

File ID 112926
Filename Chapter 5 Update semantics framework

SOURCE (OR PART OF THE FOLLOWING SOURCE):

Type Dissertation
Title Modality in typological perspective
Author F.D. Nauze
Faculty Faculty of Humanities
Year 2008
Pages x, 236
ISBN 978-90-9023343-7

FULL BIBLIOGRAPHIC DETAILS:

<http://dare.uva.nl/record/277914>

Copyright

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use.

Chapter 5

Update semantics framework

In this chapter I will develop a semantics of modality tailored to address the problem of combinations of modal items. As we have seen in the previous chapters, the interpretation of these combinations follows a certain scope order. Furthermore, this scope order is not easily accounted for from the perspective of the standard framework. The (presumably) universal nature of the scope order is the main incentive to depart from the standard framework and develop a new system. This new framework will provide a toy example of the mechanisms at hand. Obviously I do not aim for complete coverage of all the diversity found in the data chapter as well as most of the subtle nuances of meaning that are inherent to modal expressions. I will nevertheless try to hint at possible improvements whenever possible.

First, let us remember the typology of modality we begun with:

| Participant-internal | Participant-external | | Epistemic |
|----------------------|----------------------|---------------|-------------|
| | Deontic | Goal-oriented | |
| Ability | Permission | Possibility | Possibility |
| Needs | Obligation | Necessity | Necessity |

With respect to this typology, the strength of the standard framework is to offer a uniform analysis of modality. All three kinds of modality rely on the same basic interpretation. However, this strength is also its weakness when it comes to combinations of modals as we have seen that there is no direct explanation for the scope order restriction when we posit a uniform framework.

The framework I will present is in the line of update systems of (Veltman 1996). The main difference between an update framework and a truth-conditional one can be first grasped by a comparison of the slogans. The slogan of the truth-conditional semantics of the previous chapter was,

“you know the meaning of a sentence if you know the conditions under which it is true”

whereas the slogan of update semantics frameworks is,

“you know the meaning of a sentence if you know the change it brings about in the information state of anyone who accepts the news conveyed by it.”

The crucial departure from the traditional framework is that we now model the (change of) information of an (idealized) agent. That is, the update system models the effect of sentences/propositions on the information of an agent. To define an update system we need three ingredients: a language, a set of information states and an update function.

Definition 5.0.2 (Update system (Veltman 1996)). An update system is a triple $\langle L, \Sigma, [\] \rangle$ with L a language, Σ a set of information states and $[\] : L \rightarrow (\Sigma \rightarrow \Sigma)$ a function that assigns to each sentence φ an operation $[\varphi]$ from states to states.

I will first describe a simple framework for epistemic and deontic modality. This framework is simple enough to make clear what the basic idea of the system is.

5.1 Epistemic and deontic modality

This system is only meant to describe a first semantics of epistemic modality (namely epistemic *might*) and the deontic modals *may* and *must* within an update framework. The system will be propositional, like the truth-conditional framework presented in chapter 3. This is of course an idealization but it is enough to account for the main intuitions.

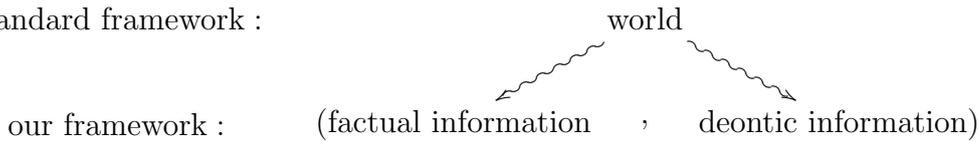
The definition of an update system is quite straightforward: the first step is to define the syntax of its (formal) language. That is in line with the traditional methodology in formal linguistics that consists in giving a precise formulation of the syntax of the language studied, then its semantics and finally turning to pragmatic considerations.

However, this methodology is not tenable in our current situation! The main intuition is that the property to be accounted for (the scope hierarchy) is intimately intertwined with the architecture of our system, and in particular with the architecture of the information states. As such, the prediction would actually be that the structure of the system and the semantics of modal expressions restrict the range of “felicitous sentences” with modal combinations. That is, predictions about syntax (or grammaticality) would actually be the end product of the analysis. It is therefore problematic and, in this setup, counterintuitive to begin with a precise syntactic definition. Formally, the syntax of the language will not be specified, although its ingredients can be explicitly given. Summing up, we will not define the notion of grammatical or syntactic well-formedness but will consider all possible strings of symbols alike. Which strings of symbols are interpretable will then be decided by the meaning of those symbols.

Definition 5.1.1 (Ingredients of the language). \mathcal{D} is a set of atomic declarative sentences. We also have the usual boolean connectives: negation $\neg\varphi$, conjunction $\varphi \wedge \psi$ and disjunction $\varphi \vee \psi$ as well as the conditional ‘if φ, ψ ’, the epistemic might φ and the deontic may φ and must φ .

As I just mentioned the crucial step in the definition of the update system is made in the design of the information states. We have seen that the main problem of the standard framework is that epistemic and deontic modals are defined in a unified fashion. This is not only problematic for an account of combinations of modal items but also, for instance, in the case of deontic conditionals. The problem as we characterized it was that deontic information should not depend on accessible worlds but should be an integral part (though of a different nature) of any world/situation. One way to implement this idea is to say that the notion of a world is too coarse for the task at hand and that under the notion of a “world” falls two distinct types of information: factual information and deontic information.

standard framework :



That kind of move has already been illustrated for imperatives in (Portner 2003) and (Mastop 2005) through the use of To-Do lists. Intuitively, the idea is that deontic modals operates on a different level of information. The basic unit is not a world anymore but a possibility, that is, a pair consisting of a *situation* and a *deontic plan*.

Definition 5.1.2 (Possibilities). A possibility is a pair (s, δ) consisting of a situation s and a non-empty deontic plan δ .

I will first explain the difference between a *world* and a *situation*. Intuitively, a world is specified by giving the totality of facts that are true in it, therefore if we take a set of basic sentences as primitive, a world is a total characteristic function of this set, i.e. a function $w : \mathcal{D} \rightarrow \{0, 1\}$. Therefore a world decides every possible declarative sentence. Obviously, this is a very strong commitment if a world is supposed to represent the information (or shared information) available for an agent (a set of agents respectively). Here, the change in terminology is also a change in practice. A situation is not meant to decide every sentence but only to represent the part of the world an agent is aware of in some circumstances. Therefore formally, a situation is a set of pairs consisting of an atomic declarative and a “truth value.”

Definition 5.1.3 (Situations). A situation is a subset s of $\mathcal{D} \times \{1, 0\}$.¹

¹As can be expected, 1 stands for true and 0 for false.

Notice that situations are not partial functions but sets of pairs. That is, a situation can contain one and the same atomic declarative in two different pairs, i.e. as true and false. Furthermore, a situation need not be informed about all possible atomic declaratives, but can represent just those declarative sentences the holder of the information state is aware off.

Deontic plans will serve a similar purpose as to-do lists in (Mastop 2005).² That is, they are the recipient of deontic information. However, to-do lists in the work of the previously cited authors are basically simple sets (of properties and of atomic imperatives respectively). This works fine as long as the intention is to model imperatives only. But this is not going to work easily now that we have two kinds of deontic statements possible, i.e. obligations and permissions. Both kinds of statements should be easily read off the deontic plan associated with a possibility. To solve this problem, it is enough to model the deontic plans as sets of to-do lists. In (Portner 2003) and (Portner 2007), to-do lists are sets of properties (bound to an agent). As we restrict ourselves to a kind of propositional framework, it is simple enough to let to-do lists be of the same form as situations.

Definition 5.1.4 (Deontic plans and to-do lists). A to-do list is a subset of $\mathcal{D} \times \{1, 0\}$. A deontic plan is a set δ of to-do lists.

Notice that there is still a major difference between Portner's definition of a to-do list and deontic plans. As I just mentioned, Portner's to-do lists are bound to an agent which, in our framework, wouldn't necessarily be the same as the one of the information state at hand. Therefore, the deontic plan as it is defined in 5.1.4 actually represents the known deontic information about possibly different agents. It is as such much more similar to the standard framework. However, the present framework can be extended without much trouble³ to deal with agents in the same fashion as the framework of (Portner 2007) (by binding deontic plans to agents). As it arguably does not influence the problem of combinations of modals, for the sake of simplicity I will not distinguish between agents' deontic plans.

Deontic plans are thus the recipients of deontic information and are made of to-do lists. The deontic plans are best seen as guidelines for actions (or at this stage of the formalization as guidelines for states of affairs to be reached). In order to be morally good, the agent must fulfill some part of its deontic plan. This means that he must at least fulfill one of the to-do lists in his deontic plan.⁴

Example 5.1.5 (Deontic plans). The following sets are examples of deontic plans. I will also use sometimes a more visual representation of to-do lists based on the

²The framework developed in (Mastop 2005) was partly inspired by the notion of to-do list of (Portner 2003) and unpublished work of Frank Veltman. There is however a difference in implementation as the last framework is a static truth-conditional framework.

³Mainly at the expense of more complicated definitions.

⁴I abstract here from the fact that there might be uncertainty about deontic plans too.

idea that a to-do list contains things you have to do $\langle d, \underline{1} \rangle$ and things you must not do $\langle d, \underline{0} \rangle$. This notation is most useful when a lot of sentences are present in the to-do list.

| | do's | don'ts |
|-------------|------|--------|
| to-do list: | ⋮ | ⋮ |
| | ⋮ | ⋮ |

1. $\{\{\langle a, 1 \rangle\}\} = \{ \boxed{a \quad \square} \}$ is a deontic plan with only one to-do list containing only one atomic declarative (positively): it represents an obligation to do a .
2. $\{\{\emptyset\}, \{\langle a, 1 \rangle\}\} = \{ \boxed{\quad \square}, \boxed{a \quad \square} \}$ contains two to-do lists: the empty one and $\{\langle a, 1 \rangle\}$. As the empty to-do list is part of the deontic plan this means that there are no obligations, i.e. doing nothing is morally acceptable/desirable. Furthermore, a is permitted.
3. In $\{\{\langle a, 1 \rangle, \langle b, 1 \rangle\}, \{\langle b, 1 \rangle, \langle c, 0 \rangle\}, \{\langle b, 1 \rangle\}\} = \{ \boxed{\begin{array}{c} a \\ b \end{array} \quad \square}, \boxed{b \quad c}, \boxed{b \quad \square} \}$, b is an obligation as it is present in all to-do lists. Furthermore, it is the only obligation and the other declaratives are thus permissions.

We have now defined possibilities and can finally turn to information states which are, as is standard in update semantics, sets of possibilities.

Definition 5.1.6 (Information states). An information state σ is a set of possibilities.

Before we turn to the definition of the update functions, I will give some examples of possibilities.

Example 5.1.7 (Possibilities and the minimal information state). Consider a simple language based on the set $\mathcal{D} = \{p, q, r\}$ of simple declarative sentences. We can single out some interesting “types” of possibilities (s, δ) :

1. The minimal possibility is $(\emptyset, \{\emptyset\})$ consisting of an empty situation \emptyset (there are no known facts) and of an empty deontic plan $\{\emptyset\}$ (there is no known deontic information), i.e. a plan having as only to-do list the empty to-do list \emptyset .
2. $(\{\langle p, 1 \rangle, \langle q, 1 \rangle, \langle r, 0 \rangle\}, \{\emptyset\})$ is a possibility with a situation with complete factual information and an empty deontic plan. The situation amounts to a world in the truth-conditional framework.
3. $(\{\langle q, 1 \rangle\}, \{\emptyset, \{\langle p, 1 \rangle\}\}), (\{\langle p, 0 \rangle, \langle r, 1 \rangle\}, \{\{\langle p, 1 \rangle, \langle r, 0 \rangle\}, \{\langle p, 1 \rangle\}\})$ are two possibilities with non-empty situations and deontic plans.

4. ($\{\langle q, 1 \rangle, \langle q, 0 \rangle, \langle r, 1 \rangle\}, \{\emptyset, \{\langle p, 1 \rangle\}\}$) is a possibility with a situation containing contradictory information (about q).
5. ($\{\langle p, 0 \rangle\}, \{\{\langle p, 1 \rangle\}, \{\langle p, 1 \rangle, \langle r, 0 \rangle\}, \{\langle p, 1 \rangle, \langle p, 0 \rangle\}\}$) is a possibility with a deontic plan containing a to-do list with contradictory information (about p).

Information states are thus sets of possibilities. A special information state is the minimal information state, that is, the information state with the least (consistent) information possible. The minimal information state $\mathbf{0}$ is the state containing only the minimal possibility, i.e. $\mathbf{0} = \{(\emptyset, \{\emptyset\})\}$.

Obviously, the last two possibilities of example 5.1.7 are special in a negative way. They both describe contradictory information. On the one hand, a situation that takes a particular sentence to be both true and false is an impossible situation. On the other hand, there is no intuitive rationale yet for deciding that a deontic plan with an impossible to-do list (remember that at this point in the formalization to-do lists and situations are the same kind of objects) is an impossible deontic plan, but this will fall out of the definition of updates with deontic sentences. The main concept at hand here is the consistency of a situation or to-do list.

Definition 5.1.8 (Consistency). A situation or to-do list s is consistent iff there is no $a \in \mathcal{D}$ such that $\langle a, 1 \rangle$ and $\langle a, 0 \rangle$ belong to s .

Example 5.1.9. Remember the list notation of to-do lists: the empty to-do list is $\boxed{\quad}\boxed{\quad}$. A to-do list that is a superset of $\boxed{a}\boxed{a}$ for some $a \in \mathcal{A}$ is not consistent.

Therefore, the last two possibilities of example 5.1.7 are such that the situation of the first is not consistent and that the deontic plan of the second contains a to-do list that is not consistent. The idea is that when an information state contains a possibility that is either not consistent because of its situation or because of a to-do list of its deontic plan, something has gone wrong with our information. In the first case, the factual information is not consistent while in the second, the deontic information is not consistent. Within the update semantics tradition, this means that there are two ways to be in an absurd information state.

Definition 5.1.10 (Absurd information state). The absurd state is a set of states denoted by \perp . It is defined as follows:

- $\emptyset \in \perp, \Lambda = \{\emptyset\} \in \perp,$
- for any information state $\sigma, \sigma \cup \Lambda \in \perp.$

The empty set characterizes failure through factual information, while Λ stands for failure through deontic information.⁵

⁵I will develop the notion of failure through deontic information later on when discussing the updates with deontic modals.

Although formally the absurd state is a set of states, I will for simplicity refer to it as an information state.

There are two differences between the update semantics framework of (Veltman 1996) and the one presented here. Firstly, factual information is partial. We moved from worlds to situations that just render some small set of facts. The consequence is that the system will not be eliminative but additive, and is best modeled with not one but two “update” functions: a positive update and a negative update.⁶ Secondly, the updates operate not only on information states but also on deontic plans as a result of the split in types of information. I will present the update functions with declarative sentences, the boolean connectives, the conditional and finally with the modal items after having introduced the key concept of acceptance.

Definition 5.1.11 (Acceptance).

- A sentence φ is accepted by the information state σ (write $\sigma \Vdash \varphi$) iff $\sigma \uparrow \varphi = \sigma$
- For φ and ψ two sentences, $\varphi \Vdash \psi$ iff for any information state σ , ψ is accepted by $\sigma \uparrow \varphi$.

Acceptance is what replaces the notion of entailment of the truth-conditional framework in update semantics. Basically, a sentence is accepted in an information state when it does not add any new information to it.

Definition 5.1.12 (Sentential update of information states). Take $p \in \mathcal{D}$, and σ an information state.⁷

$$\begin{aligned}\sigma \uparrow p &= \{(s \cup \{\langle p, 1 \rangle\}, \delta) \mid (s, \delta) \in \sigma \ \& \ s \cup \{\langle p, 1 \rangle\} \text{ consistent}\} \\ \sigma \downarrow p &= \{(s \cup \{\langle p, 0 \rangle\}, \delta) \mid (s, \delta) \in \sigma \ \& \ s \cup \{\langle p, 0 \rangle\} \text{ consistent}\}\end{aligned}$$

Therefore, updating one’s information state with a declarative sentence is adding positively the sentence consistently to the situations of the possibilities: p is the case in the resulting information state. Datedating is then a similar operation but adds the sentence negatively: p is not the case in the resulting information state. Finally, it is important to see that updating and datedating with atomic sentences does not bring any change in the deontic information.

Example 5.1.13. Consider a simple language based on the set $\mathcal{D} = \{p, q\}$ of simple declarative sentences; we have then the following:

1. $\mathbf{0} \uparrow p = \{(\{\langle p, 1 \rangle\}, \{\emptyset\})\}$; $\mathbf{0} \uparrow p = \mathbf{0} \uparrow p \uparrow p$.

⁶I will often use the bare term *update* to refer to *positive update* and the term *datedate* to refer to a *negative update*. It should however be clear to the reader that the term *datedate* is not meant to suggest a kind of revision.

⁷I will follow the traditional practice in update semantics and invert the correct notation by writing $\sigma \uparrow p$ instead of $[p]_{up}(\sigma)$ (and obviously $\sigma \downarrow p$ instead of $[p]_{down}(\sigma)$).

2. $(\mathbf{0} \downarrow q) \uparrow p = \{\{\langle p, 1 \rangle, \langle q, 0 \rangle\}, \{\emptyset\}\}$.
3. $((\mathbf{0} \downarrow q) \uparrow p) \uparrow q = \perp$ because the only possibility we obtain with the update, $\{\langle p, 1 \rangle, \langle q, 0 \rangle, \langle q, 1 \rangle\}$, is not consistent.⁸
4. $(\mathbf{0} \uparrow p) \downarrow p = (\mathbf{0} \downarrow p) \uparrow p = \perp$.

As this example makes clear, the sequential use of an update and a downdate with the same sentence leads to the absurd state. This is quite clearly the behavior that we expect for a sequential update of a sentence and its negation. That is, the crucial role of negation is to make a switch in update function. I will now introduce the up- and downdates for the boolean connectives. Notice that, in contrast to the update with atomic sentences (that only change the information of situations, i.e. about facts in the world), the up- and downdates with the boolean connectives are defined on information states as well as on deontic plans in the same manner. This is not problematic as both entities, information states and deontic plans, are themselves defined as sets of objects (possibilities and to-do lists respectively).

Definition 5.1.14 (Boolean connectives). Let χ be an information state or a plan, and φ a (possibly complex) sentence.

$$\begin{array}{llll}
 \chi \uparrow \neg\varphi & = & \chi \downarrow \varphi & \chi \downarrow \neg\varphi & = & \chi \uparrow \varphi \\
 \chi \uparrow (\varphi \vee \psi) & = & (\chi \uparrow \varphi) \cup (\chi \uparrow \psi) & \chi \downarrow (\varphi \vee \psi) & = & (\chi \downarrow \varphi) \downarrow \psi \\
 \chi \uparrow (\varphi \wedge \psi) & = & (\chi \uparrow \varphi) \uparrow \psi & \chi \downarrow (\varphi \wedge \psi) & = & (\chi \downarrow \varphi) \cup (\chi \downarrow \psi)
 \end{array}$$

The rules of update and downdate are easily explained. I will begin with the rules for negation. Updating with the negation of a sentence is downdating with the sentence without negation. Thus in the case of simple declarative sentences, if someone accepts the sentence $\neg p$, he is willing to add $\langle p, 0 \rangle$ (i.e. “ p is not the case”) to his information state. And downdating with the negation of a sentence is updating with the sentence without negation.

Example 5.1.15 (Negation). Let $p \in \mathcal{D}$, φ be a sentence and σ an information state.

1. $\mathbf{0} \uparrow \neg p = \mathbf{0} \downarrow p = \{\{\langle p, 0 \rangle\}, \{\emptyset\}\}$.
2. $\mathbf{0} \downarrow \neg p = \mathbf{0} \uparrow p = \{\{\langle p, 1 \rangle\}, \{\emptyset\}\}$.
3. $(\mathbf{0} \uparrow \varphi) \uparrow \neg\varphi = (\mathbf{0} \uparrow \neg\varphi) \uparrow \varphi = \perp$; $(\mathbf{0} \downarrow \varphi) \downarrow \neg\varphi = (\mathbf{0} \downarrow \neg\varphi) \downarrow \varphi = \perp$.

⁸Notice that formally, we should write here that $((\mathbf{0} \downarrow q) \uparrow p) \uparrow q = \emptyset \in \perp$, but as already mentioned, I will mostly treat the absurd state as an information state (even though it is a set of information states). I will only be more precise when I want to stress the type of failure that leads to the absurd state.

The update of a conjunction is also straightforward: it is the sequential update of the first and the second conjunct.

Example 5.1.16 (Update with conjunction). Let $p, q \in \mathcal{D}$, φ be a sentence and σ an information state.

1. $\mathbf{0} \uparrow (p \wedge q) = (\mathbf{0} \uparrow p) \uparrow q = \{(\{\langle p, 1 \rangle, \langle q, 1 \rangle\}, \{\emptyset\})\}$.
2. $\mathbf{0} \uparrow (\varphi \wedge \neg\varphi) = \mathbf{0} \uparrow (\neg\varphi \wedge \varphi) = \perp$.

It is important to notice that although the updates and dwnupdates so far literally add information, they are eliminative at the possibility level. There is information growth at the level of the information state that correlates with the possible elimination of possibilities. Atomic updates, negation and conjunction (of simple declarative sentences) can only decrease the number of possibilities in an information state. This is different in the case of the update with disjunction. The update with a disjunction is the union of the updates with the disjuncts and therefore, if the updates with both disjuncts do not lead to the absurd state (and add new information), it adds new possibilities to the resulting update. Disjunctions can be a source of uncertainty.

Example 5.1.17 (Disjunction). Let $p, q \in \mathcal{D}$ and σ be an information state.

1. $\mathbf{0} \uparrow (p \vee q) = (\mathbf{0} \uparrow p) \cup (\mathbf{0} \uparrow q) = \{(\{\langle p, 1 \rangle\}, \{\emptyset\}), (\{\langle q, 1 \rangle\}, \{\emptyset\})\}$.
2. $\mathbf{0} \uparrow (p \vee \neg p) = (\mathbf{0} \uparrow p) \cup (\mathbf{0} \downarrow p) = \{(\{\langle p, 1 \rangle\}, \{\emptyset\}), (\{\langle p, 0 \rangle\}, \{\emptyset\})\}$.
3. $(\mathbf{0} \uparrow \neg q) \uparrow (p \vee q) = \{(\{\langle q, 0 \rangle\}, \{\emptyset\})\} \uparrow (p \vee q) = \{(\{\langle q, 0 \rangle\}, \{\emptyset\})\} \uparrow p \cup \{(\{\langle q, 0 \rangle\}, \{\emptyset\})\} \uparrow q = \{(\{\langle p, 1 \rangle, \langle q, 0 \rangle\}, \{\emptyset\})\}$.

Finally, the dwnupdates of conjunction and disjunction are defined so as to respect the following intuitive equivalences $\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$ and $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$. In the case of the first equivalence the explanation goes as follows: the dwnupdate with $\varphi \vee \psi$ is a dwnupdate with $\neg(\neg\varphi \wedge \neg\psi)$, but a dwnupdate with negation is an update with the non-negated formula, i.e. an update with $\neg\varphi \wedge \neg\psi$. The last formula is a conjunction and therefore the original dwnupdate is actually the consecutive update with $\neg\varphi$ and $\neg\psi$, that is, the consecutive dwnupdate with φ and ψ . The same sort of explanation can be given for the second equivalence.

Definition 5.1.18 (Conditional). Let χ be an information state or a deontic plan,⁹ and φ and ψ (possibly complex) sentences.

$$\chi \uparrow \text{if } \varphi, \psi = \chi \downarrow \varphi \cup \chi \uparrow \varphi \uparrow \psi$$

i.e. if $\varphi, \psi \equiv \neg\varphi \vee (\varphi \wedge \psi)$.

⁹This will disappear from the final analysis as I do not believe that genuine conditionals appear under deontic operators, but I do not want to rule out this possible combinations yet.

The reader is probably wondering why a conditional operator is needed when a conditional could simply be defined as in the truth-conditional framework as $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$.

| | φ | ψ | $\varphi \rightarrow \psi$ | |
|-------------------------------|-----------|--------|----------------------------|------------------------|
| | 1 | 1 | 1 | $\leftrightarrow \psi$ |
| | 1 | 0 | 0 | |
| $\neg\varphi \leftrightarrow$ | 0 | 1 | 1 | $\leftrightarrow \psi$ |
| $\neg\varphi \leftrightarrow$ | 0 | 0 | 1 | |

The point is that the standard framework is in some sense an eliminative framework, i.e. we have complete information about the truth conditions of sentences in worlds (even though we might not know in which world we are). The update system I have introduced so far is however not eliminative but additive. In such a system adding the same kind of information twice matters. As we can see in the above truth table for the conditional, by defining a conditionals as $\neg\varphi \vee \psi$ we create overlapping conditions in the case the antecedent is false and the consequent is true (i.e. third line of the truth table). This has no consequence in the truth-conditional framework but is problematic in this update system: were we to learn that $\neg\varphi$ is actually the case (i.e. update with $\neg\varphi$), we would then have a bias toward ψ (versus $\neg\psi$).

Example 5.1.19. Let p and $q \in \mathcal{D}$.

1. $\mathbf{0} \uparrow (\neg p \vee q) = \mathbf{0} \downarrow p \cup \mathbf{0} \uparrow q = \{(\{ \langle p, 0 \rangle \}, \{ \emptyset \}), (\{ \langle q, 1 \rangle \}, \{ \emptyset \})\}$.
2. $(\mathbf{0} \uparrow (\neg p \vee q)) \uparrow \neg p = \{(\{ \langle p, 0 \rangle \}, \{ \emptyset \}), (\{ \langle p, 0 \rangle, \langle q, 1 \rangle \}, \{ \emptyset \})\}$.

The chosen definition of the conditional singles out all the different cases of the truth-conditional framework but does this only once each time. Therefore, the redundancy in the case the antecedent is false and the consequent is true is only expressed once.¹⁰

| | | | φ | ψ | $\varphi \rightarrow \psi$ |
|---|-------------------------------------|-------------------|-----------|--------|----------------------------|
| $\sigma \uparrow \varphi \uparrow \psi$ | update with φ and ψ | \leftrightarrow | 1 | 1 | 1 |
| | | | 1 | 0 | 0 |
| $\sigma \downarrow \varphi$ | downdate with φ leaves open | \leftrightarrow | 0 | 1 | 1 |
| | up/downdates with ψ | \leftrightarrow | 0 | 0 | 1 |

Example 5.1.20. Let p and $q \in \mathcal{D}$.

¹⁰The same effect can be achieved by defining the conditional as if $\varphi, \psi \equiv ((\neg\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)) \vee (\varphi \wedge \psi)$. But I will argue this version is less intuitive, in particular when the consequent is a non-epistemic modal.

1. $\mathbf{0} \uparrow \text{if } p, q = \mathbf{0} \downarrow p \cup (\mathbf{0} \uparrow p) \uparrow q = \{(\{\langle p, 0 \rangle\}, \{\emptyset\}), (\{\langle p, 1 \rangle, \langle q, 1 \rangle\}, \{\emptyset\})\}$.
2. $(\mathbf{0} \uparrow \text{if } p, q) \uparrow \neg p = \{(\{\langle p, 0 \rangle\}, \{\emptyset\})\}$.

We can see with examples 5.1.19 and 5.1.20 that the update with a conditional if φ , ψ is indeed different from an update with $\neg\varphi \vee \psi$. Although both formulas ($\neg\varphi \vee (\varphi \wedge \psi)$ and $\neg\varphi \vee \psi$) are truth-conditionally equivalent in propositional logic, this is not any more the case in this update system.

I will now introduce the update with epistemic possibility.¹¹ In simple terms, an update with *might* adds to the information state a copy of itself where the embedded sentence is the case. Furthermore, the definition is meant to be compatible with the insight of (Groenendijk, Stokhof and Veltman 1996) that *might* functions as a test on the information state but improves on it by stating that one can become aware of a new possibility with an epistemic possibility. The crucial property of epistemic modality is that it operates on whole information states, i.e. on the knowledge of the agent.

Definition 5.1.21 (Epistemic possibility). Let φ be a sentence.¹²

$$\sigma \uparrow \text{might } \varphi = \begin{array}{ll} \sigma \cup \sigma \uparrow \varphi & \text{if } \sigma \text{ is an information state such that } \sigma \uparrow \varphi \notin \perp, \\ \perp & \text{otherwise} \end{array}$$

We can now turn to the definitions of the updates with deontic information. Recall that we split the traditional unit of information, the world, into two distinct pieces, a situation and a deontic plan. Formally a deontic plan is made of to-do lists which, so far, are the same kind of entities as situations (that is, sets of pairs of sentences and “truth values”). The idea is that a to-do list in a deontic plan represents a set of morally desirable state of affairs. A deontic plan may contain different to-do lists and the agent may choose freely which to-do list(s) to fulfill (or not).

Notice that at this point it is not possible to represent different holders of obligations/permissions and thus any sentence in a to-do list is interpreted as a

¹¹Epistemic necessity will not be discussed and is left for future work. Its definition would probably be similar to the defaults in (Veltman 1996).

¹²We could thus stay closer to the original insight by defining for instance epistemic possibility as follows:

$$\sigma \uparrow \text{might } \varphi = \begin{array}{ll} \sigma & \text{if } \sigma \uparrow \varphi \cap \sigma \neq \emptyset \\ \sigma \cup \sigma \uparrow \varphi & \text{if } \sigma \uparrow \varphi \cap \sigma = \emptyset \text{ and } \sigma \uparrow \varphi \notin \perp \\ \perp & \text{otherwise} \end{array}$$

where the update is a test if the information is already part of the information state. There are surely many ways to model epistemic possibility even better but the main insight and property that is relevant for this work is that it operates on information states. I will therefore use this simple definition.

desirable state of affairs for the addressee (even though the obligation/permission might not be directed at the addressee).

A deontic plan is a set of to-do lists. Therefore, its update with some atomic declarative sentence will result in the set of to-do lists to which we add this declarative much as in the case of the update of a whole information state with an atomic declarative. The only difference is that the update with atomic declaratives does not check for consistency and we therefore need to define the notion of a consistent deontic plan. The dwnupdate is then defined as can be expected.

Definition 5.1.22 (Simple deontic information). Let δ be a deontic plan and $a \in \mathcal{D}$.

$$\begin{aligned}\delta \uparrow a &= \{t' \mid t' = t \cup \{a, 1\}\} \text{ for some } t \in \delta \\ \delta \downarrow a &= \{t' \mid t' = t \cup \{a, 0\}\} \text{ for some } t \in \delta\end{aligned}$$

Definition 5.1.23 (Consistent part of a deontic plan). Let δ be a deontic plan.

$$(\delta)_{cons} = \{t \in \delta \mid t \text{ is consistent}\}$$

Example 5.1.24 (Some updates of deontic plans). Let $a, b \in \mathcal{D}$ be some simple declaratives.

1. $\{ \begin{array}{|c|c|} \hline \vdots & \vdots \\ \hline \end{array}, \dots, \begin{array}{|c|c|} \hline \vdots & \vdots \\ \hline \end{array} \} \uparrow a = \{ \begin{array}{|c|c|} \hline a & \vdots \\ \hline \end{array}, \dots, \begin{array}{|c|c|} \hline a & \vdots \\ \hline \end{array} \}$
2. $\{ \begin{array}{|c|c|} \hline \vdots & \vdots \\ \hline \end{array}, \dots, \begin{array}{|c|c|} \hline \vdots & \vdots \\ \hline \end{array} \} \downarrow a = \{ \begin{array}{|c|c|} \hline \vdots & a \\ \hline \end{array}, \dots, \begin{array}{|c|c|} \hline \vdots & a \\ \hline \end{array} \}$
3. $(\{ \begin{array}{|c|c|} \hline & \\ \hline \end{array}, \begin{array}{|c|c|} \hline & a \\ \hline \end{array} \} \uparrow a)_{cons} = (\{ \begin{array}{|c|c|} \hline a & \\ \hline \end{array}, \begin{array}{|c|c|} \hline a & a \\ \hline \end{array} \})_{cons} = \{ \begin{array}{|c|c|} \hline a & \\ \hline \end{array} \}$
4. $\{ \begin{array}{|c|c|} \hline & \\ \hline \end{array} \} \uparrow (a \vee b) = \{ \begin{array}{|c|c|} \hline & \\ \hline \end{array} \} \uparrow a \cup \{ \begin{array}{|c|c|} \hline & \\ \hline \end{array} \} \uparrow b = \{ \begin{array}{|c|c|} \hline a & \\ \hline \end{array} \} \cup \{ \begin{array}{|c|c|} \hline b & \\ \hline \end{array} \} = \{ \begin{array}{|c|c|} \hline a & \\ \hline \end{array}, \begin{array}{|c|c|} \hline b & \\ \hline \end{array} \}$

The main question is thus when such an update takes place. The answer is obviously that those updates are triggered by deontic modal operators. But before turning to the update rules for modal operators I would like to make the ideas behind the formalization clear. First, remember that at this stage we abstract from the fact that we deal with a particular agent's information state and thus treat the deontic information much as in the standard framework as representing all the relevant obligations and permissions in a uniform fashion irrespectively of their bearer. Thus a deontic plan is a moral guideline for permissible and mandatory actions/states of affairs to be reached. As those deontic plans are sets of to-do lists, the guideline is that a moral agent has to try to fulfill at least one of the to-do lists in the deontic plan. A simple obligation is thus best represented by a pair of an atomic declarative and a truth value (i.e. $\langle a, 1 \rangle$) present in every

to-do list of the deontic plan whereas a permission is such a pair that is present in only some to-do list(s).

I will now give some discourse effects that need to be incorporated in the framework.

- (1)
 - a. You may take an apple... # You must not take an apple.
 - b. You must take an apple... # You're allowed not to take an apple.
- (2)
 - a. You must not go to the movies... # Maybe you may go to the movies.
 - b. You're allowed not to go to the movies... # Maybe you must go to the movies.
- (3)
 - a. If it rains, you may go to the movies. # You must not go to the movies.
 - b. If it rains, you must go to the movies. # You're allowed not to go to the movies.
- (4)
 - a. If it rains, you may go to the movies. # Maybe you may not go to the movies.
 - b. If it rains, you must go to the movies. # Maybe you don't have to go to the movies.

What is clear from examples (1), (2), (3) and (4) is that someone cannot update his information state successfully with the second sentence if his state supports the first. That is, one cannot update successfully with the second sentence without giving up somehow the first (but that is some kind of revision and is a whole different problem). For instance in example (1-a), one cannot accept to have the obligation to refrain from taking an apple when one already has the permission to take one, without giving up this permission. The same observation holds in the case of conditional obligations and permissions as in example (3). Furthermore example (3) also shows that modus tollens is not a valid argument with deontic information. More precisely the argument schema cannot be used as the update with the second sentences in (3) leads to the absurd state (which thus makes the argument vacuously valid in update semantics). Therefore deontic information cannot help decide about the antecedent of a deontic conditional. Notice that in examples (3) and (4) with conditional deontic sentences it is really the consecutive sequence of the two sentences that is problematic. If some time after the first sentence has been accepted the hearer comes to learn (i.e. accept) the negation of the antecedent, the second sentence might be accepted too.¹³

¹³There are however some examples where it would seem that sequences as in (3) are not that problematic. Consider the following example: you are at work but need to use a car. You ask a colleague whether you can use the company car and he responds.

- (i) If the boss is not around, you may use the company car. Let me check!

- (5) If it rains, you may go to the movies. . . It doesn't rain. . . You must not go to the movies.
 $\mathbf{0} \uparrow$ if $rain$, $movies \uparrow \neg rain \uparrow \text{must} \neg movies$
 $= \{(\{\langle rain, 0 \rangle\}, \{\{\langle movies, 1 \rangle\})\}$, for instance.
- (6) a. Maybe you're allowed to take an apple. . . You must not take an apple.
 b. Maybe you must take an apple. . . You're allowed not to take an apple.

Deontic modals under epistemic uncertainty can be discarded as the previous example makes clear. Uncertainty over obligations and permissions can be updated with more specific information. Finally, the deontic part of the update system has to take care of the problem of free choice permission.

- (7) a. You may take an apple or a pear \models You may take an apple.
 b. You may take an apple or a pear \models You may take a pear.
 c. You may take an apple or a pear $\not\models$ You may take an apple and a pear.

The solution to this problem will be a semantic one contrary to other recent proposals, such as for instance (Schulz 2007), which take a pragmatic approach.

To define the updates and dwnupdates with may and must, we need some additional definitions. I will proceed with the definition of the extension of a deontic plan.

Definition 5.1.25 (Extension). A plan δ' extends a plan δ iff for every $t \in \delta$, there is some $t' \in \delta'$ such that $t \subseteq t'$.

Example 5.1.26 (Extension of plans).

1. Any non-empty plan (even an inconsistent plan) extends the empty plan $\{\emptyset\}$.
2. $\{\{\langle a, 1 \rangle\}\}$, the plan where a is the only obligation and nothing else is permitted, extends $\{\emptyset, \{\langle a, 1 \rangle\}\}$, the plan where a is the only permission and nothing is mandatory.

The colleague comes back a couple of minutes later and says,

- (ii) Sorry, you may not use the company car.

I think two different pragmatic phenomena are occurring in this example. The first one is an instance of closed world reasoning on the first conditional, that is, the conditional is reinterpreted as an equivalence (the information state is thus also updated with *If the boss is around, you may not use the company car*). As such an update with the second sentence leaves to the absurd state. However under the maxim of quality, the speaker (who obviously shares the information about the deontic conditional) is saying what he thinks to be "true", which means he must have some evidence that the boss is around.

That is, a deontic plan that extends another plan is such that each to-do list of the original is included in some list of the extension. If a possibility is such that the update of its deontic plan with an obligation does not extend the original plan (i.e. the obligation contradicts some permission), it is removed from the information state. But as we have seen, in the case of an update with an obligation, we want to account for example (1). That is, adding an obligation to a possibility/deontic plan should not remove any permissions. We thus need an extra condition that guarantees that an update with a deontic obligation is only felicitous if it does not add any non-deontic information, i.e. the eventual deletion of possibilities should not change the overall content of this information. I will dub this property *factual subsistance*.

Definition 5.1.27 (Factual subsistance). An information state σ factually subsists in an information state σ' iff for all $(s, \delta) \in \sigma$ there is a δ' such that $(s, \delta') \in \sigma'$.

Example 5.1.28 (Factual subsistance of information states).

1. The minimal information state $\{(\emptyset, \{\emptyset\})\}$ (the state with no knowledge whatsoever) does not factually subsist in the state $\{(\{\langle hungry, 1 \rangle\}, \{\emptyset\})\}$ (a state where I know that the dog is hungry) but it does in a state where I know that the dog might be hungry $\{(\emptyset, \{\emptyset\}), (\{\langle hungry, 1 \rangle\}, \{\emptyset\})\}$.
2. The minimal information state $\{(\emptyset, \{\emptyset\})\}$ factually subsists in a state where the only thing I know is that the dog must eat $\{(\emptyset, \{\{\langle eat, 1 \rangle\})\})\}$.¹⁴
3. The state $\{(\{\langle hungry, 0 \rangle\}, \{\emptyset\}), (\{\langle hungry, 1 \rangle\}, \{\langle eat, 1 \rangle\})\}$ results from the update of the minimal information state with the sentence “if the dog is hungry, he must eat/be fed.” It does not subsist in the state where the dog is not hungry but you are allowed to feed it, $\{(\{\langle hungry, 0 \rangle\}, \{\emptyset, \{\langle eat, 1 \rangle\})\})\}$.

Factual subsistance is a condition that applies to any update with deontic information as we have seen in example (3). Finally, we see that a deontic plan can contain information about obligations as well as permissions by the fact that it consists of different to-do lists. When updating with a permission we therefore want to add some to-do lists that contain it as well as the current obligations of the plan. However we do not want to add any superfluous information and we thus have to add the permission not as a copy of all to-do lists but only as a copy of the minimal set of to-do lists representing the current obligations. I will call this set the *base*.

Definition 5.1.29 (Base¹⁵). The set $\delta_b = \{s \in \delta \mid \text{there is no } s' \in \delta \text{ s.t. } s' \subset s\}$ is called the base of a plan δ . It represents the duties of plan δ , that is intuitively, you have to do a in δ iff you have to do it in δ_b .

¹⁴Obviously, it is not the dog that has the obligation but the holder of the information state that has to see to it that the dog gets fed.

¹⁵The same kind of concept is termed the minimum in (Mastop 2007)

Example 5.1.30 (Bases of deontic plans).

1. The empty plan is its own base: $\{\emptyset\}_b = \{\emptyset\}$
2. A deontic plan with no obligations and only permissions has the empty plan as base: $\{\emptyset, \{\langle a, 1 \rangle\}, \{\langle b, 0 \rangle\}\}_b = \{\emptyset\}$.
3. $\{\{\langle a, 1 \rangle\}, \{\langle a, 1 \rangle, \langle b, 0 \rangle\}\}_b = \{\{\langle a, 1 \rangle\}\}$, a is the only duty of the deontic plan $\{\{\langle a, 1 \rangle\}, \{\langle a, 1 \rangle, \langle b, 0 \rangle\}\}$.
4. $\{\{\langle a, 1 \rangle\}, \{\langle a, 1 \rangle, \langle c, 1 \rangle\}, \{\langle b, 1 \rangle\}, \{\langle b, 1 \rangle, \langle c, 1 \rangle\}\}_b = \{\{\langle a, 1 \rangle\}, \{\langle b, 1 \rangle\}\}$. Notice that the base of this deontic plan does not represent a simple obligation (i.e. an obligation of a simple declarative). The base shows that the holder of this deontic plan has to do either a or b .

We can now introduce the definitions for the modal operators *may* and *must*. A final observation that needs to be made is that I will consider that the deontic modal operators *may* and *must* are each others dual as in the following sentences:

- (8) logical form: $\text{must} \neg \text{smoke}$
 - a. You must not smoke in the building.
 - b. It is forbidden to smoke in the building.
- (9) logical form: $\neg \text{may smoke}$
 - a. You may not smoke in the building.
 - b. It is not allowed to smoke in the building.

Definition 5.1.31 (Deontic updates of information states). Let α be a sentence and σ an information state. If σ factually subsists in the update,

$$\begin{aligned} \sigma \uparrow \text{may } \alpha &= \{(s, \delta \cup (\delta_b \uparrow \alpha)_{cons}) \mid (s, \delta) \in \sigma \ \& \ (\delta_b \uparrow \alpha)_{cons} \text{ extends } \{\emptyset\} \uparrow \alpha\} \\ \sigma \downarrow \text{may } \alpha &= \{(s, (\delta \downarrow \alpha)_{cons}) \mid (s, \delta) \in \sigma \ \& \ (\delta \downarrow \alpha)_{cons} \text{ extends } \delta \text{ and } \{\emptyset\} \downarrow \alpha\} \\ \\ \sigma \uparrow \text{must } \alpha &= \{(s, (\delta \uparrow \alpha)_{cons}) \mid (s, \delta) \in \sigma \ \& \ (\delta \uparrow \alpha)_{cons} \text{ extends } \delta \text{ and } \{\emptyset\} \uparrow \alpha\} \\ \sigma \downarrow \text{must } \alpha &= \{(s, \delta \cup (\delta_b \downarrow \alpha)_{cons}) \mid (s, \delta) \in \sigma \ \& \ (\delta_b \downarrow \alpha)_{cons} \text{ extends } \{\emptyset\} \downarrow \alpha\} \\ &= \Lambda \text{ otherwise.} \end{aligned}$$

First, it is important to notice the following crucial property of these updates: deontic sentences only operate directly on deontic plans although they may eliminate possibilities when their deontic update is not successful. I will now discuss the different updates. For instance, the positive update with a permission $\text{may } \alpha$ performs the following operations:

1. for every possibility (s, δ) in the information state, take the base of the deontic plan (the obligation of this plan) and update it with the embedded sentence. Consider the consistent subset of this update,

- (a) if it extends the minimal deontic plan updated with the embedded sentence, replace (s, δ) by $(s, \delta \cup (\delta_b \uparrow \alpha)_{cons})$ (i.e. add to the plan to-do lists containing the former obligations augmented with the new “permission”),
 - (b) otherwise remove (s, δ) from the information state,
2. return
- (a) the information state obtained by the steps in 1 if no factual information is lost,
 - (b) Λ otherwise.

In other words, when updating an information state σ with a deontic permission may α , we first form the set containing the possibilities consistently updated with the permission without removing any obligations, that is, $\{(s, \delta \cup \delta_b \uparrow \alpha) \mid (s, \delta) \in \sigma \ \& \ (\delta_b \uparrow \alpha)_{cons} = \delta_b \uparrow \alpha\}$. Finally, if σ factually subsists in it, this set becomes the new information state.

Successful updates with permissions give the agent the choice to choose a to-do list containing it as its moral guideline (of things to do/accomplish). This choice of the agent makes clear that once the permission is successfully added to his information state he has gained the right to perform it.

Example 5.1.32 (Updates with permissions). Here are some examples of updates with permissions.¹⁶

$$\begin{aligned}
1. \quad & \mathbf{0} \uparrow \text{may } a = \{(\emptyset, \{\emptyset\})\} \uparrow \text{may } a = \{(\emptyset, \{ \boxed{} \boxed{} \})\} \uparrow \text{may } a \\
& = \{(\emptyset, \{ \boxed{} \boxed{} \} \cup (\{ \boxed{} \boxed{} \}_b \uparrow a)_{cons})\} \quad \text{by Definition 5.1.31,}^{17} \\
& = \{(\emptyset, \{ \boxed{} \boxed{} \} \cup \{ \boxed{a} \boxed{} \})\} \quad \text{by definitions 5.1.22 and 5.1.29,} \\
& = \{(\emptyset, \{ \boxed{} \boxed{} , \boxed{a} \boxed{} \})\}
\end{aligned}$$

$$2. \quad \{(\emptyset, \{\langle a, 0 \rangle\})\} \uparrow \text{may } a = \{(\emptyset, \{ \boxed{} \boxed{a} \})\} \uparrow \text{may } a = \Lambda \text{ by Definition 5.1.31 because the information state } \{(\emptyset, \{ \boxed{} \boxed{a} \})\} \text{ does not subsist in the update:}$$

by definition 5.1.29, $\{ \boxed{} \boxed{a} \}_b = \{ \boxed{} \boxed{a} \}$ and thus,

$$\{ \boxed{} \boxed{a} \}_b \uparrow a = \{ \boxed{a} \boxed{a} \} \text{ and } (\{ \boxed{a} \boxed{a} \})_{cons} = \emptyset \text{ by definitions 5.1.22 and 5.1.23.}$$

but \emptyset does not extend $\{ \boxed{} \boxed{} \} \uparrow a = \{ \boxed{a} \boxed{} \}$ which means the following for the set of definition 5.1.31:

$$\{(\emptyset, \{ \boxed{} \boxed{a} \} \cup (\{ \boxed{} \boxed{a} \}_b \uparrow a)_{cons}) \mid (\{ \boxed{} \boxed{a} \}_b \uparrow a)_{cons} \text{ extends } \{ \boxed{} \boxed{} \}_b \uparrow a\} = \emptyset \text{ and obviously } \{(\emptyset, \{ \boxed{} \boxed{a} \})\} \text{ does not subsist in } \emptyset.$$

¹⁶I will only work out some of them completely and let the reader check that the other claims hold.

¹⁷As $(\{ \boxed{} \boxed{} \}_b \uparrow a)_{cons} = \{ \boxed{a} \boxed{} \} = \{ \boxed{} \boxed{} \}_b \uparrow a$.

3. $\{(\{\langle p, 0 \rangle\}, \{\emptyset\}), (\{\langle p, 1 \rangle\}, \{\{\langle a, 0 \rangle\}\})\} \uparrow$ may $a = \Lambda$ because the update deletes the second possibility as can be seen from the previous example (irrespective of the factual information), i.e. the update would result in the following information state $\{(\{\langle p, 0 \rangle\}, \{\emptyset, \{\{\langle a, 1 \rangle\}\})\}$ in which the original state does not factually subsist.
4. $\mathbf{0} \uparrow \text{may}(a \vee b) = \{(\emptyset, \{\square\square\} \cup (\{\square\square\}_b \uparrow (a \vee b))_{cons})\}$ by definition 5.1.31,
 $= \{(\emptyset, \{\square\square\} \cup (\{\square\square\}_b \uparrow a \cup \{\square\square\}_b \uparrow b)_{cons})\}$ by definition 5.1.14
 $= \{(\emptyset, \{\square\square\} \cup (\{\text{a}\square\} \cup \{\text{b}\square\})_{cons})\}$ by example 5.1.30 and definition 5.1.22,
 $= \{(\emptyset, \{\square\square, \text{a}\square, \text{b}\square\})\}$.
 Therefore $\mathbf{0} \uparrow \text{may}(a \vee b) \Vdash \text{may } a$ as $\{\square\square, \text{a}\square, \text{b}\square\}_b = \{\square\square\}$.

We can now turn to the update with a deontic obligation of the form ‘must α ’. The idea is that whatever the present permissions/rights are, i.e. whatever to-do lists are present in the deontic plans, they should all be made to contain α . That is, α is an element of any to-do list and thus is a mandatory action/state of affairs to be reached. Formally, the definition goes as follows:

1. for every possibility (s, δ) in the information state, update the deontic plan with the embedded sentence and consider the consistent subset of this update,
 - (a) if it extends the original deontic plan and the minimum update with the obligation (respectively: all the to-do lists of the original deontic plan are a subset of some element of the updated plan and thus no permission has been removed; and no part of the meaning of the obligation is lost),¹⁸ then replace (s, δ) by $(s, (\delta \uparrow \alpha)_{cons})$
 - (b) otherwise remove (s, δ) from the information state,
2. return
 - (a) the information state obtained by the steps in 1 if no factual information is lost,
 - (b) Λ otherwise.

In an update with a deontic obligation, a possibility is removed any time its deontic plan contains a to-do list that is not consistent with the embedded sentence.

¹⁸This condition, $(\delta \uparrow \alpha)_{cons}$ extends $\{\emptyset\} \uparrow \alpha$, accounts for a central idea of the framework in (Veltman 2007): if you are not allowed to do a , then the update with the obligation to do a or b leads to the absurd state. Therefore obligations are not completely disconnected from permissions.

Example 5.1.33 (Updates with obligations).

1. $\mathbf{0} \uparrow \text{ must } a = \{(\emptyset, \{ \square \square \})\} \uparrow \text{ must } a = \{(\emptyset, (\{ \square \square \} \uparrow a)_{cons})\} = \{(\emptyset, \{ \boxed{a} \square \})\}$.
2. $\mathbf{0} \uparrow \text{ may } \neg a \uparrow \text{ must } a = \{(\emptyset, \{ \square \square, \square \boxed{a} \})\} \uparrow \text{ must } a = \Lambda$ because:
 $(\{ \square \square, \square \boxed{a} \} \uparrow a)_{cons} = (\{ \boxed{a} \square, \boxed{a} \boxed{a} \})_{cons} = \{ \boxed{a} \square \}$ which does not extend $\{ \square \square, \square \boxed{a} \}$. Therefore the condition of definition 5.1.31 is not fulfilled for the only possibility in the information state and the original information state does not subsist in the update.
3. $\mathbf{0} \uparrow \text{ must}(a \vee b) = \{(\emptyset, \{ \boxed{a} \square, \square \boxed{b} \})\}$.
 Ross's paradox (*you must post the letter* \models *you must post the letter or burn it*) does not occur in this system, i.e. $\text{must } a \not\models \text{must}(a \vee b)$, for instance:
 $\mathbf{0} \uparrow \text{ must } a = \{(\emptyset, \{ \boxed{a} \square \})\}$

$$\neq \mathbf{0} \uparrow \text{ must } a \uparrow \text{ must}(a \vee b) = \{(\emptyset, \{ \boxed{a} \square, \begin{array}{|c|} \hline a \\ \hline b \\ \hline \end{array} \square \})\}.$$

It is also nice to notice that once you know you have the duty to do a and $a \vee b$, you actually get the right to do b :

$$\mathbf{0} \uparrow \text{ must } a \uparrow \text{ must}(a \vee b) = \mathbf{0} \uparrow \text{ must}(a \vee b) \uparrow \text{ must } a \Vdash \text{ may } b.^{19}$$

Finally the deontic modal operators are considered to be dual and therefore the downdates are equivalent to the updates with the opposite modal and a negation, i.e. the downdate with *may* is an update with *must not*, and the downdate with *must* is an update with *may not*.

Observation 5.1.34. Let σ be an information state and $a, b \in \mathcal{D}$ two declarative sentences.

1. If σ is not the absurd state and $\sigma \Vdash \neg \text{ may } a$, then $\sigma \uparrow \text{ may}(a \vee b) \in \perp$.
 For instance, $\mathbf{0} \uparrow \neg \text{ may } a \Vdash \neg \text{ may } a$ i.e. $\{(\emptyset, \{ \square \boxed{a} \})\} \Vdash \neg \text{ may } a$, but
 $\{(\emptyset, \{ \square \boxed{a} \})\} \uparrow \text{ may}(a \vee b) = \Lambda$ because $(\{ \square \boxed{a} \})_b \uparrow (a \vee b)_{cons} =$

¹⁹The system does however make a prediction that is somewhat troublesome: $\mathbf{0} \uparrow \text{ must}(a \vee b) \not\models \text{ must}(a \vee b)$ as $\mathbf{0} \uparrow \text{ must}(a \vee b) = \{(\emptyset, \{ \boxed{a} \square, \square \boxed{b} \})\}$ and $\mathbf{0} \uparrow \text{ must}(a \vee b) \uparrow (a \vee b) = \{(\emptyset, \{ \boxed{a} \square, \square \boxed{b}, \begin{array}{|c|} \hline a \\ \hline b \\ \hline \end{array} \square \})\}$. Obviously this result is not warranted but it can be surmounted in at least two ways. A first solution is to loosen the definition of acceptance in a way similar to (Mastop 2005, p105). The other solution is to change the (relevant part of the) definition of the update with deontic necessity for instance as follows:

$$\begin{aligned} \sigma \uparrow \text{ must } \alpha &= \{(s, \delta) \mid (s, \delta) \in \sigma \ \& \ (\delta \downarrow \alpha)_{cons} = \emptyset \ \& \ \delta_b \text{ extends } \{\emptyset\} \uparrow \alpha\} \cup \{(s, (\delta \uparrow \alpha)_{cons}) \mid \\ &(s, \delta) \in \sigma \ \& \ ((\delta \downarrow \alpha)_{cons} \neq \emptyset \ \text{or } \delta_b \text{ does not extend } \{\emptyset\} \uparrow \alpha) \ \text{but } (\delta \uparrow \alpha)_{cons} \\ &\text{extends } \delta\}. \end{aligned}$$

- $\{ \boxed{b \mid a} \}$
 does not extend $\{ \boxed{ \mid } \} \uparrow (a \vee b) = \{ \boxed{a \mid }, \boxed{b \mid } \}$.
2. $\mathbf{0} \uparrow \text{may}(a \vee b) \uparrow \neg \text{may}(a \wedge b) \notin \perp$,
 $\mathbf{0} \uparrow \text{may}(a \vee b) = \{(\emptyset, \{ \boxed{ \mid }, \boxed{a \mid }, \boxed{b \mid } \})\}$ as example 5.1.32 shows
 and
 $\mathbf{0} \uparrow \text{may}(a \vee b) \uparrow \neg \text{may}(a \wedge b) = \{(\emptyset, \{ \boxed{ \mid b}, \boxed{a \mid b}, \boxed{ \mid a}, \boxed{b \mid a} \})\}$.

These two examples show that the condition in the update with a permission is precisely what is needed. The condition is on $(\delta_b \uparrow \alpha)_{cons}$ where α is the embedded sentence under the deontic permission and δ_b represents the duties of deontic plan δ cleaned out of the rights. One important thing to check when updating with a permission is that it does not conflict with duties. As such, the condition $(\delta_b \uparrow \alpha)_{cons} \neq \emptyset$ would do the trick. However, the first update in this example would not fail with this condition, for instance $\mathbf{0} \uparrow \neg \text{may } a \uparrow \text{may}(a \vee b) \notin \perp$. A stronger condition would be to ask for $\delta_b \uparrow \alpha$ to be fully “consistent”, that is, $(\delta_b \uparrow \alpha)_{cons} = \delta_b \uparrow \alpha$ and the embedded permission is consistent with all the duties. Now it is the second update that goes wrong, $\mathbf{0} \uparrow \text{may}(a \vee b) \uparrow \neg \text{may}(a \wedge b) \uparrow \text{may}(a \vee b) \in \perp$. Therefore, this condition is too strong. What we need is that the sentence embedded under the permission does not conflict with the duties but also that the update does add the whole permission expressed by the sentence, i.e. $(\delta_b \uparrow \alpha)_{cons}$ extends $\{\emptyset\} \uparrow \alpha$.

It is important to notice why a specialized deontic absurd state is needed. This is best seen in the context of conditional sentences. Suppose you know you are allowed to go to the movies. This happens for instance after having updated the minimal information state with the sentence,

- (10) You may go to the movies.
 $\mathbf{0} \uparrow \text{may } go = \{(\emptyset, \{\emptyset, \{\langle go, 1 \rangle\})\}) \Vdash \text{may } go$

In such a state, going to the movies is a right, an unrestricted permission you have obtained. Therefore any conditional sentence (such as the one following) trying to restrict this right is in contradiction with it (or better said, restricting a right after it has been issued is something that a theory of revision has to deal with but is not the point of an update system).

- (11) If the weather is nice, you may not go to the movies.

Assume that the failure condition (factual subsistence) is removed from definition 5.1.31, that is, an update with a deontic sentence on an information state is either the updated information state or the empty set. Were we now to update the information state obtained after sentence (10), we would have a successful update and obtain an information state that supports the information that we may go to the movies and that the weather is not nice:

$$\begin{aligned}
& \mathbf{0} \uparrow \text{ may } go \uparrow \text{ if } nice, \neg \text{ may } go \\
= & \mathbf{0} \uparrow \text{ may } go \downarrow nice \cup \mathbf{0} \uparrow \text{ may } go \uparrow nice \uparrow \neg \text{ may } go \\
= & \{(\langle nice, 0 \rangle, \{ \boxed{}, \boxed{go } \})\} \cup \\
& \{(\langle nice, 1 \rangle, \{ \boxed{}, \boxed{go } \})\} \downarrow \text{ may } go \\
& \text{but } (\{ \boxed{}, \boxed{go } \} \downarrow go)_{cons} = (\{ \boxed{} go, \boxed{go go} \})_{cons} = \{ \boxed{} go \} \\
& \text{which does not extend } \{ \boxed{}, \boxed{go } \}, \\
= & \{(\langle nice, 0 \rangle, \{ \boxed{}, \boxed{go } \})\} \cup \emptyset \\
= & \{(\langle nice, 0 \rangle, \{ \boxed{}, \boxed{go } \})\}
\end{aligned}$$

This shows that a mechanism is needed to prevent updates with deontic information deciding about factual information. Notice that this mechanism cannot be replaced by the simplest idea of not allowing deontic updates to return the empty set. Why this is so should become clear with the study of sentences combining an epistemic modal above a deontic one.

5.1.1 Epistemic above deontic modality

As we have seen in chapter 2, not all combinations of modal items are grammatical. However epistemic modals can combine with deontic ones when they take wide scope. In particular the following type of sentence is grammatical:

- (12) a. You might have to go to Amsterdam.
b. You might be allowed to go to Amsterdam.²⁰

In view of the definition of epistemic possibility, the update with sentence (12-a) on an information state is successful just in case the deontic update is successful on this very same state:

$$\sigma \uparrow \text{ might must } go \notin \perp \text{ iff } \sigma \uparrow \text{ must } go \neq \Lambda$$

I will now present the update with those sentences on the minimal information state (for the sake of simplicity).

$$\begin{aligned}
& \mathbf{0} \uparrow \text{ might must } go \\
= & \mathbf{0} \cup \mathbf{0} \uparrow \text{ must } go \qquad \text{as } \mathbf{0} \uparrow \text{ must } go \notin \perp \\
= & \{(\emptyset, \{\emptyset\})\} \cup \{(\emptyset, \{\langle go, 1 \rangle\})\} \\
= & \{(\emptyset, \{\emptyset\}), (\emptyset, \{\langle go, 1 \rangle\})\}
\end{aligned}$$

The update of the minimal information state with sentence (12-a) results in the addition to this minimal state of a possibility in which to go to Amsterdam is an obligation. The update with sentence (12-b) is similar except that in the new possibility going to Amsterdam is a right not an obligation:

²⁰Notice that there are two different temporal interpretations of these sentences under the epistemic modal. The obligation or permission is either already holding for the agent (overlapping with the speech time) or its start is located in the future.

$$\begin{aligned} & \mathbf{0} \uparrow \text{might } \text{may } go \\ = & \{(\emptyset, \{\emptyset\}), (\emptyset, \{\emptyset, \{\langle go, 1 \rangle\})\}) \} \end{aligned}$$

Observation 5.1.35. From these updates we can deduce that all the following updates are successful:

1. $\mathbf{0} \uparrow \text{might } \text{must } go \uparrow \text{must } go = \{(\emptyset, \{\{\langle go, 1 \rangle\})\})\}$.
2. $\mathbf{0} \uparrow \text{might } \text{must } go \uparrow \text{must } \neg go = \{(\emptyset, \{\{\langle go, 0 \rangle\})\})\}$.
3. $\mathbf{0} \uparrow \text{might } \text{must } go \uparrow \neg \text{must } go = \{(\emptyset, \{\emptyset, \{\langle go, 0 \rangle\})\})\}$.
4. $\mathbf{0} \uparrow \text{might } \text{must } go \uparrow \text{may } go = \{(\emptyset, \{\emptyset, \{\langle go, 1 \rangle\})\}), (\emptyset, \{\{\langle go, 1 \rangle\})\})\}$.
5. $\mathbf{0} \uparrow \text{might } \text{must } go \uparrow \text{may } go \uparrow \neg \text{must } go = \{(\emptyset, \{\emptyset, \{\langle go, 1 \rangle\}, \{\langle go, 0 \rangle\})\})\}$.

The fourth update is the only one with two possibilities. This is the case because the permission update does not resolve the uncertainty about whether we have to go or not. This ambiguity is resolved when we update with the sentence *You don't have to go to Amsterdam* as in 5.

From the previous examples we can see why the deontic absurd state cannot be the empty set: in the case of a deontic modal embedded under an epistemic one, new deontic information can help decide between possibilities.

5.1.2 In the scope of deontic modals

Within this update system, we can now give an explanation of the fact that deontic modals cannot scope over epistemic ones. As will become clear, this property is due to the architecture of the system: epistemic modals operate on whole information states whereas deontic modals operate on deontic plans (inside possibilities). Formally, it is enough to see an example with the minimal information state to understand the failure to update with this combination of modals:

(13) #You must might go to Amsterdam.

$$\begin{aligned} & \mathbf{0} \uparrow \text{must } \text{might } go \\ = & \{(\emptyset, (\{\emptyset\} \uparrow \text{might } go)_{cons}) \mid (\{\emptyset\} \uparrow \text{might } go)_{cons} \text{ extends } \{\emptyset\}\}, \\ & \text{if it factually subsists in } \mathbf{0}, \\ = & \text{failure} \\ \text{as} & \{\emptyset\} \uparrow \text{might } go \text{ is not defined because } \{\emptyset\} \text{ is no information state.} \end{aligned}$$

In the general case, the situation remains the same, that is, the *might* update would have to be accomplished on a deontic plan which is a different kind of entity from an information state. The first is a set of to-do lists whereas the second is a set of possibilities and we have already seen that to-do lists and possibilities

are different entities. But the update with *might* is only defined on information states and therefore the update cannot be completed and fails. Notice that the problem is not that the update would lead to the absurd state (as an update with a sentence and its negation would do) but that it cannot proceed because of the mismatch between the operator and the (state/plan) argument of the update.

Maybe more surprising is the fact that we cannot stack deontic operators either. Therefore the following sentences (as combinations of two deontic modal items) leads to failure too.

- (14) a. #You must have to go to Amsterdam.
 b. #You must be allowed to go to Amsterdam.
 c. #You are allowed to have to go to Amsterdam.
 d. #You are allowed to be allowed to go to Amsterdam.

The same kind of reason as for epistemic modality can be given, that is, deontic modals operate on information states. This might seem curious at first but is all easily explained. Deontic modals give information about the world and as such operate on whole information states, however the information they deliver is of deontic nature and therefore acts upon the deontic plans of the information state's possibilities. But the deontic plans are only defined for atomic updates and boolean combinations thereof. Thus their update with a deontic modal fails. We can demonstrate this with the update of the minimal information state with sentence (14-a).

$$\begin{aligned}
 & \mathbf{0} \uparrow \text{ must must } go \\
 = & \{(\emptyset, (\{\emptyset\} \uparrow \text{ must } go)_{cons}) \mid (\{\emptyset\} \uparrow \text{ must } go)_{cons} \text{ extends } \{\emptyset\}\}, \\
 & \text{if it factually subsists in } \mathbf{0}, \\
 = & \text{ failure} \\
 \text{as } & \{\emptyset\} \uparrow \text{ must } go \text{ is not defined because } \{\emptyset\} \text{ is no information state.}
 \end{aligned}$$

It should be noted that sentence (14-b) is not necessarily ungrammatical. However I want to argue that the grammatical reading of this sentence is different from the update with *must may go*. It is important to remark that the grammatical subject needs not be the holder of the obligation. Consider first the case where the subject is the person to which the obligation is addressed, that is, the update is of the form $\sigma \uparrow \text{ must}$ (“*you are allowed to go to Amsterdam*”). Therefore the hearer has indeed to add a sentence as an obligation to its deontic plans. Intuitively the sentence to be added is not itself a permission but a state to be reached, i.e. the hearer has to see to it that he gets the permission to go to Amsterdam. As the previous formulation indicates it seems that the sentence embedded under the obligation is added as a whole to the deontic plans. Within the update system this would be made possible by the introduction of a lexical counterpart to the deontic modal operator, that is, some lexical element *permission* (and constraints on its use with respect to what must be the case in the deontic plans). In the

second case, the subject is not the addressee of the obligation but it is directed to someone else, say MrX. The sentence can thus be paraphrased by *MrX must see to it that you are allowed to go to Amsterdam*. It seems only fair to suppose that the covert holder of the obligation, MrX, is attributed this obligation because he indeed can do something about your permission, i.e. he has authority on this matter. Therefore the sentence is probably better paraphrased by *MrX must allow you to go to Amsterdam* where the verb of the embedded sentence is active. However, I have not provided any semantic definition for the active verb *to allow* but only for its passive counterpart *to be allowed to*. It seems again fair to suppose that sentence (14-b) thus means that MrX should have in all his to-do lists *I allow you to go to Amsterdam*.

In conclusion, the semantics of modal items given in this chapter make the following predictions:

1. Epistemic modals can scope over deontic ones.
2. Epistemic modals cannot be interpreted under deontic ones.
3. Deontic modal operators cannot be stacked.

5.2 Goal-oriented and participant-internal modality

For completeness, I will now provide a semantics for the two missing modal categories of goal-oriented and participant-internal modality. In our typology, goal-oriented modality is a subpart of participant-external modality, just as deontic modality is. It would therefore be appropriate to give a semantics to these modals that is in line with the semantics given above for deontic modals. For the sake of simplicity, I will however remove deontic information from the formalization in order to keep the definitions as short as possible. This means that we again have to define possibilities although information states remain the same (that is, sets of possibilities). In the previous section, possibilities were an ordered pair of a situation and of a deontic plan. We now want to be able to speak about goals and the ways to achieve them but still independently of what is the case, the situation. I will thus replace the deontic plan by the appropriate entity to account for goal-oriented modals. In the first part of this section, I will present a formalization of goal-oriented modality without ability (i.e. without incorporating participant-internal modals). In the second part, the semantics of participant-internal modality and their close relationship with goal-oriented modality will be discussed.

5.2.1 Goal-oriented modality: a first sketch

Goal-oriented modality is a part of participant-external modality just as deontic modality is. It should therefore not be surprising that the remark issued for deontic modality holds for this kind of modality too: goal-oriented modality has to do with planning and this kind of information is distinct from factual information. We can almost take over the previous picture literally:



That is, information about plans and the ways to achieve them is independent from information about facts. The question is now what kind of information is a plan exactly? To answer this question, we first have to remember the “logical” form of goal-oriented sentences.

- (15) a. To go to Texel, you have to take a boat.
 b. To go to Texel, you can take the ferry.

Those sentences have two parts. First there is a goal argument²¹ (here the goal is to go to Texel) and then a condition on the possible plans to achieve this goal. In sentence (15-a) the condition expresses that every plan to achieve the goal is a plan where you take a boat whereas in sentence (15-b) the condition expresses that a possible plan is a plan where you take the ferry.²² Formally I will stay in line with definition 5.1.4 and call a plan a set of to-do lists. However we need

²¹Notice that I will for ease of presentation only consider the case of goal-oriented sentences where the covert argument of the goal is identical to the subject position of the condition (which will be the holder of the information state designated by *you*). This is a very common kind of goal-oriented sentence but is by no means the only one. The subject of the condition can of course also be different from the addressee as in (i-a) and if an overt argument is provided to the goal the subject of the condition can be different as in (i-b). Furthermore, just as in the case of deontic modals, the covert argument of the goal can be provided by the context (e.g. the addressee). This is particularly the case when the condition features an expletive subject or a “stative” sentence.

- (i) a. To win the game, John must score two points.
 b. In order for the Netherlands to qualify for the next round, Germany has to beat France.
 c. To kitesurf, it must be windy.
 d. To play at night, the lights must be on.

This simplification is in line with the one made regarding deontic modals. The precise formalization in the general case will unfortunately have to be left for later work.

²²Notice also that the conditions of a goal-oriented sentences can be divided in two main types: actions/events and states. In sentence (15-a), for instance, the condition is an action/event, *you take the boat*, but in the following sentence it is a state, *you are older than 21*:

to represent plans to achieve a goal. This will be done by taking the ordered pair of the goal and its plan, i.e. $(goal, plan)$. Obviously there is more to a plan than just the ways to go to Texel and therefore a planning system will be a set of ordered pairs of a goal and its plan. Notice that to have a plan for a particular goal in a planning system does not mean to have the will to achieve this goal but just to have the knowledge of how to achieve it if wanted or needed.

Definition 5.2.1 (Planning system). A planning system is a set of ordered pairs consisting of a goal (a sentence of the language)²³ and a plan. As in definition 5.1.4, a plan is a set of to-do lists. I will use the uppercase Π to denote a planning system and lowercase π to refer to plans.

Example 5.2.2 (Some planning systems).

1. The minimal planning system Π_0 is the empty planning system, i.e. there is no information about goals and how to achieve them.
2. $\Pi_1 = \{(Texel, \{ \boxed{\text{boat}} \})\}$ is the planning system that, as we will see, is obtained by updating the minimal planning system with sentence (15-a).
3. $\Pi_2 = \{(Texel, \{ \boxed{\text{boat}} \text{ ferry} \}, \boxed{\text{boat}} \text{ rent} \})\}$ is the planning system with only one goal, going to Texel, where sentence (15-a) is still the case (you have to take a boat) but where you know you have a choice between taking the ferry or renting your own boat.
4. $\Pi_3 = \{(Texel, \{ \boxed{\text{boat}} \}), (Boston, \{ \boxed{\text{plane}} \})\}$ contains two goals, going to Texel and going to Boston, with their respective (partial) plans.

(i) To drink alcohol in this state, you have to be older than 21.

I will in this section only discuss the action/event type of condition. The distinction will however become important in the discussion of participant-internal modality.

²³The goal part of goal-oriented sentence is obviously not a sentence from a syntactic point of view but is more a kind of a verb phrase. To be more precise, the goal argument can be analyzed similarly to purpose clauses in (Bach 1982, p42), that is, as structures of the form:

1. *to VP*,
2. *For NP to VP*.

I will assume as a simplification that the *VP* of 1 is controlled by the subject of the condition (in the second form –which I will not discuss– the *VP* is controlled by the *NP*). Therefore I will treat the goal as a simple (untensed) declarative *Subject VP*. Notice that the goal part of goal-oriented sentences displays some differences from purpose clauses (for instance the latter cannot be preposed (Bach 1982, p36)). As such those goals look more like Bach’s *in-order-to* clauses even though the match is not perfect.

Finally it is questionable whether we should add complex *VP*’s to the analysis (with negation, conjunction etc...). Those more complex goals will not be discussed either.

It is also important to realize that the plan associated with a goal is by no means sufficient to achieve the goal. For instance, I might be aware that to go to Boston I must take the airplane, as in 4 of example 5.2.2, but still not know (as is the case in planning system Π_3) how to take the airplane.

Definition 5.2.3 (Possibilities and information states). A possibility is an ordered pair (s, Π) consisting of a situation s and a planning system Π . An information state is a set of possibilities and the minimal information state is $\mathbf{0} = \{(\emptyset, \emptyset)\}$.

As should be clear from the above examples, part of the meaning of a goal-oriented sentence is to do the following:

1. add a goal/plan pair to the planning system if there was no such pair with this goal,
2. otherwise update the plan of this goal with the expressed condition.

For instance the update of the minimal planning system with sentence (15-a) adds a new goal/plan pair to the planning system such that the goal is to go to Texel and the plan says that you have to take a boat. However this is not enough. At this stage we would add for any new goal-oriented sentence with a goal not yet present in the planning system a new pair. This is insufficient in two cases.

In the first case, the planning system contains for instance a pair $(a, \{\{\langle b, 1 \rangle\}\})$ and gets updated with a sentence of the form *to b, have to c*, that is, the goal of the sentence is present as part of a plan to achieve another goal.

$$(a, \{\{\langle \underline{b}, 1 \rangle\}\}) \in \Pi \text{ updated with } \textit{to } \underline{b}, \textit{ have to } c$$

Consider you have been told sentence (15-a) (for instance, you hold planning system Π_1 of example 5.2.2) and now you are told the following sentence:

(16) To take a boat, you have to go to the seaport.

Obviously the result of an update with this sentence will be at least to add a goal/plan pair (for instance $(\textit{boat}, \{\{\langle \textit{seaport}, 1 \rangle\}\})$ if nothing is known about this goal in the information state) to the planning system of all possibilities. However we surely get to know more than just that, in particular, the following sentence should intuitively be accepted:

(17) To go to Texel, you have to go to the seaport.

There is thus some kind of transitivity at work. Sentence (16) not only creates its own goal/plan pair but also influences the plans where its goal appears as a means to achieve another goal. Intuitively the update with sentence (16) should proceed as follows:

1. add the goal/plan pair $(\textit{boat}, \{ \boxed{\textit{seaport}} \})$ to the planning system and

2. update the pair $(Texel, \{\boxed{\text{boat}}\})$ to $(Texel, \{\begin{array}{|c|} \hline \text{boat} \\ \hline \text{seaport} \\ \hline \end{array}\})$.

Therefore an update with a goal-oriented sentence with goal a also has to update all the pairs where a appears not as a goal but as part of the plan.

In the second case, the planning system contains for instance a pair $(a, \{\{\langle b, 1 \rangle\}\})$ and gets updated with a sentence of the form $to\ c$, have to a , that is, a plan to achieve the condition of the goal-oriented sentence is already known.

$$(\underline{a}, \{\{\langle b, 1 \rangle\}\}) \in \Pi \text{ updated with } to\ c, \text{ have to } \underline{a}$$

Suppose you do not have any known plans (the planning systems are empty) and you learn sentence (16). The planning system then becomes $\{(boat, \{\boxed{\text{seaport}}\})\}$. Now you are told sentence (15-a). Just as before we would like to accept sentence (17). This means that the update should add consistently the plan that might be known for the condition of the goal-oriented sentence:

1. add to the planning system the goal/plan pair $(Texel, \{\boxed{\text{boat}}\})$ to which the already known plan to achieve $boat$ is added. Thus add to the planning system $(Texel, \{\boxed{\text{boat}} \cup \boxed{\text{seaport}}\}) = (Texel, \{\begin{array}{|c|} \hline \text{boat} \\ \hline \text{seaport} \\ \hline \end{array}\})$.

The last case shows that the basic update of a plan with a sentence (condition) within a planning system must be completed by the update with the plan of this sentence as goal if available. This is the right moment to give the definition of the basic update of a plan within a planning system.

Definition 5.2.4 (Basic update of a plan). Let π be a plan of a planning system Π and $a \in \mathcal{D}$.

$$\begin{aligned} \pi \uparrow a &= \{t' \mid t' = t \cup \{\langle a, 1 \rangle\} \cup s \text{ for some } s \in \tau, \text{ if } (a, \tau) \in \Pi, \\ &\quad \text{and } s = \emptyset \text{ otherwise, and for some } t \in \pi\} \\ \pi \downarrow a &= \{t' \mid t' = t \cup \{\langle a, 0 \rangle\} \text{ for some } t \in \pi\} \end{aligned}$$

Example 5.2.5. Let $(Boston, \{\{\langle airplane, 1 \rangle\}, \{\langle boat, 1 \rangle\}\})$ be a pair in a planning system Π . Intuitively this pair represents for instance the information that to go to Boston you have to either take the airplane or the boat. Assume further that we have to add a new pair (due to the update with a goal-oriented sentence) for which the condition is that you have to go to Boston. That is, the plan for this new goal will be the following:

$$\begin{aligned} &\{\emptyset\} \uparrow Boston \\ = &\{t' \mid t' = \emptyset \cup \{\langle Boston, 1 \rangle\} \cup s \text{ for some } s \in \{\{\langle airplane, 1 \rangle\}, \{\langle boat, 1 \rangle\}\}\} \\ &\text{because } (Boston, \{\{\langle airplane, 1 \rangle\}, \{\langle boat, 1 \rangle\}\}) \in \Pi \\ = &\{\{\langle Boston, 1 \rangle\} \cup \{\langle airplane, 1 \rangle\}, \{\langle Boston, 1 \rangle\} \cup \{\langle boat, 1 \rangle\}\} \\ = &\{\{\langle Boston, 1 \rangle, \langle airplane, 1 \rangle\}, \{\langle Boston, 1 \rangle, \langle boat, 1 \rangle\}\} \end{aligned}$$

The first case discussed above will be taken care of by the definition of an update with a goal-oriented sentence. The idea is that a goal-oriented sentence changes also pairs of the planning system where the goal appears in a to-do list.

Definition 5.2.6 (Update with *to a*, have to φ). Let σ be an information state, $a \in \mathcal{D}$ a simple declarative and φ a sentence. Π^a will denote the set containing the updated goal/plan pair with a as goal of a planning system Π :

$$\begin{aligned} \Pi^a = & \{(a, (\pi^a \uparrow \varphi)_{cons}) \mid \pi^a = \pi \text{ if } (a, \pi) \in \Pi, \{\emptyset\} \text{ otherwise}\} \\ & \text{if } (\pi^a \uparrow \varphi)_{cons} \text{ extends } \pi^a, \\ & \emptyset \text{ otherwise.} \end{aligned}$$

Furthermore, Π^d (for $d \neq a$) will denote the set containing the update of pair $(d, \pi^d) \in \Pi$. As explained before the update consists in updating the to-do lists containing a :

$$\begin{aligned} \Pi^d = & \{(d, \{t \mid t \in \pi^d \ \& \ \langle a, 1 \rangle \notin t\} \cup (\{t \mid t \in \pi^d \ \& \ \langle a, 1 \rangle \in t\} \uparrow \varphi)_{cons})\} \\ & \text{if } (\{t \mid t \in \pi^d \ \& \ \langle a, 1 \rangle \in t\} \uparrow \varphi)_{cons} \text{ extends } \{t \mid t \in \pi^d \ \& \ \langle a, 1 \rangle \in t\}, \\ & \emptyset \text{ otherwise.} \end{aligned}$$

Finally, the update of information state σ with the goal-oriented sentence *to a*, have to φ is as follows: if σ factually subsists in the update,

$$\begin{aligned} \sigma \uparrow \textit{to a, have to } \varphi &= \{(s, \Pi^a \cup \bigcup_{d \neq a}^{(d, \pi) \in \Pi} \Pi^d) \mid (s, \Pi) \in \sigma \ \& \ \Pi^a \neq \emptyset \ \& \ \bigcap_{d \neq a}^{(d, \pi) \in \Pi} \Pi^d \neq \emptyset\} \\ &= \Lambda \text{ otherwise.} \end{aligned}$$

Finally notice that $\sigma \uparrow \textit{to a, } \neg \textit{can } \varphi = \sigma \uparrow \textit{to a, have to } \neg \varphi$.

Therefore the update with *to a*, have to b is a “two-step” process: first the planning system has to be updated for the goal a as argument by b and everything that is needed to achieve b (i.e. the plan of b), then if a belongs to the to-do list of some plan of the planning system (with a goal different from a), we update this to-do list with b and everything that is needed to achieve b . As for deontic updates, we end up in the absurd state whenever contradictory information about plans is added.

Example 5.2.7 (Some updates with goal-oriented necessity).

1. $\mathbf{0} \uparrow \textit{to a, have to } b = \{(\emptyset, \emptyset)\} \uparrow \textit{to a, have to } b = \{(\emptyset, \{(a, \{\boxed{b}\})\})\}$,
2. $\mathbf{0} \uparrow \textit{to a, have to } b \uparrow \textit{to b, have to } c = \{(\emptyset, \{(a, \{\boxed{b}\})\}), (b, \{\boxed{c}\})\}$,
and $\mathbf{0} \uparrow \textit{to a, have to } b \uparrow \textit{to b, have to } c \Vdash \textit{to a, have to } c$,

3. $\mathbf{0} \uparrow to a$, have to $b \uparrow to c$, have to $a = \{(\emptyset, \{(a, \{\boxed{b} \ \square\})\}), (c, \{\boxed{\begin{smallmatrix} a \\ b \end{smallmatrix}} \ \square\})\}\}$,
and $\mathbf{0} \uparrow to a$, have to $b \uparrow to c$, have to $a \Vdash to c$, have to b ,
4. administrative loophole: $\mathbf{0} \uparrow to a$, have to $b \uparrow to b$, have to $a = \{(\emptyset, \{(a, \{\boxed{\begin{smallmatrix} a \\ b \end{smallmatrix}} \ \square\})\}), (b, \{\boxed{\begin{smallmatrix} a \\ b \end{smallmatrix}} \ \square\})\}\}$.

Definition 5.2.8 (Update with $to a$, can φ). Let σ be an information state, $a \in \mathcal{D}$ a simple declarative and φ a sentence. Π^a will denote the set containing the updated goal/plan pair with a as goal of a planning system Π :

$$\begin{aligned} \Pi^a = & \{(a, \pi^a \cup (\pi_b^a \uparrow \varphi)_{cons}) \mid \pi^a = \pi \text{ if } (a, \pi) \in \Pi, \{\emptyset\} \text{ otherwise}\} \\ & \text{if } (\pi_b^a \uparrow \varphi)_{cons} \text{ extends } \{\emptyset\} \uparrow \varphi, \\ & \emptyset \text{ otherwise.} \end{aligned}$$

Furthermore, Π^d (for $d \neq a$) will denote the set containing the update of pair $(d, \pi^d) \in \Pi$:

$$\begin{aligned} \Pi^d = & \{(d, \pi^d \cup (\{t \mid t \in \pi^d \ \& \ \langle a, 1 \rangle \in t\}_b \uparrow \varphi)_{cons})\} \\ & \text{if } (\{t \mid t \in \pi^d \ \& \ \langle a, 1 \rangle \in t\}_b \uparrow \varphi)_{cons} \text{ extends } \{\emptyset\}_b \uparrow \varphi, \\ & \emptyset \text{ otherwise.} \end{aligned}$$

Finally, the update of information state σ with the goal-oriented sentence $to a$, can φ is as follows: if σ factually subsists in the update,

$$\begin{aligned} \sigma \uparrow to a, \text{ can } \varphi = & \{(s, \Pi^a \cup \bigcup_{d \neq a}^{\langle d, \pi \rangle \in \Pi} \Pi^d) \mid (s, \Pi) \in \sigma \ \& \ \Pi^a \neq \emptyset \ \& \ \bigcap_{d \neq a}^{\langle d, \pi \rangle \in \Pi} \Pi^d \neq \emptyset\} \\ = & \Lambda \text{ otherwise.} \end{aligned}$$

Finally notice that $\sigma \uparrow to a, \neg\text{have to } \varphi = \sigma \uparrow to a, \text{ can } \neg\varphi$.

Example 5.2.9.

1. $\mathbf{0} \uparrow to a$, can $b = \{(\emptyset, \{(a, \{\ \square \ \square \}), \boxed{b} \ \square\})\}\}$,
2. $\mathbf{0} \uparrow to a$, can $b \uparrow to a$, can $c = \{(\emptyset, \{(a, \{\ \square \ \square \}), \boxed{b} \ \square \ \square, \boxed{c} \ \square\})\}\}$,
3. $\mathbf{0} \uparrow to a$, can $b \uparrow to b$, can $c = \{(\emptyset, \{(a, \{\ \square \ \square \}), \boxed{b} \ \square \ \square, \boxed{\begin{smallmatrix} b \\ c \end{smallmatrix}} \ \square\}), (b, \{\ \square \ \square, \boxed{c} \ \square\})\}\}$.

Epistemic and goal-oriented modality

As the reader might already expect, combinations of epistemic and goal-oriented modality are only meaningful when the epistemic modal scopes over the deontic one.

- (18) a. To go to Texel, you might have to rent a boat.
 b. #To go to Texel, you have to maybe rent a boat.

Formally this means that a sentence of the form of (18-a) corresponds to an update with $\text{might}(to\ Texel, \text{have to } rent)$.

$$\begin{aligned}
 & \mathbf{0} \uparrow \text{might}(to\ Texel, \text{have to } rent) \\
 = & \{(\emptyset, \emptyset)\} \uparrow \text{might}(to\ Texel, \text{have to } rent) \\
 = & \{(\emptyset, \emptyset)\} \cup \{(\emptyset, \emptyset)\} \uparrow to\ Texel, \text{have to } rent \\
 = & \{(\emptyset, \emptyset)\} \cup \{(\emptyset, \{(Texel, \{\{\langle rent, 1 \rangle\})\})\})\} \\
 = & \{(\emptyset, \emptyset), (\emptyset, \{(Texel, \{\{\langle rent, 1 \rangle\})\})\})\}
 \end{aligned}$$

The update with a sentence of the form $to\ a, \text{have to } \text{might } \varphi$ fails for the same reason as in the deontic case: plans cannot be updated with a might-sentence (which operates on information states). Stacking goal-oriented operators also causes a failure in interpretation as those sentences need to be interpreted on information states, not on plans. We can therefore complete the predictions made in the previous section as follows:

1. Epistemic modals can scope over participant-external ones.
2. Epistemic modals cannot be interpreted under participant-external ones.
3. Participant-external modal operators cannot be stacked.²⁴

5.2.2 Participant-internal modality

Participant-internal modality is the last level of the system. The main intuition I would like to convey is that the contribution of an ability sentence is twofold: on the one side it is just a simple declarative sentence stating a fact about the world we are in, but on the other side it triggers a process of control of this information with respect to the information contained in the planning system. I will thus add an operator for ability to the language:

²⁴At this point this statement is too strong as I have not yet provided the general system that models both deontic and goal-oriented modals. This is however quite easy to see that such a system is obtained by redefining possibilities as triples (s, δ, Π) of a situation s , a deontic plan δ and a planning system Π (and by reformulating the updates accordingly).

Definition 5.2.10. For any simple declarative sentence $a \in \mathcal{D}$, the sentence able a is also in the language.²⁵

As we will see, some of the simplifying assumptions concerning deontic and goal-oriented modals made earlier turn out to be counterintuitive for ability. In particular it is difficult to avoid altogether the topic of agency. So far we mainly restricted ourselves to sentences involving the agent or holder of the information state as being the addressee of obligation/permissions or as being the agent of the goal-oriented sentences. I claim that this simplification is harmless in the sense that it is only meant to highlight the ideas behind the definitions. I would like to continue the presentation of the system with the same assumptions. However this means that in the case of participant-internal modality, we would have to deal with a sentence such as the following one:

(19) You are able to go to Texel.

It seems clear that you are the best judge to confirm or not this sentence and that my saying so to you should not change your information state much. This is not the case if, for instance, the subject of the same sentence is John.

(20) John is able to go to Texel.

In particular by updating your information state with this sentence you should also be able to conclude (remembering sentence (15-a)) that John is able to take a boat. This means that we also need a planning system for John in our information state.²⁶ Instead of a simple possibility $(s, \delta, \Pi) \in \sigma$ where the deontic plan and the planning system are those of the holder of the state, we would need to expand the notions of deontic plans and planning systems to sets of agents δ 's and Π 's, i.e. something like the following:

$$(s, \{(me, \delta_{me}), (John, \delta_{john}), \dots\}, \{(me, \Pi_{me}), (John, \Pi_{John}), \dots\})^{27}$$

²⁵We might change the definition of consistency depending on the strength we want to attribute to ability sentences. By that I mean that we might add to the definition of consistency that a consistent situation or to-do list may not contain simultaneously $\langle a, 1 \rangle$ and $\langle \text{able } a, 0 \rangle$. However I make the choice to assume that we cannot in general conclude statements about ability from the (possibly accidental) occurrence of an event. Notice for instance that the Lillooet participant-internal modal *ka-...-a* can express both meanings, accidental and ability, and that we surely cannot conclude an ability sentence from the accidental reading.

²⁶The reader might already have realized that the *you* in sentence (15-a) is most often interpreted generically. That is, this planning information applies to all agent and is therefore by default on all agent's planning systems. Of course, some planning information is agent-specific. Notice that we might as well have deontic information pertaining to John. This move which would be necessary in a complete formalization provides a solution to the problem of symmetric predicates as the addressee of the obligation/permission is singled out. Therefore the sentence *John must shake hands with Bob* with the symmetric predicate *shake hands with* does not entail *Bob must shake hands with John*.

²⁷Notice that we would probably need a kind of generic planning system for planning informa-

In order to keep things simple I will neglect this issue and continue the exposition of the system without mentioning agency as the mechanisms at hand only depend on the fact that an agent and “his” planning system can be identified.

We need to come back to the problems of the standard framework with participant-internal modality noted in the previous chapter. There were three main problems:

1. The asymmetry of participant-internal modality: possibility vs necessity (where the first is pervasive and the second marginal).
2. Embedded disjunction (Kenny 1976).
3. Epistemic possibility entails participant-internal possibility.

I will not provide a radical solution for the first two problems. In the case of the asymmetry, I propose to analyze participant-internal necessity as a dummy category corresponding to the use of two negations with participant-internal possibility: participant-internal *have to* := \neg able \neg . However, only simple declaratives will be allowed in this section under the ability operator. The full formalization will thus have to be left as future work. This being said Kenny’s problem is taken care of in a trivial but of course not satisfying way: boolean combinations are not yet allowed under ability. Contrary to the two first problems, the last one is solved in a non-trivial way. Ability statements as *able a* are statements as simple declaratives: they are the case or not. Epistemic modality does not say anything about ability unless it embeds an ability modal. Therefore we cannot in general conclude an ability statement from an epistemic one.

I will now turn to the formalization. Goal-oriented and participant-internal modality are intimately connected. In particular, it is natural to pose the following conditions for participant-internal modality given the interpretation of goal-oriented sentences:²⁸

1. if you are able to do something, you are able to do the things that are necessary to do it. For all $a, b \in \mathcal{D}$: *to a*, *have to b*; *able a* \Vdash *able b*.
2. If you are not able to do something, then you are not able to do the things which necessitate it to be done. For all $a, b \in \mathcal{D}$: *to a*, *have to b*; \neg able *b* \Vdash \neg able *a*.

tion that is not linked to an agent (world knowledge). Remember the example of last chapter, *Hydrangeas can grow here*, where we surely do not want to analyze hydrangeas as agents.

²⁸At this stage the list of conditions does not aim at exhaustivity. For instance, the following conditions all make sense too:

1. For all $a, b, c \in \mathcal{D}$: \neg able *b*; *to a*, *have to (b \vee c)* \Vdash *to a*, *have to c*.
2. For all $a, b \in \mathcal{D}$: *able a*; *to a*, *have to b* \Vdash *able b*.

Those conditions suggest in particular that the update for goal-oriented sentences with *have to* should be amended. This is easily done with the help of definition 5.2.11. Replace $(\pi^a \uparrow \varphi)_{cons}$ in definition 5.2.6 by $(\pi^a_{exec[s]} \uparrow \varphi \cup \overline{\pi_{exec[s]}})_{cons}$.

3. For all $a, b, c \in \mathcal{D}$: $to\ a$, have to $(b \vee c)$; \neg able b ; able $a \Vdash$ able c .

The first two conditions speak for themselves but the last one is in need of a deeper analysis. In the last case, we know what is necessary to achieve a goal a : either do b or do c . By the fact that we also know that the goal can be achieved but b can't, we neglect the to-do lists containing b and conclude that c can be achieved. That is, we concentrate on the to-do lists that are executable.

Definition 5.2.11 (Executable to-do lists). Let (s, Π) be a possibility. The set of executable to-do lists of a goal/plan pair in Π with respect to the situation is the set of all to-do lists that do not contain an action that the agent is not able to perform in this situation:

$$\pi_{exec[s]} = \{t \mid t \in \pi \ \& \ \text{if } \langle \text{able } a, 0 \rangle \in s, \langle a, 1 \rangle \notin t\},$$

with $\pi - \pi_{exec[s]} = \overline{\pi_{exec[s]}}$.

The main difference between a normal declarative sentence and a participant-internal modal is that the participant-internal modal triggers a kind of consistency check with respect to the planning system that will add the relevant participant-internal information that can be derived from the update. That is, the update with a participant internal sentence triggers a planning system check:

Definition 5.2.12 (Planning system check). Let s be a situation and Π a planning system. The set $(s)_{\Pi}$ is the situation that obtains by the addition of relevant participant-internal sentences deduced from participant-internal sentences in s and the planning system Π :

$$\begin{aligned} (s)_{\Pi} = & s \cup \\ & \bigcup_{\substack{a \in \mathcal{D}, \\ \langle \text{able } a, 1 \rangle \in s}} \{ \langle \text{able } d, 1 \rangle \mid \langle d, 1 \rangle \in \bigcap \pi_{exec[s]}^a, \pi^a = \begin{array}{l} \pi, \text{ if } (a, \pi) \in \Pi, \\ \{\emptyset\} \text{ otherwise} \end{array} \} \cup \\ & \bigcup_{\substack{a \in \mathcal{D}, \\ \langle \text{able } a, 0 \rangle \in s}} \{ \langle \text{able } d, 0 \rangle \mid \langle a, 1 \rangle \in \bigcap \pi_{exec[s - \{\langle \text{able } a, 0 \rangle\}]}^d, \\ & \text{for some } d \text{ such that } (d, \pi^d) \in \Pi \} \end{aligned}$$

We can now give a definition of participant-internal modality.

Definition 5.2.13 (Participant-internal update). Let σ be an information state, $a \in \mathcal{D}$ a simple declarative.

$$\begin{aligned} \sigma \uparrow \text{able } a &= \{ ((s \cup \{\langle \text{able } a, 1 \rangle\})_{\Pi}, \Pi) \mid \begin{array}{l} (s, \Pi) \in \sigma \ \& \\ (s \cup \{\langle \text{able } a, 1 \rangle\})_{\Pi} \text{ consistent} \end{array} \} \\ \sigma \downarrow \text{able } a &= \{ ((s \cup \{\langle \text{able } a, 0 \rangle\})_{\Pi}, \Pi) \mid \begin{array}{l} (s, \Pi) \in \sigma \ \& \\ (s \cup \{\langle \text{able } a, 0 \rangle\})_{\Pi} \text{ consistent} \end{array} \} \end{aligned}$$

Example 5.2.14. Suppose you know the following goal-oriented sentences:

- (21) a. To go to Texel, you have to take the ferry or go with a rental boat.
 b. To go with a rental boat, you must drive the boat.

This gives the information state (as update on the minimal information state),

$$\begin{aligned} & \mathbf{0} \uparrow (21\text{-a}) \uparrow (21\text{-b}) \\ = & \{(\emptyset, \{ (Texel, \{ \boxed{\text{ferry}} \mid \boxed{\text{boat}} \mid \boxed{\text{drive}} \}) , (boat, \{ \boxed{\text{drive}} \}) \}) \} \end{aligned}$$

which we will abbreviate as $\{(\emptyset, \Pi_0)\}$. I will now go through the steps of the update with *John is able to go to Texel*:

$$\mathbf{0} \uparrow (21\text{-a}) \uparrow (21\text{-b}) \uparrow \text{able } Texel = \{ (\{ \langle \text{able } Texel, 1 \rangle \}, \Pi_0) \}$$

Proof sketch:

$$\begin{aligned} & \mathbf{0} \uparrow (21\text{-a}) \uparrow (21\text{-b}) \uparrow \text{able } Texel \\ = & \{ ((\emptyset \cup \{ \langle \text{able } Texel, 1 \rangle \})_{\Pi_0}, \Pi_0) \mid (\emptyset \cup \{ \langle \text{able } Texel, 1 \rangle \})_{\Pi_0} \text{ consistent} \} \\ & \text{by definition 5.2.13} \\ = & \{ ((\{ \langle \text{able } Texel, 1 \rangle \})_{\Pi_0}, \Pi_0) \mid (\{ \langle \text{able } Texel, 1 \rangle \})_{\Pi_0} \text{ consistent} \} \\ = & \{ (\{ \langle \text{able } Texel, 1 \rangle \}, \Pi_0) \} \end{aligned}$$

because

$$\begin{aligned} & (\{ \langle \text{able } Texel, 1 \rangle \})_{\Pi_0} = (s_1)_{\Pi_0} \\ = & \{ \langle \text{able } Texel, 1 \rangle \} \cup \bigcup_{\substack{a \in \mathcal{D} \\ \langle \text{able } a, 1 \rangle \in s_1}} \{ \langle \text{able } d, 1 \rangle \mid \langle d, 1 \rangle \in \bigcap \pi_{exec[s_1]}^a, \\ & \pi^a = \pi \text{ if } (a, \pi) \in \Pi, \{ \emptyset \} \text{ otherwise} \} \\ = & \{ \langle \text{able } Texel, 1 \rangle \} \cup \{ \langle \text{able } d, 1 \rangle \mid \langle d, 1 \rangle \in \bigcap \pi_{exec[s_1]}^{Texel}, (Texel, \pi^{Texel}) \\ & \in \Pi_0 \} \text{ with } \bigcap \pi_{exec[s_1]}^{Texel} = \{ \langle \text{ferry}, 1 \rangle \} \cap \{ \langle \text{boat}, 1 \rangle, \langle \text{drive}, 1 \rangle \} = \emptyset \\ = & \{ \langle \text{able } Texel, 1 \rangle \} \end{aligned}$$

Q.E.D

Therefore, by learning that John is able to go to Texel, we cannot derive extra information. But now if we learn that John is not able to drive a boat, we obtain this information state:

$$\begin{aligned} & \mathbf{0} \uparrow (21\text{-a}) \uparrow (21\text{-b}) \uparrow \text{able } Texel \uparrow \neg \text{able } drive \\ = & \{ (\{ \langle \text{able } Texel, 1 \rangle, \langle \text{able } drive, 0 \rangle, \langle \text{able } ferry, 1 \rangle, \langle \text{able } boat, 0 \rangle \}, \Pi_0) \} \end{aligned}$$

Proof sketch:

$$\begin{aligned}
& \mathbf{0} \uparrow (21\text{-a}) \uparrow (21\text{-b}) \uparrow \text{able } Texel \uparrow \neg\text{able } drive \\
= & \{ (\{ \langle \text{able } Texel, 1 \rangle \} \cup \{ \langle \text{able } drive, 0 \rangle \})_{\Pi_0}, \Pi_0 \} \mid (\{ \langle \text{able } Texel, 1 \rangle \} \cup \\
& \{ \langle \text{able } drive, 0 \rangle \})_{\Pi_0} \text{ consistent} \} \quad \text{by definition 5.2.13} \\
= & \{ (\{ \langle \text{able } Texel, 1 \rangle, \langle \text{able } drive, 0 \rangle \})_{\Pi_0}, \Pi_0 \} \mid (\{ \langle \text{able } Texel, 1 \rangle, \\
& \langle \text{able } drive, 0 \rangle \})_{\Pi_0} \text{ consistent} \} \\
= & \{ (\{ \langle \text{able } Texel, 1 \rangle, \langle \text{able } drive, 0 \rangle, \langle \text{able } ferry, 1 \rangle, \langle \text{able } boat, 0 \rangle \}, \Pi_0) \}
\end{aligned}$$

because

$$\begin{aligned}
& (\{ \langle \text{able } Texel, 1 \rangle, \langle \text{able } drive, 0 \rangle \})_{\Pi_0} = (s_2)_{\Pi_0} \\
= & \{ \langle \text{able } Texel, 1 \rangle, \langle \text{able } drive, 0 \rangle \} \cup \\
& \left\{ \bigcup_{\substack{a \in \mathcal{D} \\ \langle \text{able } a, 1 \rangle \in s_2}} \{ \langle \text{able } d, 1 \rangle \mid \langle d, 1 \rangle \in \bigcap \pi_{exec[s_2]}^a, \pi^a = \begin{cases} \pi & \text{if } (a, \pi) \in \Pi, \\ \emptyset & \text{otherwise} \end{cases} \} \right. \\
& \left. \cup \bigcup_{\substack{a \in \mathcal{D} \\ \langle \text{able } a, 0 \rangle \in s_2}} \{ \langle \text{able } d, 0 \rangle \mid \langle a, 1 \rangle \in \bigcap \pi_{exec[s_1]}^d, \text{ for some } d \text{ s.t. } (d, \pi^d) \in \Pi \} \right\} \\
= & \{ \langle \text{able } Texel, 1 \rangle, \langle \text{able } drive, 0 \rangle \} \cup \\
& \{ \langle \text{able } d, 1 \rangle \mid \langle d, 1 \rangle \in \bigcap \pi_{exec[s_2]}^{Texel}, \text{ with } \pi_{exec[s_1]}^{Texel} = \{ \{ \langle ferry, 1 \rangle \} \} \} \} \cup \\
& \{ \langle \text{able } boat, 0 \rangle \mid \langle drive, 1 \rangle \in \bigcap \pi_{exec[s_1]}^{boat} = \{ \langle drive, 1 \rangle \} \} \\
= & \{ \langle \text{able } Texel, 1 \rangle, \langle \text{able } drive, 0 \rangle, \{ \langle \text{able } ferry, 1 \rangle, \langle \text{able } boat, 0 \rangle \} \}
\end{aligned}$$

Q.E.D

Combinations with participant-internal modality

Given the definition of participant-internal modality it is quite obvious that it should be possible to embed it under the other types of modality but that the reverse should be uninterpretable. This is due to the fact that, when interpreted on information states, ability sentences are basically simple declarative sentences that trigger a checking mechanism on the planning systems. This mechanism is just not triggered when an ability sentence updates plans. In this case the ability sentence is just treated as a simple declarative sentence.

Therefore epistemic modals can embed participant-internal sentences and the reverse is not interpretable.

- (22) a. John might be able to go to Texel.
b. #John is able to maybe go to Texel.

It is also easy to realize that deontic modals easily embed participant-internal modals²⁹ but that the reverse is not interpretable either.³⁰

²⁹To be formal and precise, we need to reformulate definition 5.1.22 to allow ability sentences.

³⁰Notice that the system cannot yet predict the fact that permission sentences embedding an

- (23) a. (In view of what his contract provides) John must be able to speak Dutch for his new job.
 b. #John is able to have to speak Dutch for his new job.

Finally although combinations with goal-oriented modality need some more work, notice first that ability cannot scope over goal-oriented modals just as it could not scope over other modals. Goal-oriented modals may embed ability statements:

- (24) a. To play polo, you must be able to ride a horse.
 b. #To play polo, you are able to have to ride a horse.

As such this sentence is not a problem, that is, the update works as it should. However, we run into trouble if we further update with the sentence *John can play polo*. The constraint on participant-internal modality fails as it should add the ill-formed $\langle \text{able able } polo, 1 \rangle$. This problem is easily solved although it comes at the expense of more involved definitions. Notice first that the problem is in some sense not limited to participant-internal modality. Remember the hydrangeas example of last chapter:³¹

- (25) a. In order for hydrangeas to grow, the climate must be temperate.
 b. Hydrangeas can grow here.

From these two sentences we do not want to conclude (26-a), as the definitions would do, but (26-b).

- (26) a. #The climate is able to be temperate.³²
 b. The climate is temperate.

The sentence embedded under the goal-oriented sentence, (26-b), is clearly stative. I claim that ability sentences also are stative and that this property explains the data. The idea is that when a stative sentence is required to achieve a goal and when we know that this goal can be achieved by an agent then we know that this

ability often sound strange.

- (i) John may be able to speak Dutch.

This kind of sentence mainly makes sense for a theater director that is giving shape to a character. A possible explanation for this problem is that the deontic modal gives to the grammatical subject *John* some choice but that having an ability is not something that is decided by an agent but something that needs to be acquired.

³¹If the reader is distracted by the fact that, as already mentioned, hydrangeas are not easily seen as agents, he can use the following variant:

- (i) a. To read a book in the tent at night, John must have a flashlight.
 b. John can read a book in the tent at night.

³²I use the modal *able to* here on purpose as the variant with *can*, *the climate can be temperate*, does not sound as bad although it loses its pure ability reading and has an occasional reading.

sentence, and not the sentence under ability, is the case:

$$\begin{aligned} & \{ (\emptyset, \{(a, \{ \begin{array}{|l|} \hline \text{state} \\ \hline \text{event} \\ \hline \end{array} \})\}) \} \uparrow \text{able } a \\ = & \{ (\{ \langle \text{able } a, 1 \rangle, \langle \text{state}, 1 \rangle, \langle \text{able event}, 1 \rangle \}, \{ \begin{array}{|l|} \hline \text{state} \\ \hline \text{event} \\ \hline \end{array} \}) \} \} \end{aligned}$$

We therefore need to improve the second part of definition 5.2.12 of the planning system check by differentiating between types of sentences (states vs events). Events add an ability statement to the situation whereas states add themselves. We can rephrase the original condition as follows:

1. If you are able to do something, you are able to do the actions (perform the events) that are necessary to do it and the preconditions/states that need to be the case are the case.

I will not give the precise formulation of the new rule as it should be obvious that the change is not problematic as soon as we can differentiate in the language between states and events. That is what the following definition does.

Definition 5.2.15. We make a distinction between events and states in the language. Within the set of simple declarative sentences \mathcal{D} , we therefore have a split between events and states: $\mathcal{D} = \mathcal{D}_{state} \cup \mathcal{D}_{event}$. For any simple declarative sentence $a \in \mathcal{D}$, the sentence able a is a state.

The last remark is that the second part of definition 5.2.12 also needs to account for this new distinction in the language. The intuitive formulation of the condition becomes thus:

2. If you are not able to perform some event x or some precondition/state y is not the case, then you are not able to do the things which necessitate x to be done or y to be the case.

$$\begin{aligned} & \{ (\emptyset, \{(a, \{ \begin{array}{|l|} \hline \text{state} \\ \hline \text{event} \\ \hline \end{array} \})\}) \} \uparrow \neg\text{state} \\ = & \{ (\{ \langle \text{able } a, 0 \rangle, \langle \text{state}, 0 \rangle \}, \{ \begin{array}{|l|} \hline \text{state} \\ \hline \text{event} \\ \hline \end{array} \}) \} \} \end{aligned}$$

Obviously definition 5.2.12 can be amended without problem to take this idea into account. However, this is a departure from the idea expressed at the beginning of this section that the ability modals trigger a planning system check. This last step necessitates also performing the planning system check when we update with (negated) stative sentences. We could for instance generalize the planning system check to any update with simple declaratives (including ability sentences) in definition 5.1.12. I will unfortunately have to leave this for future work.