Logics of communication and knowledge
Sietsma, F.A.G.

Citation for published version (APA):

General rights
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, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 3
Message Passing in Dynamic Epistemic Logic

3.1 Introduction

In this chapter I show how one can model the dynamics of knowledge during communication using epistemic logic. I will focus on a situation where a number of agents communicate using messages from a finite set which is known by all agents. This set is not fixed: during the message exchange, new messages may be added to the set. Such a set up is relevant in numerous situations. For example, one could think of computers communicating in accordance with a fixed protocol, or people playing a game where they have to give certain signals every round. The example of the two generals mentioned in the introduction could also be modeled this way, where the possible messages are the possible days of attack. However, in this chapter I will assume that the communication channel is reliable, so every message that is sent is also received. This is clearly not the case for the generals. I also assume that the communication is synchronous, so all the messages that are sent are immediately received.

I will use Kripke models to model the state of knowledge at a particular moment. Given some message that is sent and received, I use its structure to generate the Kripke model that represents the new state of knowledge after reception of the message. This way, sequences of Kripke models can be constructed from sequences of messages. These sequences show how the knowledge of the agents changes over time.

The system is designed for reasoning about sequences of messages that have been sent and received, given some initial situation represented by a Kripke model. The semantics is designed in a back-and-forth fashion: a Kripke model of the current situation determines which communication steps are successful on that model, and each communication step gives rise to an adaptation of the model to a new Kripke model, which again determines which successive communication steps are possible, etcetera.
In this chapter I only consider truthful communication. This means that the content of all the messages that are sent is true. Furthermore, a message can only be sent if the sender of the message knows that its content is true. Also, all messages are accepted as true, so if an agent receives a message she gains knowledge of the contents.

The semantics presented here can be used to model reasoning about the way the communication took place: agents remember which messages they sent or received, but are uncertain about which other messages were sent. This engenders uncertainties about what other agents know and about what messages they may have exchanged. The construction given in this chapter models these uncertainties in a very precise way.

The semantics allows for checking properties and effects of communication sequences that took place in the past, and allows a limited amount of reasoning about counterfactual situations, like “suppose instead of actual message \( m \), another message, \( m' \) had been sent.” Also, it allows for reasoning about properties and effects of new communication steps.

In the next section I start out with defining a logical language based on messages with a certain internal structure. In Section 3.3 I show how I use Kripke models to interpret this language and I introduce the update that models the communicative action of sending a message. In Section 3.4 I define a class of Kripke models that are a realistic result of a sequence of messages. In Section 3.5 I axiomatize the language and the two new modalities I have introduced. Finally, in Section 3.6 I discuss some related work and I conclude this chapter in Section 3.7.

### 3.2 The Language of Knowledge and Messages

In this section I will show how to incorporate messages in the epistemic language introduced in Chapter 2. Including these messages in the language allows for reasoning about how the knowledge of agents is affected by messages and the knowledge of the agents about these messages.

I will first define a simple language \( \mathcal{L}^{MPD}_0 \) that does not contain any knowledge modalities. I will use this language to represent the semantic content of the messages. Later on, I will define a richer language that can be used to reason about the messages and the knowledge of the agents.

Let \( P \) be a set of proposition letters. Let \( Ag \) be a finite set of agents.

**3.2.1. Definition.** Let \( \mathcal{L}^{MPD}_0 \) be the following language:

\[
\psi ::= p \mid (a, \psi, G) \mid \neg \psi \mid (\psi \lor \psi)
\]

where \( p \in P, a \in G \subseteq Ag \).
This is propositional logic enriched with messages. A **message** is represented by a tuple \((a, \psi, G)\) where \(a \in Ag\) is the sender of the message, \(\psi \in L_{0}^{MPD}\) is the contents of the message and \(G \subseteq Ag\) is the group of recipients of the message. The formula \((a, \psi, G)\) expresses that message \((a, \psi, G)\) was sent at some moment in the past.

I adopt the convention that a sender always receives a copy herself: any message \((a, \psi, G)\) has \(a \in G\). I will abbreviate \((a, \psi, \{a, b\})\) (a message with a single recipient, plus a copy to the sender) as \((a, \psi, b)\).

I adopt the usual abbreviations: \(\psi_1 \land \psi_2\) for \(\neg (\neg \psi_1 \lor \neg \psi_2)\) and \(\psi_1 \rightarrow \psi_2\) for \(\neg \psi_1 \lor \psi_2\).

The following is a first example of what these messages look like and how they can mention previous messages.

**3.2.2. Example.** Reply on a message \((a, p, b)\) with a quotation of the original message and some new information \(q\) can be expressed as \((b, q \land (a, p, b), a)\). Forwarding of \((a, p, b)\) by agent \(b\) to some other agent \(c\) can be expressed as \((b, (a, p, b), c)\).

This example already shows that notation can become a bit thick when nesting messages. Therefore I will often shorten notation by naming the messages \(m, m', m_1, \text{etc.}\) These names should be seen as pure abbreviations. If a message \((a, \psi, G)\) is abbreviated as \(m\) then I mean with \(s_m = a\) the sender of the message, \(c_m = \psi\) the content of the message and \(r_m = G\) the group of recipients of the message. I also use these abbreviations in the content of other messages: for example, \((b, m, c)\) is an abbreviation for the message \((b, (a, \psi, G), c)\).

**3.2.3. Example.** If \(m\) is a message, then the message \((a, \neg m, b)\) quotes message \(m\). The formula \(\neg m\) expresses that \(m\) was not sent. With the message \((a, \neg m, b)\), agent \(a\) informs agent \(b\) that \(m\) was not sent. The formula \(\neg (a, \neg m, b)\) expresses that the message \((a, \neg m, b)\) was not sent.

Note that the definition of \(L_{0}^{MPD}\) contains mutual recursion: formulas may contain messages which contain formulas. Due to this mutual recursion the language \(L_{0}^{MPD}\) is already quite expressive. Even though the content of the messages cannot contain epistemic operators, a considerable number of useful communicative situations can be expressed.

**3.2.4. Example.** Send Communication step consisting of a single message \(m\).

**Acknowledgement** Acknowledgement of the receipt of a message \(m\) can be expressed as \((b, m, s_m)\) where \(b \in r_m\).

**Reply** Reply to sending of \(m\) with reply-contents \(\psi\) can be expressed as \((b, m \land \psi, s_m)\) where \(b \in r_m\).
Forward Forwarding of \( m \) can be expressed as \((b, m, c)\) where \( b \in r_m \) and \( c \notin r_m \).

Forward with annotation Forwarding of \( m \) with annotation \( \psi \) can be expressed as \((b, m \land \psi, c)\) where \( b \in r_m \) and \( c \notin r_m \).

CC There is no distinction between addressee list and CC-list. The distinction between addressee and CC-recipient is in general a subtle matter of etiquette: usually, an addressee is supposed to reply to a message while someone on a CC-list incurs no such obligation. I think it is safe to ignore the difference here.

BCC A message \( m \) with BCC recipients \( b_1, \ldots, b_n \) can be treated as a sequence of messages \( m, (s_m, m, b_1), \ldots, (s_m, m, b_n) \). Each member on the bcc list of \( m \) gets a separate message from the sender of \( m \) to the effect that message \( m \) was sent. In Chapter 5 I will discuss a subtle difference between such a “sequence of forwards” and the actual BCC feature. I will prove in Theorem 3.3.8 that the order in which the list \((s_m, m, b_1), \ldots, (s_m, m, b_n)\) is sent does not matter.

I will set up the semantics in such a way that I can prove that any message that is forwarded was already sent at some earlier stage, and an acknowledgement never precedes a send. The fact that these properties follow from the epistemic effects of message passing is a corroboration of the appropriateness of my set-up.

The truth value of an \( \mathcal{L}_{0}^{MPD} \) formula depends not only on the truth value of the propositions in \( P \), but also on the truth value of the messages mentioned in the formula. For messages, a positive truth value means that the message was sent, and a negative truth value that it was not sent. In order to know which messages should be considered I first assign a vocabulary to every formula. This is the set of all propositions and messages that are relevant to the truth value of the formula.

3.2.5. DEFINITION. The vocabulary \( \text{voc}(\varphi) \) of a formula \( \varphi \) is defined as follows:

\[
\begin{align*}
\text{voc}(p) & := \{p\} \\
\text{voc}((a, \psi, G)) & := \{(a, \psi, G)\} \cup \text{voc}(\psi) \\
\text{voc}(\neg \psi) & := \text{voc}(\psi) \\
\text{voc}(\psi_1 \lor \psi_2) & := \text{voc}(\psi_1) \cup \text{voc}(\psi_2)
\end{align*}
\]

The following example shows how this definition works out:

3.2.6. EXAMPLE. If \( m = (a, p \lor q, b) \) and \( m' = (b, m, c) \), then

\( \text{voc}(m') = \{p, q, m, m'\} \).
There is an obvious partial order on the vocabulary of a formula. Note that vocabulary elements are either proposition letters or messages. These can be viewed as formulas, which have a vocabulary themselves. Letting $x, y$ range over vocabulary elements, I set $x \preceq y$ if $x \in \text{voc}(y)$. I set $x < y$ if $x \preceq y$ and $x \neq y$. This partially orders a vocabulary by ‘depth of embedding’. For example 3.2.6, this gives $p, q \prec m \prec m'$.

This can be used to define vocabularies per se. A vocabulary is a set of messages and proposition letters that is closed under applications of $\text{voc}$. Intuitively, what this means is that if a vocabulary contains $m$, then it also contains every proposition or message that is mentioned in $m$.

It is easy to see from this definition that the vocabulary of a formula, and hence also the vocabulary of a finite set of formulas, is always finite. Now I can give a truth definition for formulas of $\mathcal{L}_0^{MPD}$ given some valuation of their vocabulary:

**3.2.7. Definition.** Let $\Psi$ be a set of $\mathcal{L}_0^{MPD}$ formulas. Let $v$ be a subset of $\text{voc}(\Psi)$, representing the propositions that are true and the messages that are sent. Call $v$ a valuation for $\Psi$. Then truth at $v$ is defined as follows for all formulas in $\Psi$:

$$
\begin{align*}
v &\models \top & \text{always} \\
v &\models p & \text{iff } p \in v \\
v &\models m & \text{iff } m \in v \\
v &\models \neg \psi & \text{iff } v \not\models \psi \\
v &\models \psi_1 \land \psi_2 & \text{iff } v \models \psi_1 \text{ and } v \models \psi_2
\end{align*}
$$

Truth of $m$ at $v$ expresses that according to $v$ message $m$ was sent (at some time in the past).

As mentioned above, I will use a richer language with knowledge modalities to reason about the knowledge of agents and how this is influenced by message passing. I adapt the language from Chapter 2 to include messages, which leads to the following definition of the language $\mathcal{L}^{MPD}$.

$$
\phi \ ::= \ \psi \mid \neg \phi \mid \phi \lor \phi \mid (\alpha)\phi \quad \text{where } \psi \in \mathcal{L}_0^{MPD} \\
\alpha \ ::= \ a \mid ?\phi \mid \alpha; \alpha \mid \alpha \cup \alpha \mid \alpha^* \quad \text{where } a \in Ag.
$$

The semantics of this language interpreted on the world of a Kripke model is as follows. For the base case $\phi = \psi \in \mathcal{L}_0^{MPD}$ it is given by Definition 3.2.7, with respect to the valuation of the world under consideration. For the other clauses it is as in Chapter 2. Of course, this depends on a vocabulary of propositions and messages. Therefore, I will introduce Kripke models with vocabularies in the next section.
3.3 Modeling Message Passing

I will use Kripke models to represent the knowledge of agents during a sequence of message exchanges. Usual Kripke models only consider the agents’ knowledge about basic propositions. Now I also want to consider their knowledge about messages that may have been sent. In order to do this, I will explicitly add messages to the models. Because the size of the models usually increases drastically with the number of messages in the model, I propose the following modeling procedure. Model the initial situation where no messages are sent with a model with no messages. Then gradually add messages to the model as they are sent, and update the models with information concerning who knows about the messages and who does not. The following example illustrates the idea.

3.3.1. Example. Suppose the initial state is the model $\mathcal{M}_0$ from Example 2.1.9 where Alice knows about $h$, while Bob and Carol do not. Let $m$ be the message $(a, h, b)$ sent by Alice, informing Bob that $h$ is the case. Let $m'$ be the message $(b, m, c)$ sent by Bob, informing Carol that $m$ was sent. If the model $\mathcal{M}_0$ represents the initial situation, the messages can only be sent with $m$ preceding $m'$, for I assume that all messages are truthful, and the formula $m$ is not true before $m$ is sent. This gives:

$$\mathcal{M}_0 \xrightarrow{m} \mathcal{M}_1 \xrightarrow{m'} \mathcal{M}_2$$

What do the models look like? $\mathcal{M}_0$ is the Kripke model from Example 2.1.9. Omitting the names of the world, it looks as follows:

$\mathcal{M}_0$:

- $h$
- $b, c$
- $\overline{h}$

Sending message $m$ will inform Bob about $h$, while Carol still considers it possible that nothing has happened. I will not only model the knowledge that Bob gains about $h$, but also the message itself and Bob’s knowledge of it.

$\mathcal{M}_1$:

- $h, m$
- $b, c$
- $\overline{h}, \overline{m}$
- $c$

- $h, m$
3.3. Modeling Message Passing

This model has three worlds: one where \( h \) is true and the message \( m \) is sent, one where \( h \) is true and the message \( m \) is not sent, and one where \( h \) is false and \( m \) is not sent. Since I only consider truthful communication, it is not possible that \( h \) is false and \( m \) is sent. Alice and Bob know that \( h \) is true and \( m \) is sent, therefore they do not confuse the actual world with any other world. However, Carol thinks it possible that \( m \) was not sent, and confuses the actual world with the situation where both Bob and Carol are uncertain about the value of \( h \).

Now Bob sends Carol the message \( m' \), informing her that \( h \) is true. Alice does not know that this message is sent. I model this as follows:

![Diagram](image)

\[ M_2 : \]

In the actual world, Alice, Bob and Carol all know that \( h \) is true. Bob and Carol know that \( m' \) was sent, but Alice does not know this. Therefore she considers it possible that \( m' \) was not sent, and that Carol does not know about \( m \) and \( h \). She confuses the actual world with the situation from model \( M_1 \).

In order to vary the set of messages that are considered in each model, I will use vocabulary-based Kripke models. These models were introduced in [van Eijck et al., 2011]. Every vocabulary-based Kripke model has a finite vocabulary. In my set-up these vocabularies consist of the propositions and messages that are under consideration. They are the same vocabularies that I defined in the previous section. The formal definition is as follows.

3.3.2. Definition. Let a set of agents \( Ag \), a set of propositional atoms \( P \) and a set of messages \( M \) be given. A \textit{vocabulary-based Kripke model} for \( Ag, P, M \) is a tuple \( M = (W, R, Val, Voc, W_0) \) where \( W \) is a set of worlds, \( R \) is a function that assigns to each \( a \in Ag \) an equivalence relation \( R_a \) on \( W \), \( Voc \subseteq P \cup M \) is a vocabulary of propositions and messages under consideration, \( Val \) is a function that assigns to each world in \( W \) a subset of \( Voc \) (its valuation), and \( W_0 \subseteq W \) is
the set of actual worlds. I will sometimes use \( \sim_a \) for \( R_a \). Given a Kripke model \( \mathcal{M} \), I will use \( W^\mathcal{M}, R^\mathcal{M}, Val^\mathcal{M}, Voc^\mathcal{M}, W_0^\mathcal{M} \) to denote its elements.

When a message is sent, this should be modeled by a vocabulary extension combined with a knowledge update. First I will add the new message to the vocabulary of the Kripke model. It is not yet in the valuation of any world, so it is false in all worlds. Then I will use an action model to both set the truth value of the new message in the different worlds and immediately model its epistemic effects. In order to set the truth value of the new message, I need an action model that can actually change the valuation of the worlds, instead of just the relations between them. Such models are defined in [van Benthem et al., 2006]. The following definition follows the same lines.

First of all, I define a substitution that can be used to change the valuation of a world.

3.3.3. Definition. Let a set of agents \( Ag \) and a set of propositional atoms \( P \) and a set of messages \( M \) be given. A substitution over \( P, M \) is a partial function \( \sigma : (P \cup M) \rightarrow \{\top, \bot\} \) that assigns a new truth value to a subset of all propositions and messages. Given a valuation \( Val \subseteq P \cup M \), the result of applying \( \sigma \) to \( Val \) is given by

\[
Val \cdot \sigma := Val \setminus \text{dom}(\sigma) \cup \{x \in \text{dom}(\sigma) \mid \sigma(x) = \top\}.
\]

Let \( \text{sub}_{P, M} \) be the set of all substitutions over \( P, M \).

A substitution changes the truth value of a number of elements of a vocabulary. It leaves the truth value of the elements that are not in its domain unchanged. I will add a substitution to every event of the action model.

3.3.4. Definition. An action model with substitution for \( Ag, P \cup M \) is a tuple \( A = (E, R, Pre, Sub, E_0) \) where \( E, R, Pre, E_0 \) are defined like the corresponding elements of an action model and \( Sub : E \mapsto \text{sub}_{P, M} \) is a function that assigns to each event a substitution over \( P, M \).

The purpose of these action models with substitution is that the substitution of an event is applied to the valuation of all worlds matched to the event. This is reflected in the new definition of the product update:

3.3.5. Definition. Given a Kripke model \( \mathcal{M} \) and an action model with substitution \( \mathcal{A} \) over \( Voc^\mathcal{M} \), the result of updating \( \mathcal{M} \) with \( \mathcal{A} \) is the model \( \mathcal{M} \otimes \mathcal{A} = (W', R', Val', Voc', W'_0) \) given by

\[
W' := \{(w, e) \mid w \in W^\mathcal{M}, e \in E^\mathcal{A}, \mathcal{M} \models_w Pre(e)\},
\]

\[
(w, d)R'_a(v, e) \quad \text{iff} \quad wR_a^\mathcal{M}v \text{ and } dR_a^e,
\]

\[
Val'((w, e)) := (Val^\mathcal{M} : Sub^\mathcal{A}(e))(w),
\]

\[
Voc' := Voc^\mathcal{M},
\]

\[
W'_0 := \{(w, e) \in W' \mid w \in W_0^\mathcal{M} \text{ and } e \in E_0^\mathcal{A}\}.
\]
Now I am ready to define the action model that represents the act of sending a message. It should reflect a number of properties of messages. First of all, I assume that all communication is truthful. Therefore the message may only be sent if the sender knows its contents to be true. Furthermore, all recipients of the message should get to know that the message was sent, and outsiders should not get to know this. The following action model ensures these properties.

\[
\mathcal{A}_m : \begin{cases}
    e_m : [s_m]c_m, m := \top \\
    e_m : m := \bot
\end{cases}
\]

Here \(G = s_m \cup r_m\) is the set of senders and recipients of the message. The action model has two possible events. In the first one, \(m\) is set to true so the message is sent. It has precondition \([s_m]c_m\), so the message can only be sent if the sender knows its contents. In the second one \(m\) is set to false, so the message is not sent. The only agents who confuse the two worlds (and thus do not know whether the message is sent) are those agents that are not involved in the message.

Note that both events of the action model are in the set of actual events. This means that this action model does not determine whether the message was sent or not. It only extends the model with the possibility of sending the message, taking its content and epistemic consequences into account.

An event in an action model will only be matched with a world in a Kripke model if this world satisfies the precondition of the event. Therefore one could wonder whether the events in \(\mathcal{A}_m\) will match the worlds in some Kripke model it is applied to.

Because the event \(e_m\) has no precondition, for every world \(w\) from the original model \(\mathcal{M}\) there will be a world \((w, m)\) in the model \(\mathcal{M} \otimes \mathcal{A}_m\). For the other event \(e_m\), things are not so easy. If a world \(w \in W^M\) does not satisfy \([s_m]c_m\) then it will not match the event \(e_m\) and there will be no world \((w, e_m)\) in the final model \(\mathcal{M} \otimes \mathcal{A}_m\). This matches the intuition of the models: it is always possible not to send a message \(m\) but if the sender of \(m\) does not know its contents, then it is not possible to send it so the event representing the situation where the message is sent does not match any worlds in the Kripke model.

For the sake of brevity, I will define the result of adding a message \(m\) to a model \(\mathcal{M}\) as

\[
\mathcal{M} \bullet m := (W^M, R^M, Val^M, Voc^M \cup \{m\}, W_0^M) \otimes \mathcal{A}_m.
\]

I will also abbreviate \((w, e_m)\) with \((w, m)\) and \((w, e_m)\) with \((w, m)\).

The following lemma shows that this operation does not change any basic facts about the world:
3.3.6. Lemma. For any model $\mathcal{M}$, message $m \notin \text{Voc}^{\mathcal{M}}$ and formula $\psi \in \mathcal{L}_{0}^{\text{MPD}}$ such that $\text{voc}(\psi) \subseteq \text{Voc}^{\mathcal{M}}$,

$$\mathcal{M} \models_w \psi \iff \mathcal{M} \bullet m \models_{(w, \overline{m})} \psi.$$  

Furthermore, if $(w, m) \in W^{\mathcal{M} \bullet m}$ then

$$\mathcal{M} \models_w \psi \iff \mathcal{M} \bullet m \models_{(w, m)} \psi.$$ 

Proof. A simple induction on $\psi$.  

The following theorem shows that in the case that $m$ was not sent, the knowledge of the agents about basic facts does not change. Furthermore, even if $m$ was sent the knowledge of the agents who did not receive the message does not change.

3.3.7. Theorem. For any model $\mathcal{M}$, message $m \notin \text{Voc}^{\mathcal{M}}$ and formula $\psi \in \mathcal{L}_{0}^{\text{MPD}}$ such that $\text{voc}(\psi) \subseteq \text{Voc}^{\mathcal{M}}$,

$$\mathcal{M} \models_w [a] \psi \iff \mathcal{M} \bullet m \models_{(w, \overline{m})} [a] \psi.$$ 

Furthermore, if $a \notin r_m$ then

$$\mathcal{M} \models_w [a] \psi \iff \mathcal{M} \bullet m \models_{(w, m)} [a] \psi.$$ 

Proof. Suppose $\mathcal{M} \models_w [a] \psi$. Suppose $(w, \overline{m}) \sim_a (w', x)$. Then $w \sim_a w'$ so $\mathcal{M} \models_w \psi$ and by Lemma 3.3.6, $\mathcal{M} \bullet m \models_{(w', x)} \psi$. So $\mathcal{M} \bullet m \models_{(w, \overline{m})} [a] \psi$. Suppose $\mathcal{M} \bullet m \models_{(w, \overline{m})} [a] \psi$. Suppose $w \sim_a w'$. As noted above, certainly $(w', \overline{m}) \sim_a (w, \overline{m})$. Then because $w \sim_a w'$, it also holds that $(w, \overline{m}) \sim_a (w', \overline{m})$ so $\mathcal{M} \bullet m \models_{(w', \overline{m})} \psi$. Then by Lemma 3.3.6, $\mathcal{M} \models_{w'} \psi$. So $\mathcal{M} \models_w [a] \psi$.

Let $a \notin r_m$. Suppose $\mathcal{M} \bullet m \models_{(w, m)} [a] \psi$. Let $w \sim_a w'$. Then since $i \notin r_m$, $(w, m) \sim_a (w', \overline{m})$ so $\mathcal{M} \bullet m \models_{(w', \overline{m})} \psi$. Then by Lemma 3.3.6, $\mathcal{M} \models_{w'} \psi$. So $\mathcal{M} \models_w [a] \psi$.

Suppose $\mathcal{M} \models_w [a] \psi$. Let $(w, m) \sim_a (w', x)$. Then $w \sim_a w'$ so $\mathcal{M} \models_{w'} \psi$. Then by Lemma 3.3.6, $\mathcal{M} \bullet m \models_{(w', x)} \psi$. So $\mathcal{M} \bullet m \models_{(w, m)} [a] \psi$.  

Using this framework, I can now show formally that BCCs are unordered.

3.3.8. Theorem. Let $\mathcal{M}, w$ be such that $\mathcal{M} \models_w m$. Let $m' = (s_m, m, a)$ and $m'' = (s_m, m, b)$. Then

$$\mathcal{M} \bullet m' \bullet m'' = \mathcal{M} \bullet m'' \bullet m', ((w, m'), (w, m'')).$$
3.3. **Modeling Message Passing**

**Proof.** Check that

\[
\{(((w, x), y), ((w, y), x)) \mid w \in W^M, x \in \{m', m''\}, y \in \{m'', m''\}\}
\]

is a bisimulation. □

Theorem 3.3.8 and its (easy) proof illustrate how this framework can be used to formalize and prove subtle properties of message passing.

There is one other problem I have to tackle. Suppose a message is sent that mentions some other message which is not in the vocabulary of the model. Then both messages have to be added to the vocabulary: not only the message that is sent at that moment, but also the message that is mentioned in the first message. Therefore, I propose the following modeling procedure. When a message \( m \) is considered that mentions some messages of which \( m_1 \leq ... \leq m_n \) are the ones that are not in the vocabulary of the Kripke model \( M \), I define the result of the update of \( M \) with \( m \) as

\[
M \odot m := M \bullet m_1 \bullet m_2 \bullet \ldots \bullet m_n \bullet m.
\]

The next example shows how this framework can be used to model the establishment of ‘common knowledge of learning’. Agent \( b \) learns whether \( p \) is true from agent \( a \), and this fact becomes common knowledge, but outsiders do not learn whether \( p \) is true from the interaction.

**3.3.9. Example.** Consider a situation where agent \( a \) knows whether \( p \), while agent \( b \) and \( c \) do not (and this is common knowledge). Actually, \( p \) is true. This can be represented with the following model:

![Diagram](image)

Let \( m_1 \) be the message \((a, p, b)\) and let \( m_2 \) be the message \((a, \neg p, b)\). The result \( M \odot m_1 \) of updating with \( m_1 \):

![Diagram](image)

The result \( M \odot m_1 \odot m_2 \) of consecutively updating with \( m_2 \):

![Diagram](image)
Notice that agent $c$ confuses all worlds, since she would not receive either message $m_1$ or message $m_2$ if they were sent. On the other hand, in the worlds where $m_1$ or $m_2$ was sent, agent $a$ and $b$ have common knowledge of the truth value of $p$. Now suppose that agent $a$ wants to create common knowledge among the three agents that $a$ and $b$ know the truth value of $p$, without revealing that truth value to agent $c$. Then he could send a third message $m_3$ of the form $(a, m_1 \lor m_2, \{a, b, c\})$ that informs the three of them that either $m_1$ or $m_2$ was sent without revealing which of the two was actually sent. When the model was updated with this third message, the resulting model would show that in those worlds where $m_3$ was sent, it holds that $[c]((b)p \lor (b)\neg p)$, so agent $c$ knows that agent $b$ knows whether $p$, but neither $[c]p$ nor $[c]\neg p$ hold, so agent $c$ does not know the value of $p$ herself.

It can be very interesting to reason about messages in a hypothetical way. For example, one could wonder whether the agents know what the epistemic consequences of sending a certain message would be. In order to express these questions I add two new constructs to the language $\mathcal{L}_{MPD}$. The formula $[m] \varphi$ stands for “if message $m$ is sent, $\varphi$ holds”. The formula $[\overline{m}] \varphi$ stands for “if the model is extended with the possibility of sending $m$ but it is not sent, $\varphi$ holds”. The semantics of these constructs is defined as follows:

$$\mathcal{M} \models_w [m] \varphi \iff \mathcal{M} \circ m \models_{(w, m)} \varphi,$$

$$\mathcal{M} \models_w [\overline{m}] \varphi \iff \mathcal{M} \circ m \models_{(w, \overline{m})} \varphi.$$ 

Note that I use double brackets for modalities that express something about a different model, for example a model obtained by updating the current model with an action model, while I use single brackets for modalities that express something about different worlds in the current model, for example worlds related by an agent’s relation.

As mentioned before, in the update with $\mathcal{A}_m$ I do not assume the message is actually sent. Both the world where $m$ is sent and the world where $m$ is not sent are actual worlds. In some situations, it is useful to denote in the model that actually the message was sent. For this purpose I use another action model.
3.4. Models with Realistic Properties

\[ A^+_m : \quad \text{Ag} \quad e_m : m \quad e_m : \neg m \]

This action model divides the worlds of any Kripke model updated with it into those that satisfy \( m \) and those that do not. The worlds that satisfy \( m \) and that are actual worlds remain actual, while those that do not satisfy \( m \) become non-actual worlds. Because there are relations between \( e_m \) and \( e_m \) for all agents, all relations that are present in the original model are preserved. So the only thing this model does is that it makes actual worlds that do not satisfy \( m \) non-actual.

The corresponding update is defined as follows. Suppose a message \( m \) is actually sent and it mentions messages \( m_1 \preceq \ldots \preceq m_n \) that are not in the vocabulary of \( \mathcal{M} \). The result of the positive update of \( \mathcal{M} \) with \( m \) is defined as follows:

\[ \mathcal{M} \oplus m := \mathcal{M} \bullet m_1 \bullet \ldots \bullet m_n \bullet m \otimes A^+_m. \]

In situations where \( m \) was actually not sent, this can also be denoted in the model. For this purpose I define yet another action model:

\[ A^-_m : \quad \text{Ag} \quad e_m : m \quad e_m : \neg m \]

This model is very similar to \( A^+_m \), only now the worlds that do not satisfy \( m \) remain actual. Again, there is also a corresponding update. The result of the negative update of \( \mathcal{M} \) with \( m \) is defined as follows:

\[ \mathcal{M} \ominus m := \mathcal{M} \bullet m_1 \bullet \ldots \bullet m_n \bullet m \otimes A^-_m. \]

With these three action models, I have set up a framework that can be used to model a large variety of message passing situations and the agents’ knowledge in them. I imagine a typical modeling task as a situation where messages may be sent in a sequence of rounds. This may be the case when, for example, two agents communicate according to a set protocol. Another example is a game of poker where every player has the possibility to call, raise or fold in every round. The modeling procedure I propose is to start out with an initial model that has no messages in the vocabulary, and then gradually update the model whenever a message is sent (using \( \oplus \)) or could have been sent but was not (using \( \ominus \)).

In the next section I will show that not all possible Kripke models represent a realistic situation and I will define a class of models that do.

3.4 Models with Realistic Properties

In this section I will take a closer look at the axiomatic properties of the models I introduced. As mentioned above, I assume that all communication is truthful and reliable. This is also reflected in the update mechanism I proposed, as is shown by the following theorem.
3.4.1. Theorem. For any model $M$ and any sequence of messages $m_1, \ldots, m_n \notin \text{Voc}^M$ such that $m_1 \preceq \ldots \preceq m_n$, the following formulas are valid in $M \bullet m_1 \bullet \ldots \bullet m_n$ for any $m_i$:

- $m_i \rightarrow c_{m_i}$,
- $m_i \rightarrow [a]m_i$ for all $a \in r_{m_i}$.

Proof. I claim that for any $1 \leq i \leq n$, the above formulas hold in $M \bullet m_1 \bullet \ldots \bullet m_i$ for all $m_j$ with $1 \leq j \leq i$. I will prove this by induction on $i$. Suppose $i = 1$. I consider $M \bullet m_1$. Every world in $M \bullet m_1$ must be the result of matching a world from $M$ with an event from $A_{m_1}$. Because $e_{\pi m_1}$ sets the truth value of $m_1$ to $\bot$, the worlds matched with that event will not satisfy $m_1$ so they will certainly satisfy $m_1 \rightarrow c_{m_1}$ and $m_1 \rightarrow [r_{m_1}]m_1$. Now consider the other event $e_{m_1}$. It has precondition $[s_{m_1}]c_{m_1}$ so it satisfies $c_{m_1}$ and thereby $m_1 \rightarrow c_{m_1}$. For the second formula I have to check that the worlds matched with $e_{m_1}$ satisfy $[a]m_1$ for any $a \in r_{m_1}$. Take such $a$. Because there is no relation from $e_{m_1}$ to $e_{\pi m_1}$ for agents in $r_{m_1}$, the only worlds that are $a$-related to worlds matched with $e_{m_1}$ are other worlds matched with $e_{m_1}$. Because $e_{m_1}$ sets the truth value of $m_1$ to $\top$, these worlds satisfy $m_1$. So all worlds matched with $e_{m_1}$ satisfy $[a]m_1$ for all $a \in r_{m_1}$.

For the induction step, suppose $M \bullet m_1 \bullet \ldots \bullet m_i$ satisfies both formulas for all $m_j$ with $1 \leq j \leq i$. Consider $M \bullet m_1 \bullet \ldots \bullet m_{i+1}$. With a reasoning analogous to that for the previous case I can show that the formulas hold for $m_{i+1}$. All that is left is to show that the formulas for $m_1, \ldots, m_i$ are preserved in the transition from $M \bullet m_1 \bullet \ldots \bullet m_i$ to $M \bullet m_1 \bullet \ldots \bullet m_{i+1}$. For the first formula this follows from Lemma 3.3.6: note that $m_j \rightarrow c_m$ is a formula from $L_0^{MPD}$ that does not contain $m_{i+1}$, for $j \leq i$. For the second formula, note that by Lemma 3.3.6 the truth value of $m_j$ is preserved in the update. Also, the update does not add any relations between worlds, it only possibly removes some relations. So if all $a$-related worlds satisfy $m_j$ in $M \bullet m_1 \bullet \ldots \bullet m_i$, this will also hold in $M \bullet m_1 \bullet \ldots \bullet m_{i+1}$. Therefore both formulas are preserved for all $m_j$ with $1 \leq j \leq i$. \qed

But these properties are not enough to ensure that the Kripke models are realistic. There are more subtle requirements for reasonable models, as the following example shows.

3.4.2. Example. Consider the following model with three agents $a, b$ and $c$ and a message $m = (b, p, c)$:

[Diagram]

There are two possible situations, one where $m$ was sent and one where it was not sent, and none of the agents confuse the two situations. All communication in this model is truthful and reliable but still there is something strange about
the model: agent $a$ knows whether the message from $b$ to $c$ was sent, even though she should not have received it.

It is hard to express the above property in the language $L^{MPD}$: it will not do to simply state that agents that are not recipients should not know about a message, for they may have received a forward of this message and in that case they should know about it. Problems like the one in the above model would not occur if one started out with a model without messages in the vocabulary and then sequentially added new messages. Therefore, the class of models I would like to consider is the class of properly generated models:

**3.4.3. Definition.** A model $M$ is **properly generated** iff there is some model $M_0$ and a list of messages $m_1, ..., m_n$ such that there are no messages in the vocabulary of $M_0$ and

$$M \models M_0 \bullet m_1 \bullet ... \bullet m_n.$$  

So a model is properly generated if it can be built from a model containing no messages (I call such a model an **initial** model) by adding messages. These are the models I consider realistic. Therefore, I want to find a procedure to check whether a Kripke model is properly generated. The rest of this section will be devoted to this task.

Consider a model $M$ that is updated with a message $m$. As mentioned in the previous section, for every world $w \in W^M$ there will be a world $(w, \overline{m}) \in W^{M \bullet m}$. The only difference between $w$ and $(w, \overline{m})$ is that the message $m$ is added to the vocabulary. The relations between $\neg m$ worlds in $M \bullet m$ are the same as the relations between the worlds in $M$. The only difference is in the relations to and between $m$ worlds. Therefore, if one cuts off all worlds that satisfy $m$ and only considers the $\neg m$ worlds, this gives the original model again. This can be done with the following action model:

$$A_{m^-} : \boxed{ e : \overline{m}}$$

I will show how this works out with the following example.

**3.4.4. Example.** Consider the model from Example 3.3.9 again.

$$\begin{array}{c}
p \quad \Downarrow \quad b, c \quad \Downarrow \quad \overline{p} \\
\end{array}$$

Updating with $m = (a, p, b)$ gives the following result:
Chapter 3. Message Passing in Dynamic Epistemic Logic

Now when I update with the action model $A_{m^-}$ I get a model which is very much like the original, but with $m$ in the vocabulary:

Apart from the addition of $m$, the third model is identical to the first one.

The following theorem shows that updating with $A_{m^-}$ really gives the original model from before the update with $m$, if one does not consider the fact that $m$ is now in the vocabulary.

**3.4.5. Theorem.** For any model $\mathcal{M}$ such that $m \not\in \text{Voc}^\mathcal{M}$, $\mathcal{M} \models \mathcal{M} \bullet m \otimes A_{m^-}$.

**Proof.** Let $w \in W^\mathcal{M}$. Then $(w, \overline{m}) \in W^\mathcal{M} \bullet m$ and possibly $(w, m) \in W^\mathcal{M} \bullet m$. But since $(w, m)$ satisfies $m$ if it exists, $(w, m) \not\in W^\mathcal{M} \bullet m \otimes A_{m^-}$. I define the relation $Z$ between $W^\mathcal{M}$ and $W^\mathcal{M} \bullet m \otimes A_{m^-}$ as follows.

For any $w \in W^\mathcal{M}$, $wZ(w, \overline{m})$.

Clearly, $Z$ is a bisimulation if one does not consider $m$ so $\mathcal{M} \models \mathcal{M} \bullet m \otimes A_{m^-}$.

With this action model, I can check whether a model is the result of an update with the message $m$ by first “undoing” the update by updating with $A_{m^-}$ and then “redoing” it by updating with $m$. If the result is bisimilar to the original model then I know that it is the result of the message update. I will extend this to sequences of messages. In order to do this I first need the following lemma.

**3.4.6. Lemma.** For any sequence of messages $m_1, ..., m_n$ such that $m_1 \preceq ... \preceq m_n$ and for any two models $\mathcal{M}, \mathcal{N}$ such that $\mathcal{M} \models \mathcal{M} \bullet m_1 \bullet ... \bullet m_n \equiv \mathcal{N} \bullet m_1 \bullet ... \bullet m_n$.
3.4. Models with Realistic Properties

Proof. Let \( Z \) be a bisimulation between \( \mathcal{M} \) and \( \mathcal{N} \). I define a relation \( X \) between \( \mathcal{M} \cdot m_1 \cdot \ldots \cdot m_n \) and \( \mathcal{N} \cdot m_1 \cdot \ldots \cdot m_n \) as follows. For any two worlds \( w \in W^\mathcal{M} \) and \( v \in W^\mathcal{N} \) and any sequence \( x = x_1, \ldots, x_n \) where \( x_i = m_i \) or \( x_i = \overline{m_i} \),

\[ ((w, x_1), x_2), \ldots, x_n) X ((v, x_1), x_2), \ldots, x_n) \text{ iff } w Z v \]

Note that the question of whether \( (w, x) \) exists depends on whether \( \mathcal{M} \models s_{m_1} c_{m_1} \) if \( x_1 = m_1 \), and whether \( \mathcal{M} \models (w, x_1) \) \( s_{m_2} c_{m_2} \) if \( x_2 = m_2 \), etcetera. Similarly for \( (v, x) \) and \( \mathcal{N} \). But because \( m_1 \preceq \ldots \preceq m_n \), these things only depend on the propositions and messages that are true in \( w \) and in \( v \) (which are the same because \( w Z v \)) and the earlier messages. So \( (w, x) \) exists iff \( (v, x) \) exists. So \( X \) is total. It is clear from the definition of message update that \( X \) is a bisimulation. \( \square \)

Now I can characterize the class of properly generated models using the action model \( A_{m^-} \) and the message update:

3.4.7. Theorem. A model \( \mathcal{M} \) is properly generated iff there is an order \( m_1, \ldots, m_n \) listing all messages in the vocabulary of \( \mathcal{M} \) such that \( m_1 \preceq \ldots \preceq m_n \) and

\[ \mathcal{M} \models \mathcal{M} \otimes A_{-m_n} \otimes \ldots \otimes A_{-m_1} \cdot m_1 \cdot \ldots \cdot m_n. \]

Proof. \( \Rightarrow \): Suppose \( \mathcal{M} \) is properly generated. Then there is some initial model \( \mathcal{M}_0 \) and a list of messages \( m_1, \ldots, m_n \) such that \( \mathcal{M} \models \mathcal{M}_0 \cdot m_1 \cdot \ldots \cdot m_n. \)

By repeated use of Theorem 3.4.5, I have

\[ \mathcal{M}_0 \models \{m_1, \ldots, m_n\} \mathcal{M} \otimes A_{m_1^-} \otimes \ldots \otimes A_{m_n^-}. \]

Then by Lemma 3.4.6,

\[ \mathcal{M} \models \mathcal{M} \otimes A_{m_1^-} \otimes \ldots \otimes A_{m_n^-} \cdot m_1 \cdot \ldots \cdot m_n. \]

\( \Leftarrow \): Suppose there is such an order \( m_1, \ldots, m_n \). Let \( \mathcal{N} \) be the model like \( \mathcal{M} \otimes A_{m_1^-} \otimes \ldots \otimes A_{m_n^-} \), but with \( m_1, \ldots, m_n \) not in the vocabulary. Clearly,

\[ \mathcal{M} \otimes A_{m_1^-} \otimes \ldots \otimes A_{m_n^-} \models \{m_1, \ldots, m_n\} \mathcal{N} \]

so by Lemma 3.4.6,

\[ \mathcal{M} \otimes A_{m_1^-} \otimes \ldots \otimes A_{m_n^-} \cdot m_1 \cdot \ldots \cdot m_n \models \mathcal{N} \cdot m_1 \cdot \ldots \cdot m_n. \]

But because \( m_1, \ldots, m_n \) are all the messages in the vocabulary of \( \mathcal{M} \), I conclude that \( \mathcal{N} \) is an initial model. This implies that \( \mathcal{M} \otimes A_{m_1^-} \otimes \ldots \otimes A_{m_n^-} \cdot m_1 \cdot \ldots \cdot m_n \) is properly generated, and then so is \( \mathcal{M} \). \( \square \)
### 3.5 Axiomatization

I have added two modalities $[m]$ and $[m']$ to the language $\mathcal{L}^{MPD}$. In [van Benthem et al., 2006] a technique is developed for translating a language with action modalities to epistemic PDL. I will use the same technique to show that these three modalities do not increase the expressive power of $\mathcal{L}^{MPD}$. For each formula containing a modality I will give a reduction axiom that shows that the formula with the modality is equivalent to a formula without it. For the Boolean cases, these reduction axioms look as follows:

$$
\begin{align*}
[m]p & \iff [s_m]c_m \rightarrow p \\
[m']m' & \iff [s_m]c_m \rightarrow m' \quad m' \neq m \\
[m]m & \iff T \\
[m]\neg \phi & \iff \neg[m]\phi \\
[m](\phi_1 \lor \phi_2) & \iff [m]\phi_1 \lor [m]\phi_2
\end{align*}
$$

The reduction axioms for formulas containing epistemic programs (the PDL modalities $\alpha$, corresponding to relations in the model) are more complicated. This is because when a relation is followed in the Kripke model with an epistemic program, the same relation can only be followed in the model which is the result of the message update if this relation is not removed by the update.

Recall that the message updates correspond to an update with the following action model:

$$
\mathcal{A}_m : e_m : [s_m]c_m, m := T \quad e_{m'} : m := \bot
$$

A relation will be present in the result of the update if it is both in the original model and in the action model. So I have to check whether the epistemic program can be executed in the updated model by checking whether it can be executed both in the original model and in the action model "concurrently".

For all epistemic programs, I will compute an epistemic program that is the equivalent of the original program together with a concurrent step in the action model $\mathcal{A}_m$. With $T_{xy}(\alpha)$ I mean the program that is the equivalent of doing $\alpha$ in the original model and concurrently moving from state $e_x$ to state $e_y$ in the action model. I define it inductively as follows:
3.5. Axiomatization

\[ T_{mm}(a) := ?[s_m]c_m; a; ?[s_m]c_m \]

\[ T_{m\overline{m}}(a) := \begin{cases} \bot & \text{if } a \in r_m \\ ?[s_m]c_m; a & \text{otherwise} \end{cases} \]

\[ T_{\overline{m}m}(a) := \begin{cases} \bot & \text{if } a \in r_m \\ a; ?[s_m]c_m & \text{otherwise} \end{cases} \]

\[ T_{\overline{m}\overline{m}}(a) := a \]

\[ T_{mm}(?\psi) := ?([s_m]c_m \land \psi) \]

\[ T_{m\overline{m}}(?\psi) := \bot \]

\[ T_{\overline{m}m}(?\psi) := \bot \]

\[ T_{\overline{m}\overline{m}}(?\psi) := ?\psi \]

\[ T_{mm}(\alpha_1; \alpha_2) := (T_{mm}(\alpha_1); T_{mm}(\alpha_2)) \cup (T_{m\overline{m}}(\alpha_1); T_{m\overline{m}}(\alpha_2)) \]

\[ T_{m\overline{m}}(\alpha_1; \alpha_2) := (T_{mm}(\alpha_1); T_{m\overline{m}}(\alpha_2)) \cup (T_{m\overline{m}}(\alpha_1); T_{m\overline{m}}(\alpha_2)) \]

\[ T_{\overline{m}m}(\alpha_1; \alpha_2) := (T_{\overline{m}m}(\alpha_1); T_{\overline{m}m}(\alpha_2)) \cup (T_{\overline{m}m}(\alpha_1); T_{\overline{m}m}(\alpha_2)) \]

\[ T_{\overline{m}\overline{m}}(\alpha_1; \alpha_2) := (T_{\overline{m}m}(\alpha_1); T_{\overline{m}m}(\alpha_2)) \cup (T_{\overline{m}m}(\alpha_1); T_{\overline{m}m}(\alpha_2)) \]

\[ T_{mm}(\alpha_1 \cup \alpha_2) := T_{mm}(\alpha_1) \cup T_{mm}(\alpha_2) \]

\[ T_{m\overline{m}}(\alpha_1 \cup \alpha_2) := T_{m\overline{m}}(\alpha_1) \cup T_{m\overline{m}}(\alpha_2) \]

\[ T_{\overline{m}m}(\alpha_1 \cup \alpha_2) := T_{\overline{m}m}(\alpha_1) \cup T_{\overline{m}m}(\alpha_2) \]

\[ T_{\overline{m}\overline{m}}(\alpha_1 \cup \alpha_2) := T_{\overline{m}\overline{m}}(\alpha_1) \cup T_{\overline{m}\overline{m}}(\alpha_2) \]

The final case is the reduction for \( \alpha^* \). Note that the action model can be seen as the following automaton:

\[
\begin{array}{c}
\xymatrix{
T_{mm} \ar@/^/[r] & T_{m\overline{m}} \ar@/^/[l] & T_{\overline{m}m} \ar@/_/[l] & T_{\overline{m}\overline{m}} \ar@/_/[r]
}
\end{array}
\]

Then the epistemic program giving all finite paths through the action model starting in \( e_m \) and ending in \( e_m \) is:

\[ T_{mm}^* (T_{m\overline{m}} T_{m\overline{m}}^* T_{\overline{m}m} T_{\overline{m}\overline{m}}^*)^*. \]
Similarly, if I take $e_m$ as start state and $e_m$ as final state, I get:

$$T_{mm}T_{mm}^* (T_{mm}T_{mm}^*)^* .$$

For $e_m$ as start and as stop state:

$$T_{mm}^* (T_{mm}T_{mm}^*)^* .$$

And finally, if I take $e_m$ as start state and $e_m$ as stop state:

$$T_{mm}^* T_{mm}T_{mm}^* (T_{mm}T_{mm}^*)^* .$$

All in all I get the following recipe for transforming an epistemic expression of the form $\alpha^*$:

$T_{mm}(\alpha^*) := (T_{mm}(\alpha))^*; (T_{mm}(\alpha); (T_{mm}(\alpha))^*; (T_{mm}(\alpha))^*; (T_{mm}(\alpha))^*; (T_{mm}(\alpha))^*; T_{mm}(\alpha); (T_{mm}(\alpha))^*)^*,$

$T_{mm}(\alpha^*) := (T_{mm}(\alpha))^*; (T_{mm}(\alpha); (T_{mm}(\alpha))^*; (T_{mm}(\alpha))^*; (T_{mm}(\alpha))^*; (T_{mm}(\alpha))^*; (T_{mm}(\alpha))^*; T_{mm}(\alpha); (T_{mm}(\alpha))^*; T_{mm}(\alpha); (T_{mm}(\alpha))^*; T_{mm}(\alpha); (T_{mm}(\alpha))^*; T_{mm}(\alpha); (T_{mm}(\alpha))^*; T_{mm}(\alpha); (T_{mm}(\alpha))^*; T_{mm}(\alpha); (T_{mm}(\alpha))^*; T_{mm}(\alpha); (T_{mm}(\alpha))^*; T_{mm}(\alpha); (T_{mm}(\alpha))^*; T_{mm}(\alpha); (T_{mm}(\alpha))^*; T_{mm}(\alpha); (T_{mm}(\alpha))^*; T_{mm}(\alpha); (T_{mm}(\alpha))^*); T_{mm}(\alpha); (T_{mm}(\alpha))^*)^*.$

Now I can give the reduction axioms for the case of epistemic programs:

$$[\langle m \rangle][\langle \alpha \rangle] \phi \leftrightarrow [T_{mm}(\alpha)] [\langle m \rangle] \phi \land [T_{mm}(\alpha)] [\langle m \rangle] \phi$$

$$[\langle m \rangle][\langle \alpha \rangle] \phi \leftrightarrow [T_{mm}(\alpha)] [\langle m \rangle] \phi \land [T_{mm}(\alpha)] [\langle m \rangle] \phi$$

This gives:

3.5.1. Theorem. The language $L^{MPD}$ and the language $L^{MPD}$ with message modalities added have the same expressive power.

Proof sketch. Take any formula $\varphi$ from $L^{MPD}$ with message modalities. Any message modality in $\varphi$ can be replaced with an equivalent subformula that contains no message modalities. The correct equivalent subformulas are prescribed by the reduction axioms given above. This way, I can find for any formula that contains message modalities an equivalent formula that does not contain them. Therefore, the message modalities do not add expressive power to $L^{MPD}$.
3.6 Related Work

The work presented in this chapter was inspired by the wish to incorporate explicit messages in Dynamic Epistemic Logic (DEL). I will clarify what the added value of my approach is compared to the usual DEL as in [Baltag and Moss, 2004, van Benthem et al., 2006, van Ditmarsch et al., 2006]. In the usual DEL, there is no mention of any messages and the only atoms in the models are propositions. The models can be updated with so-called action models, of which my message update is a special case. In my approach I have tailored an action model for a specific kind of group messages with a sender and a set of recipients. This is very useful in modeling since it is no longer up to the user of the framework to come up with the right action model: this is automatically “generated” when defining the message. This way, I make a step towards formalizing the modeling procedure which makes it easier and less error-prone.

I have combined DEL with the vocabulary expansion proposed in [van Eijck et al., 2011] and used this to introduce messages explicitly in the models. This has the great advantage that it is possible to model agents who reason about messages that have been sent and even messages about other messages. This allows for constructions like forward, acknowledgement, BCC recipients etcetera.

In my approach every model has a vocabulary of propositions and messages that the agents are aware of. The vocabulary of a Kripke model can be viewed as a global awareness function, indicating the set of propositions and messages that the agents are aware of across the model. A more extended study of awareness in a similar setting can be found in [Fagin and Halpern, 1988, van Ditmarsch and French, 2011]. There, a more subtle notion of awareness is presented, where different agents may be aware of different vocabularies in different worlds.

My work can be compared to interpreted systems as presented in e.g. [Fagin et al., 1995]. There, the focus is on a global state that is constructed by combining local states of the agents. In this set up, two global states are related for an agent if the corresponding local states of that agent are equivalent. In my approach, there is no clear distinction between one agent’s and another agent’s information. One possible such distinction would be to say that an agent’s local state is her “inbox” of messages she sent or received up to that moment. Then one would somehow also have to incorporate the messages forwarded to the agent.

The idea of time is clearly incorporated in interpreted systems. In my framework this is less explicit: I can show how the model evolves over time by doing a sequence of message updates, but once these updates have been done the only information that is preserved in the model is whether the message has been sent at some point in time, not when it was sent exactly or an ordering between them. Of course there is the vocabulary embedding relation $\prec$, but this only partially orders the messages. This has the advantage of keeping the model simple, and in a lot of applications the exact ordering between messages is not so relevant.
3.7 Conclusion

I have shown how epistemic models can be used to represent the influence of message passing on the knowledge of agents. The models presented in this chapter directly show the agent’s knowledge using relations between possible worlds. The models are finite and I have given an axiomatization. A nice property of this approach is that the models can be generated automatically given a sequence of messages that have been sent.

This system has the curious property that agents are affected by an update with messages that are not addressed to them: they consider the fact that such a message was sent possible. The history-based system of Parikh and Ramanujam has the same property, as does the process of updating with S5 action models for group announcements (see, e.g., [Baltag and Moss, 2004]).

In some situations this property is perfectly realistic, for example in a game where in every new round the agents know which new messages may be sent. However, when modeling everyday communication it is less realistic: when two people are communicating and a third person does not know what they are communicating about the third person usually thinks any message is possible, and does not have a specific possible message in mind.

One possible solution that comes to mind is to give every agent a personal set of messages she is aware of. However, this does not solve the problem. For consider an agent $a$ that does not know whether $p$ is the case, and suppose a message $m$ is sent to some other agent $b$, informing her that $p$ is the case. Then, even if $a$ is not aware of $m$, something changes in the model that $a$ can notice: after $m$ was sent $a$ must hold it for possible that the other agent $b$ has learnt something about $p$.

Look at this informally. How can an agent $i$ ever know for sure that another agent $j$ does not know whether $p$? Suppose initially $[i](\neg[j]p \land \neg[j]\neg p)$. Suppose $i$ holds it for possible that some other agent $k$ knows whether $p$. In other words, $\langle i \rangle((k)p \lor (k)\neg p)$ holds. How can this situation persist? How can $i$ be sure that $k$ does not send a secret message $(k, p, j)$ or $(k, \neg p, j)$?

One possible solution would be to always start from initial models where $[i]((\neg[j]p \land \neg[j]\neg p)$ does not hold, for any $i, j, p$. However, this has the disadvantage of blowing up the size of the initial models. In Chapters 5 and 6 I will present two different approaches that immediately take all possible messages into account, instead of using a limited vocabulary of messages. This is more realistic in some situations, but it will become clear that this comes with a price in the form of infinitely large models. Especially in game-theoretic situations where there is a limited number of messages or signals that can be sent in each round, or when the agents are following some known protocol consisting of a limited number of possible messages, the approach given in this chapter is a lot more appropriate and efficient.