Pablo Rivas-Robledo*

HYPER-REF: A General Model of Reference for First-Order Logic and First-Order Arithmetic

<https://doi.org/10.1515/krt-2021-0033>

Published online January 10, 2022

Abstract: In this article I present HYPER-REF, a model to determine the referent of any given expression in First-Order Logic (FOL). I also explain how this model can be used to determine the referent of a first-order theory such as First-Order Arithmetic (FOA). By reference or referent I mean the non-empty set of objects that the syntactical terms of a well-formed formula (wff) pick out given a particular interpretation of the language. To do so, I will first draw on previous work to make explicit the notion of reference and its hyperintensional features. Then I present HYPER-REF and offer a heuristic method for determining the reference of any formula. Then I discuss some of the benefits and most salient features of HYPER-REF, including some remarks on the nature of self-reference in formal languages.

Keywords: self-reference, diagonalization, Gödel sentences, hyperintensionality

1 Introduction

If Lakatos were to write *Proofs and Refutations* (Imre 1976) today, he would need to do something different. Indeed, much of mathematical practice today not only has shifted towards more formal ways of dealing with mathematical objects but has been the concern of an ever growing literature which tries to philosophically inform mathematical notions. How philosophy of science and ontology have done so is pretty well documented, given the focus that philosophy of mathematics had during the twentieth century. But there is still much that can be gained if we use resources from other areas of philosophy.

Here I am not trying to re-write *Proofs and Refutations*. Far from it, my goal here is to formally characterize a notion very important in mathematics and logic using insights from philosophy of language, bringing with me tools and intuitions from recent work on hyperintensionality (Leitgeb 2019; Sedlár 2019). This notion is

***Corresponding author: Pablo Rivas-Robledo**, Tarello Institute for Legal Philosophy (UniGe)/ Munich Center for Mathematical Philosophy (LMU), Via Balbi, 30, 16126 Genova, Italy, E-mail: pablo_rivas_robledo@hotmail.com

that of reference, and explaining how it behaves in formal languages to better understand mathematical and metamathematical results is what I take to be Lakatos' Undone Work.

There is a long standing position in analytic philosophy according to which the reference of a sentence is its truth-value. Indeed, a defense of this stance was presented by Frege himself in *Über Sinn und Bedeutung*. However, in contemporary analytic philosophy the word *reference* is also used to talk about the object(s) which the sentence is about (Leitgeb 2002). This paper is about the latter notion, which is also called referent to precisely distinguish it from reference as truth-value (Dummett 1973, pp. 409–412). HYPER-REF is a model that takes a formula in First Order Logic (FOL) and helps us determine the referent(s) of such formula with the aid of (1) a reference function r that maps formulae to its referents and (2) algorithms to decompose such formulae into its parts to facilitate the process. With little changes, we can also show how this model can be used to determine the referent of formulae of first-order theories, such as First Order Arithmetic (FOA).

Understanding reference as the referent of a sentence was implicit in much of the talk about self-reference in arithmetic¹ and formal semantics,² but the study of reference in formal languages has only recently begun. Indeed, even if one can trace back debates about self-reference in arithmetic to Gödel's incompleteness theorems and, more recently, debates on whether Yablo's Paradox is self-referent, it was not until the recent work of Lavinia Picollo that we were able to better understand the behavior of reference in formal languages (Picollo 2018, 2020a, 2020b).

Here I offer several criticisms to Picollo's account and present a new one with a model to prove its effectiveness, which is HYPER-REF. Here I define the reference of a sentence or formula as the non-empty set of objects that the syntactical terms of a well-formed formula (wff) pick out given a particular interpretation. *Which* objects it picks out depends on the interpretation given to the language. A more familiar way to put this is to take reference as the object(s) the sentence is about, and so HYPER-REF will not exactly point out which particular object of the domain is being talked about, but give a general procedure to determine the referent.

Reference in formal languages as it has been defined here displays some hyperintensional features which HYPER-REF necessarily needs to address (Leitgeb 2002; Picollo 2018, 2020a). That is, referential equivalence violates

¹ See Leitgeb (2002), Heck (2007), Halbach and Visser (2014a, 2014b), Milne (2007) and Detlefsen (2001).

² See, for instance, Dekker (2016).

logical equivalence: there can be a pair of formulae that although they have the same extension across possible worlds, they refer to distinct objects (Bjørn and Duzí 2015). Thus, when we say that the reference of two formulae is the same (they are referentially equivalent), we are establishing an equivalence between them finer than logical equivalence. For example, although $P(a) \vee \neg P(a)$ and $P(b) \vee \neg P(b)$ have the same truth value (in a model), they are not referentially equivalent (that is, the item(s) that the syntactical terms pick out is not necessarily the same): the former picks out the individual object denoted by a , while the latter picks out the individual object denoted by b . Here we can see hyperintensionality at work, as the truth-value of both formulae remains the same, but not their reference.

One can also observe the hyperintensional behavior of reference when one analyzes symmetric predicates. To understand this I will introduce a syntactical restriction according to which the subject of a sentence is always the first term of the formula: for example, a dyadic predicate $L(a, b)$ such that expresses the property ‘to love’ we understand that a is the subject and b the object. Thus, in the previous example we are talking about a , and we are saying of it that it loves b . The same applies for any n -ary predicate, the first term will always represent the subject of the formula, something which will be the process of establishing the referents of atomic formulae. This is known as the relational style of writing formulae, that is used a lot in formal semantics (Coppock and Champollion 2021) and also reflects the noun phrase–verb phrase (NP–VP) distinction used in phrase structure grammar (Chomsky 2002/1957).

So, where generative grammar breaks down a sentence such as ‘Romeo loves Juliet’ into its simpler and simpler syntactic categories using phrase structure rules (see Figure 1), I am interested only in taking the noun in the main noun phrase

Figure 1: Generative parsing tree for the sentence ‘Romeo loves Juliet’.

(the subject of the sentence) and putting it as the first term of the formula. This is of course not accidental, and the point is that if we can identify the subject of sentence in natural language this way, we can also put it as the first term of a formula.

Given this, we can notice that a formula $Q(a, b)$ that expresses the property ‘ a is greater than b ’ is not talking about both elements of the relation, but of the first. That is, it is stating that the first element is greater than the second one. The same applies to a formula $R(a, b, c)$ that represents the property ‘ a is greater than b but smaller than c ’: it is referring to a given the fact that it is stating that the object denoted by a is greater than b and smaller than c , and so on.³ Now, in those cases it is clear that neither $Q(b, a)$ is referentially equivalent to $Q(a, b)$ nor $R(b, a, c)$ is referentially equivalent to $R(a, b, c)$, and this is presumably because Q and R are asymmetric. Yet, I hold that this even happens with symmetric predicates.

Indeed, even if $P(a, b)$ is logically equivalent to $P(b, a)$ when $P(x, y)$ expresses, for instance ‘ x is the twin of y ’, the reference of both expressions is different, i.e. they are not referentially equivalent. In the former we are picking out (referring to) a and saying of it that b is its twin, while in the latter we are saying of b that a is its twin. Although both have the same truth conditions, their referents are different just from the fact that their construction is syntactically different, what amounts to a shift in reference. And so, even if $P(a, b) \models P(b, a)$, that is not enough to say that they have the same reference: precisely because the object picked out by the syntactical terms of the formulae are different. If one would want to say, for instance, that they are both twins (not necessarily) of one another, one could write $T(a) \wedge T(b)$; to construct a formula that refers to both elements and one could even write $P(a, b) \wedge P(b, a)$ as a formula that refers to both elements and where is it said of them that they are twins with one another.

As it will be shown, HYPER-REF takes into account these hyperintensional features of reference. In the next section, I will inductively define the reference of formulae in FOL and present a heuristic procedure based on this definition that allows us to take any given formula in FOL and to determine the object(s) its syntactical terms pick out. This heuristic procedure is based on previous applications of graph theory to the analysis of semantic paradoxes found in Davis (1979), Beringer and Schindler (2017) and Rossi (2019). The graph theoretical notation is based on Bondy and Ram Murty (2008).

³ This must not be confused with the process of determining the extension of relational predicates as such, which is the set of n -tuples between which the relation holds.

2 Presenting HYPER-REF

Based on the definition of reference provided in the previous section, I will now inductively define the reference for the different formulae of FOL. The language of FOL ($\mathcal{L}_{\text{FOL}}$) contains a set of variables $\{x_1, x_2, x_3 \dots\}$, a set of constants $\{a_1, a_2, a_3 \dots\}$, a set of logical connectives $\{\neg, \wedge, \vee, \rightarrow\}$, a set of quantifiers $\{\forall, \exists\}$, a set of n -ary function symbols $\{f_1, f_2, f_3 \dots\}$ and a set of n -place predicate symbols $\{P^1, P^2, P^3 \dots\}$ including a binary predicate for identity that can be written as $=$.

For simplicity's sake, I use x, y, z for arbitrary variables, g as a metavariable, a, b, c for arbitrary constants and P, Q, S for arbitrary predicates. I also use Greek lowercase for arbitrary formulae and uppercase for arbitrary sets of formulae.

Wffs are generated by the following BNF, where $\circ$ is any dyadic connective:

$$\begin{aligned} t &::= x_i | a_i | f(t_1, \dots, t_m) \\ \phi &::= P(t_1, \dots, t_n) | \neg \phi | (\psi \circ \gamma) | \forall g_1, \dots, g_n (\psi) | \exists g_1, \dots, g_n (\psi) \end{aligned}$$

We assume a non-empty domain D for the language and a function val such that

- for all $d \in D$ we add a constant a_d to the language such that $\text{val}(a_d) = d$.
- if P is predicate symbol of arity n , then $\text{val}(P) \subseteq D^n$.
- if f is a function symbol of arity n , then $\text{val}(f)$ is a function $D^n \mapsto D$.

An interpretation of the language is a pair $\mathfrak{S} = \langle D, \text{val} \rangle$. This ensures that every member of the domain has a constant that represents it in the language, which is very important for a theory of reference, as we will soon see. Truth is defined the usual way.

Based on the language presented, I will define the reference of the different formulae for $\mathcal{L}_{\text{FOL}}$. This will be done via function r that maps wff of FOL (For_{FOL}) to its referents, the set $\mathfrak{P}(D)^{\setminus \{\emptyset\}} \cup \{x_1, \dots, x_n\}$, such that $r : \text{For}_{\text{FOL}} \mapsto \mathfrak{P}(D)^{\setminus \{\emptyset\}} \cup \{x_1, \dots, x_n\}$. Please remember that g is a metavariable.

Definition 2.1. *Reference of $\mathcal{L}_{\text{FOL}}$ formulae:* For any wff ϕ we can define a function r with domain For_{FOL} and co-domain $\mathfrak{P}(D)^{\setminus \{\emptyset\}} \cup \{x_1, \dots, x_n\}$ that, given a wff, it returns the reference inductively by:

1. $r(P(t_1, \dots, t_n)) = \{\text{val}(t_i)\}$
2. $r(\neg P(t_1, \dots, t_n)) = r(P(t_1, \dots, t_n))$
3. $r(\psi \circ \gamma) = r(\psi) \cup r(\gamma)$
4. $r(\neg(\psi \circ \gamma)) = r(\psi \circ \gamma)$
5. $r(\forall g_1, \dots, g_n (\psi)) = D$

6. $r(\neg \exists g_1, \dots, g_n(\psi)) = r(\forall g_1, \dots, g_n(\psi))$
7. $r(\neg \forall g_1, \dots, g_n(\psi)) = \{x_i\}$
8. $r(\exists g_1, \dots, g_n(\psi)) = r(\neg \forall g_1, \dots, g_n(\psi))$

In the next section I will explain the rationale behind of all the cases of the function, but it is important that for now we can understand what I am defending here. The first thing to address is why reference on atomic formulae works as it does. According to the previous definition, $P(t_1, \dots, t_m)$ refers to the object denoted by the first term in the formula, t_1 . This only works if we take into account the syntactic restriction mentioned above, according to which the subject of the sentence is always written as the first of any given formula. Using the function r , we can see that the reference of $P(t_1, \dots, t_n)$ is $\{d\}$ such that t_1 is the name of d according to the interpretation of the language, and so I write $r(P(t_1, \dots, t_n)) = \{\text{val}(t_1)\}$. Something similar happens with negated atomic formulae: according to r , any atomic formulae $\neg P_n(t_1, \dots, t_m)$ refers to the same object that the non-negated formula refers to. This is because here we are still referring to the same element, but just saying that it lacks a property.

Complex formulae of the form, $(\psi \circ \gamma)$ refer to whatever ψ refers to and whatever γ refers to, regardless if they are quantified or non-quantified formula. In the next section I will defend that this is even true when for formulae of the form $\psi \rightarrow \gamma$. And something similar as what happened with negated atomic formulae happens when we have negated complex formulae. That is, the reference stays the same. Thus, the reference of $\forall xP(x) \wedge \forall yQ(y)$ or $\forall xP(x) \rightarrow \forall yQ(y)$ is the union of $r(\forall xP(x))$ and $r(\forall yQ(y))$. Via this function we can already see how the model reflects the other hyperintensional feature mentioned above: $r(P(a) \vee \neg P(a)) \equiv r(P(b) \vee \neg P(b))$, as the reference of the former is $r(P(a)) \cup r(\neg P(a))$, i.e. $\{\text{val}(a)\}$; while the reference of the latter is $r(P(b)) \cup r(\neg P(b))$, i.e. $\{\text{val}(b)\}$.

Regarding quantified formulae, there are two remarks that need to be made at this point: (1) universally quantified formulae refer to every object of the domain; (2) existentially quantified formulae refer to an element of the domain, *some* element of the domain; to tell which one is impossible if we analyze this type of formula in its abstract form $(\exists g_1, \dots, g_n(\psi))$, but we at least say it is referring to an element of the domain. More on this in the next section.

I will now show how from function r we can construct a method to decompose more complex formulae. To do so I use a particular type of graphs that I will call reference graphs or reference trees. Reference graphs are branching trees that are branched out from a root, which in this case will be the formula whose reference we wish to know and therefore they do not contain loops.

For example, say we wanted to know the reference of a formula such as $\forall x \exists y ((Px \wedge Qy) \wedge Pa)$. What I want is to take this formula and have a tree that has this formula as its root and whose branching is a decomposition of the formula based on some guiding principles provided by τ . Such a tree would look like this, with some vertices in between (Figure 2).

However, before presenting the algorithms it must be noted that formulae to be evaluated with τ need to be of a particular form to ease the task of determining their reference. For instance, a formula such as $\forall x \exists y ((Px \wedge Qy) \wedge Pa)$ is in prenex form and thus hinders our ability to see what the formula is about: as you can see, the formula Pa is stating something about a that is independent of whatever is being said about x , as a is not bound by the quantifier.

What we need then is to find a logically equivalent formula that helps us make this intuition explicit. That is, a formula that has the same truth value in every model but where every possible subformula has been *pushed out*. In the case of the previous formula, $\forall x \exists y ((Px \wedge Qy) \wedge Pa)$ would be such formula. And so, only when all the subformulae with no bound variables of a formula have been successfully *pushed out* and quantifiers are *pushed in* next to the subformulae whose variables they can bind, can we begin applying Definition 2.1.

I will call this syntactic restriction the *postnex* form of a formula.⁴ To achieve this I will introduce an algorithm very similar to Picollo's (2018) to show how to translate any formula into its *postnex* form. To that end, I introduce the notion of primes:

Definition 2.2. Prime: A formula ϕ of $\mathcal{L}_{\text{FOL}}$ is a prime if and only if it is an atomic formula, the negation of an atomic formula, a universally quantified expression whose nested quantifiers (if any) have the same domain, the negation of a

Figure 2: Sketch of a reference tree for a formula $\forall x \exists y ((Px \wedge Qy) \wedge Pa)$ of FOL.

⁴ What follows is heavily inspired by Picollo's own *postnex* disjunctive normal form and the more familiar *prenex* normal form (Picollo 2018).

universally quantified expression whose nested quantifiers (if any) have the same domain, an existentially quantified whose nested quantifiers (if any) have the same domain, or the negation of a existentially quantified expression whose nested quantifiers (if any) have the same domain.

With “whose nested quantifiers (if any) have the same domain” I mean that formulae with nested quantifiers can indeed be primes, but only if the nested quantifiers cannot be pushed inwardly any further. For example, while $\forall x \exists y (Py \wedge (Qx \vee Qy))$ is not a prime, $\exists y \forall z (Pzy \wedge \forall x (Qzx \vee Qzy))$ is.

Formulae such as $\neg(P(a) \wedge P(b))$ and $\forall x \exists y ((Px \wedge Qy) \wedge Pa)$ are not primes. Based on this, I will now introduce the definition of the postnex form of a formula. Herein I will use the formula $Cg(\phi)$, where C stands for either $\forall$ or $\exists$ and g is our metavariable.

Definition 2.3. *Postnex form of a formula:* A formula ϕ of $\mathcal{L}_{\text{FOL}}$ is in postnex form if and only if:

1. ϕ is a prime, or
2. every subformula is a disjunction of conjunctions of primes and
3. ϕ does not contain quantifiers that don't bind any variable and
4. every subformula of the form $Cg_1, \dots, g_n (\gamma_1 \vee \dots \vee \gamma_m)$ is such that g_i is free in γ_j for each $1 \leq i \leq n$ and $1 \leq j \leq m$.

To arrive at the postnex of a formula it is necessary for it to undergo a process called postnexation which I will specify now and that explains how to comply with the restrictions set by the definition. To obtain the postnex form of a formula, nested quantifiers sometimes need to be taken into account. To that end, I introduce the notion of depth.

Definition 2.4. *Depth:* Let dep be an assignment of numbers to formulae of $\mathcal{L}_{\text{FOL}}$ such that

1. $\text{dep}(\phi) = 0$ if ϕ does not contain quantifiers
2. $\text{dep}(Cg(\psi)) = 1$
3. $\text{dep}(Cg \neg \psi) = \text{dep}(Cg\psi)$
4. $\text{dep}(Cg(\psi \circ \gamma)) = \max\{\text{dep}(Cg\psi), \text{dep}(Cg\gamma)\}$
5. $\text{dep}(Cg_m Cg_n \psi) = \text{dep}(Cg_n \psi) + 1$

The dep of a formula ϕ is just the length of the longest chain of nested quantifiers occurring in ϕ . Now we proceed to define the process by which we can turn a formula into its postnex form.

Definition 2.5. *i-postnexation:* The *i-postnexation* $[\phi]^i$ of a formula ϕ such that $\phi \in \mathcal{L}_{\text{FOL}}$ is the result of applying the following transformations to each subformula $Cg\psi$ of ϕ of depth i . The process of postnexation of ϕ must be performed starting with formulae of depth $i = 0$ and ends after performing postnexing the formulae with the maximum depth, $i = \max\{\text{dep}(Cg\psi) : Cg\psi \text{ is a subformula of } \phi\}$.

1. Replace every subformula of the form $\psi \rightarrow \gamma$ with $\neg\psi \vee \gamma$ until they don't occur any longer, starting with the innermost.
2. Erase double negations.
3. Replace every subformula of the form $\neg(\psi \wedge \gamma)$ and $\neg(\psi \vee \gamma)$ for $(\neg\psi \vee \neg\gamma)$ and $(\neg\psi \wedge \neg\gamma)$ respectively until they don't occur any longer, starting with the innermost.
4. Erase double negations.
5. Replace every subformula of the form $Cg_1, \dots, g_n(\psi \circ \gamma)$ where γ is a prime containing no bound variables by $Cg_1, \dots, g_n$, for $Cg_1, \dots, g_n(\psi) \circ \gamma$ until they don't occur any longer, starting with the innermost.
6. Replace every subformula of the form $\psi \wedge (\gamma \vee \delta)$ and $(\gamma \vee \delta) \wedge \psi$ for $(\psi \wedge \gamma) \vee (\psi \wedge \delta)$ and $(\gamma \wedge \psi) \vee (\delta \wedge \psi)$ respectively until they don't occur any longer, starting with the innermost.
7. Replace every subformula of the form $Cg_1, \dots, g_n(\gamma_1 \vee \dots \vee \gamma_m)$ where g_1 isn't free in some γ_j and $1 \leq j \leq m$, with $Cx_2, \dots, x_n(\gamma_1 \circ \dots \circ \forall x_1 \gamma_j \circ \dots \circ \gamma_m)$ until such subformulae don't occur any longer.

As the process of postnexation only involves the application of semantical equivalences, it produces a logically equivalent formula that allows us to apply function r successfully.

For example, consider the formula

$$\forall z \exists y \forall x ((P(x, y) \rightarrow Qy) \wedge Pz) \wedge Pa$$

Its 0-postnexation, $[\phi]^0$, consists in applying transformations 2.5 to 2.5 in the definition above to its subformulae with depth 0 i.e. $((P(x, y) \rightarrow Qy) \wedge Pz) \wedge P(a)$. This results in:

$$\forall z \exists y \forall x (((\neg P(x, y) \wedge Pz) \vee (Qy \wedge Pz)) \wedge Pa)$$

Its 1-postnexation, $[\phi]^1$, does the same to its subformulae with depth 1, i.e. $\forall x ((\neg P(x, y) \wedge Pz) \vee (Qy \wedge Pz)) \wedge Pa$. This results in:

$$\forall z \exists y ((\forall x (\neg P(x, y) \wedge Pz) \vee (Qy \wedge Pz)) \wedge Pa)$$

Its 2-postnexation, $[\phi]^2$, consists in applying the same procedure to subformulae of depth 2, i.e. $\exists y((\forall x(\neg P(x, y) \wedge Pz) \vee (Qy \wedge Pz)) \wedge Pa)$. This results in a very slight change where the existential moves one place to the right:

$$\forall z(\exists y(\forall x(\neg P(x, y) \wedge Pz) \vee (Qy \wedge Pz)) \wedge Pa)$$

Its 3-postnexation, $[\phi]^3$, would be:

$$(\exists y(\forall z \forall x(\neg P(x, y) \wedge Pz) \vee (Qy \wedge (Pz)))) \wedge Pa$$

By this process we are identifying the main quantifier of the formula, (the determiner of the sentence) and also other independent formulae that are not in the domain of said quantifier. And now, with its 3-postnexation we can move to draw a reference tree that takes from the formula to referents of it.

As I stated before, the trees of graphs that I will be working here will not have cycles or loops and will be branching trees. That means they are a type of directed, rooted graphs whose starting point is a node from which different nodes will branch out called the root of the tree. From this root we will specify how to get to the reference of a formula thanks to a series of algorithms defined below. In what follows I borrow heavily from Rossi (2019).

So first I will show how to expand vertices that are labeled with formulae that are not complex and do not contain quantifiers (Definition 2.6). Then I will move to explain how to expand vertices that are labeled with complex formulae (Definition 2.7). Lastly I will explain how to expand vertices labeled with quantified formulae (Definition 2.8).

And so, for every $\phi \in \mathcal{L}_{\text{FOL}}$, I will define a labeled tree $\langle V_\phi, A_\phi, \mathcal{Q}_\phi \rangle$, where V_ϕ is the set of vertices $\{v_1, \dots, v_n\}$ of the tree, A_ϕ is the set of arcs or branches between vertices $\{(v_x, v_y), \dots, (v_z, v_w)\}$ and $\mathcal{Q}_\phi$ is the labeling function of the tree that assigns FOL formulae to the nodes of the tree such that $\mathcal{Q} : V \mapsto \text{For}_{\text{FOL}}$. These trees will have a starting vertex which is labeled with ϕ itself in postnex form that will be the root or the starting point of tree from which the tree is expanded. I will refer to this vertex as *root*.

To expand them I will use new trees that add nodes and arcs. That is, for example, given a tree $\langle V, A \rangle$ and a label function $\mathcal{Q}$, I will define a new tree, say $\langle V', A' \rangle$, and a new labeling function, say $\mathcal{Q}'$, which extends the labeled tree $\langle V, A, \mathcal{Q} \rangle_\phi$ such that this new function assigns either referents or expressions of FOL generated by the grammar to the new vertices according to Definitions 2.6–2.8. That this $\mathcal{Q}' : V' \mapsto \{(\mathfrak{P}(D)^{\setminus \{\emptyset\}} \cup \{x_1, \dots, x_n\}) \cup \text{Exp}_{\text{FOL}}\}$. So let us start with operations for vertices labeled with expressions that do not contain quantifiers or dyadic connectives (quantifier and dyadic free expressions). This next definition will show how to expand a tree that contain vertices labeled with this type of subformulae.

Definition 2.6. *Operations with quantifier and dyadic free formulae:* For every tree $\langle V, A \rangle$ labeled with For_{FOL} and a root vertex root, and for every function $\mathcal{Q}$, define the following set by induction: $v_i \in V^U$, $(v, v_i) \in A^U$ and $(v_i, \sigma) \in \mathcal{Q}^U$ if

1. $v \in V$, $\mathcal{Q}(v) = t_1$ and $\sigma = \{\text{val}(t_1)\}$, or
2. $v \in V$, $\mathcal{Q}(v) = P(t_1, \dots, t_n)$ and $\sigma = t_1$, or
3. $v \in V$, $\mathcal{Q}(v) = \neg P(t_1, \dots, t_n)$ and $\sigma = P(t_1, \dots, t_n)$, or
4. $v \in V$, $\mathcal{Q}(v) = \neg(\psi \circ \gamma)$ and $\sigma = \psi \circ \gamma$.

What this is saying is that, for example, you can take a tree and generate a new one with an extra vertex (v_i) if the original tree contains a vertex v such that is labeled as t_1 ($\mathcal{Q}(v) = t_1$). In this case the label (σ) of the new vertex (v_i) would be $\{\text{val}(t_1)\}$.

Now we move to formulae with dyadic connectives, which require a different process.

Definition 2.7. *Operations with formulae with dyadic connectives:* For every tree $\langle V, A \rangle$ labeled with For_{FOL} and a root vertex root, and for every function $\mathcal{Q}$, define the following set by induction: $v_i, v_j \in V^B$, $(v, v_i), (v, v_j) \in A^B$ and $(v_i, \sigma_0), (v_j, \sigma_1) \in \mathcal{Q}^B$ if

1. $v \in V$, $\mathcal{Q}(v) = (\psi \circ \gamma)$, and $\sigma_1 = \psi$ and $\sigma_2 = \gamma$

Here what happens is that when we get a tree in which a vertex's label is $(\psi \circ \gamma)$, then we get to define a new tree with two new branches. In each of those there is a new vertex $(v_i$ and v_j , respectively) whose labels are ψ and γ . Let us finish with nodes labeled with quantified formulae.

Definition 2.8. *Operations with quantified formulae:* For every tree $\langle V, A \rangle$ labeled with For_{FOL} and a root vertex root, and for every function $\mathcal{Q}$, define the following set by induction: $v_i \in V^Q$, $(v, v_i) \in A^Q$ and $(v_i, \sigma) \in \mathcal{Q}^Q$ if

1. $v \in V$, $\mathcal{Q}(v)$ is either $\forall g_1, \dots, g_n \psi$ or $\neg \exists g_1, \dots, g_n \psi$, and $\sigma = D$, or
2. $v \in V$, $\mathcal{Q}(v)$ is either $\neg \forall g_1, \dots, g_n \psi$ or $\exists g_1, \dots, g_n \psi$, and $\sigma = \{x\}$.

Here what happens is similar to Definition 2.6. A single new vertex is branched out containing a new label depending if the formula is universally or existentially quantified.

Lastly, in order to define all the new trees, we just need to put together Definitions 2.6, 2.7 and 2.8. Such that for every formula of $\mathcal{L}_{\text{FOL}}$, the labeled reference tree $\langle V_\phi, A_\phi, \mathcal{Q}_\phi \rangle$ is the result of applying Definitions 2.6, 2.7 and 2.8 to a graph consisting only in the root (labeled with ϕ) until a fixed point is reached.

Definition 2.9. For every $\phi \in \mathcal{L}_{\text{FOL}}$, the labeled reference tree $T = \langle V_\phi, A_\phi, \mathcal{L}_\phi \rangle$ generated from ϕ is the least fixed points of the following definition:

- At stage 0, put:

$$V_\phi^0 = \{\text{root}\} \quad | \quad A_\phi^0 = \emptyset \quad | \quad \mathcal{L}_\phi^0 = \{(\text{root}, \phi)\}$$

- For an arbitrary successor stage $m + 1$, put:

$$\begin{aligned} V_\phi^{m+1} &= (V_\phi^m)^U \cup (V_\phi^m)^B \cup (V_\phi^m)^Q \\ A_\phi^{m+1} &= (A_\phi^m)^U \cup (A_\phi^m)^B \cup (A_\phi^m)^Q \\ \mathcal{L}_\phi^{m+1} &= (\mathcal{L}_\phi^m)^U \cup (\mathcal{L}_\phi^m)^B \cup (\mathcal{L}_\phi^m)^Q \end{aligned}$$

We can put all this to use by establishing the reference of the formula ϕ such as $\forall z \exists y \forall x ((P(x, y) \rightarrow Qy) \wedge Pz) \wedge Pa$, whose postnexation we presented earlier. And so, we replace the postnex of ϕ with its postnexation: $(\exists y (\forall z \forall x (\neg P(x, y) \wedge Pz) \vee (Qy \wedge (Pz)))) \wedge Pa$.

We start by assigning $\langle V_\phi, A_\phi, \mathcal{L}_\phi \rangle$ its root, which, according to our labeling function, must be ϕ (Figure 3).

Given that ϕ is of the form $(\psi \circ \gamma)$, $r(\phi) = r(\psi) \cup r(\gamma)$. We can show this in the graph by applying Clause 1 of Definition 2.7 (Figure 4).

But we can still apply r further: we can apply it to the formula on the left and the formula on the right and represent it applying Clauses 2 of Definition 2.8 and 2 of Definition 2.6 respectively (Figure 5).

Finally, the new branch on the right can be expanded using Clause 1 of Definition 2.6 to obtain (Figure 6).

Given the function r , the reference of the formula would be the union of the sets $\{y\}$, and $\{\text{val}(a)\}$.

This concludes the presentation of HYPER-REF. Based on the model presented thus far, one could establish the reference of any formula in FOL. In what follows I

$$(\exists y (\forall z \forall x (\neg P(x, y) \wedge Pz) \vee (Qy \wedge (Pz)))) \wedge Pa$$

Figure 3: Root of $\langle V_\phi, A_\phi, \mathcal{L}_\phi \rangle$.

Figure 4: Branching the root of $\langle V_\phi, A_\phi, \mathcal{L}_\phi \rangle$.

Figure 5: Further branching of $\langle V_\phi, A_\phi, \mathcal{L}_\phi \rangle$.

Figure 6: Reference graph for ϕ .

discuss in further detail the construction of the r function and how HYPER-REF solves some problems that previous theories of reference in formal languages.

3 Using HYPER-REF to Determine the Reference of FOL Formulae

In the last section, I presented HYPER-REF, which consisted in a function r , which establishes the reference of formulae in their postnext form, and a procedure to construct reference graphs. In this section I will motivate and explain the cases of the function for FOL formulae. In the following section I do the same for formulae in FOA.

The most basic formulae are atomic formulae, $P(a)$. This formula is saying of a that it has the property P , for instance, ‘ a is Powerful’.⁵ Setting $P(a)$ as the root and applying rules 2 and 1 of definition 2.6 one obtains the following graph (Figure 7).

⁵ This should not be confused with the extension of predicates as such, which is the set of objects that have the property denoted by the predicate.

Figure 7: Reference graph for an atomic formula $P(a)$ of FOL, which is an instance of case 1 of τ .

The reference of a formula expressing a relational predicate can be determined similarly. Given any n -place predicates of the form $P(t_1, \dots, t_n)$, a wff expressing a n -ary relation between $(t_1, \dots, t_n)$ refers to the object denoted by the first term. This may look counter-intuitive, as pointed out in the previous section, but it is just the consequence of the syntactic constraint mentioned earlier. The graph for this formula is just Figure 7 with $P(t_1, \dots, t_n)$ as its root, t_1 as the next vertex and $\{val(t_1)\}$ as the last one.

We may continue with formulae with logical connectives. We start with negation. A formula $\neg P(a)$ expresses that a certain object denoted by a lacks the property P . It may be decomposed by first applying Clause 3 of Definition 2.6, and then Clauses 2 and 1 of Definition 2.6 (Figure 8).

Figure 8: Reference graph for a negated atomic formula $\neg P(a)$ of FOL, which is an instance of case 2 of τ .

Now it is time for the dyadic connectives $\wedge$, $\vee$, and $\rightarrow$. The connective $\leftrightarrow$ can be defined the usual way. It is the case that for each formula of the form $\phi \wedge \psi$, $\phi \vee \psi$, and $\phi \rightarrow \psi$, they refer to both the object that the formula ϕ refers to and the object that the formula ψ refers to. For instance, think of the formula $Pa \wedge Pb$ as stating that both a and b have the property P , and as such it refers to both of them. And so is the case with the other formulae with dyadic connectives. Indeed, formulae such as $Pa \vee Pb$ state that either a or b or both have the property P , to even establish which has the property we have to acknowledge which elements may have such property, that is, which elements are we talking about; a formula such that $Pa \rightarrow Pb$ states that if a has the property P , then b has said property too.

And so, any formula $(\psi \circ \gamma)$ (where $\circ$ represents either $\wedge$, $\vee$ or $\rightarrow$) refers to whatever object(s) ϕ refers and to whatever object(s) ψ refers to. And so, a graph for a formula $P(a) \wedge P(b)$ can be drawn using rules 1 of Definition 2.7 and the rules 2 and 1 of Definition 2.6 (Figure 9).

Now we move to quantified formulae, which allow us to refer to bound variables through the use of quantifiers. Establishing reference for this type of formulae is slightly more complicated than with atomic formulae, but not as cumbersome as Picollo (2018, 2020b) seems to think.

The simplest quantified formulae of FOL are $\forall xP(x)$ and $\exists xP(x)$. The former of these refers to everything and it states that every element has the property P , the latter refers to an element and it states that it has the property P . At this point we must again remember that we are analyzing reference and not truth, and such

Figure 9: Reference graph for complex formula $P(a) \wedge P(b)$ of FOL, which is an instance of case 3 of τ .

we are only interested in determining what a given quantified formula of FOL is talking about.

Thus, it is because we are referring with the universal quantifier that we can tell that we are referring to everything; and it is because we are referring with the existential quantifier that we can tell we are referring to some object, which one? again, that is not the point of referring with the existential quantifier, but to state that there is an object that satisfies some conditions expressed the formula. This will be relevant very soon for a discussion on Picollo's own view. Graphs for basic quantified formulae can be obtained using rules 1 and 2 of 2.8 respectively (Figures 10 and 11).

Here we can see that by introducing formulae with quantifiers we are introducing a new mechanism of reference altogether. Previously we had only been able to refer to single, determined objects via the constants that denoted them thanks to the interpretation of the language (m-reference). Through reference via quantifiers (q-reference) we can do even more: we can talk of all of the elements in the domain, some of them and even just one (think of definite descriptions), so here we are not referring as concisely as with m-reference, as with the universal quantifier we are not talking about one determinate object of the domain, but of all elements of the domain; with the existential quantifier we are referring to some element of the domain. Indeed, when we say $\exists xP(x)$ we are not referring to an

Figure 10: Reference graph for a simple quantified universal formula $\forall xP(x)$ of FOL, which is an instance of case 5 of τ .

Figure 11: Reference graph for a simple quantified existential formula $\exists xP(x)$ of FOL, which is an instance of case 8 of τ .

element in particular, we are just establishing the fact that there is one element of the domain that satisfies $P(x)$.⁶

Now, a possible objection to my treatment of reference via quantifiers would be to say that it leads to some undesirable results, namely that function r forces us to consider that $r(\exists x\phi) \neq r(\exists y\phi)$, since $r(\exists x\phi) = \{x\} \neq \{y\} = r(\exists y\phi)$, but that's implausible since we know that the reference of both formulae is the same, they refer to *some* element of the domain. On the contrary, I think this an interesting feature of HYPER-REF that precisely reflects how we use variables in FOL. When we write $\exists x\phi$, we want to talk about an element of the domain, and we use the variable x to represent this. When we write $\exists x\phi$ and also write $\exists y\phi$, in both cases we want to say something about an element of the domain, but maybe not the same element of the domain.⁷ In cases such as these, there is no syntactic restriction that could force us to believe that $x = y$ such that $\{x\} = \{y\}$, therefore there is no reason to believe that concluding $r(\exists x\phi) \neq r(\exists y\phi)$ is a disadvantage of the system. So $\{x\}$ and $\{y\}$ still represent some element of the domain, the fact that we use different variables is just there to represent that there is perhaps reason to think that they are not the same. This is why I think the set theoretic notation is important, x may just be a variable of the language as much as y is, by the same coin, $\{x\}$ is some element of the domain as much as $\{y\}$ is.⁸

Returning to the issue of nested quantifiers, we can also see that formulae with nested quantifiers whose quantifiers cannot be pushed inwardly q -refer to what the first or main quantifier refers. For instance, formulae such as $\forall x_1 \forall x_2\phi$ and $\forall x_1 \exists x_2\phi$ where quantifiers cannot be pushed further in still refer to every $x \in D$. This is because in each case the second quantifier is not used to determine what it is being talked about, but for predication. For instance, a formula of $\forall x \forall y (Px, y)$ where P states that for all xs (the subject of the formula) it is true that they stand in a relation P with regard to all the ys . P could be ' x greater than y ' and thus the formula would state that all the xs are greater than all the ys .

Now we add negations to simple quantified formulae. Even if $\forall xP(x)$ refers to every element in the domain, $\neg \forall xP(x)$ does not, because of the semantics of the

⁶ The original usage of the terms m -reference and q -reference are found in Picollo (2018).

⁷ In a very Finean spirit, we could hold that $(\exists x\phi)$ and $(\exists y\phi)$ are precisely not synonymous when they jointly occur in a formula (say, $\exists x \exists y\phi$) (Fine 2007, ch. 1); yet, they may be synonymous in other instances, when they do not appear jointly, but the point here is that it need not be.

⁸ Though similar, I think this is not strictly related to other area of Kit Fine's work regarding models for arbitrary objects (Fine 1985). Indeed, there Fine was interested in establishing a theory of arbitrary objects, but the reference of $\exists g\phi$ is not an arbitrary object, precisely because its reference is part of the domain, whereas arbitrary objects are not.

universal quantifier: negating a universal quantifier results in referring not to all elements of the domain, but only some elements (at least one) of the domain, which was exactly to what $\exists xP(x)$ referred. As such, they refer to the same thing(s), some $x \in D$. As expected, something very similar happens with expressions such as $\forall xP(x)$ and $\neg \exists xP(x)$, as they refer to every element in the domain.

However, quantified formulae not always come so simple. Universal quantified formulae often come in the form $\forall x(Px \rightarrow Qx)$ and existential formulae often come in the form $\exists x(Px \wedge Qx)$. But of course that is not the end of the story, as it is possible to construct quantified formulae with any connective (not just ones just said), nested quantifiers and every conceivable syntactic combination.

This is where I think that Picollo over complicates things with q-reference. Picollo's starting point is simple quantified formulae such as $\forall xP(x)$ and $\exists xP(x)$, for which she says respectively that refer to every element of the domain and some element(s) of the domain. But then she jumps to formulae such as $\forall x(Px \rightarrow Qx)$, and says that it refers to every element that has the property Q provided that said element has the property P (Picollo 2018, pp. 584–585).⁹

Both formulae $\forall xP(x)$ and $\forall x(Px \rightarrow Qx)$ have the same determiner, i.e. the same quantifier, and so I don't think there is really a shift in reference. What determines the reference of a quantified formula is the quantifiers, not the predicate. And if they have the same quantifier, I think this shift complicates things. The predicate of a quantified formula has no bearing in what the formula refers to in the sense that changing the predicate does not change what we are referring to. Changing the predicate changes what we are saying about the referent, sure, but it does not change what we are talking about. And so, Picollo is introducing a novel way of reference which she is not explaining, and that's reference through predicates: if $\forall xP(x)$ and $\forall x(Px \rightarrow Qx)$ refer to different objects there is a shift in reference for which she is not accounting.

Here is where things get a tad harder for Picollo and others who agree with her (Beringer and Schindler 2017, p. 448; Leitgeb 2005, pp. 180–181), for these amounts to saying that every semantically distinct formula of FOL refers to different objects. And so she is forced to devise algorithms to transform any formula into its *postnex disjunctive normal form* and only then can we establish what a sentence is referring to. That is, to take a formula ϕ and rewriting it such that (1) it does not contain $\rightarrow, \leftrightarrow, \exists$ nor superfluous quantifiers, (2) every subformula of ϕ is a disjunction of conjunctions of primes (an atomic formula, its

⁹ Picollo in part attributes this stance to Leitgeb (2002), but I cannot find any specific mention of this by him here but rather in Leitgeb (2002, pp. 180–181).

negation, a universal formula, or its negation), and (3) in every subformula of the form $\forall g_1 \dots \forall g_n (\phi_1 \vee \dots \vee \phi_n)$, with g_i free in ϕ_i .¹⁰

And so, for instance, Picollo states that even if formulae such as $\forall x (\phi \rightarrow \psi)$ and $\forall x (\neg\psi \rightarrow \neg\phi)$ (which I will call Π) refer to the same objects, and even if formulae such as $\forall x \phi$ and $\forall x \neg \neg\phi$ (which I will call Σ) refer to the same objects, formulae Π and Σ do not refer to the same objects (Picollo 2020a, p. 428). But if we take into account what has been said so far, one can see that both pairs of sentences refer to the same objects: all the objects in the domain. Again, here the quantifier (what actually counts when establishing the referent of a formula) says the same, so I see no reason as to why say that Π and Σ have different referents. What makes them different is what are they attributing or predicating of such objects, but they still refer to every object in the domain because that is the role that the universal quantifier fulfills in regards to reference.

What Picollo is ultimately doing is a very robust and precise theory of predication. One that allows us to determine when two expressions are predicating the same of the same objects. But this is different to what I have sought to explain, and so a theory of predication needs to be deferred until we have a theory of reference.

4 Using HYPER-REF to Determine the Reference of FOA Formulae

In the previous section I used HYPER-REF to determine the reference of a number of formulae in FOL to explain the function r . In this section I will do the same for the FOA formulae, a language which is an instance of FOL. My interest here is to explain how I can apply the theory to other language of first-order logic that can afford self-reference. To that end I will explain the particularities r for FOA formulae, and then review the strong and weak diagonal lemmas to understand the extend to which they are self-referential.

The language of FOA ($\mathcal{L}_{\text{FOA}}$) is an instance of the language of FOL with a different interpretation. For that, we need to introduce 0 as the only individual constant, S as the monadic function symbol for the successor operation, $+$ and $\times$ as dyadic function symbols and the axioms of Robinson Arithmetic ($\mathcal{Q}$), the weakest

¹⁰ Picollo expanded the concept of postnex disjunctive normal form to the concept alethic disjunctive normal form to account for reference of languages with a truth predicate. See Picollo (2020a, Definition 2).

theory in which self-reference is possible. The new interpretation is $\mathbb{N} = \langle \omega, \text{val} \rangle$, comprised of a domain ω of positive integers and now the function val adjusted to the new domain.

To do that, I will introduce gödelization. Gödelization of $\mathcal{L}_{\text{FOA}}$ is a function that assigns to each symbol of our language (and therefore every string of symbols) a unique natural number (which belongs to the domain). Thus, if θ is a string of symbols, $\# \theta$ will be its code or Gödel number. To do so, we need to understand that for every individual term consisting n occurrences of the symbol S followed by 0 is called the numeral of n . And so, if $\# \theta$ is the code or Gödel number of θ , $\ulcorner \theta \urcorner$ is the numeral of its code.

Thus,

- for all $n \in \omega$ we add a constant $\ulcorner n \urcorner$ to the language such that $\text{val}(\ulcorner n \urcorner) = \# n$
- if P is predicate symbol of arity n , then $\text{val}(P) \subseteq \omega^n$
- if f is a function symbol of arity n , then $\text{val}(f)$ is a function $\omega^n \mapsto \omega$

As $\ulcorner \psi \urcorner$ is just a term of FOA (an instance of FOL), any formula that uses numerals to refer (such as $P(\ulcorner \psi \urcorner)$) is just a special instance of case 2.1 of Definition 2.1: that is $(P \ulcorner \psi \urcorner)$ refers to $\# \psi$. Consequently, something very similar applies with Clause 2.6 of Definition 2.6, where the new node contains $\{\text{val}(\ulcorner \psi \urcorner)\}$, that is $\{\# \psi\}$.

This is also a departure from Picollo, for whom $P \ulcorner \psi \urcorner$ m-refers to ψ and not to $\# \psi$ (Picollo 2018, §4; 2020a, sec. 3.2). But I think this is not correct. For once, she is supposing that m-reference has instances of transitivity which I will deny in a minute and also simply because ψ is not an object of the domain, $\# \psi$ is. Although her model has its merits, I think it is problematic to say that you can refer to objects outside the domain.

As our domain is ω , the reference of 0 is the empty set. Furthermore, the reference of a function is the range of values it can have given its domain, since our domain is ω , then the reference of any instance of the functions just introduced (S , $+$ and $\times$) is the value the function has: the reference of $S0$ is the value of the function, which is 1; the reference of the function $5 + 4$ (which is short for $+(5, 4)$) is 9. This means that reference through functions is just another case of m-reference.

Now, even if $S0$ and $SS0$ appear in $SSS0$, neither of them should be taken as subterms of $SSS0$ in way such that $SSS0$ refers to whatever $S0$ and $SS0$ refer to. This is so because with $SSS0$ we are solely interested in determining the successor of $SS0$, and so claiming that the reference of $SSS0$ also is 1 and 2 is wrong, because the successor of $SS0$ is neither (they are not identical to 3). And so only because $S0$ and

SS0 occur in SSS0 that does not mean we are referring to them because we are not taking them as values of the function.¹¹

We now move to self-reference. For this we need to first detail the conditions for a successful gödelization as it cannot be arbitrary on pain of triviality.¹² As such, the coding must be effective, ε -adequate and strongly monotonic. A numbering is effective if there is an algorithm through which we can establish which expression σ a number n codifies, and vice versa (Picollo 2018, p. 574). A numbering is ε -adequate if it represents a large portion of syntactic relations and operations by elementary relations and operations on ω (Grabmayr and Visser 2020, p. 3). A numbering is monotonic if for a sub-expression σ of σ' , it assigns codes such that $\#\sigma \leq \#\sigma'$, and its strong if, in addition, the code of the Gödel numeral of an expression is larger than the code of the expression itself (Grabmayr and Visser 2020, p. 3).¹³ This is needed as we need the codings to preserve m-reference (Grabmayr and Visser 2020, sec. 6.2). These constraints may seem exaggerated, but they allow for to introduce self-reference in the system with relative ease.¹⁴

All this technicality aside, let us analyze some features of reference via numerals. First of all, we must observe how reference works here. Given an expression ϕ of $\mathcal{L}_{\text{FOA}}$, our coding must assign it a code which is a number $\#\phi$ that belongs to the domain whose numeral is $\ulcorner \phi \urcorner$, which are used in the wffs.

Given that our gödelization is effective and strongly monotonic, we can take any numeral $\ulcorner \phi \urcorner$ and from there establish which number it refers to, which is the expression ϕ . After all, with gödelization we are just assigning names to the expressions, such that the name, refers to the expressions of the language itself. As said earlier, the case of $\ulcorner \urcorner$ for this expression is just a special instance of case 2.1 where $\ulcorner (P(\ulcorner \phi \urcorner)) \urcorner = \phi$. The rules for graphs containing Gödel numbers are just

11 A very different issue is the fact that to establish the value of SSS0 we need to establish the value of SS0 and S0, which is the whole point of induction and an epistemological claim independent from the fact of the reference of SSS0 is neither SS0 nor S0.

12 See Heck (2007, sec. 2.1) and Halbach and Visser (2014a) for some difficulties of more traditional numberings. Examples of non-standard Gödel numberings for which self-reference fails are introduced in Grabmayr (2021), sec. 2.3).

13 This concept is actually attributed to Volker Halbach, who formulated to either Albert Visser and/or Balhasar Grabmayr in a personal communication. Mortals as I who have no access to such correspondence are obliged to quote it as it has been done here.

14 Indeed. As has been argued by Grabmayr (2021), these constraints assure us relative invariance in respect to main results that are provable in first-order Robinson arithmetic. This is a huge source of intensionality in the system (Halbach and Visser 2014a, §2.1), as with two different codes we could referring to the same expression, or with the same code be referring to different expressions, but for the present purposes it is only relevant that one is chosen.

special instances of rule 2.6 of Definition 2.6, such that from $P(\ulcorner \phi \urcorner)$ we must branch to $\{\ulcorner \phi \urcorner\}$ and then to $\{\#\phi\}$.

According to an inconvenience noted by Picollo (2018, pp. 582–583) and later corrected and presumably solved in Picollo (2020a, pp. 424–425) an expression with code $\ulcorner \phi \urcorner$ m-refers (besides to the sentence coded) to every well-formed expression whose code is smaller. However if we see these codes as names of expressions (as it will be done here) we can see that there is no such problem.

M-reference is not a transitive relation: if ϕ m-refers to ψ and ψ m-refers to a , ϕ does not m-refer to a , even if we are using gödelization. Indeed, when one m-refers to an object of the domain, one does so by using a name: although an expression such as $P(\ulcorner Q(\ulcorner 0 = 0 \urcorner) \urcorner)$ refers to $Q(\ulcorner 0 = 0 \urcorner)$ we do not want to say $P(\ulcorner Q(\ulcorner 0 = 0 \urcorner) \urcorner)$ refers to $0 = 0$ (the reference of $Q(\ulcorner 0 = 0 \urcorner)$). If this was the case, then this accounts to saying that $Q(\ulcorner 0 = 0 \urcorner)$ is identical to $0 = 0$, as they would both be the reference of $P(\ulcorner Q(\ulcorner 0 = 0 \urcorner) \urcorner)$. However, they are not, because they are different formulae: the former states that a formula with code $\ulcorner 0 = 0 \urcorner$ has the property Q , the later states that 0 is equal to 0 .

Another example may be in order. Consider the formula $P(\ulcorner \phi \urcorner)$, this formula refers to whatever formula $\ulcorner \phi \urcorner$ stands for, but not for whatever ϕ refers to, as the reference of ϕ can be perfectly different to ϕ . A similar phenomenon can be observed in natural language: if I say ‘the grass is green’ and right after say ‘the sentence I just said has 4 words’, with my last utterance I am not referring to grass nor stating that the grass satisfies the property of having four words. With the latter, I am referring to the expression ‘the grass is green’ but not to grass itself.

Returning to self-reference, there are two standard ways to enrich the language so it can afford it. One way will lead us to diagonalization (strong and weak) and other related functions such as normalization (Smullyan 1957; 1994, chaps. 1 and 5), the other leads us to self-referential numberings, which were hinted in Kripke (1975, footnote 6) and Feferman (1984, p. 80) and developed later on.¹⁵

Here I will only focus on strong and weak diagonalization. Self-referential numberings have been recently frowned upon for their lack of monotonicity, and so are subject to a number of criticisms, such as the fact that m-self-reference is not attainable in the language (Heck 2007). This, however was recently challenged by Grabmayr and Visser (2020), who showed that there are self-referential numberings for expressively weak languages.¹⁶

¹⁵ Kripke actually constructed such numberings in his 1982 seminar on truth. Allen Hazen allowed me to see his notes on the seminar and for that I am grateful.

¹⁶ See also Kripke (2021).

Self-reference through strong diagonalization is achieved by enriching the language with function symbols for primitive recursive functions, such that the resulting language $\mathcal{L}_{\text{FOA}}^+$ contains a term $\dot{d}$ which represents a diagonal function. Given a formula ϕ with x free, the diagonalization of ϕ returns a formula where all instances of x have been replaced in ϕ for $\ulcorner \phi \urcorner$. Let $\vec{y}$ abbreviate $y_1, \dots, y_n$, a possibly empty sequence of individual variables.

Lemma 4.1. *Strong diagonalization: For every formula $P(x, \vec{y})$ where x is different to each $y_1, \dots, y_n$, there is a term t such that $\mathcal{Q} \vdash t = \ulcorner P(t, \vec{y}) \urcorner$.*

Proof. Let $t = \dot{d}(\ulcorner P(\dot{d}(x), \vec{y}) \urcorner)$. Then $\dot{d}(\ulcorner P(\dot{d}(x), \vec{y}) \urcorner) =_{\mathcal{Q}} \ulcorner P(\dot{d}(\ulcorner P(\dot{d}(x), \vec{y}) \urcorner), \vec{y}) \urcorner$. Indeed, $\ulcorner P(\dot{d}(\ulcorner P(\dot{d}(x), \vec{y}) \urcorner), \vec{y}) \urcorner$ is the diagonalization of $\dot{d}(\ulcorner P(\dot{d}(x), \vec{y}) \urcorner)$.¹⁷ $\square$

From the perspective of HYPER-REF, we can see that strong diagonalization does not exactly offer us self-reference upfront, at least not as the indexical ‘I’ does in natural language. Indeed, $r(P(x, \vec{y}))$ is $\{\text{val}(x)\}$, but $\{\text{val}(x)\}$ is equivalent neither to x nor to $P(x, \vec{y})$, but rather to $\{\#x\}$, an object of the domain (a number). Consequently $r(P(t, \vec{y}))$ such that $t = \ulcorner P(t, \vec{y}) \urcorner$ is not $P(t, \vec{y})$, but rather $\{\#P(t, \vec{y})\}$, a number that is an object of the domain, but not the formula itself, which is not part of the domain. Thus, t is not equivalent to the sentence that says of itself that has the property P , but rather equivalent to that sentence’s name. It is not self-referential because it literally refers to itself ($r(t) \neq t$), but rather because we can prove (because we know) that the sentence whose name t is identical refers to t ($r(P(t, \vec{y})) = t$). We can create a graph to show this better (Figure 12).

Figure 12: Reference graph for an atomic formula $P(t, y)$, which is not identical to t .

¹⁷ This result is due to Jeroslow (1973). This particular way of presenting the proof is due to Picollo (2018, p. 582).

And so, the graph never takes us again to $P(t, \vec{y})$, but to $val(t)$ (or $val(\ulcorner P(t, \vec{y}) \urcorner)$ in this case), which is identical to $\#P(t, \vec{y})$, a name, and not the formula itself.

On the other hand, the weak diagonal lemma states that for every predicate P there is a formula ϕ materially equivalent to $P(\ulcorner \phi \urcorner)$ (that is, $\mathcal{Q} \vdash \phi \leftrightarrow P(\ulcorner \phi \urcorner)$). The sentences that fulfill these requirements are either of the form $\forall x \forall y (x = \ulcorner \forall y (\text{Diag}(x, y) \rightarrow P(y)) \urcorner \rightarrow (\text{Diag}(x, y) \rightarrow P(y)))$ or of the form $\exists x \exists y (x = \ulcorner \exists y (\text{Diag}(x, y) \wedge P(y)) \urcorner \wedge (\text{Diag}(x, y) \wedge P(y)))$ depending of the method you are choosing to define diagonalization (see Boolos, Burgess, and Jeffrey 1974; Milne 2007, §17.2). Notice however, that d represents diagonalization and Diag strongly represents it.

In both cases however, self-reference is not obtained upfront, because in the former case the sentence is referring to every x and in the latter is referring to some x . What really enables us to establish that these sentences refer to themselves is actually what is being said of them. That is, that there is a formula y that is the diagonalization of a formula x which has the property P and that we know that ϕ is that formula.

Graphs for formulae obtained via the weak diagonalization lemma are pretty straightforward, since its postnex is just $\exists x (x = \ulcorner \exists y (\text{Diag}(x, y) \wedge P(y)) \urcorner \wedge \exists y (\text{Diag}(x, y) \wedge P(y)))$. Its graph is obtained labeling the root to this formula and applying Clause 2 of Definition 2.8 (Figure 13).

The simplicity of the graph is due to the fact that there are no independent subformulae that can be pushed out of the main formula, and so the formula is only referring via the first existential quantifier. This helps to show in a heuristic way what I just stated regarding self-reference through the weak diagonal lemma: the claim that formulae like this one are self-referential is a claim that rests on the epistemological fact that we know what is being ascribed to the referent of the formula that we can know that the formula is actually referring to itself.

There has been an enduring claim in the literature according to which diagonalization (either strong or weak) is equivalent to other mechanisms of self-

Figure 13: Reference graph for the existential version of the diagonal lemma.

reference in natural languages, such as indexicals (Kleene 1986, p. 134; Gaifman 2006, p. 713; Raatikainen 2018, sec. 4.3; Smullyan 1992, p. 11) and this appears to be the received view (Milne 2007, Appendix); others claim that they are equivalent to demonstratives (Priest 1995); while some others think of them as definite descriptions (Beringer and Schindler 2017; Halbach 2016, p. 162; Heck 2007, p. 5). However, in the light of HYPER-REF these claims look ill-founded, as this sort of epistemic self-reference is not evidently mirrored by any of these mechanisms for self-reference. This is without a doubt an interesting and open research topic.

5 Conclusion

The remarks made in this paper can at least help us to understand three things: the first, is that there are formulae that are referentially identical but semantically distinct, such as ϕ and $\neg\phi$. The second is that under HYPER-REF there is also room for referentially distinct yet semantically equivalent formulae, such as $P(a, b)$ and $P(b, a)$, whenever P is symmetric. Lastly, although the possibility of self-reference in formal languages has been acknowledged, claims of self-reference through diagonalization (strong and weak) rest upon epistemological claims.

The study of reference and self-reference in formal languages is still nascent. Important philosophical and logical questions are open whose answer this article is admittedly presupposing. For me, the most pressing of these is the very possibility of reference in formal languages: reference is a term that was introduced to discuss certain aspects of natural language, a language that does not depend on the interpretation assigned to it. The fact that in formal languages we need not only the language itself, but also an interpretation of it in terms of an ordered pair $\langle D, \text{val} \rangle$, makes me question the role that the interpretation of the language has in making reference in formal languages possible.

Here I assumed that an interpretation of the language is sufficient to establish relations of reference between formulae and objects of the domain, but this is hardly the same relation that occurs with sentences of natural language and objects of the world. If reference in formal languages is possible after all, then there is much to say about the differences between reference in formal and natural languages, including how different or similar are mechanisms of self-reference in both languages. As was hinted in the previous section, strong diagonalization cannot be equated with self-reference via indexicals and as of now I see no reason to think that weak diagonalization does.

Acknowledgments: Without a doubt, I am indebted beyond words to Lorenzo Rossi, who acted as my mentor during the Lakatos' Undone Work project. I would

like to also thank the other members of the program (both mentors and mentees) for their valuable comments, especially Jonas Raab. My gratitude is also extended to the organizers of the event -Sophie Nagler, Hannah Pillin and Deniz Sarikaya- and the two anonymous reviewers from *Kriterion*.

References

- Beringer, T., and T. Schindler. 2017. "A Graph-Theoretic Analysis of the Semantic Paradoxes." *Bulletin of Symbolic Logic* 23 (4): 442–92.
- Bondy, A., and M. Ram Murty. 2008. *Graph Theory*. Germany: Springer.
- Boolos, G. S., J. P. Burgess, and R. C. Jeffrey. 1974/2007. *Computability and Logic*, 5th ed. New York: Cambridge University Press. (originally published in 1974).
- Bjørn, J., and M. Duzí. 2015. "Introduction." *Synthese* 192 (3): 525–34.
- Chomsky, N. 2002/1957. *Syntactic Structures*, 2nd ed. Mouton de Gruyter (originally published in 1957).
- Coppock, E., and L. Champollion. 2021. In preparation. *Invitation to Formal Semantics (Formerly Known as Semantics Boot Camp)* Self-published. Available at: <https://eecoppock.info/bootcamp/semantics-boot-camp.pdf>.
- Kleene, S. C. 1986. "Introductory Note to 1930b, 1931 and 1932b." In *Kurt Gödel – Collected Works. Volume I. Publications 1929–1936*, edited by S. Feferman, et al., 126–41. New York: Oxford University Press & Clarendon Press.
- Davis, L. 1979. "An Alternate Formulation of Kripke's Theory of Truth." *Journal of Philosophical Logic* 8 (1): 289–96.
- Dekker, P., and T. E. Zimmermann. 2016. "Reference." In *The Cambridge Handbook of Formal Semantics*, edited by M. Aloni, and P. Dekker. Cambridge Handbooks in Language and Linguistics, 173–205. United Kingdom: Cambridge University Press.
- Detlefsen, M. 2001. "What Does Gödel's Second Theorem Say?" *Philosophia Mathematica* 9 (1): 37–71.
- Dummett, M. 1973. "Truth-Value and Reference." In *Frege: Philosophy of Language*, 401–29. Great Britain: Harper & Row Publishers.
- Feferman, S. 1984. "Toward Useful Type-free Theories. I." *Journal of Symbolic Logic* 49 (1): 75–111.
- Fine, K. 1985. *Reasoning with Arbitrary Objects*. Oxford: Blackwell.
- Fine, K. 2007. *Semantic Relationism*. Singapore: Blackwell.
- Gaifman, H. 2006. "Naming and Diagonalization, from Cantor to Gödel to Kleene." *Logic Journal of the IGPL* 14 (5): 709–28.
- Grabmayr, B. 2021. "On the Invariance of Gödel's Second Theorem with Regard to Numberings." *Review of Symbolic Logic* 14 (1): 51–84.
- Grabmayr, B., and A. Visser. 2020. Self-reference Upfront: A Study of Self-Referential Gödel Numberings. Unpublished manuscript. arXiv: 2006.12178. Also available at <https://arxiv.org/abs/2006.12178>.
- Halbach, V., and A. Visser. 2014a. "Self-reference in Arithmetic I." *The Review of Symbolic Logic* 7 (4): 671–91.
- Halbach, V., and A. Visser. 2014b. "Self-Reference in Arithmetic II." *The Review of Symbolic Logic* 7 (4): 692–712.

- Halbach, V. 2016. "The Root of Evil: A Self-Referential Play in One Act." In *Liber Amicorum Alberti: A Tribute to Albert Visser, Vol. 30. Tributes*, 155–63. London: College Publications.
- Heck, R. 2007. "Self-Reference and the Languages of Arithmetic." *Philosophia Mathematica* 15 (1): 1–29.
- Imre, L. 1976. *Proofs and Refutations: The Logic of Mathematical Discovery*. Cambridge: Cambridge University Press.
- Jeroslow, R. G. 1973. "Redundancies in the Hilbert–Bernays Derivability Conditions for Gdel's Second Incompleteness Theorem." *Journal of Symbolic Logic* 38 (3): 359–67.
- Kripke, S. 1975. "Outline of a Theory of Truth." *Journal of Philosophy* 72 (19): 690–716.
- Kripke, S. A. 2021. "Gödel's Theorem and Direct Self-Reference." *The Review of Symbolic Logic* 1–6. <https://doi.org/10.1017/s1755020321000526>, in press.
- Leitgeb, H. 2002. "What is a Self-Referential Sentence? Critical Remarks on the Alleged (Non-) Circularity of Yablo's Paradox." *Logique and Analyse* 177 (178): 3–14.
- Leitgeb, H. 2005. "What Truth Depends on." *Journal of Philosophical Logic* 34 (2): 155–92.
- Leitgeb, H. 2019. "HYPE: A System of Hyperintensional Logic." *Journal of Philosophical Logic* 48 (2): 305–405.
- Milne, P. 2007. "On Gödel Sentences and What They Say." *Philosophia Mathematica III* 15: 193–226.
- Piccolo, L. 2018. "Reference in Arithmetic." *Review of Symbolic Logic* 11 (3): 573–603.
- Piccolo, L. 2020a. "Alethic Reference." *Journal of Philosophical Logic* 49: 417–38.
- Piccolo, L. 2020b. "Reference and Truth." *Journal of Philosophical Logic* 49: 4389–474.
- Priest, G. 1995/2002. *Beyond the Limits of Thought*, 2nd ed. Great Britain: Oxford University Press (originally published in 1995).
- Raatikainen, P. 2018. "Gödel's Incompleteness Theorems." In *The Stanford Encyclopedia of Philosophy*, edited by E. N. Zalta. Fall. Stanford: Metaphysics Research Lab. Stanford University. (Accessed 2018).
- Rossi, L. 2019. "A Unified Theory of Truth and Paradox." *Review of Symbolic Logic* 12 (2): 209–54.
- Sedlár, I. 2019. "Hyperintensional Logics for Everyone." *Synthese* 198 (2): 933–56.
- Smullyan, R. M. 1957. "Languages in Which Self-Reference is Possible." *The Journal of Symbolic Logic* 22 (1): 55–67.
- Smullyan, R. M. 1992. *Gödel's Incompleteness Theorems*. New York: Oxford University Press.
- Smullyan, R. M. 1994. *Diagonalization and Self-Reference*, Vol. 27. Oxford Logic Guides – Oxford Science Publications. Clarendon Press.