Monads as a Solution for Generalized Opacity

Gianluca Giorgolo
University of Oxford

Ash Asudeh
University of Oxford and Carleton University

{gianluca.giorgolo,ash.asudeh}@ling-phil.ox.ac.uk

Abstract

In this paper we discuss a conservative extension of the simply-typed lambda calculus in order to model a class of expressions that generalize the notion of opaque contexts. Our extension is based on previous work in the semantics of programming languages aimed at providing a mathematical characterization of computations that produce some kind of *side effect* (Moggi, 1989), and is based on the notion of *monads*, a construction in category theory that, intuitively, maps a collection of "simple" values and "simple" functions into a more complex value space, in a canonical way. The main advantages of our approach with respect to traditional analyses of opacity are the fact that we are able to explain in a uniform way a set of different but related phenomena, and that we do so in a principled way that has been shown to also explain other linguistic phenomena (Shan, 2001).

1 Introduction

Opaque contexts have been an active area of research in natural language semantics since Frege's original discussion of the puzzle (Frege, 1892). A sentence like (1) has a non-contradictory interpretation despite the fact that the two referring expressions *Hesperus* and *Phosphorus* refer to the same entity, the planet we know as Venus.

(1) Reza doesn't believe Hesperus is Phosphorus.

The fact that a sentence like (1) includes the modal *believe* has influenced much of the analyses proposed in the literature, and has linked the phenomenon with the notion of modality. In this paper we challenge this view and try to position data like (1) inside a larger framework that also includes other types of expressions.

We decompose examples like (1) along two dimensions: the presence or absence of a modal expression, and the way in which we multiply refer to the same individual. In the case of (1), we have a modal and we use two different co-referring expressions. Examples (2)-(4) complete the landscape of possible combinations:

(2) Dr. Octopus punched Spider-Man but he didn't punch Spider-Man.

(3) Mary Jane loves Peter Parker but she doesn't love Spider-Man.

(4) Reza doesn't believe Jesus is Jesus.

(2) is clearly a contradictory statement, as we predicate of Dr. Octopus that he has the property of having punched Spider-Man and its negation. Notice that in this example there is no modal and the exact same expression is used twice to refer to the object individual. In the case of (3) we still have no modal and we use two different but co-referring expressions to refer to the same individual. However in this case the statement has a non-contradictory reading. Similarly (4) has a non-contradictory reading, which states that, according to the speaker, Reza doesn't believe that the entity he (Reza) calls Jesus is the entity that the speaker calls Jesus (e.g., is not the same individual or does not have the same properties). This case is symmetrical to (3), as here we have a modal expression but the same expression is used twice to refer to the same individual.

If the relevant reading of (4) is difficult to get, consider an alternative kind of scenario, as in (5), in which the subject of the sentence, Kim, suffers from Capgras Syndrome[1] and thinks that Sandy is an impostor. The speaker says:

[1]From Wikipedia: "[Capgras syndrome] is a disorder in which a person holds a delusion that a friend, spouse, par-

Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS), pages 19–27,
Gothenburg, Sweden, April 26-30 2014.

(5) Kim doesn't believe Sandy is Sandy

Given the definition of Capgras Syndrome (in fn. 1), there is a clear non-contradictory reading available here, in which the speaker is stating that Kim does not believe that the entity in question, that the speaker and (other non-Capgras-sufferers) would call Sandy, is the entity that she associate with the name Sandy.

We propose an analysis of the non-contradictory cases based on the intuition that the apparently co-referential expressions are in fact interpreted using different interpretation functions, which correspond to different perspectives that are pitted against each other in the sentences. Furthermore, we propose that modal expressions are not the only ones capable of introducing a shift of perspective, but also that verbs that involve some kind of mental attitude of the subject towards the object have the same effect.

Notice that this last observation distinguishes our approach from one where a sentence like (3) is interpreted as simply saying that Mary Jane loves only one *guise* of the entity that corresponds to Peter Parker but not another one. The problem with this interpretation is that if it is indeed the case that different co-referring expressions simply pick out different guises of the same individual, then a sentence like (6) should have a non-contradictory reading, while this seems not to be the case.

(6) Dr. Octopus killed Spider-Man but he didn't kill Peter Parker.

While a guise-based interpretation is compatible with our analysis[2], it is also necessary to correctly model the different behaviour of verbs like *love* with respect to others like *punch* or *kill*. In fact, we need to model the difference between, for example, *kill* and *murder*, since *murder* does involve a mental attitude of intention and the corresponding sentence to (6) is not necessarily contradictory:

(7) Dr. Octopus murdered Spider-Man but he didn't murder Peter Parker.

The implementation of our analysis is based on monads. Monads are a construction in category theory that defines a canonical way to map a set of objects and functions that we may consider as simple into a more complex object and function space. They have been successfully used in the semantics of programming languages to characterise computations that are not "pure". By pure we mean code objects that are totally referentially transparent (i.e. do not depend on external factors and return the same results given the same input independently of their execution context), and also that do not have effects on the "real world". In contrast, monads are used to model computations that for example read from or write to a file, that depend on some random process or whose return value is non-deterministic.

In our case we will use the monad that describe values that are made dependent on some external factor, commonly known in the functional programming literature as the Reader monad.[3] We will represent linguistic expressions that can be assigned potentially different interpretations as functions from interpretation indices to values. Effectively we will construct a different type of lexicon that does not represent only the linguistic knowledge of a single speaker but also her (possibly partial) knowledge of the language of other speakers. So, for example, we will claim that (4) can be assigned a non-contradictory reading because the speaker's lexicon also includes the information regarding Reza's interpretation of the name *Jesus* and therefore makes it possible for the speaker to use the same expression, in combination with a verb such as *believe*, to actually refer to two different entities. In one case we will argue that the name *Jesus* is interpreted using the speaker's interpretation while in the other case it is Reza's interpretation that is used.

Notice that we can apply our analysis to any natural language expression that may have different interpretations. This means that, for example, we can extend our analysis, which is limited to referring expressions here for space reasons, to other cases, such as the standard examples involving ideally synonymous predicates like *groundhog* and *woodchuck* (see, e.g., Fox and Lappin (2005)).

The paper is organised as follows: in section 2 we discuss the technical details of our analysis; in section 3 we discuss our analyses of the motivating examples; section 4 compares our approach with a standard approach to opacity; we conclude in section 5.

ent, or other close family member has been replaced by an identical-looking impostor."

[2]Indeed, one way to understand guises is as different ways in which we interpret a referring term (Heim, 1998).

[3]Shan (2001) was the first to sketch the idea of using the Reader monad to model intensional phenomena.

2 Monads and interpretation functions

To avoid introducing the complexities of the categorical formalism, we introduce monads as they are usually encountered in the computer science literature. A monad is defined as a triple $\langle \Diamond, \eta, \star \rangle$. $\Diamond$ is what we call a functor, in our case a mapping between types and functions. We call the component of $\Diamond$ that maps between types $\Diamond_1$, while the one that maps between functions $\Diamond_2$. In our case $\Diamond_1$ will map each type to a new type that corresponds to the original type with an added interpretation index parameter. Formally, if i is the type of interpretation indices, then $\Diamond_1$ maps any type τ to $i \rightarrow \tau$. In terms of functions, $\Diamond_2$ maps any function $f : \tau \rightarrow \delta$ to a function $f' : (i \rightarrow \tau) \rightarrow i \rightarrow \delta$. $\Diamond_2$ corresponds to function composition:

$$\Diamond_2(f) = \lambda g.\lambda i.f(g(i)) \tag{8}$$

In what follows the component $\Diamond_2$ will not be used so we will use $\Diamond$ as an abbreviation for $\Diamond_1$. This means that we will write $\Diamond\tau$ for the type $i \rightarrow \tau$.

η (pronounced "unit") is a polymorphic function that maps inhabitants of a type τ to inhabitants of its image under $\Diamond$, formally $\eta : \forall\tau.\tau \rightarrow \Diamond\tau$. Using the computational metaphor, η should embed a value in a computation that returns that value without any side-effect. In our case η should simply add a vacuous parameter to the value:

$$\eta(x) = \lambda i.x \tag{9}$$

$\star$ (pronounced "bind") is a polymorphic function of type $\forall\tau.\forall\delta.\Diamond\tau \rightarrow (\tau \rightarrow \Diamond\delta) \rightarrow \Diamond\delta$, and acts as a sort of enhanced functional application.[4] Again using the computational metaphor, $\star$ takes care of combining the side effects of the argument and the function and returns the resulting computation. In the case of the monad we are interested in, $\star$ is defined as in (10).

$$a \star f = \lambda i.f(a(i))(i) \tag{10}$$

Another fundamental property of $\star$ is that, by imposing an order of evaluation, it will provide us with an additional scoping mechanism distinct from standard functional application. This will allow us to correctly capture the multiple readings associated with the expressions under consideration.

We thus add two operators, η and $\star$, to the lambda calculus and the reductions work as expected for (9) and (10). These reductions are implicit in our analyses in section 3.

2.1 Compositional logic

For composing the meanings of linguistic resources we use a logical calculus adapted for the linear case[5] from the one introduced by Benton et al. (1998). The calculus is based on a language with two connectives corresponding to our type constructors: $\multimap$, a binary connective, that corresponds to (linear) functional types, and $\Diamond$, a unary connective, that represents monadic types.

The logical calculus is described by the proof rules in figure 1. The rules come annotated with lambda terms that characterise the Curry-Howard correspondence between proofs and meaning terms. Here we assume a Lexical Functional Grammar-like setup (Dalrymple, 2001), where a syntactic and functional grammar component feeds the semantic component with lexical resources already marked with respect to their predicate-argument relationships. Alternatively we could modify the calculus to a categorial setting, by introducing a structural connective, and using directional versions of the implication connective together with purely structural rules to control the compositional process.

We can prove that the Cut rule is admissible, therefore the calculus becomes an effective (although inefficient) way of computing the meaning of a linguistic expression.

A key advantage we gain from the monadic approach is that we are not forced to generalize all lexical entries to the "worst case". With the logical setup we have just described we can in fact freely mix monadic and non monadic resources. For example, in our logic we can combine a pure version of a binary function with arguments that are either pure or monadic, as the following are all

[4] We use for $\star$ the argument order as it is normally used in functional programming. We could swap the arguments and make it look more like standard functional application. Also, we write $\star$ in infix notation.

[5] Linearity is a property that has been argued for in the context of natural language semantics by various researchers (Moortgat (2011), Asudeh (2012), among others).

$$\frac{}{x : A \vdash x : A}\, id \qquad \frac{\Gamma \vdash B \qquad B, \Delta \vdash C}{\Gamma, \Delta \vdash C}\, Cut$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B} \multimap R \qquad \frac{\Delta \vdash t : A \qquad \Gamma, x : B \vdash u : C}{\Gamma, \Delta, y : A \multimap B \vdash u[y(t)/x] : C} \multimap L$$

$$\frac{\Gamma \vdash x : A}{\Gamma \vdash \eta(x) : \Diamond A} \Diamond R \qquad \frac{\Gamma, x : A \vdash t : \Diamond B}{\Gamma, y : \Diamond A \vdash y \star \lambda x.t : \Diamond B} \Diamond L$$

Figure 1: Sequent calculus for a fragment of multiplicative linear logic enriched with a monadic modality, together with a Curry-Howard correspondence between formulae and meaning terms.

provable theorems in this logic.

$$A \multimap B \multimap C, A, B \vdash \Diamond C \quad (11)$$
$$A \multimap B \multimap C, \Diamond A, B \vdash \Diamond C \quad (12)$$
$$A \multimap B \multimap C, A, \Diamond B \vdash \Diamond C \quad (13)$$
$$A \multimap B \multimap C, \Diamond A, \Diamond B \vdash \Diamond C \quad (14)$$

In contrast, the following is not a theorem in the logic:

$$A \multimap B \multimap C, I \multimap A, I \multimap B \nvdash I \multimap C \quad (15)$$

In general, then, if we were to simply lift the type of the lexical resources whose interpretation may be dependent on a specific point of view, we would be forced to lift all linguistic expressions that may combine with them, thus generalizing to the worst case after all.

The monadic machinery also achieves a higher level of compositionality. In principle we could directly encode our monad using the $\rightarrow$ type constructor, with linear implication, $\multimap$, on the logical side. However this alternative encoding wouldn't have the same deductive properties. Compare the pattern of inferences we have for the monadic type, in (11)–(14), with the equivalent one for the simple type:

$$A \multimap B \multimap C, A, B \vdash C \quad (16)$$
$$A \multimap B \multimap C, I \multimap A, B \vdash I \multimap C \quad (17)$$
$$A \multimap B \multimap C, A, I \multimap B \vdash I \multimap C \quad (18)$$
$$A \multimap B \multimap C, I \multimap A, I \multimap B \vdash I \multimap I \multimap C \quad (19)$$

In the case of the simple type, the final formula we derive depends in some non-trivial way on the entire collection of resources on the left-hand side of the sequent. In contrast in the case of the monadic type, the same type can be derived for all configurations. What is important is that we can predict the final formula without having to consider the entire set of resources available. This shows that the compositionality of our monadic approach cannot be equivalently recapitulated in a simple type theory.

3 Examples

The starting point for the analysis of examples (1)-(4) is the lexicon in table 1. The lexicon represents the linguistic knowledge of the speaker, and at the same time her knowledge about other agents' grammars.

Most lexical entries are standard, since we do not need to change the type and denotation of lexical items that are not involved in the phenomena under discussion. So, for instance, logical operators such as *not* and *but* are interpreted in the standard non-monadic way, as is a verb like *punch* or *kill*. Referring expressions that are possibly contentious, in the sense that they can be interpreted differently by the speaker and other agents, instead have the monadic type $\Diamond e$. This is reflected in their denotation by the fact that their value varies according to an interpretation index. We use a special index σ to refer to the speaker's own perspective, and assume that this is the default index used whenever no other index is specifically introduced. For example, in the case of the name Spider-Man, the speaker is aware of his secret identity and therefore interprets it as another name for the individual Peter Parker, while Mary Jane and Dr. Octopus consider Spider-Man a different entity from Peter Parker.

We assume that sentences are interpreted in a model in which all entities are mental entities, i.e. that there is no direct reference to entities in the world, but only to mental representations. Entities are therefore relativized with respect to the agent that mentally represents them, where non-contentious entities are always relativized according to the speaker. This allows us to represent the

WORD	DENOTATION	TYPE
Reza	$\mathbf{r}_\sigma$	e
Kim	$\mathbf{k}_\sigma$	e
Dr. Octopus	$\mathbf{o}_\sigma$	e
Mary Jane	$\mathbf{mj}_\sigma$	e
Peter Parker	$\mathbf{pp}_\sigma$	e
not	$\lambda p.\neg p$	$t \to t$
but	$\lambda p.\lambda q.p \wedge q$	$t \to t \to t$
is	$\lambda x.\lambda y.x = y$	$e \to e \to t$
punch	$\lambda o.\lambda s.\mathbf{punch}(s)(o)$	$e \to e \to t$
believe	$\lambda c.\lambda s.\lambda i.\mathbf{B}(s)(c(\kappa(s)))$	$\Diamond t \to e \to \Diamond t$
love	$\lambda o.\lambda s.\lambda i.\mathbf{love}(s)(o(\kappa(s)))$	$\Diamond e \to e \to \Diamond t$
Hesperus	$\lambda i.\begin{cases}\mathbf{es}_r & \text{if } i = \mathbf{r},\\ \mathbf{v}_\sigma & \text{if } i = \sigma\end{cases}$	$\Diamond e$
Phosphorus	$\lambda i.\begin{cases}\mathbf{ms}_r & \text{if } i = \mathbf{r},\\ \mathbf{v}_\sigma & \text{if } i = \sigma\end{cases}$	$\Diamond e$
Spider-Man	$\lambda i.\begin{cases}\mathbf{sm}_i & \text{if } i = \mathbf{o} \text{ or } i = \mathbf{mj},\\ \mathbf{pp}_\sigma & \text{if } i = \sigma\end{cases}$	$\Diamond e$
Jesus	$\lambda i.\begin{cases}\mathbf{j}_r & \text{if } i = \mathbf{r},\\ \mathbf{j}_\sigma & \text{if } i = \sigma\end{cases}$	$\Diamond e$
Sandy	$\lambda i.\begin{cases}\mathbf{imp}_k & \text{if } i = \mathbf{k},\\ \mathbf{s}_\sigma & \text{if } i = \sigma\end{cases}$	$\Diamond e$

Table 1: Speaker's lexicon

fact that different agents may have distinct equivalencies between entities. For example, Reza in our model does not equate the evening star and the morning star, but the speaker equates them with each other and with Venus. Therefore, the speaker's lexicon in table 1 represents the fact that the speaker's epistemic model includes what the speaker knows about other agents' models, e.g. that Reza has a distinct denotation (from the speaker) for Hesperus, that Mary Jane has a distinct representation for Spider-Man, that Kim has a distinct representation for Sandy, etc.

The other special lexical entries in our lexicon are those for verbs like *believe* and *love*. The two entries are similar in the sense that they both take an already monadic resource and actively supply a specific interpretation index that corresponds to the subject of the verbs. The function κ maps each entity to the corresponding interpretation index, i.e. $\kappa : e \to i$. For example, in the lexical entries for *believe* and *love*, κ maps the subject to the interpretation index of the subject. Thus, the entry for *believe* uses the subject's point of view as the perspective used to evaluate its entire complement, while *love* changes the interpretation of its object relative to the perspective of its subject. However we will see that the interaction of these lexical entries and the evaluation order imposed by $\star$ will allow us to let the complement of a verb like *believe* and the object of a verb like *love* escape the specific effect of forcing the subject point of view, and instead we will be able to derive readings in which the arguments of the verb are interpreted using the speaker's point of view.

Figure 2 reports the four non-equivalent readings that we derive in our system for example (1), repeated here as (20).[6]

(20) Reza doesn't believe that Hesperus is Phosphorus.

Reading (21) assigns to both Hesperus and Phosphorus the subject interpretation and results, after contextualising the sentence by applying it to the standard σ interpretation index, in the truth conditions in (25), i.e. that Reza does not believe that the evening star is the morning star. This

[6]The logic generates six different readings but the monad we are using here has a commutative behaviour, so four of these readings are pairwise equivalent.

$$[\![\text{believe}]\!]\,([\![\text{Hesperus}]\!] \star \lambda x.\,[\![\text{Phosphorus}]\!] \star \lambda y.\eta([\![\text{is}]\!]\,(x)(y)))([\![\text{Reza}]\!]) \star \lambda z.\eta([\![\text{not}]\!]\,(z)) \quad (21)$$

$$[\![\text{Hesperus}]\!] \star \lambda x.\,[\![\text{believe}]\!]\,([\![\text{Phosphorus}]\!] \star \lambda y.\eta([\![\text{is}]\!]\,(x)(y)))([\![\text{Reza}]\!]) \star \lambda z.\eta([\![\text{not}]\!]\,(z)) \quad (22)$$

$$[\![\text{Phosphorus}]\!] \star \lambda x.\,[\![\text{believe}]\!]\,([\![\text{Hesperus}]\!] \star \lambda y.\eta([\![\text{is}]\!]\,(y)(x)))([\![\text{Reza}]\!]) \star \lambda z.\eta([\![\text{not}]\!]\,(z)) \quad (23)$$

$$[\![\text{Hesperus}]\!] \star \lambda x.\,[\![\text{Phosphorus}]\!] \star \lambda y.\,[\![\text{believe}]\!]\,(\eta([\![\text{is}]\!]\,(x)(y)))([\![\text{Reza}]\!]) \star \lambda z.\eta([\![\text{not}]\!]\,(z)) \quad (24)$$

Figure 2: Non-equivalent readings for *Reza doesn't believe Hesperus is Phosphorus.*

reading would not be contradictory in an epistemic model (such as Reza's model) where the evening star and the morning star are not the same entity.

$$\neg\mathbf{B}(\mathbf{r})(\mathbf{es}_r = \mathbf{ms}_r) \quad (25)$$

In the case of (22) and (23) we get a similar effect although here we mix the epistemic models, and one of the referring expressions is interpreted under the speaker perspective while the other is again interpreted under Reza's perspective. For these two readings we obtain respectively the truth conditions in (26) and (27).

$$\neg\mathbf{B}(\mathbf{r})(\mathbf{v}_\sigma = \mathbf{ms}_r) \quad (26)$$

$$\neg\mathbf{B}(\mathbf{r})(\mathbf{v}_\sigma = \mathbf{es}_r) \quad (27)$$

Finally for (24) we get the contradictory reading that Reza does not believe that Venus is Venus, as both referring expressions are evaluated using the speaker's interpretation index.

$$\neg\mathbf{B}(\mathbf{r})(\mathbf{v}_\sigma = \mathbf{v}_\sigma) \quad (28)$$

The different contexts for the interpretation of referring expressions are completely determined by the order in which we evaluate monadic resources. This means that, just by looking at the linear order of the lambda order, we can check whether a referring expression is evaluated inside the scope of a perspective changing operator such as *believe*, or if it is interpreted using the standard interpretation.

If we consider a case like sentence (2), we ought to get only a contradictory reading as the statement is intuitively contradictory. Our analysis produces a single reading that indeed corresponds to a contradictory interpretation:

$$[\![\text{Spider-Man}]\!] \star \lambda x.\,[\![\text{Spider-Man}]\!] \star \lambda y.\eta([\![\text{but}]\!]\,([\![\text{punch}]\!]\,([\![\text{Dr. Octopus}]\!])(x))\,([\![\text{not}]\!]\,([\![\text{punch}]\!]\,([\![\text{Dr. Octopus}]\!])(y)))) \quad (29)$$

The verb *punch* is not a verb that can change the interpretation perspective and therefore the potentially controversial name Spider-Man is interpreted in both instances using the speaker's interpretation index. The result are unsatisfiable truth conditions, as expected:

$$\mathbf{punch}(\mathbf{o}_\sigma)(\mathbf{pp}_\sigma) \wedge \neg\mathbf{punch}(\mathbf{o}_\sigma)(\mathbf{pp}_\sigma) \quad (30)$$

In contrast a verb like *love* is defined in our lexicon as possibly changing the interpretation perspective of its object to that of its subject. Therefore in the case of a sentence like (3), we expect one reading where the potentially contentious name Spider-Man is interpreted according to the subject of *love*, Mary Jane. This is in fact the result we obtain. Figure 3 reports the two readings that our framework generates for (3).

Reading (31), corresponds to the non contradictory interpretation of sentence (3), where Spider-Man is interpreted according to Mary Jane's perspective and therefore is assigned an entity different from Peter Parker:

$$\mathbf{love}(\mathbf{mj}_\sigma)(\mathbf{pp}_\sigma) \wedge \neg\mathbf{love}(\mathbf{mj}_\sigma)(\mathbf{sm}_{mj}) \quad (33)$$

Reading (32) instead generates unsatisfiable truth conditions, as Spider-Man is identified with Peter Parker according to the speaker's interpretation:

$$\mathbf{love}(\mathbf{mj}_\sigma)(\mathbf{pp}_\sigma) \wedge \neg\mathbf{love}(\mathbf{mj}_\sigma)(\mathbf{pp}_\sigma) \quad (34)$$

Our last example, (4), repeated here as (35), is particularly interesting as we are not aware of previous work that discusses this type of sentence. The non-contradictory reading that this sentence has seems to be connected specifically to two different interpretations of the same name, *Jesus*, both under the syntactic scope of the modal *believe*.

(35) Reza doesn't believe Jesus is Jesus.

$$[\![\text{love}]\!]\,(\eta([\![\text{Peter Parker}]\!]))([\![\text{Mary Jane}]\!]) \star \lambda p.\,[\![\text{love}]\!]\,([\![\text{Spider-Man}]\!])([\![\text{Mary Jane}]\!]) \star \lambda q.\eta([\![\text{but}]\!]\,(p)([\![\text{not}]\!]\,(q))) \quad (31)$$

$$[\![\text{love}]\!]\,(\eta([\![\text{Peter Parker}]\!]))([\![\text{Mary Jane}]\!]) \star \lambda p.\,[\![\text{Spider-Man}]\!] \star \lambda x.\,[\![\text{love}]\!]\,(\eta(x))([\![\text{Mary Jane}]\!]) \star \lambda q.\eta([\![\text{but}]\!]\,(p)([\![\text{not}]\!]\,(q))) \quad (32)$$

Figure 3: Non-equivalent readings for *Mary Jane loves Peter Parker but she doesn't love Spider-Man.*

$$[\![\text{believe}]\!]\,([\![\text{Jesus}]\!] \star \lambda x.\,[\![\text{Jesus}]\!] \star \lambda y.\eta([\![\text{is}]\!]\,(x)(y)))([\![\text{Reza}]\!]) \star \lambda z.\eta([\![\text{not}]\!]\,(z)) \quad (36)$$

$$[\![\text{Jesus}]\!] \star \lambda x.\,[\![\text{Jesus}]\!] \star \lambda y.\,[\![\text{believe}]\!]\,(\eta([\![\text{is}]\!]\,(x)(y)))([\![\text{Reza}]\!]) \star \lambda z.\eta([\![\text{not}]\!]\,(z)) \quad (37)$$

$$[\![\text{Jesus}]\!] \star \lambda x.\,[\![\text{believe}]\!]\,([\![\text{Jesus}]\!] \star \lambda y.\eta([\![\text{is}]\!]\,(x)(y)))([\![\text{Reza}]\!]) \star \lambda z.\eta([\![\text{not}]\!]\,(z)) \quad (38)$$

Figure 4: Non-equivalent readings for *Reza doesn't believe Jesus is Jesus.*

Our system generates three non-equivalent readings, reported here in figure 4.[7]

Reading (36) and (37) corresponds to two contradictory readings of the sentence: in the first case both instances of the name Jesus are interpreted from the subject perspective and therefore attribute to Reza the non-belief in a tautology, similarly in the second case, even though in this case the two names are interpreted from the perspective of the speaker. In contrast the reading in (38) corresponds to the interpretation that assigns two different referents to the two instances of the name Jesus, producing the truth conditions in (39) which are satisfiable in a suitable model.

$$\neg\mathbf{B}(\mathbf{r})(\mathbf{j}_\sigma = \mathbf{j}_r) \quad (39)$$

The analysis of the Capgras example (5), repeated in (40), is equivalent; the non-contradictory reading is shown in (41).

(40) Kim doesn't believe Sandy is Sandy.

$$\neg\mathbf{B}(\mathbf{k})(\mathbf{s}_\sigma = \mathbf{imp}_k) \quad (41)$$

We use $\mathbf{imp}_k$ as the speaker's representation of the "impostor" that Kim thinks has taken the place of Sandy.

More generally, there are again three non-equivalent readings, including the one above, which are just those in figure 4, with [[Jesus]] replaced by [[Sandy]] and [[Reza]] replaced by [[Kim]].

[7] Again, there are six readings that correspond to different proofs, but given the commutative behaviour of the Reader monad, the fact that equality is commutative, and the fact that we have in this case two identical lexical items, only three of them are non-equivalent readings.

4 Comparison with traditional approaches

In this section we try to sketch how a traditional approach to opaque contexts, such as one based on a *de dicto/de re* ambiguity with respect to a modal operator, would fare in the analysis of (4), our most challenging example.

To try to explain the two readings in the context of a standard possible worlds semantics, we could take (4) to be ambiguous with respect to a *de dicto/de re* reading. In the case of the *de dicto* reading (which corresponds to the non-satisfiable reading) the two names are evaluated under the scope of the doxastic operator *believe*, i.e. they both refer to the same entity that is assigned to the name Jesus in each accessible world. Clearly this is always the case, and so (4) is not satisfiable. In the case of the *de re* reading, we assume that the two names are evaluated at different worlds that assign different referents to the two names. One of these two worlds will be the actual world and the other one of the accesible worlds. The reading is satisfiable if the doxastic modality links the actual world with one in which the name Jesus refers to a different entity. Notice that for this analysis to work we need to make two assumptions: 1. that names behave as quantifiers with the property of escaping modal contexts, 2. that names can be assigned different referents in different worlds, i.e. we have to abandon the standard notion that names are rigid designators (Kripke, 1972). In contrast, in our approach we do not need to abandon the idea of rigid designation for names (within each

agent's model).

However, such an approach would present a number of rather serious problems. The first is connected with the assumption that names are scopeless. This is a common hypothesis in natural language semantics and indeed if we model names as generalized quantifiers they can be proven to be scopeless (Zimmermann, 1993). But this is problematic for our example. In fact we would predict that both instances of the name *Jesus* escape the scope of *believe*. The resulting reading would bind the quantified individual to the interpretation of *Jesus* in the actual world. In this way we only capture the non-satisfiable reading. To save the scopal approach we would need to assume that names in fact are sometimes interpreted in the scope of modal operators.

One way to do this would be to set up our semantic derivations so that they allow different scopal relations between quantifiers and other operators. The problem with this solution is that for sentences like (4) we generate twelve different derivations, some of which do not correspond to valid readings of the sentence.

Even assuming that we find a satisfactory solution for these problems, the scopal approach cannot really capture the intuitions behind opacity in all contexts. Consider again (4) and assume that there are two views about Jesus: Jesus as a divine being and Jesus as a human being. Assume that Jesus is a human being in the actual world and that Reza is an atheist, then the only possible reading is the non-satisfiable one, as the referent for Jesus will be the same in the actual world and all accessible Reza-belief-worlds. The problem is that the scopal approach assumes a single modal model, while in this case it seems that there are two doxastic models, Reza's model and the speaker's model, under discussion. In contrast, in our approach the relevant part of Reza's model is embedded inside the speaker's model and interpretation indices indicate which interpretation belongs to Reza and which to the speaker.

Finally an account of modality in terms of scopal properties is necessarily limited to cases in which modal operators are present. While this may be a valid position in the case of typical intensional verbs like *seek* or *want*, it would not be clear how we could extend this approach to cases like 3, as the verb *love* has no clear modal connotation. Thus, the scopal approach would not be sufficiently general.

5 Conclusion

We started by discussing a diverse collection of expressions that share the common property of showing nontrivial referential behaviours. We have proposed a common analysis of all these expressions in terms of a combination of different interpretation contexts. We have claimed that the switch to a different interpretation context is triggered by specific lexical items, such as modal verbs but also verbs that express some kind of mental attitude of the subject of the verb towards its object. The context switch is not obligatory, as witnessed by the multiple readings that the sentences discussed seem to have. We implemented our analysis using monads. The main idea of our formal implementation is that referring expressions that have a potential dependency from an interpretation context can be implemented as functions from interpretation indices to fully interpreted values. Similarly, the linguistic triggers for context switch are implemented in the lexicon as functions that can modify the interpretation context of their arguments. Monads allow us to freely combine these "lifted" meanings with standard ones, avoiding in this way to generalize our lexicon to the worst case. We have also seen how more traditional approaches, while capable of dealing with some of the examples we discuss, are not capable of providing a generalised explanation of the observed phenomena.

Acknowledgements

The authors would like to thank our anonymous reviewers for their comments. This research is supported by a Marie Curie Intra-European Fellowship from the Europoean Commision under contract number 327811 (Giorgolo) and an Early Researcher Award from the Ontario Ministry of Research and Innovation and NSERC Discovery Grant #371969 (Asudeh).

References

Ash Asudeh. 2012. *The Logic of Pronominal Resumption*. Oxford Studies in Theoretical Linguistics. Oxford University Press, New York.

Nick Benton, G. M. Bierman, and Valeria de Paiva. 1998. Computational types from a logical perspective. *Journal of Functional Programming*, 8(2):177–193.

Mary Dalrymple. 2001. *Lexical Functional Grammar*. Academic Press, San Diego, CA.

Chris Fox and Shalom Lappin. 2005. *Foundations of Intensional Semantics*. Blackwell, Oxford.

Gottlob Frege. 1892. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, 100:25–50.

Gottlob Frege. 1952. On sense and reference. In Peter T. Geach and Max Black, editors, *Translations from the Philosophical Writings of Gottlob Frege*, pages 56–78. Blackwell, Oxford. Translation of Frege (1892).

Irene Heim. 1998. Anaphora and semantic interpretation: A reinterpretation of Reinhart's approach. In Uli Sauerland and Orin Percus, editors, *The Interpretive Tract*, volume 25 of *MIT Working Papers in Linguistics*, pages 205–246. MITWPL, Cambridge, MA.

Saul Kripke. 1972. Naming and necessity. In Donald Davidson and Gilbert Harman, editors, *Semantics of Natural Language*, pages 253–355. Reidel.

Eugenio Moggi. 1989. Computational lambda-calculus and monads. In *LICS*, pages 14–23. IEEE Computer Society.

Michael Moortgat. 2011. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 95–179. Elsevier, second edition.

Chung-chieh Shan. 2001. Monads for natural language semantics. In Kristina Striegnitz, editor, *Proceedings of the ESSLLI-2001 Student Session*, pages 285–298. 13th European Summer School in Logic, Language and Information.

Thomas Ede Zimmermann. 1993. Scopeless quantifiers and operators. *Journal of Philosophical Logic*, 22(5):545–561, October.