The Automatic Deduction of Classificatory Systems from Linguistic Theories ⋆

PAUL JOHN KING
Seminar für Sprachwissenschaft, Eberhard-Karls-Universität, Tübingen, Germany
E-mail: king@sfs.nphil.uni-tuebingen.de

KIRIL IVANOV SIMOV
Laboratory for Linguistic Modelling, Bulgarian Academy of Sciences, Sofia, Bulgaria
E-mail: kivs@bgcict.acad.bg

Abstract. Classifying linguistic objects is a widespread and important linguistic task, but hand deducing a classificatory system from a general linguistic theory can consume much effort and introduce pernicious errors. We present an abstract prototype device that effectively deduces an accurate classificatory system from a finite linguistic theory.

Key words: classification, constraint grammars, knowledge management

Abbreviations: HPSG – head-driven phrase structure grammar; SRL – speciate re-entrant logic

1. Introduction

A *classificatory system* over a set U of objects comprises

a *classification* C over U, a set of labels that indicate subsets of U, and

an *index* from U to C, an algorithm that effectively deduces from limited information about an object in U a label in C that indicates a set to which the object belongs.

Classifying linguistic objects is a widespread and important linguistic task, e.g., the benefits of classifying words according to their inflectional paradigms are apparent for an inflectionally simple language, such as English, and even more so for an inflectionally complex language, such as Bulgarian, in which inflection is a ubiquitous and indispensable part of syntactic processing. Indeed, the MORPHO-ASSISTANT system of Simov et al. (1990) and Simov et al. (1992) builds a dictionary-acquisition module by means of a classificatory system over Bulgarian inflectional morphology built by Paskaleva, Avgustinova, Damova and Simov.

Some linguistic theories give classificatory systems implicitly, others explicitly. Linguists usually prefer implicit classificatory systems, since they enable the expression of linguistic intuitions as general theories, but computer scientists usually

⋆ This paper is a substantially improved version of King and Simov (1997), and is due in equal parts to both authors.

Grammars **1:** 103–153, 1998.

prefer explicit classificatory systems, since they enable such tasks as fast look-up of tabulated data, automated knowledge acquisition, controlled linguistic inference, etc. Thus, computational linguistics often requires the deduction of a classificatory system from a general linguistic theory. However, hand deducing a classificatory system from a theory can consume much effort, and risks the system unwittingly violating the theory from which it is deduced, with disastrous consequences if, say, the index assigns some object a label that indicates a set to which the object does not belong. Thus, a device that effectively deduces an accurate classificatory system from a general linguistic theory would greatly benefit computational linguistics. We present here an abstract prototype of such a device.

Our experiences show that this device should ideally deduce a classificatory system possessing at least the following five qualities:

(Q1) Distinct labels in the classification should inherently indicate disjoint sets of objects, to ensure that each object receives a unique label.

(Q2) No label in the classification should inherently indicate an empty set of objects, to prune out intrinsically useless labels.

(Q3) The labels of the classification should indicate sets that are neither too small nor too big, to ensure that the classification truly captures the intuitions of the theory from which it is deduced.

(Q4) The index should correctly assign each object the label that indicates the set to which the object belongs, for obvious reasons.

(Q5) The index should efficiently assign each object the label that indicates the set to which the object belongs, to facilitate rapid classification of objects.

Qualities (Q3) and (Q5) require control mechanisms that would deduce an optimal classificatory system from a linguistic theory. However, in this paper we address only qualities (Q1), (Q2) and (Q4) in order to establish that effectively deducing a classificatory system from a general linguistic theory is at all possible.

We structure this paper as follows. Section 2 first illustrates a classificatory system by giving a simple linguistic example, then motivates the need for a device that automatically deduces classificatory systems from general linguistic theories by sketching a linguistically important classificatory system that cannot be easily hand deduced. Section 3 reviews the formal language in which our linguistic theories are written. Section 4 shows that a special disjunctive normal form, called an *exclusive matrix*, constitutes a classification over the denotation of the matrix with qualities (Q1) and (Q2). Section 5 shows that a fixed algorithm, called `Clause`, and a finite structure of queries and responses, called an *index tree*, that analyzes an exclusive matrix together constitute an index from the denotation of the matrix to the matrix with quality (Q4). Section 6 gives our device as two algorithms, called `Class` and `Index`. `Class` effectively deduces from each finite theory an exclusive matrix that is semantically equivalent to the theory, and `Index` effectively deduces from each exclusive matrix an index tree that analyzes the matrix. Thus, our device effectively deduces from each finite theory a classificatory system over the denotation of the theory with qualities (Q1), (Q2) and (Q4). Section 7 gives a simple example of our

device in action, and Section 8 concludes the paper. We use the following notation throughout. For each set X, for each set Y, we write

$X \subseteq Y$ iff X is a subset of Y, and

$X \subset Y$ iff X is a proper subset of Y.

For each set X, we write

$\mathrm{card}(X)$ for the cardinality of X.

$\mathrm{pow}(X)$ for the set of subsets of X,

$\mathrm{finpow}(X)$ for the set of finite subsets of X,

$X^\sharp$ for the set of finite strings of members of X,

X^* for the set of finite sequences of members of X, and

X^+ for the set of finite and nonempty sequences of members of X.

We write ε for the empty string. For each finite string ω, we write $\mathrm{length}(\omega)$ for the length of ω. Thus, $\mathrm{length}(x_1 \ldots x_n) = n$. For each nonempty and finite $X \subseteq \mathbb{N}$, we write

$\max(X)$ for the largest member of X, and

$\min(X)$ for the smallest member of X.

2. Illustration and Motivation

Classifying words according to their inflectional paradigms is a simple classificatory system, e.g. each German noun has eight declensions, a singular and a plural declension for each of the cases nominative, accusative, genitive and dative. The declension of each German noun can be classified according to how its nominative singular declension is modified by affixation and umlauting, and almost all German nouns exhibit one of the following patterns:[1]

	Singular				Plural			
	Nom.	Acc.	Gen.	Dat.	Nom.	Acc.	Gen.	Dat.
(D1)	—	—	—s	—	—	—	—	—n
(D2)	—	—	—(s)	—	¨—	¨—	¨—	¨—n
(D3)	—	—	—(s)	—	—e	—e	—e	—en
(D4)	—	—	—(s)	—	¨—e	¨—e	¨—e	¨—en
(D5)	—	—	—s	—	—er	—er	—er	—ern
(D6)	—	—	—s	—	¨—er	¨—er	¨—er	¨—ern
(D7)	—	—	—(s)	—	—s	—s	—s	—s
(D8)	—	—	—(s)	—	—n	—n	—n	—n
(D9)	—	—n	—n(s)	—n	—n	—n	—n	—n

Thus, (D1)–(D9) is a classification over German nouns in which each pattern is a label that indicates the set of German nouns whose declensions exhibit the pattern. There is also an index, since the nominative singular and genitive singular declen-

sions of a noun determine all of its singular declensions, and the nominative plural declension further determines all of its plural declensions.[2]

However, not all German nouns exhibit a regular declension pattern, e.g., the German noun ‘Herz’ (heart) has the following declensions:

Singular				Plural			
Nom.	Acc.	Gen.	Dat.	Nom.	Acc.	Gen.	Dat.
Herz	Herz	Herzens	Herzen	Herzen	Herzen	Herzen	Herzen

No pattern in (D1)–(D9) indicates a set containing ‘Herz’. Even worse, the index wrongly assigns ‘Herz’ pattern (D9), giving the accusative singular declension of ‘Herz’ as ‘Herzen’. Clearly, we can modify our classification to provide a label that indicates a set containing ‘Herz’ by adding the following pattern:

(D10)	—	—	—ns	—n	—n	—n	—n	—n

We can also modify our index in order to assign ‘Herz’ the appropriate pattern. To classify a noun, if the genitive singular declension appends nothing or ‘s’ to the nominative singular form then the declension pattern is among (D1)–(D8), and the nominative plural declension determines all declensions of the noun. If the genitive singular declension appends ‘n’ to the nominative singular declension, then the declension pattern of the noun must be (D9). If the genitive singular declension appends ‘ns’ to the nominative singular declension then the declension pattern is (D9) or (D10), and the accusative singular declension determines all declensions of the noun.

In order to better motivate the need for a device to automatically deduce classificatory systems from general linguistic theories, we now give an example of a linguistically important classificatory system that does not presently exist, and whose hand deduction would involve an inordinate amount of work. In a highly lexicalized grammar, the lexical entries in a theory of a natural language constrain word behavior. Such constraints apply both internally to limit the structure of individual words, and externally to limit the possible sister constituents that can occupy specific positions within larger linguistic structures. Subcategorization is an example of the external application of such constraints, e.g. in the HPSG (head-driven phrase structure grammar) of Pollard and Sag (1987) and Pollard and Sag (1994), the principle type of linguistic structure is the *sign*, where signs include both words and phrases. Each sign bears, among other things, a complex of syntactic and semantic objects called a *synsem object*. In turn, each synsem object bears, among other things, a finite list of synsem objects called a *subcat list*. The subcat

list serves to restrict which signs a given sign can take as sister constituents within a larger sign.

> 'The SUBCAT value of a sign is in essence the sign's valence, that is, a specification of what other signs the sign in question must combine with in order to become *saturated*. More precisely, the SUBCAT value is a list of *synsem* objects, corresponding to the SYNSEM values of the other signs selected as complements by the sign in question.' Pollard and Sag (1994, p. 23)

Suppose that a new lexical entry, constraining a new word, is added to a HPSG theory. Two converse problems may arise. The new constraints on the subcat list of the new word and the existing constraints on the synsem objects of the existing words may wrongly allow or forbid some existing words to occupy certain positions on the subcat list of the new word. Conversely, the new constraints on the synsem object of the new word and the existing constraints on the subcat lists of the existing words may wrongly allow or forbid the new word to occupy certain positions on the subcat lists of some existing words. To prevent both problems, the writer of a lexicon for a HPSG theory must bear in mind the synsem objects and subcat lists that the theory currently licenses, and how they may interact with the synsem objects and subcat lists that each new lexical entry licenses. This is a daunting, perhaps overwhelming, task as the lexicon grows to license tens or hundreds of thousands of words. In practice, the lexicon writer informally deduces from the current theory tacit classifications over synsem objects and subcat lists, and ensures that each new lexical entry licenses words whose synsem objects and subcat lists fall into sets indicated by appropriate labels in the classifications. However, the informality of the classifications can lead the lexicon writer to overlook subtle or unexpected interactions between synsem objects and subcat lists, and the tacitness of the classifications can allow a lexicon writer working as part of a team to add a lexical entry that accords with their idiosyncratic classifications but conflicts with those of their colleagues. Thus, it is good practice for lexicon writers to work with explicit and formally proven classifications, particularly if the lexicon is intended to be large or produced by a team of writers. The utility of a device, such as ours, that automatically deduces classificatory systems from general linguistic theories is obvious in such circumstances.

3. Speciate Re-entrant Logic

The SRL (speciate re-entrant logic) of King (1989) has a formal language designed specifically to express linguistic theories formulated within the HPSG paradigm of Pollard and Sag (1994). Here we review only the formal language of SRL and its use to express HPSG linguistic theories. We direct readers interested in the logic and model-theory of SRL to Aldag (1997), Kepser (1994), King (1989), King (1994), King et al. (1998) and Pollard (1998).

Underlying a typical interpretable logic is the intuition that an assertion is a finite and syntactically well-formed string of symbols that is either true or false

when interpreted. Underlying SRL is the intuition that a description is a finite and syntactically well-formed string of symbols that is either true or false *of an object* when interpreted. For example, the English description 'it is black' is true of a soot particle but false of a snow flake when interpreted. To capture these intuitions, SRL provides a class of formal languages, each comprising a signature and a class of interpretations. Each signature provides the nonlogical symbols from which descriptions are syntactically constructed. Each interpretation provides a universe of objects, and assigns each nonlogical symbol a meaning from which is generated a denotation function that assigns each description the subset of the universe comprising those objects of which the description is true. In this paper we work in a subclass of SRL comprising those formal languages with what we call a *computable* signature, since this ensures the termination of our subsequent algorithms.

Definition 1. *Triple* $\langle \mathsf{Spec}, \mathsf{Feat}, \mathsf{Appr} \rangle$ *is a* computable signature *iff*

Spec *is a finite set,*

Feat *is a countable set, and*

Appr *is a total recursive function from* $\mathsf{Spec} \times \mathsf{Feat}$ *to* $\mathrm{pow}(\mathsf{Spec})$.

If $\mathsf{Sign} = \langle \mathsf{Spec}, \mathsf{Feat}, \mathsf{Appr} \rangle$ is a computable signature then we call each member of Spec a *species* in Sign, each member of Feat a *feature* in Sign, and Appr the *appropriateness function* in Sign. For notational facility, we henceforth work with respect to an implicit computable signature $\mathsf{Sign} = \langle \mathsf{Spec}, \mathsf{Feat}, \mathsf{Appr} \rangle$, and assume that none of the symbols :, $\sim$, $\approx$, $\not\approx$, $\neg$, $\wedge$, $\vee$, $\rightarrow$, [and] is a species or a feature.

Definition 2. *Triple* $I = \langle U, S, F \rangle$ *is an* interpretation *iff*

U *is a set,*

S *is a total function from* U *to* Spec,

F *is a total function from* Feat *to the set of partial functions from* U *to* U, *and*

for each $\varphi \in \mathsf{Feat}$, *for each* $\upsilon \in U$,

$F(\varphi)(\upsilon)$ *is defined iff* $\mathsf{Appr}\langle S(\upsilon), \varphi \rangle \neq \emptyset$, *and*

if $F(\varphi)(\upsilon)$ *is defined then* $S(F(\varphi)(\upsilon)) \in \mathsf{Appr}\langle S(\upsilon), \varphi \rangle$.

Suppose that $I = \langle U, S, F \rangle$ is an interpretation. We call U the *universe* in I, each member of U an *object* in I, S the *species assignment function* in I, and F the *feature denotation function* in I. Each species denotes one member of a set of disjoint sets that cover U, and S assigns each object in I the species that denotes the unique set to which the object belongs: for each $\sigma \in \mathsf{Spec}$, σ denotes $\{\upsilon \in U \mid S(\upsilon) = \sigma\}$. Each feature denotes a partial function from U to U, and F assigns each feature the partial function it denotes: for each $\varphi \in \mathsf{Feat}$, φ denotes $F(\varphi)$. The appropriateness function encodes – and the last clause in the definition of an interpretation enforces – a relationship between the denotations of species and features: for each $\sigma \in \mathsf{Spec}$,

for each $\varphi \in \mathsf{Feat}$, if $\mathsf{Appr}\langle\sigma, \varphi\rangle = \emptyset$ then the denotation of φ acts upon no object in the denotation of σ, and if $\mathsf{Appr}\langle\sigma, \varphi\rangle \neq \emptyset$ then the denotation of φ acts upon each object in the denotation of σ to yield an object in the denotation of some species in $\mathsf{Appr}\langle\sigma, \varphi\rangle$.

Definition 3. Term *and* Desc *are the smallest sets such that*
$: \in \mathsf{Term}$,
for each $\tau \in \mathsf{Term}$, *for each* $\varphi \in \mathsf{Feat}$, $\tau\varphi \in \mathsf{Term}$,
for each $\tau \in \mathsf{Term}$, *for each* $\sigma \in \mathsf{Spec}$, $\tau \sim \sigma \in \mathsf{Desc}$,
for each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$, $\tau_1 \approx \tau_2 \in \mathsf{Desc}$,
for each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$, $\tau_1 \not\approx \tau_2 \in \mathsf{Desc}$,
for each $\delta \in \mathsf{Desc}$, $\neg\delta \in \mathsf{Desc}$,
for each $\delta_1 \in \mathsf{Desc}$, *for each* $\delta_2 \in \mathsf{Desc}$, $[\delta_1 \wedge \delta_2] \in \mathsf{Desc}$,
for each $\delta_1 \in \mathsf{Desc}$, *for each* $\delta_2 \in \mathsf{Desc}$, $[\delta_1 \vee \delta_2] \in \mathsf{Desc}$, *and*
for each $\delta_1 \in \mathsf{Desc}$, *for each* $\delta_2 \in \mathsf{Desc}$, $[\delta_1 \rightarrow \delta_2] \in \mathsf{Desc}$.

We call each member of Term a *term*, each member of Desc a *description*, and each subset of Desc a *theory*.

Definition 4. For each interpretation $I = \langle U, S, F\rangle$,
T_I *is the total function from* Term *to the set of partial functions from* U *to* U *such that*

for each $\upsilon \in U$,
$T_I(:)(\upsilon)$ *is defined and* $T_I(:)(\upsilon) = \upsilon$, *and*
for each $\tau \in \mathsf{Term}$, *for each* $\varphi \in \mathsf{Feat}$, *for each* $\upsilon \in U$,
$T_I(\tau\varphi)(\upsilon)$ *is defined iff* $T_I(\tau)(\upsilon)$ *and* $F(\varphi)(T_I(\tau)(\upsilon))$ *are defined, and if* $T_I(\tau\varphi)(\upsilon)$ *is defined then* $T_I(\tau\varphi)(\upsilon) = F(\varphi)(T_I(\tau)(\upsilon))$,

D_I *is the total function from* Desc *to* $\mathrm{pow}(U)$ *such that*

for each $\tau \in \mathsf{Term}$, *for each* $\sigma \in \mathsf{Spec}$,
$$D_I(\tau \sim \sigma) = \left\{ \upsilon \in U \;\middle|\; \begin{array}{l} T_I(\tau)(\upsilon) \text{ is defined, and} \\ S(T_I(\tau)(\upsilon)) = \sigma \end{array} \right\},$$
for each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$,
$$D_I(\tau_1 \approx \tau_2) = \left\{ \upsilon \in U \;\middle|\; \begin{array}{l} T_I(\tau_1)(\upsilon) \text{ is defined,} \\ T_I(\tau_2)(\upsilon) \text{ is defined, and} \\ T_I(\tau_1)(\upsilon) = T_I(\tau_2)(\upsilon) \end{array} \right\},$$
for each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$,
$$D_I(\tau_1 \not\approx \tau_2) = \left\{ \upsilon \in U \;\middle|\; \begin{array}{l} T_I(\tau_1)(\upsilon) \text{ is defined,} \\ T_I(\tau_2)(\upsilon) \text{ is defined, and} \\ T_I(\tau_1)(\upsilon) \neq T_I(\tau_2)(\upsilon) \end{array} \right\},$$
for each $\delta \in \mathsf{Desc}$, $D_I(\neg\delta) = U \setminus D_I(\delta)$,

for each $\delta_1 \in \mathsf{Desc}$, *for each* $\delta_2 \in \mathsf{Desc}$,
$D_I([\delta_1 \wedge \delta_2]) = D_I(\delta_1) \cap D_I(\delta_2)$,
for each $\delta_1 \in \mathsf{Desc}$, *for each* $\delta_2 \in \mathsf{Desc}$,
$D_I([\delta_1 \vee \delta_2]) = D_I(\delta_1) \cup D_I(\delta_2)$, *and*
for each $\delta_1 \in \mathsf{Desc}$, *for each* $\delta_2 \in \mathsf{Desc}$,
$D_I([\delta_1 \rightarrow \delta_2]) = (U \setminus D_I(\delta_1)) \cup D_I(\delta_2)$, *and*

Θ_I *is the total function from* pow(Desc) *to* pow(U) *such that*
for each $\theta \subseteq \mathsf{Desc}$, $\Theta_I(\theta) = \left\{ \upsilon \in U \;\middle|\; \text{for each } \delta \in \theta,\ \upsilon \in D_I(\delta) \right\}$.

If I is an interpretation then we call T_I the *term denotation function* in I, D_I the *description denotation function* in I, and Θ_I the *theory denotation function* in I. A term is the symbol : followed by a finite string of features, and denotes the functional composition of the denotations of its constituent features (applied in order of occurrence). Description $\tau \sim \sigma$ is true of object υ iff term τ denotes a function defined on υ to yield an object in the denotation of species σ. Description $\tau_1 \approx \tau_2$ is true of object υ iff terms τ_1 and τ_2 denote functions defined on υ to yield one single object. Description $\tau_1 \not\approx \tau_2$ is true of object υ iff terms τ_1 and τ_2 denote functions defined on υ to yield two distinct objects. Description $\neg\delta$ is true of object υ iff δ is false of υ. Description $[\delta_1 \wedge \delta_2]$ is true of object υ iff δ_1 is true of υ and δ_2 is true of υ. Description $[\delta_1 \vee \delta_2]$ is true of object υ iff δ_1 is true of υ or δ_2 is true of υ (or both). Description $[\delta_1 \rightarrow \delta_2]$ is true of object υ iff δ_2 is true of υ whenever δ_1 is true of υ. Each description denotes the set of objects of which it is true. Theory θ is true of object υ iff each description in θ is true of υ. Each theory denotes the set of objects of which it is true.

Notice that for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$, though description $\tau_1 \not\approx \tau_2$ is not in the SRL syntax of King (1989), it is semantically equivalent to

$$[\tau_1 \approx \tau_1 \wedge \tau_2 \approx \tau_2 \wedge \neg\tau_1 \approx \tau_2],$$

and is thus a harmless meta-syntactic addition that enables us to eliminate negation ($\neg$) and implication ($\rightarrow$) in the exclusive matrices of the next section, while later ensuring that every finite theory has a semantically-equivalent exclusive matrix. Notice also that each finite theory is semantically equivalent to the conjunction of its members:

Proposition 1. For each $\{\delta_1, \ldots, \delta_n\} \subseteq \mathsf{Desc}$, *for each interpretation* I,
$\Theta_I(\{\delta_1, \ldots, \delta_n\}) = D_I([: \approx : \wedge\, \delta_1 \wedge \ldots \wedge \delta_n])$.

A theory is satisfiable iff it is true of some object in some interpretation:

Definition 5. For each $\theta \subseteq \mathsf{Desc}$,
θ *is* satisfiable *iff for some interpretation* I, $\Theta_I(\theta) \neq \emptyset$.

How SRL formally expresses a HPSG linguistic theory, or *grammar*, needs a little explanation. According to Pollard and Sag (1994), a grammar

> consists of a *sort hierarchy* and a set of *principles*. The sort hierarchy is presented as a taxonomic tree, with the root labeled *object* (the sort of all linguistic entities with which the grammar deals). For each local tree in the hierarchy, the sorts $\sigma_1, \ldots, \sigma_n$, which label the daughters, partition the sort σ, which labels the mother; that is, they are necessarily disjoint subsorts of σ that exhaust σ. For two sorts σ and τ, τ is a *subsort* of σ if and only if it is dominated by σ; sorts that label terminal nodes are called *maximal* (in the sense of maximally informative or maximally specific). A *feature declaration* of the form
>
> $$\sigma : \begin{bmatrix} \mathrm{F}_1\ \tau_1 \\ \ldots \\ \mathrm{F}_n\ \tau_n \end{bmatrix}$$
>
> where $\sigma, \tau_1, \ldots, \tau_n$ are sorts and $\mathrm{F}_1, \ldots, \mathrm{F}_n$ are feature labels, signifies that for each $i = 1, \ldots, n$, (1) the feature F_i is appropriate for all objects of sort σ, and (2) for any such object, the value of the F_i feature must be an object of sort τ_i.
>
> If sorts σ_1 and σ_2 bear declarations [F τ_1] and [F τ_2] for the same feature F and σ_2 is a subsort of σ_1, then τ_2 must be a subsort of τ_1. A sort inherits the feature declarations of its supersorts; hence any feature that is defined for a given sort is defined for all of that sort's subsorts. By convention, the features that are declared for a maximal sort (including those inherited from its supersorts) are the only features defined for that sort. A special case is that of *atomic* sorts or simply *atoms*, maximal sorts for which no features are defined.
>
> Pollard and Sag (1994, pp. 395–396)

Moreover,

> The (finite) set of all sort symbols is assumed to be partially ordered, with sort symbols corresponding to more inclusive types lower in the ordering. For example, the sort *sign* is ordered below the sort *phrase* or *word* because signs include both phrases and words; we say, for example, that *word* is a *subsort* of *sign* and *accusative* is a subsort of *case*. Pollard and Sag (1994, pp. 17–18)

Notice two points from the foregoing citations. Firstly, given that the sort hierarchy is a finite partial order, sort *object* is a supersort of all other sorts, sort *object* denotes the universe of linguistic objects, and the denotations of the immediate subsorts of a nonmaximal sort partition the denotation of the sort, it follows that the denotations of the maximal sorts partition the universe of linguistic objects. Thus, each linguistic object is in the denotation of one and only one maximal sort. Maximal sorts are like species in this respect. Secondly, there are three possibilities for each nonmaximal sort and each feature: the denotation of the feature is defined either on all objects in the denotation of the sort (as with feature PHON and sort *sign*), on some but not all objects in the denotation of the sort (as with feature DTRS and sort *sign*), or on no objects in the denotation of the sort (as with feature MAJ

and sort *sign*). However, there are only two possibilities for each maximal sort and each feature: the denotation of the feature is defined either on all objects in the denotation of the sort (as with feature DTRS and sort *phrase*), or on no objects in the denotation of the sort (as with feature MAJ and sort *phrase*). The maximal sorts are like species in this respect also.

In fact, each maximal HPSG sort is a SRL species, each nonmaximal HPSG sort is the disjunction of its maximal subsorts, and each HPSG feature is a SRL feature. For example, maximal HPSG sorts *phrase* and *word* are SRL species, and nonmaximal HPSG sort *sign* is the disjunction *phrase* $\vee$ *word*. A SRL appropriateness function Appr can express the HPSG feature declarations for the maximal sorts: for each maximal sort σ and each feature φ, if σ bears feature declaration $[\varphi\ \varsigma]$ then $\mathsf{Appr}(\sigma, \varphi)$ is the set of maximal subsorts of ς else $\mathsf{Appr}(\sigma, \varphi) = \emptyset$. The HPSG feature declarations for the nonmaximal sorts are easily inferred from Appr. Thus, a SRL signature can readily express all of the apparent complexity of the sort hierarchy component of a HPSG grammar.

A SRL theory can just as readily express the set of principles that constitute the other component of a HPSG grammar. For example, noting that the SRL rendering of nonmaximal HPSG sort *headed-structure* is the disjunction

head-complement-structure $\vee$ *head-marker-structure* $\vee$
head-adjunct-structure $\vee$ *head-filler-structure*

of SRL species, the HPSG head feature principle

> In a headed phrase, the values of SYNSEM | LOCAL | CATEGORY | HEAD and DAUGHTERS | HEAD-DAUGHTER | SYNSEM | LOCAL | CATEGORY | HEAD are token-identical
> Pollard and Sag (1994, p. 399)

can be expressed as the SRL description

$$\left[\begin{array}{l} \left[\begin{array}{l} :\sim \textit{phrase} \wedge \left[\begin{array}{l} :\textsc{daughters} \sim \textit{head-complement-structure} \vee \\ :\textsc{daughters} \sim \textit{head-marker-structure} \vee \\ :\textsc{daughters} \sim \textit{head-adjunct-structure} \vee \\ :\textsc{daughters} \sim \textit{head-filler-structure} \end{array}\right] \end{array}\right] \\ \rightarrow\ :\textsc{synsem local category head} \\ \quad \approx :\textsc{daughters head-daughter synsem local category head} \end{array}\right].$$

SRL can also express such HPSG exotica as lexical rules and linear precedence (see respectively Meurers and Minnen (1997) and Richter and Sailer (1995)).[3]

In preparation for the exclusive matrices of the next section, we now move beyond the syntax and semantics of King (1989) and introduce some supplementary definitions of our own.

Definition 6.

$$\begin{aligned} \mathsf{Atom} = \{&\tau \sim \sigma \mid \tau \in \mathsf{Term}\ \textit{and}\ \sigma \in \mathsf{Spec}\} \\ &\cup \{\tau_1 \approx \tau_2 \mid \tau_1 \in \mathsf{Term}\ \textit{and}\ \tau_2 \in \mathsf{Term}\} \\ &\cup \{\tau_1 \not\approx \tau_2 \mid \tau_1 \in \mathsf{Term}\ \textit{and}\ \tau_2 \in \mathsf{Term}\}, \end{aligned}$$

$$\mathsf{Lite} = \mathsf{Atom} \cup \{\neg\delta \mid \delta \in \mathsf{Atom}\},$$

$$\mathsf{Clau} = \mathrm{finpow}(\mathsf{Lite}),\ \textit{and}$$

$\mathsf{Matr} = \mathrm{finpow}(\mathsf{Clau})$.

We call each member of Atom an *atomic description*, each member of Lite a *literal*, each member of Clau a *clause*, and each member of Matr a *matrix*.

Definition 7. For each interpretation $I = \langle U, S, F\rangle$, M_I *is the total function from* Matr *to* $\mathrm{pow}(U)$ *such that for each* $\mu \in \mathsf{Matr}$,

$$M_I(\mu) = \{\upsilon \in U \mid \textit{ for some } \theta \in \mu, \upsilon \in \Theta_I(\theta)\}.$$

If I is an interpretation then we call M_I the *matrix denotation function* in I. Matrix μ is true of object υ iff each literal in some clause in μ is true of υ. Each matrix denotes the set of objects of which it is true. Clearly, each matrix represents a disjunctive normal form description:

Proposition 2. For each $\{\delta_{i,j} \mid 1 \leqslant i \leqslant m \text{ and } 1 \leqslant j \leqslant n_m\} \subseteq \mathsf{Lite}$, *for each interpretation* I,

$$M_I(\{\{\delta_{i,j} \mid 1 \leqslant j \leqslant n_m\} \mid 1 \leqslant i \leqslant m\}) = D_I(\textstyle\bigvee_{i=1}^{m}(\bigwedge_{j=1}^{n_m}\delta_{i,j})).$$

Definition 8. For each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$,

τ_1 *is a* subterm *of* τ_2 *iff for some* $\omega \in \mathsf{Feat}^\sharp$, $\tau_1\omega = \tau_2$.

A first term is a subterm of a second term iff the first is a prefix of the second.

Definition 9. Term is the total function from Matr *to* finpow(Term) *such that for each* $\mu \in \mathsf{Matr}$,

$$\mathit{Term}(\mu) = \left\{ \tau \in \mathsf{Term} \;\middle|\; \begin{array}{l} \textit{for some } \theta \in \mu, \textit{for some } \omega \in \mathsf{Feat}^\sharp, \\ \quad \textit{for some } \sigma \in \mathsf{Spec}, \\ \quad\quad \tau\omega \sim \sigma \in \theta \textit{ or } \neg\tau\omega \sim \sigma \in \theta, \textit{ or} \\ \quad \textit{for some } \tau' \in \mathsf{Term}, \\ \quad\quad \tau\omega \approx \tau' \in \theta, \neg\tau\omega \approx \tau' \in \theta, \\ \quad\quad \tau' \approx \tau\omega \in \theta, \neg\tau' \approx \tau\omega \in \theta, \\ \quad\quad \tau\omega \not\approx \tau' \in \theta, \neg\tau\omega \not\approx \tau' \in \theta, \\ \quad\quad \tau' \not\approx \tau\omega \in \theta \textit{ or } \neg\tau' \not\approx \tau\omega \in \theta \end{array} \right\}.$$

If $\mu \in \mathsf{Matr}$ then $\mathit{Term}(\mu)$ is the set of subterms of terms occurring in literals in clauses in μ.

4. Classification

An exclusive matrix is a matrix that meets certain syntactic conditions. We show that any clause in an exclusive matrix is satisfiable, but that the union of any two

distinct clauses in an exclusive matrix is unsatisfiable. Thus, if each clause in an exclusive matrix is considered to be a label that indicates the denotation of the clause then the matrix constitutes a classification over the denotation of the matrix with qualities (Q1) and (Q2).

Definition 10. μ *is an* exclusive matrix *iff*

$\mu \in \mathsf{Matr}$, *and*
for each $\theta \in \mu$,
$\theta \subseteq \mathsf{Atom}$,
$: \approx : \in \theta$,
for each $\tau \in \mathsf{Term}$, *for each* $\varphi \in \mathsf{Feat}$,
if $\tau\varphi \approx \tau\varphi \in \theta$ *then* $\tau \approx \tau \in \theta$,
for each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$,
if $\tau_1 \approx \tau_2 \in \theta$ *then* $\tau_2 \approx \tau_1 \in \theta$,
for each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$, *for each* $\tau_3 \in \mathsf{Term}$,
if $\tau_1 \approx \tau_2 \in \theta$ *and* $\tau_2 \approx \tau_3 \in \theta$ *then* $\tau_1 \approx \tau_3 \in \theta$,
for each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$, *for each* $\varphi \in \mathsf{Feat}$,
if $\tau_1 \approx \tau_2 \in \theta$, $\tau_1\varphi \approx \tau_1\varphi \in \theta$ *and* $\tau_2\varphi \approx \tau_2\varphi \in \theta$
then $\tau_1\varphi \approx \tau_2\varphi \in \theta$,
for each $\tau \in \mathsf{Term}$,
$\tau \approx \tau \in \theta$ *iff for some* $\sigma \in \mathsf{Spec}$, $\tau \sim \sigma \in \theta$,
for each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$, *for each* $\sigma_1 \in \mathsf{Spec}$,
for each $\sigma_2 \in \mathsf{Spec}$,
if $\tau_1 \approx \tau_2 \in \theta$, $\tau_1 \sim \sigma_1 \in \theta$ *and* $\tau_2 \sim \sigma_2 \in \theta$ *then* $\sigma_1 = \sigma_2$,
for each $\tau \in \mathsf{Term}$, *for each* $\sigma_1 \in \mathsf{Spec}$, *for each* $\varphi \in \mathsf{Feat}$,
for each $\sigma_2 \in \mathsf{Spec}$,
if $\tau \sim \sigma_1 \in \theta$ *and* $\tau\varphi \sim \sigma_2 \in \theta$ *then* $\sigma_2 \in \mathsf{Appr}\langle\sigma_1, \varphi\rangle$,
for each $\tau \in \mathsf{Term}$, *for each* $\sigma \in \mathsf{Spec}$, *for each* $\varphi \in \mathsf{Feat}$,
if $\tau \sim \sigma \in \theta$, $\mathsf{Appr}\langle\sigma, \varphi\rangle \neq \emptyset$ *and* $\tau\varphi \in \mathit{Term}(\mu)$
then $\tau\varphi \approx \tau\varphi \in \theta$,
for each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$,
if $\tau_1 \not\approx \tau_2 \in \theta$ *then* $\tau_1 \approx \tau_1 \in \theta$ *and* $\tau_2 \approx \tau_2 \in \theta$,
for each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$,
if $\tau_1 \approx \tau_1 \in \theta$ *and* $\tau_2 \approx \tau_2 \in \theta$
then $\tau_1 \approx \tau_2 \in \theta$ *or* $\tau_1 \not\approx \tau_2 \in \theta$, *and*
for each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$,
$\tau_1 \approx \tau_2 \notin \theta$ *or* $\tau_1 \not\approx \tau_2 \notin \theta$.

We write Excl for the set of exclusive matrices. Clearly, each subset of an exclusive matrix is itself an exclusive matrix:

Proposition 3. For each $\mu \in \mathsf{Excl}$, *for each* $\mu' \subseteq \mu$, $\mu' \in \mathsf{Excl}$.

In order to gain an intuitive grasp of an exclusive matrix μ, consider the binary relations $\equiv$ and $\not\equiv$ on terms induced by $\approx$ and $\not\approx$ in some clause θ in μ: for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$,

$\tau_1 \equiv \tau_2$ iff $\tau_1 \approx \tau_2 \in \theta$, and

$\tau_1 \not\equiv \tau_2$ iff $\tau_1 \not\approx \tau_2 \in \theta$.

$\equiv$ is an equivalence relation on a finite, nonempty and subterm-closed domain of terms. $\equiv$ also has a right-invariant 'growth' property: for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$, for each $\varphi \in \mathsf{Feat}$,

if $\tau_1 \equiv \tau_2$,
  $\tau_1\varphi$ is in the domain of $\equiv$, and
  $\tau_2\varphi$ is in the domain of $\equiv$
then $\tau_1\varphi \equiv \tau_2\varphi$.

Moreover, each equivalence class in $\equiv$ is labeled with a unique species, and $\equiv$ must never violate the appropriateness function Appr: for each $\tau \in \mathsf{Term}$, for each $\sigma_1 \in \mathsf{Spec}$, for each $\varphi \in \mathsf{Feat}$, for each $\sigma_2 \in \mathsf{Spec}$,

if σ_1 labels the $\equiv$ equivalence class of τ, and
  σ_2 labels the $\equiv$ equivalence class of $\tau\varphi$
then $\sigma_2 \in \mathsf{Appr}\langle\sigma_1, \varphi\rangle$.

However, fulfilling Appr is compulsory only if each term involved is in *Term*(μ): for each $\tau \in \mathsf{Term}$, for each $\sigma \in \mathsf{Spec}$, for each $\varphi \in \mathsf{Feat}$,

if σ labels the $\equiv$ equivalence class of τ,
  $\mathsf{Appr}\langle\sigma, \varphi\rangle \neq \emptyset$, and
  $\tau\varphi$ is in *Term*(μ)
then $\tau\varphi$ is in the domain of $\equiv$.

Finally, $\not\equiv$ is the complement of $\equiv$: for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$,

$\tau_1 \not\equiv \tau_2$
iff τ_1 is in the domain of $\equiv$,
  τ_2 is in the domain of $\equiv$, and
  it is not the case that $\tau_1 \equiv \tau_2$.

The ability of an exclusive matrix to constitute a classification over its own denotation follows from two facts. Firstly, the union of any two distinct clauses in an exclusive matrix is unsatisfiable:

Theorem 1. For each $\mu \in \mathsf{Excl}$, *for each* $\theta_1 \in \mu$, *for each* $\theta_2 \in \mu$,
if $\theta_1 \neq \theta_2$ *then* $\theta_1 \cup \theta_2$ *is unsatisfiable.*

Proof. For each $\mu \in \mathsf{Excl}$, for each $\theta_1 \in \mu$, for each $\theta_2 \in \mu$,
if $\theta_1 \cup \theta_2$ is satisfiable
then for each $\tau \in \mathsf{Term}$,
$\tau \approx \tau \in \theta_1$ iff $\tau \approx \tau \in \theta_2$. by induction on the length of τ
Thus, for each $\mu \in \mathsf{Excl}$, for each $\theta_1 \in \mu$, for each $\theta_2 \in \mu$,
if $\theta_1 \cup \theta_2$ is satisfiable
then for each $\delta \in \mathsf{Atom}$,
$\delta \in \theta_1$ iff $\delta \in \theta_2$.
Thus, for each $\mu \in \mathsf{Excl}$, for each $\theta_1 \in \mu$, for each $\theta_2 \in \mu$,
if $\theta_1 \cup \theta_2$ is satisfiable then $\theta_1 = \theta_2$. □

Secondly, any individual clause in an exclusive matrix is satisfiable:

Theorem 2. For each $\mu \in \mathsf{Excl}$, for each $\theta \in \mu$,
θ is satisfiable.
Proof. Suppose that
$\mathsf{Atom}^{\mathsf{K}} = \{\tau \sim \sigma \mid \tau \in \mathsf{Term} \text{ and } \sigma \in \mathsf{Spec}\}$
$\cup \{\tau_1 \approx \tau_2 \mid \tau_1 \in \mathsf{Term} \text{ and } \tau_2 \in \mathsf{Term}\}$,
$\mathsf{Lite}^{\mathsf{K}} = \mathsf{Atom}^{\mathsf{K}} \cup \{\neg\delta \mid \delta \in \mathsf{Atom}^{\mathsf{K}}\}$, and
$\mathsf{Clau}^{\mathsf{K}}$ is the unique set such that for each θ, $\theta \in \mathsf{Clau}^{\mathsf{K}}$ iff
$\theta \in \mathrm{finpow}(\mathsf{Lite}^{\mathsf{K}})$,
$: \approx : \in \theta$,
for each $\tau \in \mathsf{Term}$, for each $\varphi \in \mathsf{Feat}$,
if $\tau\varphi \approx \tau\varphi \in \theta$ then $\tau \approx \tau \in \theta$,
for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$,
if $\tau_1 \approx \tau_2 \in \theta$ then $\tau_2 \approx \tau_1 \in \theta$,
for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$, for each $\tau_3 \in \mathsf{Term}$,
if $\tau_1 \approx \tau_2 \in \theta$ and $\tau_2 \approx \tau_3 \in \theta$ then $\tau_1 \approx \tau_3 \in \theta$,
for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$, for each $\varphi \in \mathsf{Feat}$,
if $\tau_1 \approx \tau_2 \in \theta$, $\tau_1\varphi \approx \tau_1\varphi \in \theta$ and $\tau_2\varphi \approx \tau_2\varphi \in \theta$
then $\tau_1\varphi \approx \tau_2\varphi \in \theta$,
for each $\tau \in \mathsf{Term}$,
$\tau \approx \tau \in \theta$ iff for some $\sigma \in \mathsf{Spec}$, $\tau \sim \sigma \in \theta$,
for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$, for each $\sigma_1 \in \mathsf{Spec}$,
for each $\sigma_2 \in \mathsf{Spec}$,
if $\tau_1 \approx \tau_2 \in \theta$, $\tau_1 \sim \sigma_1 \in \theta$ and $\tau_2 \sim \sigma_2 \in \theta$ then $\sigma_1 = \sigma_2$,
for each $\tau \in \mathsf{Term}$, for each $\sigma_1 \in \mathsf{Spec}$, for each $\varphi \in \mathsf{Feat}$,
for each $\sigma_2 \in \mathsf{Spec}$,

if $\tau \sim \sigma_1 \in \theta$ and $\tau\varphi \sim \sigma_2 \in \theta$ then $\sigma_2 \in \mathsf{Appr}\langle\sigma_1, \varphi\rangle$,

for each $\tau \in \mathsf{Term}$, for each $\sigma \in \mathsf{Spec}$, for each $\varphi \in \mathsf{Feat}$,

if $\tau \sim \sigma \in \theta$, $\mathsf{Appr}\langle\sigma, \varphi\rangle \neq \emptyset$ and $\tau\varphi \in \mathit{Term}(\{\theta\})$

then $\tau\varphi \approx \tau\varphi \in \theta$, and

for each $\delta \in \mathsf{Desc}$,

$\delta \notin \theta$ or $\neg\delta \notin \theta$.

Clearly,

$\mathsf{Atom}^{\mathsf{K}} \subseteq \mathsf{Atom}$, $\mathsf{Lite}^{\mathsf{K}} \subseteq \mathsf{Lite}$ and $\mathsf{Clau}^{\mathsf{K}} \subseteq \mathsf{Clau}$.

Suppose that K is the total function from Atom to $\mathsf{Lite}^{\mathsf{K}}$ such that

for each $\tau \in \mathsf{Term}$, for each $\sigma \in \mathsf{Spec}$, $K(\tau \sim \sigma) = \tau \sim \sigma$,

for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$, $K(\tau_1 \approx \tau_2) = \tau_1 \approx \tau_2$, and

for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$, $K(\tau_1 \not\approx \tau_2) = \neg\tau_1 \approx \tau_2$.

Suppose that $\overline{K}$ is the total function from Atom to $\mathrm{finpow}(\mathsf{Lite}^{\mathsf{K}})$ such that

for each $\tau \in \mathsf{Term}$, for each $\sigma \in \mathsf{Spec}$, $\overline{K}(\tau \sim \sigma) = \{\tau \sim \sigma\}$,

for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$, $\overline{K}(\tau_1 \approx \tau_2) = \{\tau_1 \approx \tau_2\}$, and

for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$,

$\overline{K}(\tau_1 \not\approx \tau_2) = \{\tau_1 \approx \tau_1, \tau_2 \approx \tau_2, \neg\tau_1 \approx \tau_2\}$.

Abusing notation somewhat, suppose also that K and $\overline{K}$ are the total functions from $\mathrm{finpow}(\mathsf{Atom})$ to $\mathrm{finpow}(\mathsf{Lite}^{\mathsf{K}})$ such that for each $\theta \in \mathrm{finpow}(\mathsf{Atom})$,

$K(\theta) = \{K(\delta) \mid \delta \in \theta\}$, and

$\overline{K}(\theta) = \bigcup\{\overline{K}(\delta) \mid \delta \in \theta\}$.

Firstly, for each $\mu \in \mathsf{Excl}$, for each $\theta \in \mu$, for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$,

if $\tau_1 \not\approx \tau_2 \in \theta$ then $\overline{K}(\tau_1 \not\approx \tau_2) \subseteq K(\theta)$.

Thus, for each $\mu \in \mathsf{Excl}$, for each $\theta \in \mu$,

$\overline{K}(\theta) = K(\theta)$.

Moreover, for each $\mu \in \mathsf{Excl}$, for each $\theta \in \mu$, for each interpretation I,

$\Theta_I(\theta) = \Theta_I(\overline{K}(\theta))$.

Thus, for each $\mu \in \mathsf{Excl}$, for each $\theta \in \mu$, for each interpretation I,

$\Theta_I(\theta) = \Theta_I(K(\theta))$. ... (1)

Secondly, for each $\mu \in \mathsf{Excl}$, for each $\theta \in \mu$,

$\mathit{Term}(\{\theta\}) = \mathit{Term}(\{K(\theta)\})$, and

for each $\delta \in \mathsf{Desc}$,

$\delta \in K(\theta)$ and $\neg\delta \in K(\theta)$

$\Longrightarrow$ for some $\tau_1 \in \mathsf{Term}$, for some $\tau_2 \in \mathsf{Term}$,

$\tau_1 \approx \tau_2 \in \theta$ and $\tau_1 \not\approx \tau_2 \in \theta$

$\Longrightarrow$ contradiction.

Thus, for each $\mu \in \mathsf{Excl}$, for each $\theta \in \mu$,

$K(\theta) \in \mathsf{Clau}^{\mathsf{K}}$. ... (2)

Thirdly, by Kepser (1994, theorem 77),

for each $\theta \in \mathsf{Clau}^{\mathsf{K}}$, θ is satisfiable. ... (3)

Therefore, by (1)–(3), for each $\mu \in \mathsf{Excl}$, for each $\theta \in \mu$,

θ is satisfiable. □

Thus, if each clause in an exclusive matrix is considered to be a label that indicates the denotation of the clause then Theorem 1 shows that distinct labels in the matrix inherently indicate disjoint sets of objects, and Theorem 2 shows that no label in the matrix inherently indicates an empty set of objects. Thus, each exclusive matrix constitutes a classification over the denotation of the matrix with qualities (Q1) and (Q2).

5. Index

An index tree is a finite tree labeled with queries, responses and clauses. We show that if an index tree analyzes an exclusive matrix then an algorithm, called `Clause`, can use the tree to interrogate any accurate and effective oracle about an object in the denotation of the matrix in order to effectively deduce the unique clause in the matrix that is true of the object. Thus, `Clause` and the index tree together constitute an index from the denotation of the matrix to the matrix with quality (Q4).

Queries and responses are symbols that represent respectively certain questions that could be asked regarding objects and the answers that could be given in reply:

Definition 11.

$\mathsf{Quer} = \{\boxed{\tau} \mid \tau \in \mathsf{Term}\} \cup \{\boxed{\tau_1, \tau_2} \mid \tau_1 \in \mathsf{Term}$ *and* $\tau_2 \in \mathsf{Term}\}$, *and*

$\mathsf{Resp} = \{\boxed{\sigma} \mid \sigma \in \mathsf{Spec}\} \cup \{\boxed{-}, \boxed{\approx}, \boxed{\not\approx}, \boxed{\pm}, \boxed{\mp}, \boxed{=}\}$.

We call each member of Quer a *query* and each member of Resp a *response*.

Proposition 4. Resp *is finite and nonempty.*

For each interpretation $I = \langle U, S, F\rangle$, for each $\upsilon \in U$,

for each $\tau \in \mathsf{Term}$,

$\boxed{\tau}$ represents the query 'What is $S(T_I(\tau)(\upsilon))$?', to which

$\boxed{\sigma}$ represents, for each $\sigma \in \mathsf{Spec}$, the response '$T_I(\tau)(\upsilon)$ is defined and $S(T_I(\tau)(\upsilon)) = \sigma$', and

$\boxed{-}$ represents the response '$T_I(\tau)(\upsilon)$ is undefined', and

for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$,

$\boxed{\tau_1, \tau_2}$ represents the query 'Does $T_I(\tau_1)(\upsilon) = T_I(\tau_2)(\upsilon)$?', to which

$\boxed{\approx}$ represents the response
'$T_I(\tau_1)(\upsilon)$ is defined, $T_I(\tau_2)(\upsilon)$ is defined, and
$T_I(\tau_1)(\upsilon) = T_I(\tau_2)(\upsilon)$',
$\boxed{\not\approx}$ represents the response
'$T_I(\tau_1)(\upsilon)$ is defined, $T_I(\tau_2)(\upsilon)$ is defined, and
$T_I(\tau_1)(\upsilon) \neq T_I(\tau_2)(\upsilon)$',
$\boxed{\pm}$ represents the response
'$T_I(\tau_1)(\upsilon)$ is defined and $T_I(\tau_2)(\upsilon)$ is undefined',
$\boxed{\mp}$ represents the response
'$T_I(\tau_1)(\upsilon)$ is undefined and $T_I(\tau_2)(\upsilon)$ is defined', and
$\boxed{=}$ represents the response
'$T_I(\tau_1)(\upsilon)$ is undefined and $T_I(\tau_2)(\upsilon)$ is undefined'.

Definition 12. *For each* $\tau \in$ Term, *for each* $\tau' \in$ Term,
$\boxed{\tau}$ concerns τ *and* $\boxed{\tau, \tau'}$ concerns τ *and* τ'.

Definition 13. *Query is the total function from* Matr *to* finpow(Quer) *such that for each* $\mu \in$ Matr,

$$\begin{aligned} \mathit{Query}(\mu) = {} & \{\boxed{\tau} \mid \tau \in \mathit{Term}(\mu)\} \\ & \cup \{\boxed{\tau_1, \tau_2} \mid \tau_1 \in \mathit{Term}(\mu) \text{ and } \tau_2 \in \mathit{Term}(\mu)\}. \end{aligned}$$

If $\mu \in$ Matr then $\mathit{Query}(\mu)$ is the set of queries that concern terms in $\mathit{Term}(\mu)$.

Definition 14. *For each interpretation* $I = \langle U, S, F\rangle$, *for each* $\upsilon \in U$, $\mathit{Accurate}_I^{\upsilon}$ *is the total function from* Quer *to* Resp *such that*

for each $\tau \in$ Term,

$$\mathit{Accurate}_I^{\upsilon}(\boxed{\tau}) = \begin{cases} \boxed{\sigma} & \text{if } T_I(\tau)(\upsilon) \text{ is defined, and} \\ & S(T_I(\tau)(\upsilon)) = \sigma \\ \boxed{-} & \text{if } T_I(\tau)(\upsilon) \text{ is undefined, and} \end{cases}$$

for each $\tau_1 \in$ Term, *for each* $\tau_2 \in$ Term,

$$\mathit{Accurate}_I^{\upsilon}(\boxed{\tau_1, \tau_2}) = \begin{cases} \boxed{\approx} & \text{if } T_I(\tau_1)(\upsilon) \text{ is defined,} \\ & T_I(\tau_2)(\upsilon) \text{ is defined, and} \\ & T_I(\tau_1)(\upsilon) = T_I(\tau_2)(\upsilon) \\ \boxed{\not\approx} & \text{if } T_I(\tau_1)(\upsilon) \text{ is defined,} \\ & T_I(\tau_2)(\upsilon) \text{ is defined, and} \\ & T_I(\tau_1)(\upsilon) \neq T_I(\tau_2)(\upsilon) \\ \boxed{\pm} & \text{if } T_I(\tau_1)(\upsilon) \text{ is defined, and} \\ & T_I(\tau_2)(\upsilon) \text{ is undefined} \\ \boxed{\mp} & \text{if } T_I(\tau_1)(\upsilon) \text{ is undefined, and} \\ & T_I(\tau_2)(\upsilon) \text{ is defined} \\ \boxed{=} & \text{if } T_I(\tau_1)(\upsilon) \text{ is undefined, and} \\ & T_I(\tau_2)(\upsilon) \text{ is undefined.} \end{cases}$$

If $I = \langle U, S, F \rangle$ is an interpretation, $\upsilon \in U$ and $\kappa \in \mathsf{Quer}$ then $Accurate_I^{\upsilon}(\kappa)$ is the accurate, truthful and honest response to κ about υ in I.

An index tree is a finite tree such that each of its inner nodes bears a query, each of its edges bears a response, and each of its outer nodes bears a clause:

Definition 15. Quadruple $\langle \mathcal{N}, \mathcal{I}, \mathcal{O}, \mathcal{L} \rangle$ *is an* index tree *iff*

$\mathcal{N} \in \mathrm{finpow}(\mathsf{Resp}^{\sharp})$,

$\varepsilon \in \mathcal{N}$,

for each $\nu \in \mathsf{Resp}^{\sharp}$, *for each* $\rho \in \mathsf{Resp}$, *if* $\nu\rho \in \mathcal{N}$ *then* $\nu \in \mathcal{N}$,

$\mathcal{I} = \{\nu \in \mathcal{N} \mid$ *for some* $\rho \in \mathsf{Resp}$, $\nu\rho \in \mathcal{N}\}$,

$\mathcal{O} = \{\nu \in \mathcal{N} \mid$ *for each* $\rho \in \mathsf{Resp}$, $\nu\rho \notin \mathcal{N}\}$,

$\mathcal{L}$ *is a total function from* $\mathcal{N}$ *to* $\mathsf{Quer} \cup \mathsf{Clau}$,

for each $\nu \in \mathcal{I}$, $\mathcal{L}(\nu) \in \mathsf{Quer}$, *and*

for each $\nu \in \mathcal{O}$, $\mathcal{L}(\nu) \in \mathsf{Clau}$.

We write Tree for the set of index trees. Suppose that $\mathcal{T} = \langle \mathcal{N}, \mathcal{I}, \mathcal{O}, \mathcal{L} \rangle$ is an index tree. We call each member of $\mathcal{N}$ a *node* in $\mathcal{T}$, each member of $\mathcal{I}$ an *inner node* in $\mathcal{T}$, each member of $\mathcal{O}$ an *outer node* in $\mathcal{T}$, and $\mathcal{L}$ the *labeling* in $\mathcal{T}$. For each $\nu \in \mathcal{N}$, we say $\mathcal{L}(\nu)$ *labels* ν in $\mathcal{T}$. $\mathcal{T}$ can be pictured as a finite tree with labeled nodes and labeled directed edges. The nodes are the members of $\mathcal{N}$, and there is an edge from node ν_1 to node ν_2 labeled with response ρ iff $\nu_2 = \nu_1\rho$. Thus, the root node is ε, each inner node has some edge leaving it, and each outer node has no edge leaving it. Inner node ν is labeled with query $\mathcal{L}(\nu)$. Outer node ν is labeled with clause $\mathcal{L}(\nu)$. For example, suppose that

$\mathcal{N} = \{\varepsilon, \rho_0, \rho_1, \rho_1\rho_2, \rho_1\rho_3\}$,

$\mathcal{I} = \{\varepsilon, \rho_1\}$,

$\mathcal{O} = \{\rho_0, \rho_1\rho_2, \rho_1\rho_3\}$, and

$\mathcal{L}$ is the total function from $\mathcal{N}$ to $\mathsf{Quer} \cup \mathsf{Clau}$ such that

$\mathcal{L}(\varepsilon) = \kappa_0 \in \mathsf{Quer}$,

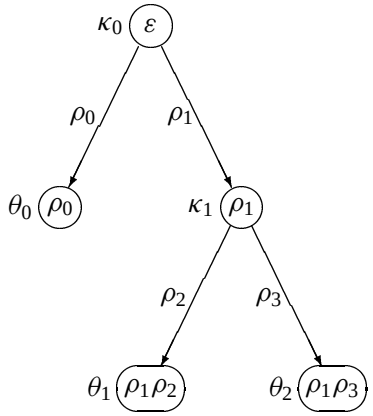
$\mathcal{L}(\rho_0) = \theta_0 \in \mathsf{Clau}$,

$\mathcal{L}(\rho_1) = \kappa_1 \in \mathsf{Quer}$,

$\mathcal{L}(\rho_1\rho_2) = \theta_1 \in \mathsf{Clau}$, and

$\mathcal{L}(\rho_1\rho_3) = \theta_2 \in \mathsf{Clau}$.

Then $\langle \mathcal{N}, \mathcal{I}, \mathcal{O}, \mathcal{L} \rangle$ is an index tree. We can picture this tree as follows, where we write labels to the left of nodes and edges.



To show how an index tree can be used to help effectively classify objects, we must first introduce the notions of a residue and a marking.

Definition 16. *Residue is the total function from* Matr × Quer × Resp *to* Matr *such that for each* $\mu \in$ Matr,

for each $\tau \in$ Term,

for each $\sigma \in$ Spec, $Residue\langle \mu, \boxed{\tau}, \boxed{\sigma} \rangle = \{\theta \in \mu \mid \tau \sim \sigma \in \theta\}$,

$Residue\langle \mu, \boxed{\tau}, \boxed{-} \rangle = \{\theta \in \mu \mid \tau \approx \tau \notin \theta\}$,

$Residue\langle \mu, \boxed{\tau}, \boxed{\approx} \rangle = \emptyset$,

$Residue\langle \mu, \boxed{\tau}, \boxed{\not\approx} \rangle = \emptyset$,

$Residue\langle \mu, \boxed{\tau}, \boxed{\pm} \rangle = \emptyset$,

$Residue\langle \mu, \boxed{\tau}, \boxed{\mp} \rangle = \emptyset$, *and*

$Residue\langle \mu, \boxed{\tau}, \boxed{=} \rangle = \emptyset$, *and*

for each $\tau_1 \in$ Term, *for each* $\tau_2 \in$ Term,

for each $\sigma \in$ Spec, $Residue\langle \mu, \boxed{\tau_1, \tau_2}, \boxed{\sigma} \rangle = \emptyset$,

$Residue\langle \mu, \boxed{\tau_1, \tau_2}, \boxed{-} \rangle = \emptyset$,

$Residue\langle \mu, \boxed{\tau_1, \tau_2}, \boxed{\approx} \rangle = \{\theta \in \mu \mid \tau_1 \approx \tau_2 \in \theta\}$,

$Residue\langle \mu, \boxed{\tau_1, \tau_2}, \boxed{\not\approx} \rangle = \{\theta \in \mu \mid \tau_1 \not\approx \tau_2 \in \theta\}$,

$Residue\langle \mu, \boxed{\tau_1, \tau_2}, \boxed{\pm} \rangle = \{\theta \in \mu \mid \tau_1 \approx \tau_1 \in \theta$ *and* $\tau_2 \approx \tau_2 \notin \theta\}$,

$Residue\langle \mu, \boxed{\tau_1, \tau_2}, \boxed{\mp} \rangle = \{\theta \in \mu \mid \tau_1 \approx \tau_1 \notin \theta$ *and* $\tau_2 \approx \tau_2 \in \theta\}$, *and*

$Residue\langle \mu, \boxed{\tau_1, \tau_2}, \boxed{=} \rangle = \{\theta \in \mu \mid \tau_1 \approx \tau_1 \notin \theta$ *and* $\tau_2 \approx \tau_2 \notin \theta\}$.

If $\mu \in$ Matr, $\kappa \in$ Quer and $\rho \in$ Resp then we call $Residue\langle \mu, \kappa, \rho \rangle$ the *residue of* μ, κ and ρ. Notice two points regarding residues:

Proposition 5. *For each* $\mu \in$ Excl, *for each* $\kappa \in$ Quer,

for each $\rho \in$ Resp, $Residue\langle \mu, \kappa, \rho \rangle \in$ Excl, *and*

for each $\theta \in \mu$, *for some* $\rho \in$ Resp, $\theta \in Residue\langle \mu, \kappa, \rho \rangle$.

Proof. Follows immediately from Proposition 3. □

Definition 17. For each $\mathcal{T} = \langle \mathcal{N}, \mathcal{I}, \mathcal{O}, \mathcal{L} \rangle \in$ Tree,

$\mathcal{M}$ *is a* marking *of* $\mathcal{T}$

iff $\mathcal{M}$ *is a total function from* $\mathcal{N}$ *to* Excl,

for each $\nu \in \mathcal{I}$, $\mathcal{L}(\nu) \in$ *Query*$(\mathcal{M}(\nu))$,

for each $\nu \in \mathcal{I}$, *for each* $\rho \in$ Resp,

if *Residue*$\langle \mathcal{M}(\nu), \mathcal{L}(\nu), \rho \rangle \neq \emptyset$

then $\nu\rho \in \mathcal{N}$ *and* $\mathcal{M}(\nu\rho) =$ *Residue*$\langle \mathcal{M}(\nu), \mathcal{L}(\nu), \rho \rangle$, *and*

for each $\nu \in \mathcal{O}$, $\mathcal{M}(\nu) = \{\mathcal{L}(\nu)\}$.

If $\mathcal{T} = \langle \mathcal{N}, \mathcal{I}, \mathcal{O}, \mathcal{L} \rangle$ is an index tree, $\nu \in \mathcal{N}$ and $\mathcal{M}$ is a marking of $\mathcal{T}$ then we say $\mathcal{M}(\nu)$ *marks* ν in $\mathcal{T}$ under $\mathcal{M}$.

The utility of index trees for effectively classifying objects stems from two facts. Firstly, the outer nodes of an index tree with a marking are labeled with clauses in the matrix that marks the root node:

Proposition 6. For each $\mathcal{T} = \langle \mathcal{N}, \mathcal{I}, \mathcal{O}, \mathcal{L} \rangle \in$ Tree, *for each* $\nu \in \mathcal{O}$, *for each marking* $\mathcal{M}$ *of* $\mathcal{T}$, $\mathcal{L}(\nu) \in \mathcal{M}(\varepsilon)$.

Secondly,

if exclusive matrix μ is true of object υ,

query κ concerns only terms in *Term*(μ), and

response ρ is the accurate response to κ about υ

then exclusive matrix *Residue*$\langle \mu, \kappa, \rho \rangle$ is also true of υ:

Proposition 7. For each $\mu \in$ Excl, *for each interpretation* I, *for each* $\upsilon \in M_I(\mu)$, *for each* $\kappa \in$ *Query*(μ),

Residue$\langle \mu, \kappa,$ *Accurate*$_I^{\upsilon}(\kappa) \rangle \neq \emptyset$, *and*

$\upsilon \in M_I($*Residue*$\langle \mu, \kappa,$ *Accurate*$_I^{\upsilon}(\kappa) \rangle)$.

Proof. For each $\mu \in$ Excl, for each $\theta \in \mu$, for each interpretation I, for each $\upsilon \in \Theta_I(\theta)$, for each $\tau \in$ *Term*(μ),

if $T_I(\tau)(\upsilon)$ is defined then $\tau \approx \tau \in \theta$. by induction on the length of τ

Thus, for each $\mu \in$ Excl, for each $\theta \in \mu$, for each interpretation I, for each $\upsilon \in \Theta_I(\theta)$, for each $\kappa \in$ *Query*(μ),

$\theta \in$ *Residue*$\langle \mu, \kappa,$ *Accurate*$_I^{\upsilon}(\kappa) \rangle$.

Thus, for each $\mu \in$ Excl, for each interpretation I, for each $\upsilon \in M_I(\mu)$, for each $\kappa \in$ *Query*(μ),

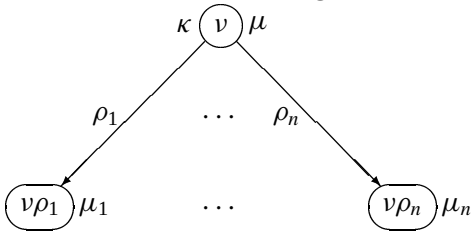
Residue$\langle \mu, \kappa,$ *Accurate*$_I^{\upsilon}(\kappa) \rangle \neq \emptyset$, and

$\upsilon \in M_I($*Residue*$\langle \mu, \kappa,$ *Accurate*$_I^{\upsilon}(\kappa) \rangle)$. □

From Proposition 7 follow two facts about any index tree with a marking. Firstly, suppose that

ν is an inner node,
query κ labels ν,
matrix μ marks ν,
response ρ_1 labels an edge from ν to node $\nu\rho_1$, ...,
response ρ_n labels an edge from ν to node $\nu\rho_n$,
matrix μ_1 marks $\nu\rho_1$, ..., and
matrix μ_n marks $\nu\rho_n$.

We can picture this fragment of the tree and marking as follows, where we write labels to the left of nodes and edges and write marks to the right of nodes.



Suppose further that

μ is true of object υ, and
ρ is the accurate response to κ about υ.

Then, by Proposition 7,

Residue$\langle \mu, \kappa, \rho \rangle \neq \emptyset$ and *Residue*$\langle \mu, \kappa, \rho \rangle$ is true of υ.

Thus, for some $i \in \{1, \ldots, n\}$,

ρ_i is ρ and μ_i is true of υ:

Proposition 8. For each $\mathcal{T} = \langle \mathcal{N}, \mathcal{I}, \mathcal{O}, \mathcal{L} \rangle \in$ Tree, *for each marking* $\mathcal{M}$ *of* $\mathcal{T}$, *for each* $\nu \in \mathcal{I}$, *for each interpretation* I, *for each* $\upsilon \in M_I(\mathcal{M}(\nu))$,

$\nu Accurate_I^{\upsilon}(\mathcal{L}(\nu)) \in \mathcal{N}$ *and* $\upsilon \in M_I(\mathcal{M}(\nu Accurate_I^{\upsilon}(\mathcal{L}(\nu))))$.

Proof. Follows immediately from Proposition 7. □

Secondly, suppose that

ν is an outer node,
clause θ labels ν, and
matrix μ marks ν.

Suppose further that

μ is true of object υ.

Then

θ is true of υ:

Proposition 9. For each $\mathcal{T} = \langle \mathcal{N}, \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle \in$ Tree*, for each marking* $\mathcal{M}$ *of* $\mathcal{T}$*, for each* $\nu \in \mathcal{O}$*, for each interpretation* I*, for each* $\upsilon \in M_I(\mathcal{M}(\nu))$*,* $\upsilon \in \Theta_I(\mathcal{L}(\nu))$*.*

Finally, from Propositions 6, 8 and 9 follows an algorithm `Clause` that can effectively classify objects. Suppose that exclusive matrix μ marks the root node of index tree $\mathcal{T}$, and is true of object υ. Informally speaking, given an accurate and effective oracle about υ, `Clause` navigates through $\mathcal{T}$, considering nodes one after another in a certain sequence that ensures each node `Clause` considers is marked by an exclusive matrix that is true of υ. `Clause` starts by considering the root node, and we have supposed that the exclusive matrix marking the root node is true of υ. If `Clause` is considering an inner node ν then `Clause` poses the query that labels ν to the oracle and after finite time receives the accurate response ρ in reply. By Proposition 8, an edge labeled ρ must leave ν for node $\nu\rho$, and the exclusive matrix that labels $\nu\rho$ must be true of υ. `Clause` turns its consideration to $\nu\rho$. Eventually, `Clause` must consider an outer node labeled with a clause θ. `Clause` outputs θ and stops. By Proposition 9, θ must be true of υ, and by Proposition 6, θ must be in μ. Thus, `Clause` has effectively deduced the unique clause in μ that is true of υ.

Notice two points about the operation of `Clause`. Firstly, `Clause` makes no use of the marking of $\mathcal{T}$ other than that it marks the root node with μ:

Definition 18. For each $\mathcal{T} \in$ Tree*, for each* $\mu \in$ Excl*,*

$\mathcal{T}$ analyzes μ *iff for some marking* $\mathcal{M}$ *of* $\mathcal{T}$*,* $\mathcal{M}(\varepsilon) = \mu$*.*

Secondly, `Clause` can accurately and effectively classify υ only if `Clause` can pose queries to an accurate and effective oracle about υ:

Definition 19. An oracle *is anything that delivers a response if posed a query.*

Our definition of an oracle is deliberately broad. It encompasses entities that deliver responses to queries by means of algorithm, intuition, genius or chance. An oracle need be neither effective nor consistent: it can deliver a response to a query after infinite time, and it can deliver different responses to the same query posed on different occasions. Of course, not all oracles enable `Clause` to accurately and effectively classify υ:

Definition 20. For each interpretation $I = \langle U, S, F \rangle$*, for each* $\upsilon \in U$*, for each oracle* `Oracle`*,*

`Oracle` discloses υ *under* I
iff for each $\kappa \in$ Quer*,*
    if `Oracle` *is posed* κ
    then `Oracle` *delivers* $\mathit{Accurate}_I^{\upsilon}(\kappa)$ *in finite time.*

Thus, an oracle that discloses an object is anything that delivers an accurate response in finite time to each query about the object. For example, an oracle that

discloses an object under a natural language might be a human native speaker of the natural language who arrives at a response to a query by consulting their linguistic intuitions, or a corpus analysis program which arrives at a response to a query by investigating corpora from the natural language. Whatever the nature of the oracle, if `Clause` can pose queries to an oracle that discloses υ then `Clause` can accurately and effectively classify υ.

We formally present `Clause` and some of our subsequent algorithms as computer programs written in an abstract imperative programming language. A program is a finite sequence of instructions, where each instruction is of one of the following five forms:

`input` *value* `into` *register*

`put` *value* `into` *register*

`output` *register*

`while` *condition* `do` *program* `end`

`if` *condition* `then` *program*$_1$ `else` *program*$_2$ `end`

Instructions

`input` *value* `into` *register*

and

`put` *value* `into` *register*

both mean 'make value *value* the content of register *register*'. However, the value for an `input` instruction must be given to the computer, whereas the value for a `put` instruction must be effectively determinable by the computer itself. Instruction

`output` *register*

means 'display the contents of register *register* in a readable form'. Instruction

`while` *condition* `do` *program* `end`

means 'perform program *program* while condition *condition* is true', and instruction

`if` *condition* `then` *program*$_1$ `else` *program*$_2$ `end`

means 'if condition *condition* is true then perform program *program*$_1$, otherwise perform program *program*$_2$'. Of course, all conditions must be effectively determinable by the computer. For legibility, we sometimes refer to registers, rather than their contents, in conditions and values. For examples, if `X` and `Y` are registers then

`put X into Y`

means 'put *the content of* register `X` into register `Y`', and

`if X` $=$ `Y then put` 0 `into X else put` 1 `into X end`

means 'if *the contents of* registers `X` and `Y` are identical then put 0 into register `X`, but if *the contents of* registers `X` and `Y` are different then put 1 into register `X`'. Some of the `put` instructions in our algorithms have complex values in which the computer

must perform effective operations on accessible arguments before making the result the content of the indicated register. We explain each of these complex values as we give the algorithms.

We can now give `Clause` formally as a program: for each oracle `Oracle`, for each $\langle \mathcal{N}, \mathcal{l}, \mathcal{O}, \mathcal{L} \rangle \in$ Tree,

```
input 𝓁 into I
input ℒ into L
put ε into N
while N ∈ I
   do put L(N) into Q
      put Oracle(Q) into R
      put suffix(N, R) into N
   end
put L(N) into C
output C,
```

where for each register `X`, for each register `Y`,

if the content of `X` is function f and the content of `Y` is y

then `X(Y)` is $\begin{cases} f(y) & \text{if } f(y) \text{ is defined} \\ \text{undefined} & \text{if } f(y) \text{ is undefined,} \end{cases}$

if the content of `X` is $x_1 \ldots x_n$ and the content of `Y` is y

then `suffix(X, Y)` is $x_1 \ldots x_n y$, and

`Oracle(X)` is the result of consulting oracle `Oracle` on the content of `X`.

Given an oracle that discloses an object in the denotation of an exclusive matrix, `Clause` effectively deduces from an index tree that analyzes the matrix the unique clause in the matrix that is true of the object:

Theorem 3. For each $\mu \in$ Excl, *for each interpretation* I, *for each* $\upsilon \in M_I(\mu)$, *for each oracle* `Oracle`,

if `Oracle` *discloses* υ *under* I

then `Clause` *computes a partial function from* Tree *to* Clau *such that for each* $\mathcal{T} \in$ Tree,

if $\mathcal{T}$ *analyzes* μ

then `Clause`$(\mathcal{T})$ *is defined,*

`Clause`$(\mathcal{T}) \in \mu$, *and*

$\upsilon \in \Theta_I($`Clause`$(\mathcal{T}))$.

Proof. Follows immediately from Propositions 8, 9 and 6. □

Though it may not be readily apparent, theorem 3 shows that if an index tree analyzes an exclusive matrix then `Clause` and the index tree together constitute

an index from the denotation of the matrix to the matrix. Recall that such an index should effectively deduce from limited information about each object in the denotation of the matrix the unique clause in the matrix that is true of the object. However, notice that the index should not itself provide this limited information, but should merely deduce the correct clause on the basis of limited information provided by some external source. In our case, this information is provided by an oracle. If index tree $\mathcal{T}$ analyzes exclusive matrix μ then theorem 3 shows that `Clause` started with input $\mathcal{T}$ can effectively deduce for each object υ in the denotation of μ the unique clause in μ that is true of υ, provided `Clause` can consult an oracle that discloses υ. In short, `Clause` and an index tree that analyzes an exclusive matrix together constitute an index from the denotation of the matrix to the matrix with quality (Q4).

6. Device

We now give an abstract prototype device that effectively deduces from each finite theory a classificatory system over the denotation of the theory. The device comprises two algorithms, called `Class` and `Index`. `Class` effectively deduces from each finite theory an exclusive matrix that is semantically equivalent to the theory, and `Index` effectively deduces from each nonempty exclusive matrix an index tree that analyzes the matrix. Since each exclusive matrix constitutes a classification over the denotation of the matrix with qualities (Q1) and (Q2), and `Clause` and an index tree that analyzes an exclusive matrix together constitute an index from the denotation of the matrix to the matrix with quality (Q4), it immediately follows that `Class` effectively deduces from each finite theory an exclusive matrix that constitutes a classification over the denotation of the theory with qualities (Q1) and (Q2), and – provided the matrix is nonempty – `Index` then effectively deduces from the matrix an index tree that, together with `Clause`, constitutes an index from the denotation of the theory to the matrix with quality (Q4).[4] Figure 1 schematically depicts our device for any interpretation I. We give `Class` and `Index` in turn in separate subsections.

6.1. THE `Class` ALGORITHM

`Class` uses three component algorithms, `Matrix`, `Goetz` and `Atomic`. Informally, `Class` operates as follows. An application of `Matrix` first turns finite theory θ into semantically-equivalent matrix μ via standard techniques for converting arbitrary propositional-logic formulae into semantically-equivalent disjunctive-normal-form formulae. Recursive applications of `Goetz` then turn μ into semantically-equivalent and *almost* exclusive matrix μ' by deleting and adding clauses. An application of `Atomic` finally turns μ' into semantically-equivalent and *fully* exclusive matrix μ''

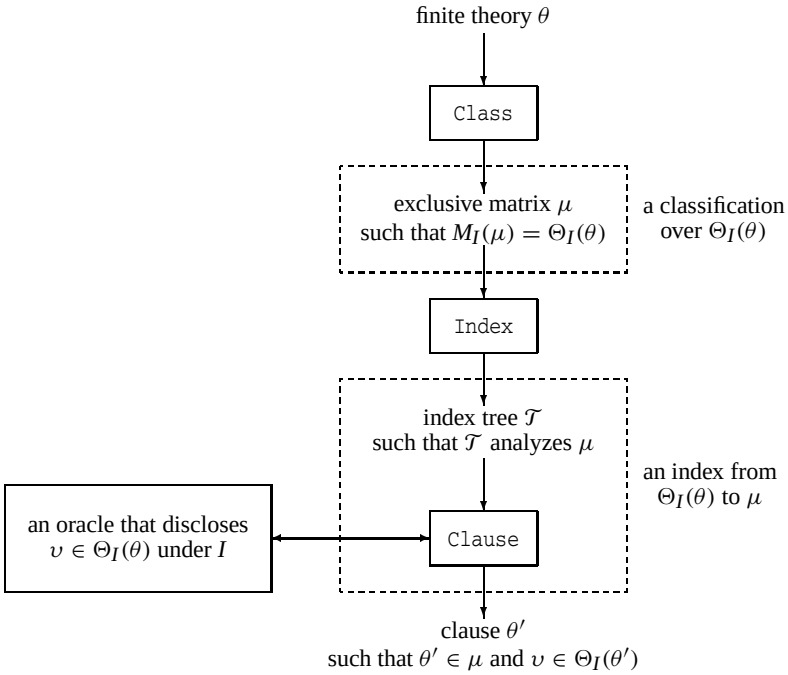


Figure 1. A schematic depiction of the device.

by deleting each nonatomic literal in each clause in μ'. We give each of `Matrix`, `Goetz`, `Atomic` and `Class` in turn.

Since SRL descriptions can be viewed as propositional-logic formulae with their propositional variables replaced by atomic SRL descriptions, any algorithm that converts each propositional-logic formula into a semantically-equivalent disjunctive normal form readily yields an algorithm that converts each SRL description into a semantically-equivalent disjunctive normal form. Thus, there is an algorithm `Matrix` that computes a total function from finpow(Desc) to Matr that preserves denotations:

Proposition 10. For each $\theta \in$ finpow(Desc)*, for each interpretation* I*,*

$\Theta_I(\theta) = M_I(\texttt{Matrix}(\theta))$.

Proof. Follows immediately from Propositions 1 and 2. □

We now give `Goetz`.

Definition 21. $\rightharpoonup$ *is the smallest binary relation on* Matr $\times$ Matr *such that for each* $\mu \in$ Matr*, for each* $\theta \in$ Clau*,*

for each $\delta \in$ Atom*,*

if $\theta \notin \mu$, $\delta \in \theta$ *and* $\neg\delta \in \theta$ *then* $\mu \cup \{\theta\} \rightharpoonup \mu$*,*

if $\theta \notin \mu$ *and* $: \approx : \notin \theta$ *then* $\mu \cup \{\theta\} \rightharpoonup \mu \cup \{\theta \cup \{: \approx :\}\}$*,*

for each $\tau \in$ Term, *for each* $\varphi \in$ Feat,
if $\theta \notin \mu$, $\tau\varphi \approx \tau\varphi \in \theta$ *and* $\tau \approx \tau \notin \theta$ *then* $\mu \cup \{\theta\} \rightharpoonup \mu \cup \{\theta \cup \{\tau \approx \tau\}\}$,

for each $\tau_1 \in$ Term, *for each* $\tau_2 \in$ Term,
if $\theta \notin \mu$, $\tau_1 \approx \tau_2 \in \theta$ *and* $\tau_2 \approx \tau_1 \notin \theta$ *then* $\mu \cup \{\theta\} \rightharpoonup \mu \cup \{\theta \cup \{\tau_2 \approx \tau_1\}\}$,

for each $\tau_1 \in$ Term, *for each* $\tau_2 \in$ Term, *for each* $\tau_3 \in$ Term,
if $\theta \notin \mu$, $\tau_1 \approx \tau_2 \in \theta$, $\tau_2 \approx \tau_3 \in \theta$ *and* $\tau_1 \approx \tau_3 \notin \theta$
then $\mu \cup \{\theta\} \rightharpoonup \mu \cup \{\theta \cup \{\tau_1 \approx \tau_3\}\}$,

for each $\tau_1 \in$ Term, *for each* $\tau_2 \in$ Term, *for each* $\varphi \in$ Feat,
if $\theta \notin \mu$, $\tau_1 \approx \tau_2 \in \theta$, $\tau_1\varphi \approx \tau_1\varphi \in \theta$, $\tau_2\varphi \approx \tau_2\varphi \in \theta$, *and*
$\tau_1\varphi \approx \tau_2\varphi \notin \theta$
then $\mu \cup \{\theta\} \rightharpoonup \mu \cup \{\theta \cup \{\tau_1\varphi \approx \tau_2\varphi\}\}$,

for each $\tau \in$ Term,
if $\theta \notin \mu$, $\tau \approx \tau \in \theta$ *and for each* $\sigma \in$ Spec, $\tau \sim \sigma \notin \theta$
then $\mu \cup \{\theta\} \rightharpoonup \mu \cup \{\theta \cup \{\tau \sim \sigma\} \mid \sigma \in \text{Spec}\}$,

for each $\tau \in$ Term, *for each* $\sigma \in$ Spec,
if $\theta \notin \mu$, $\tau \sim \sigma \in \theta$ *and* $\tau \approx \tau \notin \theta$ *then* $\mu \cup \{\theta\} \rightharpoonup \mu \cup \{\theta \cup \{\tau \approx \tau\}\}$,

for each $\tau_1 \in$ Term, *for each* $\tau_2 \in$ Term, *for each* $\sigma_1 \in$ Spec,
for each $\sigma_2 \in$ Spec,
if $\theta \notin \mu$, $\tau_1 \approx \tau_2 \in \theta$, $\tau_1 \sim \sigma_1 \in \theta$, $\tau_2 \sim \sigma_2 \in \theta$ *and* $\sigma_1 \neq \sigma_2$
then $\mu \cup \{\theta\} \rightharpoonup \mu$,

for each $\tau \in$ Term, *for each* $\sigma_1 \in$ Spec, *for each* $\varphi \in$ Feat,
for each $\sigma_2 \in$ Spec,
if $\theta \notin \mu$, $\tau \sim \sigma_1 \in \theta$, $\tau\varphi \sim \sigma_2 \in \theta$ *and* $\sigma_2 \notin \text{Appr}\langle\sigma_1, \varphi\rangle$
then $\mu \cup \{\theta\} \rightharpoonup \mu$,

for each $\tau \in$ Term, *for each* $\sigma \in$ Spec, *for each* $\varphi \in$ Feat,
if $\theta \notin \mu$, $\tau \sim \sigma \in \theta$, $\text{Appr}\langle\sigma, \varphi\rangle \neq \emptyset$, $\tau\varphi \in \mathit{Term}(\mu \cup \{\theta\})$, *and*
$\tau\varphi \approx \tau\varphi \notin \theta$
then $\mu \cup \{\theta\} \rightharpoonup \mu \cup \{\theta \cup \{\tau\varphi \approx \tau\varphi\}\}$,

for each $\tau_1 \in$ Term, *for each* $\tau_2 \in$ Term,
if $\theta \notin \mu$, $\tau_1 \not\approx \tau_2 \in \theta$ *and* $\tau_1 \approx \tau_1 \notin \theta$
then $\mu \cup \{\theta\} \rightharpoonup \mu \cup \{\theta \cup \{\tau_1 \approx \tau_1\}\}$,

for each $\tau_1 \in$ Term, *for each* $\tau_2 \in$ Term,
if $\theta \notin \mu$, $\tau_1 \not\approx \tau_2 \in \theta$ *and* $\tau_2 \approx \tau_2 \notin \theta$
then $\mu \cup \{\theta\} \rightharpoonup \mu \cup \{\theta \cup \{\tau_2 \approx \tau_2\}\}$,

for each $\tau_1 \in$ Term, *for each* $\tau_2 \in$ Term,
if $\theta \notin \mu$, $\tau_1 \approx \tau_1 \in \theta$, $\tau_2 \approx \tau_2 \in \theta$, $\tau_1 \approx \tau_2 \notin \theta$ *and* $\tau_1 \not\approx \tau_2 \notin \theta$

then $\mu \cup \{\theta\} \rightharpoonup \mu \cup \{\theta \cup \{\tau_1 \approx \tau_2\}, \theta \cup \{\tau_1 \not\approx \tau_2\}\}$, *and*

for each $\tau_1 \in \mathsf{Term}$, *for each* $\tau_2 \in \mathsf{Term}$,

if $\theta \notin \mu$, $\tau_1 \approx \tau_2 \in \theta$ *and* $\tau_1 \not\approx \tau_2 \in \theta$ *then* $\mu \cup \{\theta\} \rightharpoonup \mu$.

$\rightharpoonup$ is adapted from the rewrite rules of Götz (1994) used in Kepser (1994), and is best understood intuitively as a set of left-to-right rewrite rules on matrices. Some rule in $\rightharpoonup$ applies to a matrix only if the matrix is not exclusive, and the application of the rule better conforms the matrix to the exclusive matrix conditions by always deleting and sometimes adding clauses to the matrix. $\rightharpoonup$ preserves denotations:

Proposition 11. *For each* $\mu \in \mathsf{Matr}$, *for each* $\mu' \in \mathsf{Matr}$,

if $\mu \rightharpoonup \mu'$ *then for each interpretation* I, $M_I(\mu) = M_I(\mu')$.

A matrix is terminal iff no rewrite rule in $\rightharpoonup$ applies to it:

Definition 22. *For each* $\mu \in \mathsf{Matr}$,

μ *is* terminal *iff for no* $\mu' \in \mathsf{Matr}$, $\mu \rightharpoonup \mu'$.

There clearly exists an algorithm `Goetz` that computes a total function from Matr to $\{\textit{provisional}, \textit{terminal}\} \times \mathsf{Matr}$ such that for each $\mu \in \mathsf{Matr}$,

if μ is terminal

then $\texttt{Goetz}(\mu) = \langle \textit{terminal}, \mu \rangle$

else for some $\mu' \in \mathsf{Matr}$,

$\mu \rightharpoonup \mu'$ and $\texttt{Goetz}(\mu) = \langle \textit{provisional}, \mu' \rangle$.

Algorithm `Atomic` deletes the nonatomic literals from the clauses in a matrix. Thus, `Atomic` computes the total function from Matr to Matr such that for each $\mu \in \mathsf{Matr}$,

$\texttt{Atomic}(\mu) = \{\theta \cap \mathsf{Atom} \mid \theta \in \mu\}$.

`Atomic` applied to a terminal matrix yields an exclusive matrix and preserves the denotation:

Proposition 12. *For each terminal* $\mu \in \mathsf{Matr}$,

$\texttt{Atomic}(\mu) \in \mathsf{Excl}$, *and*

for each interpretation I, $M_I(\mu) = M_I(\texttt{Atomic}(\mu))$.

Proof. Firstly, for each terminal $\mu \in \mathsf{Matr}$,

$\texttt{Atomic}(\mu) \in \mathsf{Excl}$.

Secondly, for each terminal $\mu \in \mathsf{Matr}$, for each $\theta \in \mu$, for each interpretation I, for each $\upsilon \in \Theta_I(\theta \cap \mathsf{Atom})$, for each $\tau \in \mathit{Term}(\mu)$,

if $T_I(\tau)(\upsilon)$ is defined then $\tau \approx \tau \in \theta$. by induction on the length of τ

Thus, for each terminal $\mu \in \mathsf{Matr}$, for each $\theta \in \mu$, for each interpretation I, for each $\upsilon \in \Theta_I(\theta \cap \mathsf{Atom})$,

for each $\tau \in \mathsf{Term}$, for each $\sigma \in \mathsf{Spec}$,
$\neg\tau \sim \sigma \in \theta$
$\Longrightarrow \tau \in \mathit{Term}(\mu)$ and $\tau \sim \sigma \notin \theta$
$\Longrightarrow T_I(\tau)(\upsilon)$ is undefined or ($\tau \approx \tau \in \theta$ and $\tau \sim \sigma \notin \theta$)
$\Longrightarrow T_I(\tau)(\upsilon)$ is undefined, or
for some $\sigma' \in \mathsf{Spec} \setminus \{\sigma\}$, $\tau \sim \sigma' \in \theta$
$\Longrightarrow \upsilon \notin D_I(\tau \sim \sigma)$
$\Longrightarrow \upsilon \in D_I(\neg\tau \sim \sigma)$,

for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$,
$\neg\tau_1 \approx \tau_2 \in \theta$
$\Longrightarrow \tau_1 \in \mathit{Term}(\mu)$, $\tau_2 \in \mathit{Term}(\mu)$ and $\tau_1 \approx \tau_2 \notin \theta$
$\Longrightarrow T_I(\tau_1)(\upsilon)$ is undefined,
$T_I(\tau_2)(\upsilon)$ is undefined, or
($\tau_1 \approx \tau_1 \in \theta$, $\tau_2 \approx \tau_2 \in \theta$ and $\tau_1 \approx \tau_2 \notin \theta$)
$\Longrightarrow T_I(\tau_1)(\upsilon)$ is undefined,
$T_I(\tau_2)(\upsilon)$ is undefined, or
$\tau_1 \not\approx \tau_2 \in \theta$
$\Longrightarrow \upsilon \notin D_I(\tau_1 \approx \tau_2)$
$\Longrightarrow \upsilon \in D_I(\neg\tau_1 \approx \tau_2)$, and similarly

for each $\tau_1 \in \mathsf{Term}$, for each $\tau_2 \in \mathsf{Term}$,
if $\neg\tau_1 \not\approx \tau_2 \in \theta$ then $\upsilon \in D_I(\neg\tau_1 \not\approx \tau_2)$.

Thus, for each terminal $\mu \in \mathsf{Matr}$, for each $\theta \in \mu$, for each interpretation I, for each $\upsilon \in \Theta_I(\theta \cap \mathsf{Atom})$, for each $\delta \in \mathsf{Atom}$,
if $\neg\delta \in \theta$ then $\upsilon \in D_I(\neg\delta)$.

Thus, for each terminal $\mu \in \mathsf{Matr}$, for each $\theta \in \mu$, for each interpretation I,
$\Theta_I(\theta) = \Theta_I(\theta \cap \mathsf{Atom})$.

Thus, for each terminal $\mu \in \mathsf{Matr}$, for each interpretation I,
$M_I(\mu) = M_I(\mathtt{Atomic}(\mu))$. □

`Class` is the following program: for each $\theta \in \mathrm{finpow}(\mathsf{Desc})$,

```
input θ into T
put provisional into F
put Matrix(T) into M
while F = provisional
   do put Goetz(M) into G
      put left(G) into F
      put right(G) into M
   end
```

`put Atomic(M) into E`
`output E`,

where for each register `X`,

if the content of `X` is $\langle y, z \rangle$ then `left(X)` is y and `right(X)` is z,

`Matrix(X)` is the result of applying algorithm `Matrix` to the content of `X`,

`Goetz(X)` is the result of applying algorithm `Goetz` to the content of `X`, and

`Atomic(X)` is the result of applying algorithm `Atomic` to the content of `X`.

`Class` effectively deduces from each finite theory an exclusive matrix that is semantically equivalent to the theory:

Theorem 4. `Class` *computes a total function from* finpow(Desc) *to* Excl *such that for each* $\theta \in$ finpow(Desc), *for each interpretation* I,

$\Theta_I(\theta) = M_I(\texttt{Class}(\theta))$.

Proof. Suppose that $\lhd$ is the binary relation on Matr $\times$ Matr such that for each $\mu_1 \in$ Matr, for each $\mu_2 \in$ Matr,

$\mu_1 \lhd \mu_2$
iff for some $\mu \in$ Matr,
$\mu \subset \mu_1$,
$\mu \subseteq \mu_2$, and
for each $\theta_2 \in \mu_2 \setminus \mu$, for some $\theta_1 \in \mu_1 \setminus \mu$, $\theta_1 \subset \theta_2$.

Then $\lhd$ is transitive and irreflexive:

for each $\mu_1 \in$ Matr, for each $\mu_2 \in$ Matr, for each $\mu_3 \in$ Matr,
$\mu_1 \lhd \mu_2$ and $\mu_2 \lhd \mu_3$
$\Longrightarrow$ for some $\mu \in$ Matr, for some $\hat{\mu} \in$ Matr,
$\mu \subset \mu_1$, $\mu \subseteq \mu_2$, $\hat{\mu} \subset \mu_2$, $\hat{\mu} \subseteq \mu_3$,
for each $\theta_2 \in \mu_2 \setminus \mu$, for some $\theta_1 \in \mu_1 \setminus \mu$, $\theta_1 \subset \theta_2$, and
for each $\theta_3 \in \mu_3 \setminus \hat{\mu}$, for some $\theta_2 \in \mu_2 \setminus \hat{\mu}$, $\theta_2 \subset \theta_3$
$\Longrightarrow$ for some $\mu \in$ Matr, for some $\hat{\mu} \in$ Matr,
$\mu \cap \hat{\mu} \subset \mu_1$, $\mu \cap \hat{\mu} \subseteq \mu_3$, and
for each $\theta_3 \in$ Clau,
$\theta_3 \in \mu_3 \setminus (\mu \cap \hat{\mu})$
$\Longrightarrow \theta_3 \in \mu_3 \setminus \hat{\mu}$ or $\theta_3 \in \hat{\mu} \setminus \mu$
$\Longrightarrow$ for some $\theta_2 \in \mu_2 \setminus \hat{\mu}$, $\theta_2 \subset \theta_3$, or
$\theta_3 \in \mu_2 \setminus \mu$
$\Longrightarrow$ for some $\theta_2 \in \mu_2 \setminus \mu$, $\theta_2 \subset \theta_3$,
for some $\theta_2 \in \mu \setminus \hat{\mu}$, $\theta_2 \subset \theta_3$, or
for some $\theta_1 \in \mu_1 \setminus \mu$, $\theta_1 \subset \theta_3$
$\Longrightarrow$ for some $\theta_1 \in \mu_1 \setminus \mu$, $\theta_1 \subset \theta_3$, or
for some $\theta_1 \in \mu_1 \setminus \hat{\mu}$, $\theta_1 \subset \theta_3$

$\Longrightarrow$ for some $\theta_1 \in \mu_1 \setminus (\mu \cap \hat{\mu})$, $\theta_1 \subset \theta_3$

$\Longrightarrow \mu_1 \lhd \mu_3$, and

for each $\mu_0 \in$ Matr,

$\mu_0 \lhd \mu_0$

$\Longrightarrow$ for some $\mu \in$ Matr,

$\mu \subset \mu_0$, and

for each $\theta_2 \in \mu_0 \setminus \mu$, for some $\theta_1 \in \mu_0 \setminus \mu$, $\theta_1 \subset \theta_2$

$\Longrightarrow \mu_0$ is infinite

$\Longrightarrow$ contradiction.

Thus,

for some $\{\mu_n \mid n \in \mathbb{N}\} \subseteq$ Matr, for each $n \in \mathbb{N}$, $\mu_n \rightharpoonup \mu_{n+1}$

$\Longrightarrow$ for some $\{\mu_n \mid n \in \mathbb{N}\} \subseteq$ Matr, for each $n \in \mathbb{N}$,

$\mu_n \rightharpoonup \mu_{n+1}$ and $\mathit{Term}(\mu_n) \subseteq \mathit{Term}(\mu_0) \cup \{:\}$ by induction on n

$\Longrightarrow$ for some finite $\{\mu_n \mid n \in \mathbb{N}\} \subseteq$ Matr, for each $n \in \mathbb{N}$, $\mu_n \rightharpoonup \mu_{n+1}$

$\Longrightarrow$ for some finite $\{\mu_n \mid n \in \mathbb{N}\} \subseteq$ Matr, for each $n \in \mathbb{N}$, $\mu_n \lhd \mu_{n+1}$

$\Longrightarrow$ for some finite $\{\mu_n \mid n \in \mathbb{N}\} \subseteq$ Matr, for each $m \in \mathbb{N}$, for each $n \in \mathbb{N}$,

if $m < n$ then $\mu_m \lhd \mu_n$

$\Longrightarrow$ for some $\mu \in$ Matr, $\mu \lhd \mu$

$\Longrightarrow$ contradiction.

Thus, by Propositions 10, 11 and 12,

`Class` computes a total function from finpow(Desc) to Excl such that for each $\theta \in$ finpow(Desc), for each interpretation I,

$\Theta_I(\theta) = M_I(\texttt{Class}(\theta))$. □

6.2. THE `Index` ALGORITHM

Given a nonempty exclusive matrix μ, `Index` must construct an index tree that analyzes μ. In describing the operation of `Index` it is helpful to have a term to refer to a node marked with a nonempty exclusive matrix but not yet labeled with a query or a clause. We call such a node a *pending* node.

`Index` begins with a single pending node, ε marked with μ. From this pending node, `Index` constructs an index tree by recursively converting each pending node into an inner or outer node of the tree. The conversion of a pending node may itself introduce new pending nodes. Faced with a pending node ν and its mark μ', `Index` first considers μ' in order to judge whether ν should be converted to an inner or an outer node. If μ' contains exactly one clause θ then `Index` judges ν to be an outer node and labels ν with θ. However, if μ' contains more than one clause then `Index` judges ν to be an inner node and applies two algorithms, `Select` and `Grow`, to ν.

`Select` selects a query κ in $\mathit{Query}(\mu')$, and `Index` labels ν with κ. `Grow` then helps `Index` create a number of labeled directed edges leaving ν for marked nodes in the following manner. By Proposition 4, there are only finitely many responses. For each response ρ, `Grow` ascertains whether $\mathit{Residue}\langle\mu', \kappa, \rho\rangle$ is empty. If so then `Index` creates no directed edge labeled ρ leaving ν. If not then `Index` creates a directed edge labeled ρ from ν to node $\nu\rho$ and marks $\nu\rho$ with $\mathit{Residue}\langle\mu', \kappa, \rho\rangle$. Notice two points. Firstly, by Proposition 5,

for some $\rho \in \mathsf{Resp}$, $\mathit{Residue}\langle\mu', \kappa, \rho\rangle \neq \emptyset$.

Thus, `Index` creates at least one directed edge leaving ν. Thus, `Index` is correct to convert ν to an inner node. Secondly, again by Proposition 5,

for each $\rho \in \mathsf{Resp}$, $\mathit{Residue}\langle\mu', \kappa, \rho\rangle \in \mathsf{Excl}$.

Thus, all of the directed edges from ν that `Index` creates go to nodes marked with nonempty exclusive matrices. Thus, `Index` only ever creates pending nodes.

If `Index` stops – that is, `Index` eventually converts every pending node into an inner or outer node – then the result is an index tree that analyzes μ. To ensure `Index` stops, `Select` selects queries carefully, and it is this careful selection that makes `Index` so complicated. Whenever `Index` judges a pending node ν with mark μ' to be an inner node and applies `Select` to ν, `Select` always selects a query κ in $\mathit{Query}(\mu')$ such that

for each $\rho \in \mathsf{Resp}$, $\mathit{Residue}\langle\mu', \kappa, \rho\rangle \subset \mu'$.

Thus, each time `Index` calls `Select` to convert a pending node to an inner node and `Grow` to create new pending nodes, the matrices marking the new pending nodes have fewer clauses than the matrix marking the old pending node. Eventually, all markings of pending nodes are reduced to singletons, whereupon `Index` converts the pending nodes to outer nodes and creates no new pending nodes. Thus, `Index` must eventually stop and output an index tree that analyzes μ.

As already mentioned, `Index` has two component algorithms, `Select` to help label pending nodes with queries, and `Grow` to help create new directed edges and pending nodes. `Select` in turn has a component algorithm called `Weight`. We give each of `Weight`, `Select`, `Grow` and `Index` in turn.

Algorithm `Weight` deduces, for each matrix and each query, the cardinality of the largest residue. Suppose that

μ is a matrix,

κ is a query, and

$\mathsf{Resp} = \{\rho_0, \ldots, \rho_n\}$.

Suppose further that

for each $i \in \{0, \ldots, n\}$, r_i is the cardinality of $\mathit{Residue}\langle\mu, \kappa, \rho_i\rangle$.

Then

`Weight`$\langle\mu, \kappa\rangle$ is the largest number in $\{r_1, \ldots, r_n\}$.

Formally, `Weight` computes the total function from $\mathsf{Matr} \times \mathsf{Quer}$ to $\mathbb{N}$ such that for each $\mu \in \mathsf{Matr}$, for each $\kappa \in \mathsf{Quer}$,

$\texttt{Weight}\langle\mu, \kappa\rangle = \max\{\text{card}(\textit{Residue}\langle\mu, \kappa, \rho\rangle) \mid \rho \in \textsf{Resp}\}$.

To ensure `Index` stops, algorithm `Select` need only select for each exclusive matrix μ that contains more than one clause (remember that `Index` only applies `Select` to a pending node if the matrix marking the node contains more than one clause) a query κ in $\textit{Query}(\mu)$ such that $\texttt{Weight}\langle\mu, \kappa\rangle$ is smaller than the cardinality of μ. $\textit{Query}(\mu)$ always contains such a query:

Proposition 13. For each $\mu \in \textsf{Excl}$,

if $\text{card}(\mu) > 1$ *then for some* $\kappa \in \textit{Query}(\mu)$, $\texttt{Weight}\langle\mu, \kappa\rangle < \text{card}(\mu)$.

Proof. For each $\mu \in \textsf{Excl}$, for each $\theta_1 \in \mu$, for each $\theta_2 \in \mu$,

for each $\tau \in \textsf{Term}$, for each $\sigma \in \textsf{Spec}$,
if $\tau \sim \sigma \in (\theta_1 \setminus \theta_2)$
then for each $\rho \in \textsf{Resp}$,
$\theta_1 \notin \textit{Residue}\langle\mu, \boxed{\tau}, \rho\rangle$ or $\theta_2 \notin \textit{Residue}\langle\mu, \boxed{\tau}, \rho\rangle$,

for each $\tau_1 \in \textsf{Term}$, for each $\tau_2 \in \textsf{Term}$,
if $\tau_1 \approx \tau_2 \in (\theta_1 \setminus \theta_2)$
then for each $\rho \in \textsf{Resp}$,
$\theta_1 \notin \textit{Residue}\langle\mu, \boxed{\tau_1, \tau_2}, \rho\rangle$ or $\theta_2 \notin \textit{Residue}\langle\mu, \boxed{\tau_1, \tau_2}, \rho\rangle$, and

for each $\tau_1 \in \textsf{Term}$, for each $\tau_2 \in \textsf{Term}$,
if $\tau_1 \not\approx \tau_2 \in (\theta_1 \setminus \theta_2)$
then for each $\rho \in \textsf{Resp}$,
$\theta_1 \notin \textit{Residue}\langle\mu, \boxed{\tau_1, \tau_2}, \rho\rangle$ or $\theta_2 \notin \textit{Residue}\langle\mu, \boxed{\tau_1, \tau_2}, \rho\rangle$.

Thus, for each $\mu \in \textsf{Excl}$,

$\text{card}(\mu) > 1$
$\Longrightarrow$ for some $\theta_1 \in \mu$, for some $\theta_2 \in \mu$, for some $\delta \in \textsf{Atom}$, $\delta \in (\theta_1 \setminus \theta_2)$
$\Longrightarrow$ for some $\theta_1 \in \mu$, for some $\theta_2 \in \mu$, for some $\kappa \in \textit{Query}(\mu)$,
for each $\rho \in \textsf{Resp}$,
$\theta_1 \notin \textit{Residue}\langle\mu, \kappa, \rho\rangle$ or $\theta_2 \notin \textit{Residue}\langle\mu, \kappa, \rho\rangle$
$\Longrightarrow$ for some $\kappa \in \textit{Query}(\mu)$, for each $\rho \in \textsf{Resp}$, $\textit{Residue}\langle\mu, \kappa, \rho\rangle \subset \mu$
$\Longrightarrow$ for some $\kappa \in \textit{Query}(\mu)$, $\texttt{Weight}\langle\mu, \kappa\rangle < \text{card}(\mu)$. □

Actually, in order to construct a shorter index tree and hence a more efficient index, `Select` selects for each nonempty exclusive matrix μ a query κ in nonempty $\textit{Query}(\mu)$ with the smallest $\texttt{Weight}\langle\mu, \kappa\rangle$. Suppose that

μ is a nonempty exclusive matrix,
$\textit{Query}(\mu) = \{\kappa_0, \ldots, \kappa_n\}$, and
for each $i \in \{0, \ldots, n\}$, $w_i = \texttt{Weight}\langle\mu, \kappa_i\rangle$

Then, for some $j \in \{0, \ldots, n\}$,

$\texttt{Select}(\mu) = \kappa_j$ and w_j is the smallest number in $\{w_0, \ldots, w_n\}$.

Formally, Select computes a total function from Excl \ {∅} to Quer such that for each $\mu \in$ Excl \ {∅},

Select$(\mu) \in$ *Query*(μ), and

Weight$\langle \mu,$ Select$(\mu)\rangle = \min\{$Weight$\langle \mu, \kappa\rangle \mid \kappa \in$ *Query*$(\mu)\}$.

We immediately have that if exclusive matrix μ contains more than one clause then Weight$\langle \mu,$ Select$(\mu)\rangle$ is smaller than the cardinality of μ:

Proposition 14. For each $\mu \in$ Excl,

if card$(\mu) > 1$ *then* Weight$\langle \mu,$ Select$(\mu)\rangle <$ card(μ).

Proof. Follows immediately from Proposition 13. □

Algorithm Grow deduces from each matrix and each query a sequence of ordered pairs comprising responses with nonempty residues and their residues. Formally, Grow computes the total function from Matr × Quer to (Resp × Matr)* such that for each $\mu \in$ Matr, for each $\kappa \in$ Quer, for some well-ordering $\langle \rho_1, \ldots, \rho_n\rangle$ of $\{\rho \in$ Resp $\mid$ *Residue*$\langle \mu, \kappa, \rho\rangle \neq \emptyset\}$,

Grow$\langle \mu, \kappa\rangle = \langle\langle \rho_1,$ *Residue*$\langle \mu, \kappa, \rho_1\rangle\rangle, \ldots, \langle \rho_n,$ *Residue*$\langle \mu, \kappa, \rho_n\rangle\rangle\rangle$.

Grow applied to a nonempty exclusive matrix and a query yields a nonempty sequence of responses and nonempty exclusive matrices:

Proposition 15. For each $\mu \in$ Excl \ {∅}, *for each* $\kappa \in$ Quer,

Grow$\langle \mu, \kappa\rangle \in$ (Resp × (Excl \ {∅}))$^+$.

Proof. Follows immediately from Proposition 5. □

Index is the following program: for each $\mu \in$ Excl \ {∅},

```
input μ into E
put ∅ into I
put ∅ into O
put ∅ into L
put ⟨⟨ε, E⟩⟩ into P
while P is a nonempty sequence
   do put head(P) into H
      put tail(P) into P
      put left(H) into N
      put right(H) into E
      if E is a singleton set
         then put member(E) into C
              put O ∪ {N} into O
              put L ∪ {⟨N, C⟩} into L
         else put Select(E) into Q
```

```
                put Grow⟨E, Q⟩ into G
                put insert⟨N, G⟩ into G
                put I ∪ {N} into I
                put L ∪ {⟨N, Q⟩} into L
                put concatenate⟨P, G⟩ into P
            end
        end
    put ⟨I ∪ O, I, O, L⟩ into T
    output T,
```

where for each register X, for each register Y,

if the content of X is $\langle x_0, \ldots, x_n \rangle$
then head(X) is x_0 and tail(X) is $\langle x_1, \ldots, x_n \rangle$,
if the content of X is $\langle y, z \rangle$ then left(X) is y and right(X) is z,
if the content of X is $\{x\}$ then member(X) is x,
if the content of X is $x_1 \ldots x_m$ and the content of Y is $\langle \langle y_1, z_1 \rangle, \ldots, \langle y_n, z_n \rangle \rangle$
then insert⟨X, Y⟩ is $\langle \langle x_1 \ldots x_m y_1, z_1 \rangle, \ldots, \langle x_1 \ldots x_m y_n, z_n \rangle \rangle$,
if the content of X is $\langle x_1, \ldots, x_m \rangle$ and the content of Y is $\langle y_1, \ldots, y_n \rangle$
then concatenate(X, Y) is $\langle x_1, \ldots, x_m, y_1, \ldots, y_n \rangle$,
Select(X) is the result of applying algorithm Select to the content of X, and
Grow⟨X, Y⟩ is the result of applying algorithm Grow to the contents of X and Y.

Index effectively deduces from each nonempty exclusive matrix an index tree that analyzes the matrix:

Theorem 5. Index *computes a total function from* Excl $\setminus \{\emptyset\}$ *to* Tree *such that for each* $\mu \in$ Excl $\setminus \{\emptyset\}$,

Index(μ) *analyzes* μ.

Proof. Informally and in outline, we first suppose that μ is an arbitrary nonempty exclusive matrix and define a number of mathematical entities, $\mathcal{T}$ among them. We then prove two technical results and use these results to show that

$\mathcal{T}$ is an index tree that analyzes μ, and
if Index starts with input μ then Index stops with output $\mathcal{T}$.

Formally and in detail, suppose that

$\mu \in$ Excl $\setminus \{\emptyset\}$.

Suppose further that

Pend $=$ Resp$^\sharp \times$ (Excl $\setminus \{\emptyset\}$), and
Nex and *Coda* are the total functions from Pend$^+$ to Excl $\setminus \{\emptyset\}$ such that for each $\langle \langle \nu_0, \mu_0 \rangle, \ldots, \langle \nu_m, \mu_m \rangle \rangle \in$ Pend$^+$,
Nex$\langle \langle \nu_0, \mu_0 \rangle, \ldots, \langle \nu_m, \mu_m \rangle \rangle = \mu_0$, and
if Grow$\langle \mu_0,$ Select$(\mu_0) \rangle = \langle \langle \rho_0, \mu_0' \rangle, \ldots, \langle \rho_n, \mu_n' \rangle \rangle$

then $Coda\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle = \langle\langle \nu_0\rho_0, \mu_0'\rangle, \ldots, \langle \nu_0\rho_n, \mu_n'\rangle\rangle$.

Proposition 15 ensures *Coda* has range Pend^+. We use *Nex* and *Coda* to inductively define six families of sets. Suppose that

$\mathcal{N}_0 = \emptyset$,
$\mathcal{I}_0 = \emptyset$,
$\mathcal{O}_0 = \emptyset$,
$\mathcal{L}_0 = \emptyset$,
$\mathcal{M}_0 = \emptyset$, and
$\mathcal{P}_0 = \langle\langle \varepsilon, \mu\rangle\rangle$.

Suppose that for each $i \in \mathbb{N}$,

if $\mathcal{P}_i \notin \mathsf{Pend}^+$
then $\mathcal{N}_{i+1} = \mathcal{N}_i$,
$\mathcal{I}_{i+1} = \mathcal{I}_i$,
$\mathcal{O}_{i+1} = \mathcal{O}_i$,
$\mathcal{L}_{i+1} = \mathcal{L}_i$,
$\mathcal{M}_{i+1} = \mathcal{M}_i$, and
$\mathcal{P}_{i+1} = \mathcal{P}_i$,

if $\mathcal{P}_i \in \mathsf{Pend}^+$ and $Nex(\mathcal{P}_i)$ is a singleton set
then $\mathcal{N}_{i+1} = \mathcal{N}_i \cup \{\nu_0\}$,
$\mathcal{I}_{i+1} = \mathcal{I}_i$,
$\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{\nu_0\}$,
$\mathcal{L}_{i+1} = \mathcal{L}_i \cup \{\langle \nu_0, \theta\rangle\}$,
$\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{\langle \nu_0, \mu_0\rangle\}$, and
$\mathcal{P}_{i+1} = \langle\langle \nu_1, \mu_1\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$,
where $\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$ and $Nex(\mathcal{P}_i) = \{\theta\}$, and

if $\mathcal{P}_i \in \mathsf{Pend}^+$ and $Nex(\mathcal{P}_i)$ is not a singleton set
then $\mathcal{N}_{i+1} = \mathcal{N}_i \cup \{\nu_0\}$,
$\mathcal{I}_{i+1} = \mathcal{I}_i \cup \{\nu_0\}$,
$\mathcal{O}_{i+1} = \mathcal{O}_i$,
$\mathcal{L}_{i+1} = \mathcal{L}_i \cup \{\langle \nu_0, \mathtt{Select}(\mu_0)\rangle\}$,
$\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{\langle \nu_0, \mu_0\rangle\}$, and
$\mathcal{P}_{i+1} = \langle\langle \nu_1, \mu_1\rangle, \ldots, \langle \nu_m, \mu_m\rangle, \langle \nu_0', \mu_0'\rangle, \ldots, \langle \nu_n', \mu_n'\rangle\rangle$,
where $\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$, and
$Coda(\mathcal{P}_i) = \langle\langle \nu_0', \mu_0'\rangle, \ldots, \langle \nu_n', \mu_n'\rangle\rangle$.

From these families of sets we define $\mathcal{T}$. Suppose that

$\mathcal{N} = \bigcup\{\mathcal{N}_i \mid i \in \mathbb{N}\}$,
$\mathcal{I} = \bigcup\{\mathcal{I}_i \mid i \in \mathbb{N}\}$,
$\mathcal{O} = \bigcup\{\mathcal{O}_i \mid i \in \mathbb{N}\}$,

$\mathcal{L} = \bigcup\{\mathcal{L}_i \mid i \in \mathbb{N}\}$,
$\mathcal{M} = \bigcup\{\mathcal{M}_i \mid i \in \mathbb{N}\}$, and
$\mathcal{T} = \langle \mathcal{N}, \mathcal{I}, \mathcal{O}, \mathcal{L} \rangle$.

We now prove two technical claims:

CLAIM 5.1. *For each* $i \in \mathbb{N}$, *for each* $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
if $\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$ *then* $\nu_0 \notin \mathcal{N}_i$.

Proof of Claim. Suppose that $\preceq$ is the binary relation on $\mathsf{Resp}^\sharp \times \mathsf{Resp}^\sharp$ such that for each $\nu_1 \in \mathsf{Resp}^\sharp$, for each $\nu_2 \in \mathsf{Resp}^\sharp$,

$\nu_1 \preceq \nu_2$ iff for some $\nu \in \mathsf{Resp}^\sharp$, $\nu_1\nu = \nu_2$.

Then for each $i \in \mathbb{N}$, for each $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
if $\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$
then for each $j \in \{0, \ldots, m\}$,
for each $k \in \{0, \ldots, m\}$, if $j \neq k$ then $\nu_j \not\preceq \nu_k$, and
for each $\nu \in \mathcal{N}_i$, $\nu_j \not\preceq \nu$. by induction on i

Thus, for each $i \in \mathbb{N}$, for each $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
if $\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$ then $\nu_0 \notin \mathcal{N}_i$. ◊

CLAIM 5.2. *For each* $i \in \mathbb{N}$, *for each* $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
if $\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$
then for each $j \in \{0, \ldots, m\}$, *for some* $\langle\langle \nu'_1, \mu'_1\rangle, \ldots, \langle \nu'_n, \mu'_n\rangle\rangle \in \mathsf{Pend}^*$,
$\mathcal{P}_{i+j} = \langle\langle \nu_j, \mu_j\rangle, \ldots, \langle \nu_m, \mu_m\rangle, \langle \nu'_1, \mu'_1\rangle, \ldots, \langle \nu'_n, \mu'_n\rangle\rangle$.

Proof of Claim. By induction on j. ◊

With the help of claims 5.1 and 5.2, we now prove three relevent claims:

CLAIM 5.3. $\mathcal{T}$ *is an index tree.*
Proof of Claim. Firstly, for each $\mu' \in \mathsf{Excl} \setminus \{\emptyset\}$,
μ' is not a singleton set
$\Longrightarrow \mathtt{Weight}\langle \mu', \mathtt{Select}(\mu')\rangle < \mathrm{card}(\mu')$ by Proposition 14
$\Longrightarrow$ for each $\rho \in \mathsf{Resp}$, $\mathrm{card}(\mathit{Residue}\langle \mu', \mathtt{Select}(\mu'), \rho\rangle) < \mathrm{card}(\mu')$
$\Longrightarrow$ for each $\langle\langle \rho_1, \mu'_1\rangle, \ldots, \langle \rho_n, \mu'_n\rangle\rangle \in (\mathsf{Resp} \times \mathsf{Matr})^*$,
if $\mathtt{Grow}\langle \mu', \mathtt{Select}(\mu')\rangle = \langle\langle \rho_1, \mu'_1\rangle, \ldots, \langle \rho_n, \mu'_n\rangle\rangle$
then for each $j \in \{1, \ldots, n\}$, $\mathrm{card}(\mu'_j) < \mathrm{card}(\mu')$.

Thus, for each $i \in \mathbb{N}$, for each $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
if $\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$
then for each $j \in \{0, \ldots, m\}$, $\mathrm{length}(\nu_j) + \mathrm{card}(\mu_j) \leqslant \mathrm{card}(\mu)$.
by induction on i

Thus, for each ν,

$\nu \in \mathcal{N}$

$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$ and $\nu = \nu_0$

$\Longrightarrow \nu \in \mathsf{Resp}^\sharp$ and $\mathrm{length}(\nu) \leqslant \mathrm{card}(\mu)$.

Thus,

$\mathcal{N} \subseteq \mathsf{Resp}^0 \cup \ldots \cup \mathsf{Resp}^{\mathrm{card}(\mu)}$.

Thus, by Proposition 4,

$\mathcal{N} \in \mathrm{finpow}(\mathsf{Resp}^\sharp)$. ... (1)

Secondly,

$\mathcal{P}_0 = \langle\langle \varepsilon, \mu\rangle\rangle \in \mathsf{Pend}^+$.

Thus,

$\varepsilon \in \mathcal{N}$. ... (2)

Thirdly, for each $\nu \in \mathsf{Resp}^\sharp$, for each $\rho \in \mathsf{Resp}$,

$\nu\rho \in \mathcal{N}$

$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$ and $\nu\rho = \nu_0$

$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
for some $\langle\langle \nu'_0, \mu'_0\rangle, \ldots, \langle \nu'_n, \mu'_n\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$,
$\mathit{Coda}(\mathcal{P}_i) = \langle\langle \nu'_0, \mu'_0\rangle, \ldots, \langle \nu'_n, \mu'_n\rangle\rangle$, and
for some $j \in \{0, \ldots, n\}$, $\nu\rho = \nu'_j$

$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$ and $\nu = \nu_0$

$\Longrightarrow \nu \in \mathcal{N}$. ... (3)

Fourthly, for each ν,

$\nu \in \mathcal{U}$

$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$,
$\mathit{Nex}(\mathcal{P}_i)$ is not a singleton set, and
$\nu = \nu_0$

$\Longrightarrow \nu \in \mathcal{N}$, and
for some $\rho \in \mathsf{Resp}$, for some $i \in \mathbb{N}$,
for some $\langle\langle \nu_1, \mu_1\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^*$,
for some $\langle\langle \nu'_0, \mu'_0\rangle, \ldots, \langle \nu'_n, \mu'_n\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_{i+1} = \langle\langle \nu_1, \mu_1\rangle, \ldots, \langle \nu_m, \mu_m\rangle, \langle \nu'_0, \mu'_0\rangle, \ldots, \langle \nu'_n, \mu'_n\rangle\rangle$, and
$\nu\rho = \nu'_0$

$\Longrightarrow \nu \in \mathcal{N}$, and

for some $\rho \in \mathsf{Resp}$, for some $i \in \mathbb{N}$,
for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$ and $\nu\rho = \nu_0$ by Claim 5.2

$\Longrightarrow \nu \in \mathcal{N}$ and for some $\rho \in \mathsf{Resp}$, $\nu\rho \in \mathcal{N}$.

Moreover, for each $\nu \in \mathcal{N}$,

for some $\rho \in \mathsf{Resp}$, $\nu\rho \in \mathcal{N}$
$\Longrightarrow$ for some $\rho \in \mathsf{Resp}$, for some $i \in \mathbb{N}$,
for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$ and $\nu\rho = \nu_0$
$\Longrightarrow$ for some $\rho \in \mathsf{Resp}$, for some $i \in \mathbb{N}$,
for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
for some $\langle\langle \nu'_0, \mu'_0\rangle, \ldots, \langle \nu'_n, \mu'_n\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$,
$\mathit{Nex}(\mathcal{P}_i)$ is not a singleton set,
$\mathit{Coda}(\mathcal{P}_i) = \langle\langle \nu'_0, \mu'_0\rangle, \ldots, \langle \nu'_n, \mu'_n\rangle\rangle$, and
for some $j \in \{0, \ldots, n\}$, $\nu\rho = \nu'_j$
$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$,
$\mathit{Nex}(\mathcal{P}_i)$ is not a singleton set, and
$\nu = \nu_0$
$\Longrightarrow \nu \in \mathcal{I}$.

Thus,

$\mathcal{I} = \{\nu \in \mathcal{N} \mid$ for some $\rho \in \mathsf{Resp}$, $\nu\rho \in \mathcal{N}\}$. ... (4)

Fifthly, for each $i \in \mathbb{N}$,

$\mathcal{I}_i \cup \mathcal{O}_i = \mathcal{N}_i$ and $\mathcal{I}_i \cap \mathcal{O}_i = \emptyset$. by induction on i and Claim 5.1

Thus,

$\mathcal{I} \cup \mathcal{O} = \mathcal{N}$ and $\mathcal{I} \cap \mathcal{O} = \emptyset$.

Thus,

$\mathcal{O} = \mathcal{N} \setminus \mathcal{I} = \{\nu \in \mathcal{N} \mid$ for each $\rho \in \mathsf{Resp}$, $\nu\rho \notin \mathcal{N}\}$. ... (5)

Sixthly, for each $i \in \mathbb{N}$,

$\mathcal{L}_i$ is a total function from $\mathcal{N}_i$ to $\mathsf{Quer} \cup \mathsf{Clau}$. by induction on i and claim 5.1

Thus,

$\mathcal{L}$ is a total function from $\mathcal{N}$ to $\mathsf{Quer} \cup \mathsf{Clau}$. ... (6)

Seventhly, for each ν,

$\nu \in \mathcal{I}$
$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$,

$\mathit{Nex}(\mathcal{P}_i)$ is not a singleton set, and
$\nu = \nu_0$
$\Longrightarrow \mathcal{L}(\nu) \in \mathsf{Quer}$. ... (7)

Eighthly, for each ν,
$\nu \in \mathcal{O}$
$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$,
$\mathit{Nex}(\mathcal{P}_i)$ is a singleton set, and
$\nu = \nu_0$
$\Longrightarrow \mathcal{L}(\nu) \in \mathsf{Clau}$. ... (8)

Therefore, by (1)–(8),
$\mathcal{T}$ is an index tree. ◊

CLAIM 5.4. *$\mathcal{T}$ analyzes μ.*

Proof of Claim. Firstly, for each $i \in \mathbb{N}$,
$\mathcal{M}_i$ is a total function from $\mathcal{N}_i$ to Excl. by induction on i and claim 5.1

Thus,
$\mathcal{M}$ is a total function from $\mathcal{N}$ to Excl. ... (1)

Secondly, for each ν,
$\nu \in \mathcal{I}$
$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$,
$\mathit{Nex}(\mathcal{P}_i)$ is not a singleton set, and
$\nu = \nu_0$
$\Longrightarrow \mathcal{L}(\nu) \in \mathit{Query}(\mathcal{M}(\nu))$. ... (2)

Thirdly, for each $\nu \in \mathcal{I}$, for each $\rho \in \mathsf{Resp}$,
$\mathit{Residue}\langle \mathcal{M}(\nu), \mathcal{L}(\nu), \rho\rangle \neq \emptyset$
$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$,
$\mathit{Nex}(\mathcal{P}_i)$ is not a singleton set,
$\nu = \nu_0$, and
$\mathit{Residue}\langle \mathcal{M}(\nu), \mathcal{L}(\nu), \rho\rangle \neq \emptyset$
$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_1, \mu_1\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^*$,
for some $\langle\langle \nu'_0, \mu'_0\rangle, \ldots, \langle \nu'_n, \mu'_n\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_{i+1} = \langle\langle \nu_1, \mu_1\rangle, \ldots, \langle \nu_m, \mu_m\rangle, \langle \nu'_0, \mu'_0\rangle, \ldots, \langle \nu'_n, \mu'_n\rangle\rangle$, and
for some $j \in \{0, \ldots, n\}$, $\langle \nu\rho, \mathit{Residue}\langle \mathcal{M}(\nu), \mathcal{L}(\nu), \rho\rangle\rangle = \langle \nu'_j, \mu'_j\rangle$
$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,

$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$, and
$\langle \nu\rho, \mathit{Residue}\langle \mathcal{M}(\nu), \mathcal{L}(\nu), \rho\rangle\rangle = \langle \nu_0, \mu_0\rangle$ by Claim 5.2

$\Longrightarrow \nu\rho \in \mathcal{N}$ and $\mathcal{M}(\nu\rho) = \mathit{Residue}\langle \mathcal{M}(\nu), \mathcal{L}(\nu), \rho\rangle$. ... (3)

Fourthly, for each ν,

$\nu \in \mathcal{O}$
$\Longrightarrow$ for some $i \in \mathbb{N}$, for some $\langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle \in \mathsf{Pend}^+$,
$\mathcal{P}_i = \langle\langle \nu_0, \mu_0\rangle, \ldots, \langle \nu_m, \mu_m\rangle\rangle$,
$\mathit{Nex}(\mathcal{P}_i)$ is a singleton set, and
$\nu = \nu_0$
$\Longrightarrow \mathcal{M}(\nu) = \{\mathcal{L}(\nu)\}$. ... (4)

Fifthly,

$\mathcal{P}_0 = \langle\langle \varepsilon, \mu\rangle\rangle \in \mathsf{Pend}^+$.

Thus,

$\mathcal{M}(\varepsilon) = \mu$. ... (5)

Therefore, by (1)–(5),

$\mathcal{T}$ analyzes μ. ◊

CLAIM 5.5. *If* `Index` *starts with input* μ *then* `Index` *stops with output* $\mathcal{T}$.

Proof of Claim. For each $i \in \mathbb{N}$,

if `Index` starts with input μ and $\mathcal{P}_i \in \mathsf{Pend}^+$
then `Index` tests the `while` condition for the $(i+1)$th time, and
the contents of registers `I`, `O`, `L` and `P` are respectively $\mathcal{I}_i$, $\mathcal{O}_i$, $\mathcal{L}_i$ and $\mathcal{P}_i$ as `Index` does so. by induction on i

Thus, for each $i \in \mathbb{N}$,

if `Index` starts with input μ and $\mathcal{P}_i \notin \mathsf{Pend}^+$
then `Index` stops with output $\langle \mathcal{I}_i \cup \mathcal{O}_i, \mathcal{I}_i, \mathcal{O}_i, \mathcal{L}_i\rangle$. by induction on i

However, for each $i \in \mathbb{N}$,

$\mathcal{P}_i \notin \mathsf{Pend}^+$
$\Longrightarrow$ for each $j \in \mathbb{N}$,
if $j \leqslant i$ then $\mathcal{I}_j \subseteq \mathcal{I}_i$, $\mathcal{O}_j \subseteq \mathcal{O}_i$ and $\mathcal{L}_j \subseteq \mathcal{L}_i$, and
if $j \geqslant i$ then $\mathcal{I}_j = \mathcal{I}_i$, $\mathcal{O}_j = \mathcal{O}_i$ and $\mathcal{L}_j = \mathcal{L}_i$
$\Longrightarrow \mathcal{I} = \mathcal{I}_i$, $\mathcal{O} = \mathcal{O}_i$ and $\mathcal{L} = \mathcal{L}_i$.
$\Longrightarrow \mathcal{T} = \langle \mathcal{I}_i \cup \mathcal{O}_i, \mathcal{I}_i, \mathcal{O}_i, \mathcal{L}_i\rangle$. by Claim 5.3

Thus, for each $i \in \mathbb{N}$,

if `Index` starts with input μ and $\mathcal{P}_i \notin \mathsf{Pend}^+$ then `Index` stops with output $\mathcal{T}$.

Moreover,

for each $i \in \mathbb{N}$, $\mathcal{P}_i \in \mathsf{Pend}^+$

$\Longrightarrow$ for each $i \in \mathbb{N}$, $\mathcal{N}_i \subset \mathcal{N}_{i+1}$ by Claim 5.1
$\Longrightarrow$ $\mathcal{N}$ is an infinite set
$\Longrightarrow$ contradiction. by Claim 5.3

Thus,

`Index` starts with input μ
$\Longrightarrow$ `Index` starts with input μ and for some $i \in \mathbb{N}$, $\mathcal{P}_i \notin \mathsf{Pend}^+$
$\Longrightarrow$ `Index` stops with output $\mathcal{T}$. ◊

Thus, by Claims 5.3, 5.4 and 5.5, for each $\mu \in \mathsf{Excl} \setminus \{\emptyset\}$,

if `Index` starts with input μ
then `Index` stops with output an index tree that analyzes μ.

Thus, `Index` computes a total function from $\mathsf{Excl} \setminus \{\emptyset\}$ to Tree such that for each $\mu \in \mathsf{Excl} \setminus \{\emptyset\}$,

`Index`(μ) analyzes μ. □

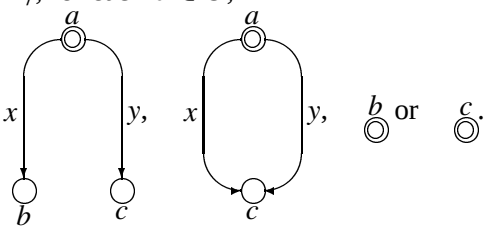
7. Example

In this section we give a simple illustration of our device and the classificatory systems it deduces. Suppose that

$\mathsf{Spec} = \{a, b, c\}$,
$\mathsf{Feat} = \{x, y\}$,
Appr is the total function from $\mathsf{Spec} \times \mathsf{Feat}$ to $\mathrm{pow}(\mathsf{Spec})$ such that
$\mathsf{Appr}\langle a, x\rangle = \{b, c\}$,
$\mathsf{Appr}\langle a, y\rangle = \{b, c\}$,
$\mathsf{Appr}\langle b, x\rangle = \emptyset$,
$\mathsf{Appr}\langle b, y\rangle = \emptyset$,
$\mathsf{Appr}\langle c, x\rangle = \emptyset$, and
$\mathsf{Appr}\langle c, y\rangle = \emptyset$, and
$\theta = \{[{:}x \sim b \rightarrow {:}y \sim c], [{:}x \sim c \rightarrow {:}x \approx {:}y]\}$.

Clearly,

$\langle \mathsf{Spec}, \mathsf{Feat}, \mathsf{Appr}\rangle$ is a computable signature,
$\theta \subseteq \mathsf{Desc}$, and
for each interpretation $I = \langle U, S, F\rangle$, for each $\upsilon \in U$,

$\upsilon \in \Theta_I(\theta)$ iff υ is of the form 

Suppose that `Class` starts with input θ. Since descriptions

$[[:x \sim b \rightarrow :y \sim c] \wedge [:x \sim c \rightarrow :x \approx :y]]$ and

$$\left[\begin{array}{l} [\neg:x \sim b \wedge \neg:x \sim c] \vee [\neg:x \sim b \wedge :x \approx :y] \\ \vee[:y \sim c \wedge \neg:x \sim c] \vee [:y \sim c \wedge :x \approx :y] \end{array}\right]$$

are semantically equivalent, we can also suppose that

$$\texttt{Matrix}(\theta) = \left\{\begin{array}{l} \{\neg:x \sim b, \neg:x \sim c\}, \{\neg:x \sim b, :x \approx :y\}, \\ \{:y \sim c, \neg:x \sim c\}, \{:y \sim c, :x \approx :y\} \end{array}\right\}.$$

Then `Class` stops with output $\{P, Q, R, S\}$, an exclusive matrix with clauses

$$P = \left\{ : \approx :, : \sim b \right\},$$

$$Q = \left\{ : \approx :, : \sim c \right\},$$

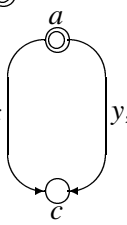
$$R = \left\{\begin{array}{l} : \approx :, :x \approx :x, :y \approx :y, :x \approx :y, :y \approx :x, \\ : \not\approx :x, :x \not\approx :, : \not\approx :y, :y \not\approx :, \\ : \sim a, :x \sim c, :y \sim c \end{array}\right\}, \text{ and}$$

$$S = \left\{\begin{array}{l} : \approx :, :x \approx :x, :y \approx :y, \\ : \not\approx :x, :x \not\approx :, : \not\approx :y, :y \not\approx :, :x \not\approx :y, :y \not\approx :x, \\ : \sim a, :x \sim b, :y \sim c \end{array}\right\}.$$

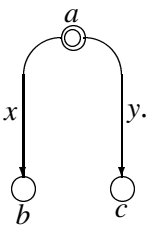
Clearly, for each interpretation $I = \langle U, S, F\rangle$, for each $\upsilon \in U$,

$\upsilon \in \Theta_I(P)$ iff υ is of the form b,

$\upsilon \in \Theta_I(Q)$ iff υ is of the form c,

$\upsilon \in \Theta_I(R)$ iff υ is of the form 

, and

$\upsilon \in \Theta_I(S)$ iff υ is of the form 

.

Thus, for each interpretation I, $\{P, Q, R, S\}$ constitutes a classification over $\Theta_I(\theta)$ with qualities (Q1) and (Q2).

Now suppose that `Index` starts with input $\{P, Q, R, S\}$. The registers of `Index` are initially

$$\texttt{I} = \emptyset, \texttt{O} = \emptyset, \texttt{L} = \emptyset \text{ and } \texttt{P} = \langle\langle \varepsilon, \{P, Q, R, S\}\rangle\rangle.$$

Since `P` is a nonempty sequence, `Index` begins the first performance of its `while` program by considering pending node $\langle \varepsilon, \{P, Q, R, S\}\rangle$. Since $\{P, Q, R, S\}$ is not

a singleton set, `Index` must calculate $\texttt{Select}(\{P, Q, R, S\})$. Thus, `Select` must calculate $\texttt{Weight}\langle\{P, Q, R, S\}, \kappa\rangle$ for each $\kappa \in \mathit{Query}(\{P, Q, R, S\})$, where

$$\mathit{Query}(\{P, Q, R, S\}) = \left\{ \begin{array}{l} \boxed{:}, \boxed{:x}, \boxed{:y}, \\ \boxed{:,:}, \boxed{:,:x}, \boxed{:,:y}, \\ \boxed{:x,:}, \boxed{:x,:x}, \boxed{:x,:y}, \\ \boxed{:y,:}, \boxed{:y,:x}, \boxed{:y,:y} \end{array} \right\}.$$

For example,

$\mathit{Residue}\langle\{P, Q, R, S\}, \boxed{:}, \boxed{a}\rangle = \{R, S\}$,
$\mathit{Residue}\langle\{P, Q, R, S\}, \boxed{:}, \boxed{b}\rangle = \{P\}$,
$\mathit{Residue}\langle\{P, Q, R, S\}, \boxed{:}, \boxed{c}\rangle = \{Q\}$,
$\mathit{Residue}\langle\{P, Q, R, S\}, \boxed{:}, \boxed{-}\rangle = \emptyset$,
$\mathit{Residue}\langle\{P, Q, R, S\}, \boxed{:}, \boxed{\approx}\rangle = \emptyset$,
$\mathit{Residue}\langle\{P, Q, R, S\}, \boxed{:}, \boxed{\not\approx}\rangle = \emptyset$,
$\mathit{Residue}\langle\{P, Q, R, S\}, \boxed{:}, \boxed{\pm}\rangle = \emptyset$,
$\mathit{Residue}\langle\{P, Q, R, S\}, \boxed{:}, \boxed{\mp}\rangle = \emptyset$, and
$\mathit{Residue}\langle\{P, Q, R, S\}, \boxed{:}, \boxed{=}\rangle = \emptyset$.

Thus

$\texttt{Weight}\langle\{P, Q, R, S\}, \boxed{:}\rangle = 2$.

In fact,

$\texttt{Weight}\langle\{P, Q, R, S\}, \boxed{:x}\rangle = 2$,
$\texttt{Weight}\langle\{P, Q, R, S\}, \boxed{:y}\rangle = 2$,
$\texttt{Weight}\langle\{P, Q, R, S\}, \boxed{:,:}\rangle = 4$,
$\texttt{Weight}\langle\{P, Q, R, S\}, \boxed{:,:x}\rangle = 2$,
$\texttt{Weight}\langle\{P, Q, R, S\}, \boxed{:,:y}\rangle = 2$,
$\texttt{Weight}\langle\{P, Q, R, S\}, \boxed{:x,:}\rangle = 2$,
$\texttt{Weight}\langle\{P, Q, R, S\}, \boxed{:x,:x}\rangle = 2$,
$\texttt{Weight}\langle\{P, Q, R, S\}, \boxed{:x,:y}\rangle = 2$,
$\texttt{Weight}\langle\{P, Q, R, S\}, \boxed{:y,:}\rangle = 2$,
$\texttt{Weight}\langle\{P, Q, R, S\}, \boxed{:y,:x}\rangle = 2$, and
$\texttt{Weight}\langle\{P, Q, R, S\}, \boxed{:y,:y}\rangle = 2$.

Thus, $\texttt{Select}(\{P, Q, R, S\})$ can be any of

$\boxed{:}$, $\boxed{:x}$, $\boxed{:y}$, $\boxed{:,:x}$, $\boxed{:,:y}$, $\boxed{:x,:}$, $\boxed{:x,:x}$, $\boxed{:x,:y}$, $\boxed{:y,:}$, $\boxed{:y,:x}$ and $\boxed{:y,:y}$.

Suppose that

$\texttt{Select}(\{P, Q, R, S\}) = \boxed{:}$.

Then

$$\texttt{Grow}\langle\{P, Q, R, S\}, \texttt{Select}(\{P, Q, R, S\})\rangle = \langle\langle\boxed{a}, \{R, S\}\rangle, \langle\boxed{b}, \{P\}\rangle, \langle\boxed{c}, \{Q\}\rangle\rangle,$$

and upon completing the first performance of its `while` program, the registers of `Index` are such that

$$\langle \mathtt{I} \cup \mathtt{O}, \mathtt{I}, \mathtt{O}, \mathtt{L}\rangle = \boxed{:}\,\bigcirc$$

and

$$\mathtt{P} = \langle\langle\boxed{a}, \{R, S\}\rangle, \langle\boxed{b}, \{P\}\rangle, \langle\boxed{c}, \{Q\}\rangle\rangle.$$

Since `P` is a nonempty sequence, `Index` begins the second performance of its `while` program by considering pending node $\langle\boxed{a}, \{R, S\}\rangle$. Since $\{R, S\}$ is not a singleton set, `Index` must calculate $\texttt{Select}(\{R, S\})$. Since

$\texttt{Weight}\langle\{R, S\}, \boxed{:}\rangle = 2,$
$\texttt{Weight}\langle\{R, S\}, \boxed{:x}\rangle = 1,$
$\texttt{Weight}\langle\{R, S\}, \boxed{:y}\rangle = 2,$
$\texttt{Weight}\langle\{R, S\}, \boxed{:, :}\rangle = 2,$
$\texttt{Weight}\langle\{R, S\}, \boxed{:, :x}\rangle = 2,$
$\texttt{Weight}\langle\{R, S\}, \boxed{:, :y}\rangle = 2,$
$\texttt{Weight}\langle\{R, S\}, \boxed{:x, :}\rangle = 2,$
$\texttt{Weight}\langle\{R, S\}, \boxed{:x, :x}\rangle = 2,$
$\texttt{Weight}\langle\{R, S\}, \boxed{:x, :y}\rangle = 1,$
$\texttt{Weight}\langle\{R, S\}, \boxed{:y, :}\rangle = 2,$
$\texttt{Weight}\langle\{R, S\}, \boxed{:y, :x}\rangle = 1,$ and
$\texttt{Weight}\langle\{R, S\}, \boxed{:y, :y}\rangle = 2,$

$\texttt{Select}(\{R, S\})$ can be any of

$\boxed{:x}$, $\boxed{:x, :y}$ and $\boxed{:y, :x}$.

Suppose that

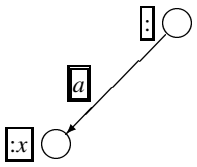
$$\texttt{Select}(\{R, S\}) = \boxed{:x}.$$

Then

$$\texttt{Grow}\langle\{R, S\}, \texttt{Select}(\{R, S\})\rangle = \langle\langle\boxed{b}, \{S\}\rangle, \langle\boxed{c}, \{R\}\rangle\rangle,$$

and upon completing the second performance of its `while` program, the registers of `Index` are such that

$\langle \mathtt{I} \cup \mathtt{O}, \mathtt{I}, \mathtt{O}, \mathtt{L}\rangle =$

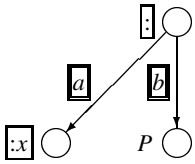


and

$$\mathtt{P} = \langle\langle\boxed{b}, \{P\}\rangle, \langle\boxed{c}, \{Q\}\rangle, \langle\boxed{a}\,\boxed{b}, \{S\}\rangle, \langle\boxed{a}\,\boxed{c}, \{R\}\rangle\rangle.$$

Since P is still a nonempty sequence, Index begins the third performance of its while program by considering pending node $\langle [b], \{P\}\rangle$. Since $\{P\}$ is a singleton set, Index need not calculate Select($\{P\}$). Instead, upon completing the third performance of its while program, the registers of Index are such that

$\langle \mathtt{I} \cup \mathtt{O}, \mathtt{I}, \mathtt{O}, \mathtt{L}\rangle =$

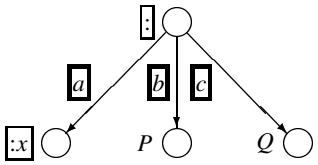


and

$\mathtt{P} = \langle\langle [c], \{Q\}\rangle,\ \langle [a]\,[b], \{S\}\rangle,\ \langle [a]\,[c], \{R\}\rangle\rangle.$

Index performs its while program three more times. Upon completing the fourth performance, the registers of Index are such that

$\langle \mathtt{I} \cup \mathtt{O}, \mathtt{I}, \mathtt{O}, \mathtt{L}\rangle =$

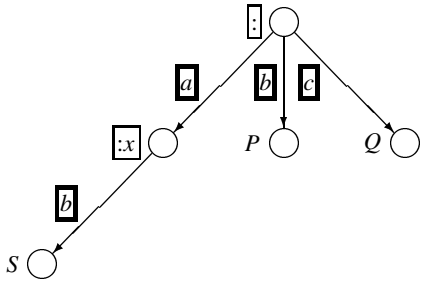


and

$\mathtt{P} = \langle\langle [a]\,[b], \{S\}\rangle,\ \langle [a]\,[c], \{R\}\rangle\rangle,$

upon completing the fifth, they are such that

$\langle \mathtt{I} \cup \mathtt{O}, \mathtt{I}, \mathtt{O}, \mathtt{L}\rangle =$

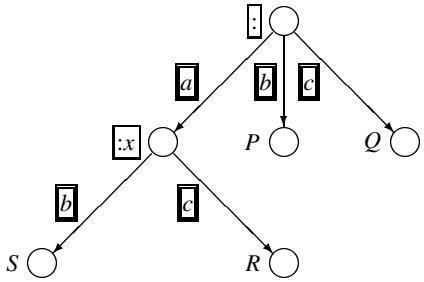


and

$\mathtt{P} = \langle\langle [a]\,[c], \{R\}\rangle\rangle,$

and upon completing the sixth, they are such that

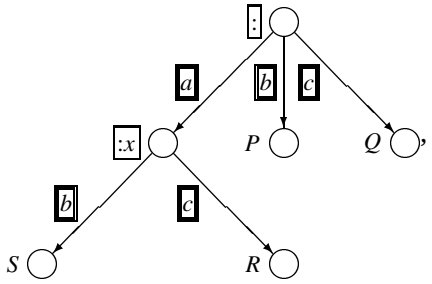
$\langle \mathtt{I} \cup \mathtt{O}, \mathtt{I}, \mathtt{O}, \mathtt{L}\rangle =$



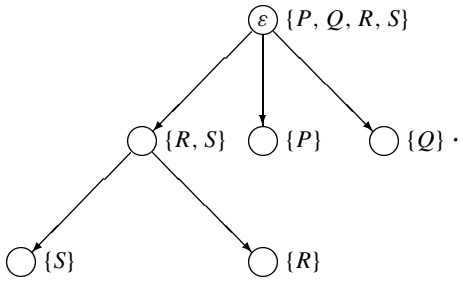
and

$\mathtt{P} = \langle\rangle.$

Since P is now the empty sequence, Index stops with output



an index tree $\mathcal{T}$ that analyzes $\{P, Q, R, S\}$ via the marking



Clearly, for each interpretation I, for each $\upsilon \in \Theta_I(\theta)$,

if Clause starts with input $\mathcal{T}$ and consults an oracle that discloses υ under I

then Clause stops with output $\theta' \in \{P, Q, R, S\}$ and $\upsilon \in \Theta_I(\theta')$.

Thus, for each interpretation I, Clause starting with input $\mathcal{T}$ constitutes an index from $\Theta_I(\theta)$ to $\{P, Q, R, S\}$ with quality (Q4). Indeed, Clause starting with input $\mathcal{T}$ can classify any object in the denotation of θ on the basis of accurate and effective responses to at most two queries: 'What is the species of the object?' and, if the response is '*a*', 'What is the species of the x feature of the object?'

8. Conclusions

In this paper we have presented an abstract prototype device that effectively deduces an accurate classificatory system from a finite linguistic theory. More specifically, we have defined two notions – exclusive matrices and index trees that analyze exclusive matrices – and three algorithms – Clause, Class and Index – and shown that

each exclusive matrix constitutes a classification over the denotation of the matrix with qualities (Q1) and (Q2),

Clause and each index tree that analyzes an exclusive matrix together constitute an index from the denotation of the matrix to the matrix with quality (Q4),

`Class` effectively deduces from each finite theory a semantically-equivalent exclusive matrix, and

`Index` effectively deduces from each nonempty exclusive matrix an index tree that analyzes the matrix.

Thus, `Class` effectively deduces from each finite theory an exclusive matrix that constitutes a classification over the denotation of the theory with qualities (Q1) and (Q2), and `Index` then effectively deduces from the matrix an index tree that, together with `Clause`, constitutes an index from the denotation of the theory to the matrix with quality (Q4). We conclude this paper by making five points about our device.

Firstly, our device is computationally expensive – `Class` and `Index` are time exponential or worse – and makes no attempt to deduce classificatory systems meeting qualities (Q3) and (Q5). Nonetheless, our device shows that the automatic deduction of classificatory systems meeting qualities (Q1), (Q2) and (Q4) is possible in principle. Only future research can show if the automatic deduction of classificatory systems meeting qualities (Q1)–(Q5) is possible in practice.

Secondly, our present device improves greatly upon our earlier device of Simov and King (1996). Certainly, our earlier device is simpler than our present device, since it has no index deducing component at all. However, this simplicity comes at the price of an index that repeatedly calculates `Select` and *Residue*. It would be simple to incorporate an algorithm into our earlier device that took over this computational burden. Starting with a nonempty exclusive matrix μ as input, the algorithm would construct a table that contains for each $\mu' \subseteq \mu$ with two or more clauses,

`Select`(μ') and for each $\rho \in$ Resp, *Residue*$\langle \mu', $ `Select`$(\mu'), \rho \rangle$.

Our earlier index would then find these values in the table rather than calculating each itself. However, constructing this table is inefficient, since it can easily be the case that no matter which object is being classified, the index would never need to calculate `Select` or *Residue* for many of the subsets of μ with two or more clauses. The index deducing component of our current device is more sophisticated, in that `Index` only ever calculates `Select` and *Residue* for subsets of μ that are actually needed to classify some object. The efficiency gains are obvious: in the example of the previous section, `Index` calculates `Select` and *Residue* for $\{P, Q, R, S\}$ and $\{R, S\}$, but for none of the nine other subsets of $\{P, Q, R, S\}$ with two or more members.

Thirdly, our index poses queries to an external oracle in order to elicit sufficient information to classify an object. However, certain inferential processes, such as parsing and generation, themselves pose queries, but to a classification, in order to further instantiate their current information about some objects. Though this is not strictly indexing, we envisage extending our index along the lines of Graf (1995) in order to yield a device that can efficiently support such inferencing.

Fourthly, given the complexity of natural language, a linguistic theory can easily overlook certain linguistic objects. If a classificatory system accurately deduced from a theory fails to classify an object then the object is identified as theoretically overlooked, thus indicating that a revision of the theory is necessary. Unfortunately, our device deduces from a theory a classificatory system that can occasionally classify an object the theory is not true of, much like the first classification of section 2 classifies 'Herz' as exhibiting pattern (D9). To overcome this shortcoming, and render our device more scientifically useful, we must modify our device so that if `Clause` classifies an object then `Clause` seeks confirmation that the clause it deduces is actually true of the object.

Lastly, our device deliberately delivers a classification with quality (Q1), but one can argue that the labels in a classification should name a hierarchy of subsets, not a partition. However, let quadruple $\langle L, \ll, \mu, \# \rangle$ be an *exclusive hierarchy* iff

$\langle L, \ll \rangle$ is a partial order,

$\mu \in \mathsf{Excl}$,

$\#$ is a total function from L to $\mathrm{pow}(\mu)$, and

for each $\lambda_1 \in L$, for each $\lambda_2 \in L$,

$\lambda_1 \ll \lambda_2$ iff $\#(\lambda_1) \subseteq \#(\lambda_2)$.

Intuitively, $\langle L, \ll \rangle$ is a hierarchical classification such that for each $\lambda \in L$,

λ is a label, and

for each interpretation I, λ indicates $M_I(\#(\lambda))$.

Meets, joins, etc. can be added to $\langle L, \ll \rangle$ as required, and our index easily modified to suit. Though not all hierarchies can be construed as exclusive hierarchies, we believe that the majority of important linguistic hierarchies can be so construed.

Acknowledgements

During periods of the writing of this paper, the Deutsche Forschungsgemeinschaft sponsored King with a postdoctoral fellowship at the Graduiertenkolleg Integriertes Linguistikstudium, Seminar für Sprachwissenschaft, Tübingen, and the Internationales Zentrum, Tübingen sponsored Simov with a visiting scholarship at the Seminar für Sprachwissenschaft, Tübingen. The authors wish to thank the Seminar für Sprachwissenschaft, the Deutsche Forschungsgemeinschaft and the Internationales Zentrum for their support. The authors also wish to thank Jennifer King for her help in constructing the example of German nominal declensions in Section 2, and two anonymous and patient referees for their valuable comments and advice on earlier drafts of this paper.

Notes

1. Some German nouns have additional letters (typically ‘e’ and ‘s’) inserted into some of their declensions, but the insertions are regularly determined by factors such as dialect, style and rhythm.

2. Indeed, all German dictionaries exploit this index, in that each entry for a noun gives only its nominative singular, genitive singular and nominative plural declensions.

3. However, SRL has no native mechanism for expressing HPSG relations such as list concatenation or set union. Though each serious HPSG relation the authors have so far encountered is expressible in SRL using a ‘junk-slot’ encoding (see Richter and Sailer (1995) for many examples), such an encoding is *ad hoc*, cumbersome and counterintuitive. Moreover, it is unclear which relations can be so encoded, and which not. To overcome this weakness, Frank Richter and Manfred Sailer have created RSRL, a relational extension of SRL with a perspicuous syntax and semantics of relations formulated specifically to meet the needs of HPSG. Unfortunately, RSRL is *too* expressive for our present needs. A device that effectively deduces a classification with quality (Q2) from a SRL or RSRL theory also effectively decides whether the theory is satisfiable. Whereas Kepser (1994) shows that the satisfiability of finite SRL theories is decidable for each computable SRL signature, a corollary of King et al. (1998) shows that the satisfiability of finite RSRL theories is undecidable for some computable, indeed finite, RSRL signature. Thus, for some finite RSRL signature, there can be no device that effectively deduces a classification with quality (Q2) from a finite RSRL theory. Nonetheless, RSRL is an extremely attractive language for formally expressing HPSG, and a number of important results have already been obtained for it. See Richter (in preparation) for preliminary details.

4. Deducing an index tree from the empty matrix is of no interest, since for each interpretation I, $M_I(\emptyset) = \emptyset$: there are simply no objects to classify.

References

Aldag, B., A proof theoretic investigation of prediction in HPSG. *Master’s Thesis, Seminar für Sprachwissenschaft*, Eberhard-Karls-Universität, Tübingen, Germany, 1997.

Götz, T., A normal form for typed feature structures. *Sonderforschungsbereich* 340 *Technical report* 40, IBM Deutschland GmbH, Heidelberg, Germany, 1994.

Graf, P., Term Indexing, No. 1053. In *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, Germany, 1995.

Kepser, S., A satisfiability algorithm for a typed feature logic. *Sonderforschungsbereich* 340 *Technical Report* 60, IBM Deutschland GmbH, Heidelberg, Germany, 1994.

King, P.J., A logical formalism for head-driven phrase structure grammar. *Ph.D. thesis*, Manchester University, Manchester, England, 1989.

King, P.J., An expanded logical formalism for head-driven phrase structure grammar. *Sonderforschungsbereich* 340 *Technical Report* 59, IBM Deutschland GmbH, Heidelberg, Germany, 1994.

King, P.J. and K.Iv. Simov, The automatic deduction of classificatory systems from linguistic theories (abridged), In C. Retore, editor, *The Proceedings of Logical Aspects of Computational Linguistics 1996, No. 1328 in Lecture Notes in Artificial Intelligence*, 248–273, Springer-Verlag, Berlin, Germany, 1997.

King, P.J., K.Iv. Simov, and B. Aldag, The complexity of modellability in finite and computable signatures of a constraint logic for head-driven phrase structure grammar. *The Journal of Logic, Language and Information*, 1998.

Meurers, D. and G. Minnen, A computational treatment of lexical rules in HPSG as covariation in lexical entries. *Computational Linguistics* **23**(4), 1997.

Pollard, C.J., Strong generative capacity in HPSG. In G. Webelhuth, J.-P. Koenig, and A. Kathol, editors, *Lexical and Constructional Aspects of Linguistic Explanation*, CSLI, Stanford, California, USA, 1998.

Pollard, C.J. and I.A. Sag, Information-Based Syntax and Semantics, No. 13 in CSLI *Lecture Notes*. CSLI, Stanford, California, USA, 1987.

Pollard, C.J. and I.A. Sag, Head-driven Phrase Structure Grammar. University of Chicago Press. Chicago, Illinois, USA, 1994.

Richter, F., A relational extension of speciate reentrant logic. *Ph.D. thesis, Seminar für Sprachwissenschaft*, Eberhard-Karls-Universität, Tübingen, Germany, in preparation.

Richter, F. and M. Sailer, Remarks on linearisation: Reflection on the treatment of LP-rules in HPSG in a typed feature logic. *Master's Thesis, Seminar für Sprachwissenschaft*, Eberhard-Karls-Universität, Tübingen, Germany, 1995.

Simov, K., G. Angelova, and E. Paskaleva, 'MORPHO-ASSISTANT: The proper treatment of morphological knowledge. In *The Proceedings of COLING'90*, Vol. 3. 453–457. Helsinki, Finland, 1990.

Simov, K.Iv. and P.J. King, Indexing of linguistic knowledge'. In *Logical Aspects of Computational Linguistics (Conference Handout)* 81–84. Villers-lès-Nancy, France, 1996.

Simov, K., E. Paskaleva, M. Damova, and M. Slavcheva, MORPHO-ASSISTANT: A knowledge-based system for Bulgarian morphology. In *The Proceedings of the Third Conference on Applied Natural Language Processing (Systems Demonstrations)*, 17–18. Trento, Italy, 1992.