG}^d(x)$. That is,

$$\mathcal{G}^d[x] = (\mathcal{NG}^d(x), E^d[x]),$$

where

$$E^d[x] = \{(u, \tau, v) \in E^d \mid u, v \in \mathcal{NG}^d(x)\}.$$

The dependency-type function on $\mathcal{G}^d[x]$ is the restriction of t^d to $E^d[x]$.

Definition (pointed dependency graph) Let C be a semantic configuration and let $x \in M_C$. The pair

$$(\mathcal{G}^d_C[x], x)$$

is called the pointed dependency graph at x.

Definition (pointed dependency-graph isomorphism) Let C and C' be semantic configurations, and let $x \in M_C$ and $x' \in M_{C'}$. A pointed dependency-graph isomorphism from $(\mathcal{G}^d_C[x], x)$ to $(\mathcal{G}^d_{C'}[x'], x')$ is a dependency-graph isomorphism

$$\psi : \mathcal{G}^d_C[x] \longrightarrow \mathcal{G}^d_{C'}[x']$$

such that $\psi(x) = x'$.

G5. State isomorphism and automorphism

Convention (indexing by state). Structural functions and data are indexed by the state when necessary; we may write, for example, $t_{\Sigma'}$, arity_{Σ}, argMap_{Σ}, props_{Σ}, and similarly for the induced sets and functions.

Definition (state isomorphism) Let $\Sigma = (V, E)$ and $\Sigma' = (V', E')$ be variorum states, not necessarily distinct, with element sets V and V', relation sets E and E', and argument-token sets A and A' induced from V and V', respectively.
A state isomorphism $\varphi : \Sigma \to \Sigma'$ is a bijective structure morphism between states Σ and Σ'. It is determined by a bijection on model elements,

$$\varphi_V : V \to V',$$

which induces corresponding bijections on relations and argument tokens,

$$\varphi_E : E \to E', \qquad \varphi_A : A \to A',$$

defined by

$$\varphi_E((v, \tau, u)) = (\varphi_V(v), \tau, \varphi_V(u)), \qquad \varphi_A(<v, i>) = <\varphi_V(v), i>.$$

When no confusion arises, we may write φ instead of φ_V, φ_E, and φ_A.
The isomorphism preserves and reflects the structure of the state as follows.

1. Element types are preserved: for all $v \in V$,

$$t_{\Sigma'}(\varphi(v)) = t_{\Sigma}(v).$$

2. Arity is preserved: for all $v \in V_{\text{c}}$,

$$\text{arity}_{\Sigma'}(\varphi(v)) = \text{arity}_{\Sigma}(v).$$

3. Typed relations are preserved and reflected: for all $v, u \in V$ and $\tau \in \mathcal{T}'$,

$$v \xrightarrow{\tau_{\Sigma}} u \quad \Longleftrightarrow \quad \varphi(v) \xrightarrow{\tau_{\Sigma'}} \varphi(u).$$

   In particular, for all $e \in E$,

$$t'_{\Sigma'}(\varphi(e)) = t'_{\Sigma}(e).$$

4. Element-valued bindings are preserved and reflected: for all $v \in V_{\text{c}}$, $a \in A$, and $x \in V_{\text{c}} \cup V_{\kappa}$,

$$<a, x> \in \text{props}_{\Sigma}(v) \Longleftrightarrow <\varphi(a), \varphi(x)> \in \text{props}_{\Sigma'}(\varphi(v)).$$

5. Literal-valued property assignments are preserved and reflected: for all $v \in V_{\text{c}}$ and $a \in A$,

$$\exists \ell \in L : <a, \ell> \in \text{props}_{\Sigma}(v) \Longleftrightarrow \exists \ell' \in L : <\varphi(a), \ell'> \in \text{props}_{\Sigma'}(\varphi(v)).$$

6. Argument correspondence is preserved: for all $v \in V_{\text{c}}$ and $j \in \{1, \ldots, \text{arity}(v)\}$,

$$\text{argMap}_{\Sigma'}(\varphi(v))(j) = \text{argMap}_{\Sigma}(v)(j).$$

Consequently, for every $a \in A$,

$$\text{argPred}_{\Sigma'}(\varphi(a)) = \varphi(\text{argPred}_{\Sigma}(a)),$$

where $\varphi(\bot) = \bot$.

When $\Sigma = \Sigma'$, a state isomorphism will be called a state automorphism.

G6. Axioms on meaning functions

Axioms A1–A8 govern the configuration-indexed meaning functions μ_C.

The axioms are divided into two groups. Axioms A1 – A5 have formed the axiomatic basis of variorum semantics since the earliest versions of this draft. Their original numbering has been preserved, although A1 is now placed in the second group.

Part I. Foundational axioms

The axioms stated here are foundational, or constitutional axioms. They express the non-negotiable commitments of variorum semantics, already latent in the preceding sections of this document.

Axiom A2. Literal inertness

Literals have no semantic role. μ is invariant under differences that involve only literals.
In particular:

- literals bear no meaning: $M \cap L = \varnothing$;
- differences only in the literals occurring as values in property assignments do not, by themselves, entail differences in meaning.

Axiom A3. No new meaning bearers

μ introduces no new meaning bearers. All semantic dependency is expressed solely in terms of members of M.

Remark This axiom is already enforced by the typing $\mu : M \longrightarrow \mathcal{D}(M)$, since $\mathcal{D}(M)$ contains only dependency terms whose meaning-bearing atoms are members of M. Thus, for every $x \in M$, $\text{atoms}(\mu(x)) \subseteq M$. A3 is stated explicitly to record the corresponding foundational commitment.

Axiom A4. Closure

All semantic dependency is internal: if y affects x, then $y \in M$.

Axiom A5. Reciprocity and irreflexivity

(i) Semantic dependency is mutual.

$$\forall x, y \in M : \quad x \text{ affects } y \iff y \text{ affects } x$$

(ii) Semantic dependency is irreflexive.

$$\forall x \in M : \quad x \text{ does not affect } x$$

Part II. Refinement axioms

The axioms stated here provide representational and structural constraints on meaning functions that satisfy the foundational axioms of Part I.

Axiom A1. Inert dependency terms

For every $x \in M$, $\mu(x)$ is an inert dependency term: it has no evaluation, reduction, or rewrite semantics.
In particular, for any $y \in M$, μ does not compute using $\mu(y)$, that is, $\mu(y)$ never appears in the expression defining $\mu(x)$.

Axiom A6. Local soundness

Within a fixed semantic activation scheme, equal meaning requires equal semantic signatures.
(i) Same-configuration local soundness
Let C be a semantic configuration. For all $x, y \in M_C$:

$$\mu_C(x) = \mu_C(y) \Longrightarrow \mathcal{S}_C^d(x) = \mathcal{S}_C^d(y)$$

(ii) Cross-configuration local soundness
Let C and C' be semantic configurations such that $\Omega_C^d = \Omega_{C'}^d$. For all $x \in M_C \cap M_{C'}$:

$$\mu_C(x) = \mu_{C'}(x) \Longrightarrow \mathcal{S}_C^d(x) = \mathcal{S}_{C'}^d(x)$$

Remark A6(ii) imposes no comparison between meaning functions defined under different semantic activation schemes.

Corollary Equal meaning requires a dependency-graph automorphism
Let C be a semantic configuration with

$$\mathcal{G}_C^d = (M_C, E^d).$$

For all $x, y \in M_C$, if

$$\mu_C(x) = \mu_C(y),$$

then there exists a dependency-graph automorphism φ of $\mathcal{G}_C^d$ such that

$$\varphi(x) = y \qquad \text{and} \qquad \varphi(y) = x.$$

Proof. We omit the fixed configuration index. Assume

$$\mu(x) = \mu(y).$$

By A6(i),

$$\mathcal{S}^d(x) = \mathcal{S}^d(y).$$

Because the signature contains the actual identities of neighbors, for every $z \in M$ and every dependency label $\tau \in \mathcal{T}^d$,

$$(x, \tau, z) \in E^d \iff (y, \tau, z) \in E^d,$$

and

$$(z, \tau, x) \in E^d \iff (z, \tau, y) \in E^d.$$

Let φ swap x and y and fix every other member of M. The two equivalences above imply

$$(u, \tau, v) \in E^d \iff \big(\varphi(u), \tau, \varphi(v)\big) \in E^d$$

for all $u, v \in M$ and every $\tau \in \mathcal{T}^d$. Therefore, φ is a dependency-graph automorphism.

Example Consider the variorum graph (referents have been omitted for simplicity).

$$x \xrightarrow{\text{pu}} k, \text{arity}(x) = 0$$
$$y \xrightarrow{\text{pu}} k, \text{arity}(y) = 0$$

and $M = V_{\text{c}} \cup V_{\kappa} = \{x, y, k\}$, and $\Omega^d = \{\text{pu}^d\}$. Then, $\mathcal{G}^d$ is:

$$x \xrightarrow{\text{pu}^d} k$$
$$y \xrightarrow{\text{pu}^d} k$$

For all $z \in M \cap V$, let $\mu(z) = [\text{pu}^d\text{-parent}(z), \text{pu}^d\text{-children}(z)]$ (with their obvious definitions), so that:

$$\mu(k) = [\bot, \{x, y\}]$$
$$\mu(x) = [k, \{\}]$$
$$\mu(y) = [k, \{\}]$$

Therefore,

$$\mu(x) = \mu(y)$$

It is straightforward to verify that the map φ that swaps x and y and fixes k, that is,

$$\varphi(x) = y,$$
$$\varphi(y) = x,$$
$$\varphi(k) = k,$$

is a dependency-graph automorphism of $\mathcal{G}^d$: it exchanges the two pu^d edges $x \xrightarrow{\text{pu}^d} k$ and $y \xrightarrow{\text{pu}^d} k$ and vice versa, and fixes every other dependency edge.

Example (the redundant witness) Consider the variorum state, with referents omitted for simplicity:

$$x \xrightarrow{\text{pu}} k, \text{arity}(x) = 0$$
$$r \xrightarrow{\text{pu}} R, \text{arity}(r) = 1$$

and perform two F5-transitions:

$$w_1 = \text{bindAndPlace}(r, k; 1 = x)$$
$$w_2 = \text{bindAndPlace}(r, k; 1 = x)$$

Thus,

$$w_1 \xrightarrow{\text{pu}} k \quad w_1 \xrightarrow{\text{df}} r$$
$$w_2 \xrightarrow{\text{pu}} k \quad w_2 \xrightarrow{\text{df}} r$$

and

$$\text{props}(w_1) = \{<<r, 1>, x>\}$$
$$\text{props}(w_2) = \{<<r, 1>, x>\}$$

Let $M = V_c \cup V_\kappa \cup A = \{x, r, k, R, w_1, w_2, <r, 1>\}$, and $\Omega^d = \{\text{pu}^d, \text{df}^d, \text{ao}^d, \text{ap}^d, \text{b}^d, \text{f}^d, \text{r}^d\}$. Then, $\mathcal{G}^d$ is:

$$x \xrightarrow{\text{pu}^d} k$$
$$r \xrightarrow{\text{pu}^d} R$$
$$w_1 \xrightarrow{\text{pu}^d} k$$
$$w_2 \xrightarrow{\text{pu}^d} k$$
$$w_1 \xrightarrow{\text{df}^d} r$$
$$w_2 \xrightarrow{\text{df}^d} r$$
$$<r, 1> \xrightarrow{\text{ao}^d} r$$
$$w_1 \xrightarrow{\text{b}^d} <r, 1>$$
$$w_1 \xrightarrow{\text{f}^d} x$$
$$w_2 \xrightarrow{\text{b}^d} <r, 1>$$
$$w_2 \xrightarrow{\text{f}^d} x$$
$$x \xrightarrow{\text{r}^d} <r, 1>$$

Suppose that besides pu^d-parent/pu^d-children information, $\mu(z)$ also encodes df-lineage and argument binding information:

$$\mu(z) = [\text{pu}^d\text{-parent}(z), \text{pu}^d\text{-children}(z), \text{df}^d\text{-parent}(z),$$
$$\text{df}^d\text{-children}(z), \text{props}(z), \text{roles}(z), \text{fillers}(z)]$$

where $\text{roles}(z)$ encodes, for $z \in M \cap V$, the places where z is used as a binding value, and $\text{fillers}(z)$ encodes, for $z \in M \cap A$, the elements v that are used as binding values when z is used in a binding. So,

$$\mu(w_1) = [k, \{\}, r, \{\}, \{<<r, 1>, x>\}, \{\}, \bot],$$

and

$$\mu(w_2) = [k, \{\}, r, \{\}, \{<<r, 1>, x>\}, \{\}, \bot],$$

so that

$$\mu(w_1) = \mu(w_2).$$

The relevant permutation φ swaps w_1 and w_2 and fixes every other element:

$$\begin{aligned} \varphi(w_1) &= w_2, \\ \varphi(w_2) &= w_1, \\ \varphi(z) &= z, \quad \text{for all } z \in M \setminus \{w_1, w_2\}. \end{aligned}$$

Because φ preserves every typed dependency edge, it is a dependency-graph automorphism of $\mathcal{G}^d$.

Axiom A7. Admissible dependency

Semantic dependency cannot be arbitrary. A meaning bearer can affect only meaning bearers that are semantically reachable from it, or meaning bearers from which it is semantically reachable. Formally,

$$\forall x \in M, \text{atoms}(\mu(x)) \subseteq \mathcal{R}^+\mathcal{G}^d(x) \cup \mathcal{R}^-\mathcal{G}^d(x)$$

Remark Axiom A7 entails axioms A3 and A4.

Axiom A8. Local structural equivariance

Let C and C' be semantic configurations such that $\Omega^d_C = \Omega^d_{C'}$, and let $x \in M_C$ and $x' \in M_{C'}$. If

$$\psi : (\mathcal{G}^d_C[x], x) \longrightarrow (\mathcal{G}^d_{C'}[x'], x')$$

is a pointed dependency-graph isomorphism, then

$$\mu_{C'}(x') = \psi(\mu_C(x)),$$

where ψ is extended to dependency terms by replacing each meaning-bearing atom y by $\psi(y)$, and leaving $\bot$ and all structural notation unchanged.

Remark By A7,

$$\text{atoms}(\mu_C(x)) \subseteq \mathcal{R}^+\mathcal{G}^d_C(x) \cup \mathcal{R}^-\mathcal{G}^d_C(x) \subseteq \mathcal{N}\mathcal{G}^d_C(x).$$

Therefore, the extension of ψ to $\mu_C(x)$ is well defined.

Remark A8 imposes no comparison between meaning functions defined under different semantic activation schemes.

Example Let $\Omega^d_C = \Omega^d_{C'}$, and suppose $\mathcal{G}^d_C$ is

$$x \xrightarrow{\text{pu}^d} k$$

and $\mathcal{G}^d_{C'}$ is

$$x' \xrightarrow{\text{pu}^d} k'$$

Let $\psi : (\mathcal{G}^d_C[x], x) \to (\mathcal{G}^d_{C'}[x'], x')$ be the pointed dependency-graph isomorphism that takes x to x' and k to k':

$$\psi(x) = x', \qquad \psi(k) = k'.$$

Then

$$\mu_{C'}(x') = \psi(\mu_C(x)).$$

Corollary (local equality under independent extension) Let C and C' be semantic configurations such that $\Omega^d_C = \Omega^d_{C'}$, and let $x \in M_C \cap M_{C'}$. If the pointed dependency graphs at x are identical, so that

$$\mathcal{G}^d_C[x] = \mathcal{G}^d_{C'}[x],$$

then

$$\mu_C(x) = \mu_{C'}(x).$$

In particular, if C' is obtained from C by adding semantic structure strictly outside $\mathcal{G}^d_C[x]$, then $\mu_C(x) = \mu_{C'}(x)$. Thus, differences between C and C' outside restricted dependency structures cannot change the meaning of x.

Proof. The identity map on $\mathcal{G}^d_C[x]$ is a pointed dependency-graph isomorphism from $(\mathcal{G}^d_C[x], x)$ to $(\mathcal{G}^d_{C'}[x], x)$. The result follows from A8. □

Example Suppose $\mathcal{G}^d_C$ is

$$x \xrightarrow{\text{pu}^d} k_1$$

and $\mathcal{G}^d_{C'}$ is

$$x \xrightarrow{\text{pu}^d} k_1 \qquad y \xrightarrow{\text{pu}^d} k_2$$

with no dependency edge connecting $\{x, k_1\}$ to $\{y, k_2\}$. Then

$$\mu_C(x) = \mu_{C'}(x), \qquad \mu_C(k_1) = \mu_{C'}(k_1).$$

Corollary (global equivariance under state isomorphism) Let C and C' be semantic configurations. If $\varphi : \Sigma_C \longrightarrow \Sigma_{C'}$ is a state isomorphism such that

$$\varphi[M_C] = M_{C'}$$

and

$$\Omega^d_C = \Omega^d_{C'},$$

then, for every $x \in M_C$,

$$\mu_{C'}(\varphi(x)) = \varphi(\mu_C(x)).$$

This follows because φ induces a dependency-graph isomorphism

$$\varphi : \mathcal{G}^d_C \longrightarrow \mathcal{G}^d_{C'}.$$

Consequently, for every $x \in M_C$, φ maps $\mathcal{NG}^d_C(x)$ bijectively onto $\mathcal{NG}^d_{C'}(\varphi(x))$. Its restriction therefore defines a pointed dependency-graph isomorphism

$$\varphi : (\mathcal{G}^d_C[x], x) \longrightarrow (\mathcal{G}^d_{C'}[\varphi(x)], \varphi(x)).$$

By A8,

$$\mu_{C'}(\varphi(x)) = \varphi(\mu_C(x)).$$

Remark In the special case $C = C'$, where φ is a state automorphism, the result does not in general imply

$$\mu_C(x) = \mu_C(\varphi(x)).$$

It says only that φ transports the meaning of x to the meaning of $\varphi(x)$.

Example (the redundant witness, continued) Let φ be the state automorphism that swaps w_1 and w_2 and fixes every other model element and argument token. Then

$$\begin{aligned}
\mu(\varphi(w_1)) &= \mu(w_2) = [k, \{\}, r, \{\}, \{<<r, 1>, x>\}, \{\}, \bot], \\
\varphi(\mu(w_1)) &= \varphi([k, \{\}, r, \{\}, \{<<r, 1>, x>\}, \{\}, \bot]) \\
&= [k, \{\}, r, \{\}, \{<<r, 1>, x>\}, \{\}, \bot] \\
&= \mu(\varphi(w_1)).
\end{aligned}$$

H. Lexical content

Each model element, and where appropriate each argument token, may be associated with lexical content such as names, labels, descriptions, comments, or documentation.

Lexical content exists solely to support human interpretation, presentation, and communication. It plays no role in the model's relational semantics.

For any object x with associated lexical content $\mathrm{lex}(x)$:

- $\mathrm{lex}(x)$ is not part of the variorum state and does not contribute to $\mu(x)$;
- $\mathrm{lex}(x)$ never occurs as a meaning-bearing atom in any dependency term;
- changes to $\mathrm{lex}(x)$ leave the variorum state, the primary dependency graph, and all meanings invariant;
- changes to the variorum state or to meaning do not, by themselves, determine changes to $\mathrm{lex}(x)$.

Conclusion

The specification of *variorum$^{\psi}$*, a novel relational-semantic model, has been presented in detail. The focus has been on the formal description of the model: its elements, relations, arguments, and bindings; the validity conditions governing its states; the transitions by which those states evolve; and the axiomatic foundation of its relational semantics.

The main text specifies the model and its semantics, while Appendix A establishes two supporting metatheoretical results. First, it gives a set of sufficient conditions for state validity. Second, it proves the validity of every state reachable from the initial state by a finite sequence of applicable transitions. Thus, the transition system preserves state validity.

Several practical and theoretical questions have deliberately been left for future work. These include the systematic study of the realizability of the axiomatic system—that is, the construction and classification of systems of configuration-indexed meaning functions μ_C satisfying axioms A1–A8—as well as the model's implementability, its representational and reasoning capabilities, and the characterization of the class of knowledge-representation problems for which a model of purely relational semantics may be useful.

Appendix B presents a realizability witness for one nontrivial class of semantic configurations.

Remark The state validity rules and transitions have already been implemented in software. Information about the current status of the project is available on the project website, `https://variorum.bilberry.gr`.

Acknowledgements

The author wishes to thank:

- *Aliki Balser*, the fellow traveler on many literal and conceptual walks through the project's nascent ideas;
- *Anna Mavrou* and *Dionysis Balser-Altanis*, for their critical feedback and support during the development of this specification, and for their broader contribution to the *variorum$^{\psi}$* project;
- *Vasiliki (Vasia) Georgiopoulou* and *Efstratios (Stratos) Mavros*, for patiently listening to its early formulations, and for their thoughtful input;
- *Aliki, Michalis, Dionysis, and* ***Mufa***, for being there.

Appendix A: State validity and its preservation

Proof of the validity lemma

- **E1.** Since the placement forest $(V_c \cup V_\kappa, E_{pu})$ is nonempty, it has at least one root. Since the roots of the forest are exactly the members of $V_{root} \subseteq V_\kappa$, it follows that $V_\kappa \neq \varnothing$.
- **E2.** Concepts $c \in V_c$ cannot be roots in the placement forest, so they must have a unique parent in a placement tree.
- **E3.** Every nonroot category is a member of the placement forest, and since it is not a member of V_{root} it has a unique pu-parent.
- **E4.** By the definition of the placement forest $(V_c \cup V_\kappa, E_{pu})$, every pu-edge has endpoints in $V_c \cup V_\kappa$.
- **E5.** Let $(u, v) \in E_{pu}$ and $v \in V_c$. By E4, $u \in V_c \cup V_\kappa$. If $u \in V_\kappa$, then the extra hypothesis gives $v \in V_\kappa$, contradicting $v \in V_c$.
- **E6.** By the placement forest acyclicity.
- **E7.** By the surjectivity of ρ.
- **E8.** Since the graph of ρ equals E_{cb}, $\forall e \in E_{cb}$: source$(e) \in V_r$, target$(e) \in V_c$.
- **E9.** Since $\rho : V_o \longrightarrow V_r$ is a function, each $o \in V_o$ has exactly one incoming cb-source, namely $\rho(o)$. Therefore, no concept can contextualize more than one referent.
- **E10.** By the definition of the derivation forest (V_c, E_{df}), every endpoint of a df-relation is a member of V_c.
- **E11.** By the definition of the derivation forest (V_c, E_{df}), every vertex in a df-tree has at most one df-parent.
- **E12.** By the fact that the df-relations form a forest.
- **E13.** Since E_{cb} is the graph of $\rho : V_o \longrightarrow V_r$, the targets of E_{cb} are exactly the members of $V_o = V_c \setminus V_{derived}$. Thus directly contextualized concepts are exactly the nonderived df-roots. Therefore, no derived concept is directly contextualized, and no directly contextualized concept is derived.
- **E14.** Let $c \in V_c$. If $c \in V_o$, then there is a unique $r = \rho(c) \in V_r$ such that $(r, c) \in E_{cb}$. If $c \notin V_o$, then $c \in V_{derived}$. Thus, c belongs to a df-tree and is not its root. Let c' be the root of this tree. Since $c' \in V_o$, there is a unique $r = \rho(c') \in V_r$ such that $(r, c') \in E_{cb}$. Hence, c has a forward-maximal df-path to a concept that directly contextualizes a referent. Uniqueness follows from the uniqueness of the root and from the functionality of ρ.

□

Proof of the preservation theorem

The result will be proved by induction on the length of the transition sequence. It will be shown, in particular, that every state reached from Σ_0 satisfies the conditions of the validity lemma.

Base case. Σ_0 contains no concepts, therefore the provisions of section D are vacuously satisfied. Hence, Σ_0 is a variorum state.

The placement graph consists of a single category κ_0 and has no edges. It is therefore a finite nonempty directed forest whose unique root is κ_0, which is exactly the unique member of V_{root}. The additional category-placement condition is vacuously satisfied.

The derivation graph and the sets V_c, V_r, and V_o are empty. Hence (V_c, E_{df}) is an empty forest whose root set is $V_o = \varnothing$, and the empty function

$$\rho : \varnothing \longrightarrow \varnothing$$

is surjective and has graph $E_{cb} = \varnothing$.

Thus, Σ_0 satisfies the sufficient conditions for state validity, hence Σ_0 is a valid variorum state.

Inductive step. Let $\Sigma = (V, E)$ be a variorum state satisfying the sufficient conditions for validity, and let an applicable transition produce the state $\Sigma' = (V', E')$. We verify that Σ' is a variorum state satisfying the sufficient conditions for validity.

F1.

The transition adds a new category κ as an isolated vertex of the placement forest:

$$V'_\kappa = V_\kappa \cup \{\kappa\}.$$

Since no pu-edge is added, κ is a new root, so

$$V'_{\text{root}} = V_{\text{root}} \cup \{\kappa\}.$$

The placement graph remains a finite nonempty forest, and the additional category-placement condition continues to hold.

The derivation and contextualization structures are unchanged:

$$V'_o = V_o, \qquad V'_r = V_r, \qquad E'_{cb} = E_{cb}.$$

Define $\rho' = \rho$. Since $V'_o = V_o$, $V'_r = V_r$, and $E'_{cb} = E_{cb}$, the function ρ' is surjective and satisfies

$$E'_{cb} = \{(\rho'(o), o) \mid o \in V'_o\}.$$

The transition creates no concept, so there are no new argument or property data to verify; the provisions of section D are vacuously satisfied.

F2.

The transition adds a new concept c as a placement leaf under p:

$$V'_c = V_c \cup \{c\},$$

so the placement graph remains a finite nonempty forest, with the same roots as before. Since $t(c) = C$, $V'_\kappa = V_\kappa$ and the additional category-placement condition continues to hold. The new concept becomes an isolated vertex of the derivation forest. Therefore,

$$V'_o = V_o \cup \{c\}.$$

A new referent r is also added:

$$V'_{\mathrm{r}} = V_{\mathrm{r}} \cup \{r\},$$

together with the cb-relation (r, c):

$$E'_{\mathrm{cb}} = E_{\mathrm{cb}} \cup \{(r, c)\}.$$

Define $\rho' : V'_{\mathrm{o}} \longrightarrow V'_{\mathrm{r}}$ by

$$\rho'(o) = \rho(o) \text{ for } o \in V_{\mathrm{o}}, \qquad \rho'(c) = r.$$

Since ρ is surjective onto V_{r}, ρ' is surjective onto V'_{r}. Moreover,

$$E'_{\mathrm{cb}} = \{(\rho'(o), o) \mid o \in V'_{\mathrm{o}}\}.$$

The transition creates the concept c according to the first case of the formation principle of section D. Hence, the argument and property data of c satisfy the provisions of section D.

F3.

The transition adds a new concept c as a placement leaf under p:

$$V'_{\mathrm{c}} = V_{\mathrm{c}} \cup \{c\},$$

so the placement graph remains a finite nonempty forest, with the same roots as before. Since $t(c) = C$, $V'_{\kappa} = V_{\kappa}$ and the additional category-placement condition continues to hold. The new concept becomes an isolated vertex of the derivation forest. Therefore,

$$V'_{\mathrm{o}} = V_{\mathrm{o}} \cup \{c\}.$$

No new referent is added:

$$V'_{\mathrm{r}} = V_{\mathrm{r}},$$

while a cb-relation (r, c) is added:

$$E'_{\mathrm{cb}} = E_{\mathrm{cb}} \cup \{(r, c)\}.$$

Define $\rho' : V'_{\mathrm{o}} \longrightarrow V'_{\mathrm{r}}$ by

$$\rho'(o) = \rho(o) \text{ for } o \in V_{\mathrm{o}}, \qquad \rho'(c) = r.$$

Since ρ is surjective onto V_{r}, ρ' is surjective onto V'_{r}. Moreover,

$$E'_{\mathrm{cb}} = \{(\rho'(o), o) \mid o \in V'_{\mathrm{o}}\}.$$

The transition creates the concept c according to the first case of the formation principle of section D. Hence, the argument and property data of c satisfy the provisions of section D.

F4.

The transition adds a new category κ as a placement leaf under the category p:

$$V'_{\kappa} = V_{\kappa} \cup \{\kappa\}, \qquad E'_{\mathrm{pu}} = E_{\mathrm{pu}} \cup \{(\kappa, p)\}.$$

Since κ is new, this cannot create a cycle, so the placement graph remains a finite nonempty forest, with the same roots as before. Because $\kappa, p \in V_{\kappa}$, the additional category-placement condition continues to

hold. The derivation and contextualization structures are unchanged:

$$V'_{\mathrm{o}} = V_{\mathrm{o}}, \qquad V'_{\mathrm{r}} = V_{\mathrm{r}}, \qquad E'_{\mathrm{cb}} = E_{\mathrm{cb}}.$$

Define $\rho' = \rho$. Since $V'_{\mathrm{o}} = V_{\mathrm{o}}$, $V'_{\mathrm{r}} = V_{\mathrm{r}}$, and $E'_{\mathrm{cb}} = E_{\mathrm{cb}}$, the function ρ' is surjective and satisfies

$$E'_{\mathrm{cb}} = \{(\rho'(o), o) \mid o \in V'_{\mathrm{o}}\}.$$

The transition creates no concept, so there are no new argument or property data to verify; the provisions of section D are vacuously satisfied.

F5 and F6.

Each transition adds a new concept c as a placement leaf under p:

$$V'_{\mathrm{c}} = V_{\mathrm{c}} \cup \{c\},$$

so the placement graph remains a finite nonempty forest, with the same roots as before. Since $t(c) = C$, $V'_{\kappa} = V_{\kappa}$ and the additional category-placement condition continues to hold.

The transition also adds c as a leaf of the derivation tree containing v, together with the df-relation (c, v):

$$E'_{\mathrm{df}} = E_{\mathrm{df}} \cup \{(c, v)\}.$$

Thus the derivation graph remains a finite forest, with the same roots as before. Since c is derived,

$$V'_{\mathrm{o}} = V_{\mathrm{o}}.$$

No referent or cb-relation is added or removed:

$$V'_{\mathrm{r}} = V_{\mathrm{r}}, \qquad E'_{\mathrm{cb}} = E_{\mathrm{cb}}.$$

Define $\rho' = \rho$. Since $V'_{\mathrm{o}} = V_{\mathrm{o}}$, $V'_{\mathrm{r}} = V_{\mathrm{r}}$, and $E'_{\mathrm{cb}} = E_{\mathrm{cb}}$, the function ρ' is surjective and satisfies

$$E'_{\mathrm{cb}} = \{(\rho'(o), o) \mid o \in V'_{\mathrm{o}}\}.$$

F5 creates the new concept according to the second case of the formation principle of section D; F6 creates the new concept according to the third case of the formation principle. Hence, the argument and property data of c satisfy the provisions of section D.

F7.

The transition removes the pu-relation from the nonroot category κ to its parent:

$$E'_{\mathrm{pu}} = E_{\mathrm{pu}} \setminus \{(\kappa, \mathrm{parent}(\kappa))\}.$$

Removing this edge splits one placement tree into two trees and cannot create a cycle. Therefore, the placement graph remains a finite nonempty forest. The category κ becomes a new root:

$$V'_{\mathrm{root}} = V_{\mathrm{root}} \cup \{\kappa\}.$$

Since no placement edge is added, the additional category-placement condition continues to hold. The

derivation and contextualization structures are unchanged:

$$V'_{\mathrm{o}} = V_{\mathrm{o}}, \qquad V'_{\mathrm{r}} = V_{\mathrm{r}}, \qquad E'_{\mathrm{cb}} = E_{\mathrm{cb}}.$$

Define $\rho' = \rho$. Since $V'_{\mathrm{o}} = V_{\mathrm{o}}$, $V'_{\mathrm{r}} = V_{\mathrm{r}}$, and $E'_{\mathrm{cb}} = E_{\mathrm{cb}}$, the function ρ' is surjective and satisfies

$$E'_{\mathrm{cb}} = \{(\rho'(o), o) \mid o \in V'_{\mathrm{o}}\}.$$

The transition creates no concept, so there are no new argument or property data to verify; the provisions of section D are vacuously satisfied.

F8.

(i) If v has a pu-parent q in the current state Σ, the transition replaces the relation (v, q) by (v, p):

$$E'_{\mathrm{pu}} = (E_{\mathrm{pu}} \setminus \{(v, q)\}) \cup \{(v, p)\}.$$

If, instead, v is a root in the current state Σ, the transition adds the relation (v, p):

$$E'_{\mathrm{pu}} = E_{\mathrm{pu}} \cup \{(v, p)\}.$$

In either case, the placement subtree induced by $\{v\} \cup R_{\mathrm{pu}}(v)$ is unchanged. Since $p \neq v$ and $p \notin R_{\mathrm{pu}}(v)$, the new relation (v, p) cannot create a cycle. Thus, the placement graph remains a finite nonempty forest.

(ii) If $v \notin V_{\mathrm{root}}$, the roots remain unchanged:

$$V'_{\mathrm{root}} = V_{\mathrm{root}}.$$

If, instead, $v \in V_{\mathrm{root}}$, then v ceases to be a root:

$$V'_{\mathrm{root}} = V_{\mathrm{root}} \setminus \{v\}.$$

The latter set remains nonempty because p already belongs to a placement tree whose root is different from v.

(iii) If $v \in V_{\kappa}$, the precondition requires $p \in V_{\kappa}$. Therefore, the additional category-placement condition continues to hold.

(iv) The derivation and contextualization structures are unchanged:

$$V'_{\mathrm{o}} = V_{\mathrm{o}}, \qquad V'_{\mathrm{r}} = V_{\mathrm{r}}, \qquad E'_{\mathrm{cb}} = E_{\mathrm{cb}}.$$

Define $\rho' = \rho$. Since $V'_{\mathrm{o}} = V_{\mathrm{o}}$, $V'_{\mathrm{r}} = V_{\mathrm{r}}$, and $E'_{\mathrm{cb}} = E_{\mathrm{cb}}$, the function ρ' is surjective and satisfies

$$E'_{\mathrm{cb}} = \{(\rho'(o), o) \mid o \in V'_{\mathrm{o}}\}.$$

The transition creates no concept, so there are no new argument or property data to verify; the provisions of section D are vacuously satisfied.

F9.

By the preconditions, there is no pu-edge or df-edge whose target is v. Thus, v has no children in either the placement forest or the derivation forest.

Removing v, together with its outgoing pu-relation if one exists, removes a leaf from the placement forest. Hence the placement graph remains a forest.

If $v \notin V_{\text{root}}$, then

$$V'_{\text{root}} = V_{\text{root}}.$$

If, instead, $v \in V_{\text{root}}$:

$$V'_{\text{root}} = V_{\text{root}} \setminus \{v\},$$

which is nonempty by the precondition $V_{\text{root}} \setminus \{v\} \neq \varnothing$. In either case, $V'_{\text{root}} \neq \varnothing$, so the placement graph remains a finite nonempty forest. Since the transition adds no placement edge, the additional category-placement condition continues to hold.

There are three cases for the derivation and contextualization structures.

(i) $v \in V_{\kappa}$. The derivation and contextualization structures are unchanged:

$$V'_{\text{o}} = V_{\text{o}}, \qquad V'_{\text{r}} = V_{\text{r}}, \qquad E'_{\text{cb}} = E_{\text{cb}}.$$

Define $\rho' = \rho$. Then ρ' is surjective and satisfies

$$E'_{\text{cb}} = \{(\rho'(o), o) \mid o \in V'_{\text{o}}\}.$$

(ii) $v \in V_{\text{c}} \setminus V_{\text{o}} = V_{\text{derived}}$. Removing v, together with its outgoing df-relation, removes a nonroot leaf from the derivation forest. Therefore the derivation forest remains a forest with the same roots:

$$V'_{\text{o}} = V_{\text{o}}.$$

Since $v \notin V_{\text{o}}$, no cb-relation has target v. Hence

$$V'_{\text{r}} = V_{\text{r}}, \qquad E'_{\text{cb}} = E_{\text{cb}}.$$

Define $\rho' = \rho$. Then ρ' is surjective and satisfies

$$E'_{\text{cb}} = \{(\rho'(o), o) \mid o \in V'_{\text{o}}\}.$$

(iii) $v \in V_{\text{o}}$. Then no df-edge has source v. Therefore, the derivation tree rooted at v consists only of v. Removing v therefore removes an isolated root:

$$V'_{\text{o}} = V_{\text{o}} \setminus \{v\}.$$

Let $r = \rho(v)$. The transition removes the relation (r, v), so

$$E'_{\text{cb}} = E_{\text{cb}} \setminus \{(r, v)\}.$$

If there exists an $o \in V'_{\text{o}}$ such that $\rho(o) = r$, then

$$V'_{\text{r}} = V_{\text{r}}.$$

Otherwise, r has no remaining contextualization relation and is removed, so

$$V'_{\text{r}} = V_{\text{r}} \setminus \{r\}.$$

In either case, define $\rho' : V'_{\mathrm{o}} \longrightarrow V'_{\mathrm{r}}$ by:

$$\rho'(o) = \rho(o), \qquad o \in V'_{\mathrm{o}}.$$

Then ρ' is surjective onto V'_{r} and satisfies

$$E'_{\mathrm{cb}} = \{(\rho'(o), o) \mid o \in V'_{\mathrm{o}}\}.$$

The transition creates no concept, so there are no new argument or property data to verify; the provisions of section D are vacuously satisfied.

Thus, every applicable transition preserves the formation requirements of sections B–D and the three sufficient conditions of the validity lemma. By induction, every state reachable from Σ_0 by a finite sequence of applicable transitions is a variorum state satisfying those three conditions. By the validity lemma, every such state is valid. □

Appendix B: A realizability witness for the semantic axioms

Let Σ be any valid state and define:

$$M = V_{\mathrm{c}} \cup V_{\kappa}, \qquad \Omega^d = \{\mathrm{pu}^d\}.$$

Then $M \cap V \neq \varnothing$ and $M \cap V_{\mathrm{r}} = M \cap A = \varnothing$, so (M, Ω^d) is an admissible semantic basis for Σ. By E4, every participant in a pu-relation is a concept or category, and therefore every participant in an active pu^d-dependency belongs to M.

For every $x \in M$, define $\mathrm{parent}^{\mathrm{d}}(x)$ by:

$$\mathrm{parent}^{\mathrm{d}}(x) = \mathrm{parent}(x),$$

and $\mathrm{children}^{\mathrm{d}}(x)$ by:

$$\mathrm{children}^{\mathrm{d}}(x) = \{\, y \in M \mid (y, \mathrm{pu}^d, x) \in E^d \,\}.$$

Then the function

$$\mu_{(\Sigma, M, \Omega^d)}(x) = [\mathrm{parent}^{\mathrm{d}}(x), \mathrm{children}^{\mathrm{d}}(x)]$$

is readily shown to satisfy axioms A1–A8:

Proof. A1–A4 hold immediately from the definition: the displayed pair is an inert dependency term containing only members of M and $\perp$, and no literals. It is independent of all literal-valued property assignments and introduces no atoms outside M. For A5(i), y occurs in $\mu(x)$ exactly when x and y are connected by a pu^d-dependency: if y is the parent of x, then x is a child of y, and conversely; for A5(ii), by E6 no element is its own parent or child. A6(i) follows because, within a configuration, equality of the displayed pairs entails equality of the corresponding incoming and outgoing pu^d-dependencies (same parent, same children). The same argument establishes A6(ii) for configurations with the common activation scheme $\Omega^d = \{\mathrm{pu}^d\}$. A7 holds because every atom occurring in $\mu(x)$ is connected to x by a single pu^d-dependency. Finally, A8 follows because, for configurations with the common activation scheme $\Omega^d = \{\mathrm{pu}^d\}$, every pointed dependency-graph isomorphism preserves and reflects pu^d-dependencies, yielding the same parent–children pair up to transport by the isomorphism. □

Publication and Project Information

Publication information

This document is the official Version 1.0 release of *The variorum$^{\psi}$ Model: Foundations*, published in July 2026 and deposited in Zenodo.

Recommended citation: Altanis, Giorgos. *The variorum$^{\psi}$ Model: Foundations*. Version 1.0, July 2026. Zenodo. https://doi.org/10.5281/zenodo.21331591.

Copyright and permissions

Project information

Project website: `https://variorum.bilberry.gr`

Contact: variorum@bilberry.gr