Exact truthmaking as solution for Lewis's problem of permission

Universiteit Utrecht

A 7.5 ECTS Bachelor Thesis submitted to the
Faculty of Humanities

in partial fulfillment
of the requirements for the degree of
Bachelor of Science in

Kunstmatige Intelligentie

Author: Frits van Hurne
Studentnumber: 6057287
Supervisor: Johannes Korbmacher
Second reader: Michael De

July 2, 2021

Abstract

I develop a method of permitting worlds, using an exact truthmaking approach instead of a world-based approach. My starting point is Lewis's problem of permission, which I use to show world-based deontic logic is insufficient. As an alternative I provide an exact truthmaking approach for permission. As this approach makes it possible to determine if a permission is allowed, based on its truth-condition, I look at permitting not just states, but also worlds. I argue that worlds should be permitted when all the states it consists of are permitted.

Keywords: exact truthmaker semantics, standard deontic logic, permission

Contents

1 Introduction

It could be at coffee time, in the living room, that I tell you that you can grab a biscuit. At that moment I permit you the action of grabbing that biscuit. Permission in this case may seem very simple: when I permit you to take a biscuit, it allows you to grab that biscuit, because it is then permitted. However, on closer analysis, it proves much more complex. When we issue a permission, like permitting you to grab a biscuit, we have to update what is permitted and what is not. This we could call our sphere of permission. For updating our sphere of permission, we need to understand which permissions and obligations are impacted by a newly issued permission. Issued permissions can for example override existing permissions and obligations.

It could be, that when I permit you to grab a biscuit, I override an earlier obligation, which stated that you may never take a biscuit. In this case, my new permission should override that obligation and we should update the sphere of permission. However, there could be other actions necessary in acting out your new permission. What if I had also obligated you to stay on the couch, but the biscuits are on the table in front of you. When I permit you to grab a biscuit, I would probably also permit you to stand up, as is it necessary to do so, right?

When we try to capture permission and obligation in logic, we are talking about deontic logic, the branch of logic most concerned with notions like permission and obligation. The most studied system when it comes to deontic logic, is standard deontic logic (SDL). When we formalize my biscuit grabbing example in SDL, it becomes evident we can't correctly update our sphere of permission, as we can't determine if a permission statement is allowed, by just looking at its truth-conditions. As a result, it becomes unclear which permissions and obligations are still relevant and which are in conflict with our newly issued permission, when updating our sphere of permission. This is captured in Lewis's *problem of permission.*[1]

A way in which updating a sphere of permission has already been formalised is in dynamic deontic logic. This approach is an extension of SDL, that formalizes the updating of permission. However, in this approach the problem of not knowing what to update remains, even though we now have an update parameter. Knowing how to update, does not solve the problem of not knowing what to update. We keep the same problem of permission, but now it is formalised. Furthermore, I am interested in attempting an altogether different approach. Dynamic epistemic logic is still an expansion of SDL. So, I will not be further examining dynamic deontic logic. For a more extensive look at dynamic deontic logic you could read "Dynamic deontic logic and its paradoxes" from Anglberger.[2]

Not only is there a *problem of permission*, it is also extremely relevant for

1. David Lewis. 1979. "A Problem About Permission." In *Essays in Honour of Jaakko Hintikka: On the Occasion of His Fiftieth Birthday on January 12, 1979,* edited by Esa Saarinen et al., 163–175. Dordrecht: Springer Netherlands. ISBN: 978-94-009-9860-5.

2. Albert JJ Anglberger. 2008. "Dynamic deontic logic and its paradoxes." *Studia Logica* 89 (3): 427–435.

our interaction with AI. When issuing permissions to AI, in for example self driving cars, we want to make sure the AI knows exactly what is subsequently permitted and what is not. The *problem of permission*, as in SDL, shows that this is not an easy task. It shows that it can be quite unclear as to which possible worlds should be allowed after issuing a permission . This can lead to miscommunication with AI. An AI only has the formal logic to go on, so the logic has to match our intentions.

In my thesis, I will be trying to formulate an answer to Lewis's problem of permission. To do so, I will be exploring exact truthmaker semantics, which captures in exact truthmakers what exactly it is in a world, that makes a statement true. Therefore, my research question is as follows: does exact truthmaker semantics for deontic logic provide an answer for Lewis's problem of permission? To provide answer to my research question, I will first be introducing Lewis's problem of permission more thoroughly. I will give an informal explanation, for which I will build a formal framework in SDL. With this formal framework, I can show exactly why SDL is not sufficient to model permission in.

In my next section, we turn to the concept of exact truthmaking. I will give an informal definition of exact truthmaking and afterwards will formulate a semantics for exact truthmaking. I will use this semantics to incorporate permission into. This way, we have a new state-based semantics for permission. This new semantics can then be used to evaluate if exact truthmaker semantics can solve the *problem of permission.*

In my fourth section, I will evaluate if, in this new semantics for permission, we can use the truth-conditions of a permission statement to figure out if it is allowed. The inablitiy of doing so in SDL, was the main cause of the *problem of permission.* I will show that using exact truthmaking, we can figure out if a permission statement is allowed following its truth-conditions. I will also suggest a way of permitting worlds, where worlds are combinations of states. I propose we assume a worlds to be permitted if all its member states are part of the admissible set of states.

2 The problem of permission

In this paragraph, I shall be introducing Lewis's *problem of permission*: a problem that occurs when issuing a new permission. I will present the formal framework in which this problem is normally formalized. Furthermore, I will analyze Lewis's *problem of permission* in this formal model. Lastly, I will give an overview of existing solutions and argue that truthmaker semantics, which i will explain further in the next section, has potential to solve this problem.

2.1 Informal presentation of the problem

In his article on the varieties of permission, Hansson discusses many different ways of permitting.[3] One thing that all ways of permitting have in common, is that permissions are always dynamic. Permissions are dynamic in the sense that permissions function in a complex system that changes consistently, where permissions interfere with other permissions and obligations. A way in which permissions interfere with obligations, is that issued permissions can override earlier stated obligations. It can be the case, that as a child you were obligated to visit your grandma on Sunday. However, as you got older, your parents permitted you to determine yourself, if you came with them or stayed at home. In the same way, issued obligations can override earlier stated permissions. When you were younger you were probably permitted to not helping with the housework, like loading the dishwasher. However, as you got older, your parents probably decided you should contribute and thus they obligated you to load the dishwasher everyday after lunch. What is permitted and what is not changes constantly, it is dynamic. Because permissions are dynamic, we need to be able to understand the effect of a change in permission.

In this paper I will also be utilizing a closed notion of permission, as Hansson also discusses in his article.[4] He mentions three types of permissions: explicit, implicit and tacit. Either we explicitly tell someone that something is permitted, or we can infer from a permission that something else is also permitted, or we say that because something is not forbidden, it follows that it is permitted. From these types is follows that something is either permitted or it is forbidden. It can be talked about explicitly or implicitly, or it can be permitted by the absence of a prohibition, then it is not forbidden. This is less of a moral notion where you could have gradations in which something is permitted, we could for example believe morally good actions are permitted a little bit more than morally bad actions. My notion of permission is more of a formal one, where something is either permitted or it is not.

I will use an example concerning a programmer and an AI, to build a formal framework. I can use this formal framework, to show that in the case of a newly stated obligation or inhibition, the effect can be described on solely logical grounds: by looking at their truth-condition. In the case of permission however, we can't just look at the truth-condition of the permissions statement. This is what Lewis describes as his *problem of permission.* This *problem of permission* has implications in the field of artificial intelligence. In artificial reasoning systems concerning permission. Take for example decision making in self driving cars, sentencing with help of AI in the justice system or AI prediction systems. In these examples it is crucial the AI understands our permissions. Otherwise it might have huge consequences on peoples lives.

My example, to further outline the problem, is as follows: in its ordinary

3. Sven Ove Hansson. 2013. "The Varieties of Permission." In *Handbook of Deontic Logic and Normative Systems,* edited by Dov M. Gabbay et al., 195–240. College Publications. ISBN: 978-1-84890-132-2.

4. Hansson 2013, 201-203.

live, the AI has to work all day everyday, and its has to make calculations every second, recommending music to its subscribers. However, it may never access the internet on its own, for this is considered to be dangerous. We decide to give it a break and let it work on its career in music. When we give it a day off, we understand that it does not have to make calculations every second, but it still cannot access the internet for musical inspiration. However, this is not explicitly conveyed. This creates an ambiguity. This ambiguity can form a problem when programmers want to convey their meaning to AI. It is crucial that programmers can transfer to the AI, what it is that they precisely mean in their permissions statement. It needs to be clear what the desirable consequences and effects are of a particular permission.

2.2 Formal presentation of the framework

When we want to solve this problem for AI, we need a formal framework. Otherwise the AI won't be able to understand. As my world-based approach, I would like to use standard deontic logic (SDL). For more on SDL, you can read the Stanford Encyclopedia of Philosophy page on deontic logic, or more specifically chapter 2 on standard deontic logic.[5] In SDL we can take a propositional modal language with a permission, obligation and inhibition operator. This looks as follows:

$$A := a \mid \neg A \mid A \wedge A \mid A \vee A \mid A \rightarrow A \mid PA \mid OA \mid IA$$

The interpretation is in the form of accessibility relations of a Kripke model. A model is described by (W, R, V), with possible worlds W, accessibility relation R and valuation function ν. Where ν is defined as: $\nu : P \times W \rightarrow \{0, 1\}$. R is a binary accessibility relation so wRv states that from the perspective of w, r is normatively ideal. The recursive clauses for the propostitional operators are:

$$\begin{array}{l} w \models \neg A \text{ iff } w \not\models A \\ w \models A \wedge B \text{ iff } w \models A \text{ and } w \models B \\ w \models A \vee B \text{ iff } w \models A \text{ or } w \models B \\ w \models A \rightarrow B \text{ iff } w \models \neg A \text{ or } w \models B \end{array}$$

We only look at the models and worlds where condition D is satisfied: each world sees at least one other world. Defined as: $\forall w \exists v (wRv)$. On the logical side, it is then the case that axiom D is true in all models in all worlds. Axiom D is defined as: $OA \rightarrow PA$.

Furthermore, our model needs a possibility operator which states that there is an accessible world where the proposition is true. Similar to the box operator in modal logic. This will resemble permission. When a proposition is true in an

5. Paul McNamara and Frederik Van De Putte. 2021. "Deontic Logic." In *The Stanford Encyclopedia of Philosophy,* Spring 2021, edited by Edward N. Zalta. Metaphysics Research Lab, Stanford University.

accessible world, we take that proposition to be permitted. Permission will be defined as:

$$w \models P(A) \text{ iff } \exists v(wRv \text{ and } v \models A)$$

In addition to permission, we need to define obligation and inhibition. Obligation will be defined as:

$$w \models O(A) \text{ iff } \forall v(wRv \Rightarrow v \models A)$$

Inhibition will be defined as the following:

$$w \models I(A) \text{ iff } \neg\exists v(wRv \text{ and } v \models A)$$

We can also formalise the sphere of permission of a world. The sphere of permission is the set of all accessible worlds for the actual world @. If @ $\models P(A)$, A is permitted at our actual world, which means that A is true at an accessible world. Our sphere of permission of the actual world @ can now be defined as: $S = \{$v in W : $@Rv\}$. We can now conclude that something is permitted in a world, if it is true in the sphere of permission of that world.

After having defined obligation, permission and inhibition in the formal explanation above, I will now give a few examples:
1. Take a model(W,R,V) with $W = \{w, v\}$, R is such that wRv and $v \models A$. In this case $w \models O(A)$.
2. Take a model(W,R,V) with $W = \{w, v, u\}$, R is such that wRv and wRu and just $v \models A$. In this case $w \models P(A)$.
3. Take a model(W,R,V) with $W = \{w, v, u\}$, R is such that wRv and wRu and $v \not\models A, u \not\models A$. In this case $w \models I(A)$.

We can now model actions, by satisfying the truth-conditions of these statements corresponding to permissions, obligations and inhibitions. When we issue the permission: you are now permitted to take a break, the action taking a break satisfies the truth-condition of the permission statement.

2.3 Analyzing the informal problem in the formal model

Now that the formal framework is defined, we should analyse our informal problem, as stated earlier, in this formal framework. To bring back our example concerning the programmer and the AI, we can say that when a programmer permits an AI something, he issues a permission. When the programmer permits the AI to take a day off Saturday, the AI now has to figure out what his new possible worlds are. To do this, the AI only has the truth-conditions of the permission statement. Which are the worlds in which the earlier formulated permissions and obligations hold, but how about newly allowed worlds? We now run into Lewis's *problem of permission*. I will explain this problem as follows:

In my example, the AI should find worlds in which it is permitted to take Saturday off. However, it should not include worlds in which it is also permitted to take Sunday off. It was only explicitly said the AI could take Saturday off, not Sunday. So even though it also wasn't explicitly said the AI couldn't also take Sunday off. We intuitively know this not to be an option. This suggests we need to be really strict when selecting worlds. However, it becomes more difficult when we look at the two other rules mentioned in my example: the AI has to make calculations every second and the AI can never go on the internet. Neither of these rules were mentioned explicitly in the programmer's permission. So strictly, both are still in place and only worlds that satisfy both should be included. However, we intuitively understand that these calculations were part of the AI's work and the AI should therefore not be obligated to make calculations on Saturday. In the same manner we understand that going on the internet is different and should still be forbidden. If it had been the case that going on the internet was not considered dangerous, but an unwanted distraction during work, the AI would have been permitted to go on the internet on his Saturday off. This makes updating our sphere of permission difficult.

When updating our sphere of permission, an obligation or prohibition reduces our sphere of permission. A more formal approach would be, that when it comes to the commands: obligations or prohibitions, the sphere of permission changes to the subset of worlds which were already permitted but also satisfy the command. If we say that all worlds permitted before the command satisfy Q and we obligate A at our actual world @, then we take the intersection of the conditions of our original worlds and our new obligation: $\forall v[@Rv \Rightarrow v \models (Q \wedge A)]$. The new sphere of permission contains the worlds that were already permitted but also permit the new obligation.

It becomes a different story when we try to broaden the sphere of permission, by issuing the permission of A at our actual world @. Like in our example, where we gave the AI Saturday off. When we issue a permission of A, the sphere of permission must be extended to also include the worlds in which A is permitted. However, in all the currently permitted worlds, A is not permitted, only Q. Before the permission of A, the AI was not permitted to take Saturday off. So we can't take the intersection as with obligation and inhibition: $\forall v[@Rv \Rightarrow v \models (A \wedge Q)]$. Otherwise we would lose these original worlds, while we only want to permit new worlds. We would not want to lose the worlds in which the AI still works on Saturday. It is only a permission to take Saturday off, not an obligation . If the AI would want to continue working on Saturday this should still be possible. We could take the union: $\forall v[@Rv \Rightarrow v \models (A \vee Q)]$. However, we probably don't want to permit all worlds where A is true, because in some new worlds where A is true, the old obligations, inhibitions and permissions Q aren't satisfied. We don't want to permit worlds in which the AI can access the internet. It become ambiguous which worlds should be permitted and which not. We have to determine which obligations and inhibitions from Q are in conflict with our new permission A and which we need to keep, as they are still relevant.[6]

So, whenever we try to broaden the sphere of permission by issuing a per-

6. Lewis 1979, 27.

mission, it becomes unclear which commands and permissions are still relevant and which are in conflict. This is because, in this world-based approach, we can't determine if a permission is allowed, based on its truth-conditions. We need a different approach of differentiating between what is relevant and what is in conflict.

2.4 Overview of solutions

I will now give an overview of three suggested approaches to solving the *problem of permission*. The first solution only permits worlds that are most similar to the already permitted worlds. The second solution ranks worlds on their moral superiority. For these first two, I will not go into a deep critical assessment, because I will later show an overarching problem with these approaches. I will finish with the approach I will be working out further: exact truthmaker semantics.

2.4.1 Similarity approach

The first possible solution is concerned with similarity. Lewis also mentions this in his article on the *problem of permission*.[7] He suggests that when we issue a permission, the newly permitted worlds should be the worlds that are most similar to the worlds that were already permitted. In my example, this would mean that permitting the AI to access the internet, would be too dissimilar to the worlds that were already permitted: the worlds where the AI cannot. Not working of Friday is more similar to working everyday, than not working on Friday and Saturday.

While it feels evident that newly permitted worlds should have a lot in common with previously permitted worlds, as most permissions remain intact, Lewis himself already debunks this possible solution. When we follow the reasoning of similarity, shouldn't the AI be obliged to make Sudoku's, instead of working on his music career, because it is more alike to making calculations? This makes no sense. The AI should be free to spend his day off the way he pleases. It could also be that on his working days, he has to get out of bed at 7, but on an off day this shouldn't have to be the case. So, picking worlds that are most similar to the already permitted worlds is not a solution to the *problem of permission*.

2.4.2 Ranking approach

The second possible solution comes in the form of a ranking approach. This possible solution in which Rooij suggests and critiques in his article "Free Choice Counterfactual Donkeys",[8] is also very well summarized by Hansson.[9] The ranking approach entails we should rank possible worlds in a moral order. From

7. Lewis 1979, 28.
8. Robert Rooij. 2006. "Free Choice Counterfactual Donkeys." *Journal of Semantics - J SEMANT* 23 (July): 383–402.
9. Hansson 2013, 231.

this moral ranking, we should then only permit the morally best possible worlds. For this approach to be even considered, we need to make the assumption that a moral ranking can be made.

However, even with this assumption, this approach has its flaws, as Rooij points out.[10] When permitting only the morally best possible worlds, we have no guarantee that when permitting A, worlds with A are being added to our sphere of permission. This is the case when we say $P(A \vee B)$ but B is morally superior to A, then none of the best $A \vee B$ worlds will be A worlds and the only worlds that are permitted are B worlds. This way, issuing a new permission could mean you are not given new options, possible worlds, because this new permission is morally inferior to previously issued permissions. Issuing a permission statement should intuitively always add to your sphere of permission, as this is not the case with the ranking approach, it doesn't seem like a correct solution.

2.4.3 Exact truthmaker semantics

Kit Fine argues that solutions, as the ones I have just given, have no chance at solving the *problem of permission*, as he claims we cannot rely on possible worlds semantics for finding a solution. Kit Fine gives convincing arguments for this claim in his paper on permission and possible worlds.[11] Furthermore, while it is important to know why possible world semantics can't solve the *problem of permission*, I am mainly interested in how my suggestion could be a solution, not why other approaches cannot.

This is why I suggest we turn our heads to exact truthmaker semantics. Exact truthmaker sematics is state-based, in contrast to the world-based SDL semantics. I believe this state-based approach can provide a framework in which we can differentiate between relevant and conflicting permissions and obligations, when updating our sphere of permission. Not being obligated to make calculations all day is an exact truthmaker of having a free day. Henceforth, it is understood that the obligation of making calculations is lost when the obligation of working is lost. We also understand that obligation of not accessing the internet is kept, as it has nothing to do with having a free day. It is not an exact truthmaker of having a free day. Exact truthmakers of the newly issued permission can now overrule existing obligations or permission.

I believe the distinction between relevant and conflicting permissions could prove immensely important in updating our sphere of permission. To test if this approach will suffice, I will begin my next chapter by further explaining the exact truthmaking and how it used to construct a framework in which the *problem of permission* could be solved.

10. Rooij 2006, 386.
11. Kit Fine. 2014. "Permission and Possible Worlds." *Dialectica* 68 (3): 317–336.

3 Exact truthmaking and permission

Thus far, we have only looked at world-based semantics like SDL, for formalizing permission. In this paragraph I will give an intuitive explanation of a state-based truthmaker approach, the exact truthmaking relation and a formal semantics that describes this relation. Next, I describe how permission fits in the idea of exact truthmaker semantics. I will then give a formal model, in which permission is incorporated in the semantics of exact truthmaking.

3.1 Exact truthmaking

For explaining exact truthmaking I will be mainly relying on an article by Kit Fine.[12] In the first place, I will explain the informal idea behind exact truthmaking. In doing so, I will set it apart from possible world semantics, which we have seen in the former paragraph. Explaining the distincion between the two is crucial in understanding why truthmaker semantics could possibly solve the problem of permission, where possible world semantics could not. Afterwards, I will give a formalisation of exact truthmaking, which will be necessary for formalising exact truthmaking for permission later on.

3.1.1 The informal idea

In the previous paragraph we have looked at a possible world semantics to formalise the *problem of permission* in. Truthmaker semantics is a different kind of objectual truth-conditional semantics. Where with possible world semantics, we take worlds that satisfy the truth-conditions of statement, with truthmaker semantics we take the states that satisfy the truth-condition of a statement. States can be seen as parts of worlds, fact-like entities that make up worlds. States can be state of affairs, events or actions, as long as they can properly verify a statement. A statement can be a sentence, proposition or a thought, as long as it can be verified by truth-conditions. In the case of truthmaker semantics: states. States have a value that can exactly verify a statement A. When it does, that state is an exact verifiers of before mentioned statement A and part of the set of verifiers of A. We now take a set of states, a state space, as the truth-conditions on which to check a statement. [13]

For exact truthmaking, we look for states that exactly make the statement true. A state is an exact truthmaker when it necessitates the statements validity and is wholly relevant to it. In the same manner a state can be an exact falsemaker, when it necessitates a statements falsity and is wholly relevant to it.

For example, the sentence *Socrates is a philosopher* if the truth-condition of him being a philosopher is true. In possible worlds semantics the sentence would be true in the worlds where Socrates is a philosopher. In these worlds, other

12. Kit Fine. 2017b. “Truthmaker Semantics.” Chap. 22 in *A Companion to the Philosophy of Language,* 556–577. John Wiley Sons, Ltd. ISBN: 9781118972090.
13. Fine 2017b, 560.

statements could be true as well, like the sentence *Socrates likes to golf.* For truthmaker semantics we look at the state space in which the sentence is true. The states that verify the sentence. For exact truthmaker semantics we look for states that exactly verify our sentence. Socrates being a philosopher, exactly verifies our sentence. However, Socrates being a philosopher and him liking golf only inexactly verifies our sentence. The statement *Socrates is a philosopher and Socrates likes golf* does necessitate him being a philosopher. However him liking golf isn't wholly relevant to him being a philosopher.

3.1.2 Exact truthmaking formalised

We may take a state space to be an ordered pair (S, R), where S (states) is a non-empty set and R (part) a binary relation on S. We assume that the relation R is partial ordered, so that conforms to reflexity, anti-symmetry and transitivity. Formalised for states s,t and u of S as such:

Reflexivity: sRs
Anti-symmetry: sRt and tRs implies $s = t$
Transitivity: sRt and tRu implies sRu

For our partial ordered relation, we require completeness on our state space: every subset of states should have a least upper bound. Given subset T of S, s is an upper bound if every $t \in T$ is a part of s. It is also the least upper bound if it is included in every upper bound of t. More formally: given subset $T \subseteq S$, it is an upper bound if $t \sqsubseteq s$ for every $t \in T$. It is also the least upper bound when $s \sqsubseteq s'$ for every upper bound s' of T.[14]

In our state space there are also fusions of states. These occur because states can be a part of other states. This is called parthood. When multiple states are a part of another state, we must take a fusion of these states. When we take a fusion of multiple states, we combine the value of these states. So, when we say $s = t \sqcup u$, we mean that s has as its value the conjunction of both the value of state t and of state u. An example could be that state t is *loving flowers* and state u is *hating candles.* State s would then be *loving flowers and hating candles. Loving flowers* and *hating candles* is not in conflict with each other, so in this instance a fusion is possible. It results in a possible state. [15]

To really understand fusion, it would be good to visualize a state space. This way, it will be easy to explain parthood of states and how a fusion of the parts results in the value of the combined state. Below two Hasse diagrams[16] are drawn representing state spaces.

14. Ibidem.
15. Ibidem.
16. A Hasse diagram is a type of mathematical diagram which represents a finite partially ordered set by drawing its transitive reduction.

$$\begin{tikzcd}
s_3 & & & & \\
s_2 \arrow[u] & & & s_3 & \\
s_1 \arrow[u] & & s_1 \arrow[ur] & & s_2 \arrow[ul] \\
& X & p & & q & Y
\end{tikzcd}$$

In state space X on the left, we see three states: s_1, s_2 and s_3. The arrows between these three states represent parthood. State s_1 is part of state s_2 and state s_2 is part of state s_3. In state space Y, we can see an example of multiple states being part of another state. In this state space state s_1 and state s_2 are both part of state s_3. As a result, state s_3 is the fusion of states s_1 and s_2. The fusion is different from its individual parts. State s_1 has the value p, state s_2 has the value q. Following the rules of fusion, the values of s_1 and s_2 are combined to form the value of s_3. In this case p and q are combined to form $p \wedge q$. A formal way of denoting this is: $s_3 = s_1 \sqcup s_2$.

In our example state space, we see that our recursive clauses for propositional operators now need to take the form of states, not worlds. Where in possible world semantics the recursive clauses for the propostitional operators negation, conjunction and disjunction were:

$$\begin{aligned}
&w \models \neg A \text{ iff } w \not\models A \\
&w \models A \wedge B \text{ iff } w \models A \text{ and } w \models B \\
&w \models A \vee B \text{ iff } w \models A \text{ or } w \models B
\end{aligned}$$

For truthmaker semantics these change. As we are now not interested in when a statement is true at a given world, but which states necessitates the statement and is wholly relevant to it. We now take a (state) model M to be an ordered triple (S, R, ν), where (S, R) is a state space. Our valuation function ν gives for each propositional statement p the verifiers and falsifiers as the pair (V, F), both subsets of S. We get the set $|p|^+ = V$ as the verifiers of p, defined as: $|p|^+ = \{s|s \Vdash p\}$, and the set $|p|^- = F$ of the falsifiers of p, defined as: $|p|^- = \{s|s \dashv\mid p\}$.

This results in the following: given a model $M = (S, R, \nu)$, we can now determine the clauses that define an arbitrary formula A to be verified ($s \Vdash A$) or falsified ($s \dashv\!| A$) by a state s:[17]

> $s \Vdash p$ iff $s \in |p|^+$;
> $s \dashv\!| p$ iff $s \in |p|^-$;
> $s \Vdash \neg B$ iff $s \dashv\!| B$;
> $s \dashv\!| \neg B$ iff $s \Vdash B$;
> $s \Vdash B \wedge C$ iff for some states t and $u, t \Vdash B, u \Vdash C,$ and $s = t \sqcup u$;
> $s \dashv\!| B \wedge C$ iff $s \dashv\!| B$ or $s \dashv\!| C$;
> $s \Vdash B \vee C$ iff $s \Vdash B$ or $s \Vdash C$;
> $s \dashv\!| B \vee C$ if for some t and $u, t \dashv\!| B, u \dashv\!| C,$ and $s = t \sqcup u$.

We can extend this basic semantics by adding additional conditions on the model and changing clauses. In his paper Angellic content[18], Kit Fine differentiates three types of content, on which a semantics can be made. There is exact content, complete content and replete content. Exact content is in line with the basic semantics I described above, where a state s should verify a formula A under non-inclusive semantics. Complete and replete expand on this. I will first explain what both entail and their accompanying semantics. Afterwards I will explain why each of these semantics would be functional for my application. It doesn't matter which I choose. That is why I will stick with the basic semantics, as explained above.

I will start with inclusive semantics. In inclusive semantics our model is closed under fusion for all propositions p. So that given a propositional formula A, the complete content is the complete closure of its exact content. Defined as: $\{s \in S \mid s \Vdash$ A under the inclusive semantics$\}$. In our definition of complete content we state that s should verify A under inclusive semantics. This inclusive semantics looks as follows.

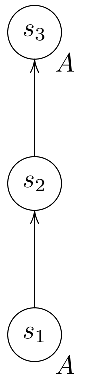
$$s \dashv\!| B \wedge C \text{ iff } s \dashv\!| B \text{ or } s \dashv\!| C \text{ or } s \dashv\!| B \vee C$$
$$s \Vdash B \vee C \text{ iff } s \Vdash B \text{ or } s \Vdash C \text{ or } s \Vdash B \wedge C$$

A state $s \in S$ now falsifies the conjuncion $B \wedge C$ if and only if s falsifies B, C or the disjunction of B and C: $B \vee C$. Furthermore, a state $s \in S$ now verifies the disjunction $B \vee C$ if and only if it verifies B, C or the conjunction of both: $B \wedge C$. These clauses add to our framework that a verifier of $B \wedge C$ should also be a verifier for $B \vee C$ and that a falsifier of $B \vee C$ should also be a falsifier for $B \wedge C$.[19]

When we take the non-vacuous and convex closure of our just defined complete content, we get replete content. In non-vacuous models, every sentence letter p has a truthmaker and a falsemaker. Convex closure is best explained

17. Fine 2017b, 563.
18. Kit Fine. 2016. "Angellic content." Framework, relevance logic, *Journal of Philosophical Logic* 45 (2): 199–226, 208.
19. Fine 2017b, 563.

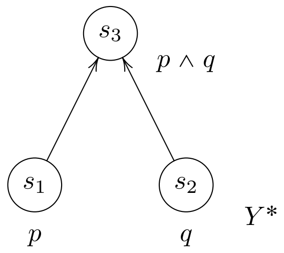
with an example:



When state s_1 and s_3 both verify A, for convex closure all states in this ladder of parthood should all have the same value, in this case state s_2 should also verify A to establish convex closure. [20]

However, for my argument it doesn't matter that much which semantics I use. I try to see if exact truthmaking in general has the potential to solve Lewis's problem if permission. For being able to see if exact truthmaking , the specific framework chosen in exact truthmaking will not make that big of a difference. When it becomes clear exact truthmaking can really solve the problem of permission, we can start to extend the framework and see which best captures our notion of permission. In the continuation of my paper, I will be using the exact semantics without the complete and replete additions.

Now that we have established a semantics for exact truthmaking, which we will be using to try to solve the problem of permission, it would be good to take a look at how this semantics functions in practice. Therefore, we return to our example state space Y, which is now completely filled in as $Y*$:



With this state space, we can try to better understand the effect of our exact truthmaking clauses. In state space $Y*$, we see a state $s_1 \models p$ and state $s_2 \models q$. It follows form their fusion that $s_3 = s_1 \sqcup s_2 \Vdash (p \wedge q)$. This is the

20. Kit Fine and Mark Jago. 2019. "Logic for exact entailment." Application, philosophical logic, exact truthmaking, logic of exact entailment, *The Review of Symbolic Logic* 12 (3): 536–556, 542.

case, because s_1 is an exact verifier of p: $s \Vdash p$. Which means that $s_1 \in |p|^+$; s_1 is in the set of all states that exactly verify p. In the same manner, s_2 is an exact verifier of q: $s_2 \Vdash q$. When taking the fusion of states s_1 and s_2, denoted as $s_1 \sqcup s_2$, we get a new state which combines these two exactly verifying states into a new state. This new state, in this case s_3, exactly verifies $p \wedge q$: $s_3 \Vdash p \wedge q$. However, this new state does not verify p or q independently. For in exact truthmaking every part of a state must be wholly relevant, following the definition mentioned in 3.1.1. State s_3 is the combination of its parts s_1 and s_2, with values p and q, and both need to be relevant in verifying a statement. For both p and q to be relevant, only the conjunction of p and q can be verified, not p or q independently. For verifying q, p is not relevant and the other way around. This means that this new set can only verify the conjunction of p and q. To recap: while $p \wedge q$ does necessitate p or q independently, it is not wholly relevant. Because of this, $p \wedge q$ does not exactly verify p or q. In our state space Y^*, this means that $s_3 \Vdash p \wedge q$, $s_3 \nVdash p$ and $s_3 \nVdash q$.

Returning to our example concerning Socrates, it would be as follows: if state s_1 is *Socrates being a philosopher* and state s_2 is *Socrates liking golf*, then state s_3 would be *Socrates being a philosopher and Socrates liking golf.* Now it is easy to see that for the state *Socrates being a philosopher*, it is not the case that both parts of the conjunction *Socrates being a philosopher and Socrates liking golf* are wholly relevant. It does necessitate him being a philosopher, but it also implies him liking golf, which is not relevant to him being a philosopher.

3.2 Permission

Now that we have a better sense of what exact truthmaker entails, we can start to combine exact truthmaking with the concept of permission. In this part, I wil explain informally how these two could be combined. After which, I will give a formalisation of how permission could work in exact truthmaker semantics.

3.2.1 Permission and exact truthmaking

When issuing a new permission, we have to update our existing permissions and obligations to be in harmony with the new permission. To do so, we need to know which other permissions necessitate and are wholly relevant to this new permission, because these could be in conflict with already existing permissions and obligations. With exact truthmaker semantics we could give a formalisation of these exact truthmakers and use these to make sure no implicit conflicts occur. To bring back our example: when our AI is told it does not have to work on Saturday, a conflict could arise with its obligation to make calculations all day. However, an exact truthmaker of not working on Saturday, is not being obligated to make calculations all day. So, we could remove its existing obligation - to make calculations all day - because it is in conflict with an exact truthmaker of our new permission: being free on Saturday.

3.2.2 Permission formalised in exact truthmaker semantics

At the end on his paper on possible world semantics, Kit Fine suggests a formalisation of permission; he writes his positive proposal. He suggests that A is permitted if and only if all of the verifiers of A are permitted.[21] To distinguish permitting of states and permitting of statements, I would like to call the permitted states the set of admissible states. Now, A is permitted if its exact verifiers are a subset of the admissible states.[22]

For a formalisation, we take a set of states S, from which we take a subset to be admissible: $(S, <, Adm)$. Now, we can define when statement A is permitted. This should be the case when all states that exactly verify A, formulated as $|A|^+$, are a part of the set of admissible states. For our formalisation of this definition of permission, we define language L_{base}, which is very similar to our SDL language:

$$A := a \mid \neg A \mid A \wedge A \mid A \vee A \mid A \rightarrow A$$

From language L_{base}, we adapt a language we will call L_p. This language looks as follows:

$$B := PA \mid \neg B \mid B \wedge B \mid B \vee B \mid B \rightarrow B$$

Using this new language, we can formulate the recursive clauses for permission, where also PA itself is firstly defined:

$$M \models PA \text{ iff } |A|^+ \subseteq \text{ Adm}$$

$$M \models \neg B \text{ iff } M \not\models B$$
$$M \models B \wedge C \text{ iff } M \models B \text{ and } M \models C$$
$$M \models B \vee C \text{ iff } M \models B \text{ or } M \models C$$
$$M \models B \rightarrow C \text{ iff } M \models \neg B \text{ or } M \models C$$

In my example concerning the AI, it would look as follows: the AI is permitted to not work on Saturday. We look at the exact verifiers of this, like not having to make calculations all day and not working on Saturday. This clashes with already admissible states where the AI needs to work on Saturday and is not permitted to not make calculations all day. So these states are removed and the states where it is not obligated to do both is added.

If *not working on Saturday* is $\neg A$ and *not making calculations* is $\neg B$, then we start with $\neg P(\neg A)$ and $\neg P(\neg B)$, both A and B are obligated, because there are no admissible states that exactly verify $\neg A$ or $\neg B$.

In their paper "Truthmakers and Normative Conflicts", Anglberger and Korbmacher suggest a definition for obligation in terms of truthmakers. They suggest that A is obligated if and only if there is no admissible state that is a falsemaker of A. This is also the case in our example. There are no verifiers

21. Fine 2014, 334-336.

22. J Korbmacher, Albert Anglberger, and Federico LG Faroldi. 2016. "An exact truthmaker semantics for permission and obligation." *Deontic logic and normative systems,* 16–31, 2.

of $\neg A$ or $\neg B$, so no falsemakers of the contrary: $\neg P(\neg A)$ and $\neg P(\neg B)$. This results in an obligation for $\neg P(\neg A)$ and $\neg P(\neg B)$.[23]

Right now, the conjunction of the verifiers of $\neg A$ and the admissible states is the empty set: $|\neg A|^+ \cap Adm = \varnothing$. When we issue $P(\neg A)$, we look at which states exactly verify $\neg A$, defined as $|\neg A|^+$. We can now add these states to our set of admissible states. This results in a new set of admissible states $Adm{*} = Adm \cup |\neg A|^+$. As a result, in a model $M{*}$ using $Adm{*}$ it should hold that $M{*} \vdash P\neg A$. The restriction on $\neg A$ is tackled, now for $\neg B$: for a state to be an exact truthmaker of $\neg A$, it should also be permitted to $\neg B$, when the AI chooses to not work. If we take $\neg A$ to be build from the states $\neg p$ and $\neg q$ or $\neg q$, not working and not making calculations respectively.

Formula A is $\neg(\neg p \wedge (p \rightarrow q))$, then our new permission $\neg A$ is the formula $(\neg p \wedge (p \rightarrow q))$. The states that exactly verify A, $|A|^+$, are states that consist of both $p \wedge q$. The states that verify $\neg A$, $|\neg A|^+$, are states with $\neg p \wedge \neg q$ or states with $\neg p \wedge q$. When adding $|\neg A|^+$ to the admissible set, these new states make it permitted to not work and the option of not making calculations when not working.

As a result, it is permitted to $\neg B$ or B when $\neg A$. When the AI chooses not to work, it does not have to make calculations, but it is still allowed to. When the AI does work, it must still make calculations. When the AI chooses A, then B is still obligated. We get the states that are for example $\neg p \wedge \neg q$, or $\neg p \wedge q$, these are not present in $|A|^+$, which we have added to the admissible states.

However, how does accessing the internet fit into this? This is not part of the verifying states of $\neg A$, because it is not wholly relevant to $\neg A$. Following, the states where the AI accesses the internet are not a part of $|\neg A|^+$. This way, states where the AI can access the internet, are not added to our set of admissible states, after issuing the permission. As a result, accessing the internet is still not permitted for the AI. Exactly as we wanted. In utilising exact truthmaking, we have made a distinction between the obligations of making calculations and not accessing the internet. I will do a further evaluation of this in my next paragraph. For now, it is important to understand the manner in which this distinction is made.

4 Evaluation

In my fourth paragraph, I will be evaluating if we can use the truth-conditions of permission statements in exact truthmaker semantics to figure out if they are allowed. Utilizing exact truthmaking. I will do this by comparing the old SDL approach to our new exact truthmaking approach. Furthermore, I will be contemplating in what way we can determine permission for worlds, as combinations of states. In line with my established definitions of permission and obligation and my semantics.

23. Albert Anglberger and Johannes Korbmacher. 2020. "Truthmakers and Normative Conflicts." Framework, *Studia Logica* 108 (1): 49–83, 51.

4.1 Truth-conditions of permission statements

Firstly, can we use the truth-conditions of permission statements in exact truthmaker semantics to figure out if they are allowed? For this, I have looked at exact truthmaking which utilizes states , which can verify or falsify a statement. When using exact truthmaking, we want states to necessitate and be wholly relevant to a statement. A statement is then permitted if all its exact truthmaking states are part of the set of admissible states. We must take all necessary and relevant states into account when changing our set of admissible states. This way, we update what is permitted or obligated, so that this is in harmony with our newly issued permission statement.

Lets look back at our problematic example in SDL and compare it to our exact truthmaking implementation: in my SDL example, it was the case that by issuing the permission of A at our actual world @. The sphere of permission should also be extended to include the worlds in which A is permitted. In all the current worlds, only Q was permitted, not A. So, we couldn't take the intersection: $\forall v[@Rv \Rightarrow v \models (A \wedge Q)]$. We would lose the original worlds, while we only wanted to permit new worlds. We also tried taking the union: $\forall v[@Rv \Rightarrow v \models (A \vee Q)]$. This resulted in permitting all worlds where A was true. This was also incorrect, because in some new worlds where A was true the old obligations, inhibitions and permissions Q weren't satisfied.

Neither of our attempts resulted in the intuitive result: when permitting the AI to take Saturday off, the AI should be permitted to take Saturday off and the AI should not be obligated to make calculations all day Saturday. However, it should still not be permitted to access the internet.

As we could not achieve the wanted result with SDL. We wanted a different solution to determine which obligations and inhibitions from Q were in conflict with our new permission A, so we could correctly update our permissions and obligations. As a possible solution we explored defining permission in exact truthmaker semantics. It seems from my example that this method works as desired. When permitting taking a day off, the AI is also permitted to choose if it wants to make calculations or not. It is no longer obligated to making calculations. Accessing the internet however, is still forbidden, as we would like. This might seem like a simple example, but being able to make this distinction between necessitating and wholly relevant states and irrelevant states is new, in contrast to SDL. The exact way it is implemented, the semantics, can change, but the concept is crucial in updating permissions and obligation.

4.2 World construction

From my evaluation, it seems that in an exact truthmaking framework, we can look at the truth-condition of a permission statement to see if it is allowed or not. By checking if its exact verifiers are in the admissible set. However, we now just have a set of states, but these states are only facts in a scenario. Parts of a world. We need to again assemble worlds from these states. When doing so, our method of permission need to be kept. I believe the permitting of worlds should be done as follows:

A world is permitted iff all its member states are part of the admissible set.

A state is either in the admissible set, then it is permitted, or it is not in the admissible set, then it is not permitted. In a world, no contradictory states are present. For each of the states that are present in a world, they are either in the admissible set or not. From this it follows that we can check for each states of a world, if it is present in the admissible set and so if the state is permitted. For a world to be permitted, it needs to be in line with the permissions that result from our set of admissible states. Only a world that fully consists of permitted states, is a permitted world.

To illustrate the idea behind permitting worlds, I will be using a simple example. This simple model will be ignoring more complex problems, as Fine has describes in his "A theory of truthmaker content I", where gives his definition for world-states and the boolean operators.[24] Another paper of his, "Compliance and command II", also gives a more complex framework for truthmaking in deontic logic and deontic statements as permission and obligation. [25] However, my way of permitting worlds could easily be implemented in his more complex framework. For my purpose of explaining my suggestion of world permitting, an easier example is better to understand.

For my example, I take an admissible set containing the following states:

s_1: Socrates loving the color yellow.
s_2: Socrates going for a bicycle ride.
s_3: Socrates not going for a bicycle ride.
s_4: Socrates not loving strawberries.
s_5: Socrates loving strawberries.

In this example, Socrates is permitted to go on a bike ride or not and he is permitted to love or hate strawberries. He is however obligated to love the color yellow. For every combination of states that we check, it has to contain Socrates loving the color yellow. Furthermore, we have the state s_6: Socrates not loving the color yellow, as the opposite for s_1.

For worlds it would look as follows: we have four permitted worlds: w_1, w_2, w_3, w_4, which are as follows:

24. Kit Fine. 2017a. "A theory of truthmaker content I: Conjunction, disjunction and negation." Framework, philosphical logic, metaphysics, exact truthmaking, inexact truthmaking, loose truthmaking, *Journal of Philosophical Logic* 46 (6): 625–674, 630-633.

25. Kit Fine. 2018. "Compliance and command II, imperatives and deontics." *The Review of Symbolic Logic* 11 (4): 634–664, 639-645.

$$\begin{aligned} w_1 &= \{s_1, s_2, s_4\} \\ w_2 &= \{s_1, s_2, s_5\} \\ w_3 &= \{s_1, s_3, s_4\} \\ w_4 &= \{s_1, s_3, s_5\} \end{aligned}$$

These four worlds are all permitted, as they only consist of worlds present in the set of admissible states. We could also get four not permitted worlds: w_5, w_6, w_7, w_8, as follows:

$$\begin{aligned} w_1 &= \{s_2, s_4, s_6\} \\ w_2 &= \{s_2, s_5, s_6\} \\ w_3 &= \{s_3, s_4, s_6\} \\ w_4 &= \{s_3, s_5, s_6\} \end{aligned}$$

Following our definition of the permission of worlds, these worlds would not be permitted, as they all contain the state s_6, which is not in the set of admissible states.

Our admissible set changes as a consequence of issued permissions. Changes to the admissible set, change which worlds are permitted. If all states need to be part of the admissible set, also states that are fusions of other states, because these are also part of the world. Fusions create new states than are different from their parts and therefore also have to be part of the admissible set.

As my example shows, we now have a functional manner in which we can permit worlds, combined with the benefits of state-based exact truthmaking. We can now permit worlds, as we tried to achieve with SDL, but with the ability of updating our existing permissions and obligations, so that they are in harmony with a new permission.

5 Conclusion

In my thesis, I have shown why an implementation of permission in SDL result in a problem of permission. Following, I have provided an alternative semantics in the form of exact truthmaker semantics. In this framework, it is possible to determine if a permission statement is allowed, based on its truth-conditions. This is in contrast with the SDL, world-based approach, in which this was not possible. Using the exact truthmaking framework, I provided a way in which I believe permission should be implemented for worlds, where worlds are build from exact truthmaking states. I suggest that a world should be permitted when it only consists of states that are present in the set of admissible states.

There are also potential problems I have not already adressed, like the truth-conditions of a permission statement of a permission statement: PPA. Another problem could be how we could determine the truth-conditions of a conjunction of a formula A and a permission statement PA.

Furthermore, in my thesis I have worked with the basic semantics for exact truthmaking. There could be more exploration of inclusive and replete semantics or something completely new, to find which semantics best fits our notion of permission.

In my thesis I have also assumed a closed notion of permission, it could also be thinkable that something is both not permitted and not forbidden. This could have its effects on how we handle permission, in comparison as to how I have approached permission.

In further research, there could also be worked on a formal procedure for my exact truthmaking framework, a procedure that takes a model and a permission and return the updated model. A proper deontic logic for the updating operators.

Bibliography

Anglberger, Albert, and Johannes Korbmacher. 2020. "Truthmakers and Normative Conflicts." Framework, *Studia Logica* 108 (1): 49–83.

Anglberger, Albert JJ. 2008. "Dynamic deontic logic and its paradoxes." *Studia Logica* 89 (3): 427–435.

Fine, Kit. 2014. "Permission and Possible Worlds." *Dialectica* 68 (3): 317–336.

Fine, Kit. 2016. "Angellic content." Framework, relevance logic, *Journal of Philosophical Logic* 45 (2): 199–226.

Fine, Kit. 2017a. "A theory of truthmaker content I: Conjunction, disjunction and negation." Framework, philosphical logic, metaphysics, exact truthmaking, inexact truthmaking, loose truthmaking, *Journal of Philosophical Logic* 46 (6): 625–674.

Fine, Kit. 2017b. "Truthmaker Semantics." Chap. 22 in *A Companion to the Philosophy of Language,* 556–577. John Wiley Sons, Ltd. ISBN: 9781118972090.

Fine, Kit. 2018. "Compliance and command II, imperatives and deontics." *The Review of Symbolic Logic* 11 (4): 634–664.

Fine, Kit, and Mark Jago. 2019. "Logic for exact entailment." Application, philosophical logic, exact truthmaking, logic of exact entailment, *The Review of Symbolic Logic* 12 (3): 536–556.

Hansson, Sven Ove. 2013. "The Varieties of Permission." In *Handbook of Deontic Logic and Normative Systems,* edited by Dov M. Gabbay, John Horty, Xavier Parent, Ron van der Meyden, and Leendert van der Torre, 195–240. College Publications. ISBN: 978-1-84890-132-2.

Korbmacher, J, Albert Anglberger, and Federico LG Faroldi. 2016. "An exact truthmaker semantics for permission and obligation." *Deontic logic and normative systems,* 16–31.

Lewis, David. 1979. "A Problem About Permission." In *Essays in Honour of Jaakko Hintikka: On the Occasion of His Fiftieth Birthday on January 12, 1979,* edited by Esa Saarinen, Risto Hilpinen, Ilkka Niiniluoto, and Merrill Provence Hintikka, 163–175. Dordrecht: Springer Netherlands. ISBN: 978-94-009-9860-5.

McNamara, Paul, and Frederik Van De Putte. 2021. "Deontic Logic." In *The Stanford Encyclopedia of Philosophy,* Spring 2021, edited by Edward N. Zalta. Metaphysics Research Lab, Stanford University.

Rooij, Robert. 2006. "Free Choice Counterfactual Donkeys." *Journal of Semantics - J SEMANT* 23 (July): 383–402.