Proving Norm Compliancy of Protocols in Electronic Institutions

Huib Aldewereld
Frank Dignum
John-Jules Ch. Meyer
Javier Vázquez-Salceda

institute of information and computing sciences, utrecht university

technical report UU-CS-2005-010

www.cs.uu.nl

Proving Norm Compliancy of Protocols in Electronic Institutions*

Huib Aldewereld Frank Dignum John-Jules Ch. Meyer
Javier Vázquez-Salceda

01-02-2005

Abstract

There is a wide agreement on the use of norms in order to specify the expected behaviour of agents in open MAS. However, in highly regulated domains, where norms dictate what can and cannot be done, it can be hard to determine whether a desired goal can actually be achieved without violating the norms. To help the agents in this process, agents can make use of predefined (knowledge-based) protocols, which are designed to help reach a goal without violating any of the norms. But how can we guarantee that these protocols are actually norm-compliant? Can these protocols really realise results without violating the norms? In this paper we introduce a formal method, based on program verification, for checking the norm compliance of (knowledge-based) protocols.

1 Introduction

Agents in open multiagent systems are sometimes as diverse as humans, as heterogeneous agents may behave in different ways in trying to complete their specified tasks. As some of this behaviour might not be desired, one needs mechanisms to constrain the behaviour of the agents joining the system by defining what is right and wrong. By doing so one can guarantee a safe and regulated environment for the agents to work in.

An Electronic Institution (eInstitution) is such an environment, where the expected behaviour of the agents joining the institution is described by means of an explicit specification of *norms* [9] [26]. As in human institutions, norms in eInstitutions are stated in such a form that allows them to regulate a wide range of situations over time without the need for modification. To achieve this stability, the formulation of norms abstracts from a variety of concrete aspects [12] [26]; i.e., norms are expressed in terms of concepts that are kept vague and ambiguous on purpose [14].

Because of their abstract nature, norms tend to be hard to understand and, as in real life, adhering to the norms that regulate the institution of which you are a part can be, at the least, a bit challenging. It is not unlikely that in highly regulated systems agents (and humans alike) might become overly cautious, trying not to violate any of the norms and thereby seriously reducing their efficiency and even influence the outcome and success of their goals, i.e., desired results, that *are* possible to achieve, might not be achieved anymore because the agent believes that performing the actions leading to the desired result could be violating the norms. In order to help agents act in such an environment and

*This research was supported by the Netherlands Organisation for Scientific Research (NWO) under project number 634.000.017

increase their efficiency as well as their chance of success one can specify *norm-compliant protocols* for the tasks that are to be accomplished in the institution.

Such protocols provide the means to complete one's tasks and achieve one's goals without the need to worry about the norms that regulate the institution. A norm-compliant protocol is a guideline that makes sure that, when followed, one does not violate any of the norms, and as such it provides a quick and efficient manner to do the tasks one is assigned, since one does not need to review the norms and check norm compliance whenever one is planning to perform an action. In order to guarantee this the protocol should be checked for norm compliance, which means that one should check that no norms are violated by the protocol during it's execution in all situations, i.e. the norm compliance of the protocol should not depend on the state of the world. Therefore, the protocol should provide a violation-free path to achieve the agent's goals. As long as the protocol is followed *to the letter* the agent should stay out of harm's way.

Please note that we are not presenting a manner of implementing norms in MAS, since that would require a more direct enforcement instead of just checking the norm compliance of the protocols used by the agents. Norms in MAS can be enforced either by the use of violation detection and sanctioning these violations, or by directly constraining the agents such that they can only do actions that do not violate norms[1] [25]. Protocols are guidelines and agents are, as such, not (necessarily) constrained to follow the protocols; protocols, however, provide norm-compliant means to achieve one's tasks but not necessarily the only norm-compliant means.

In this paper we present a formal method for checking the norm compliance of protocols based on temporal logic, using an approach used in concurrent programming [15]. We have chosen this approach over traditional techniques for verifying (sequential) programs, because verification methods for concurrent programs and temporal logics allow us to see whether norms are violated in intermediate steps as well, where traditional techniques are only for checking the input and output of a program. In section 3 we present this formal framework and we explain some of the difficulties one will encounter when formalising protocols and norms. In sections 4 and 5 we show how the formalism works on an example protocol taken from the medical domain. We end this paper with a discussion on norm compliance. Finally we draw some conclusions and propose some future work.

The example problem that we are going to use throughout this paper is a real-life protocol that describes which steps should be taken in order for a doctor to determine whether he can extract the organs of a donor or not (for the use of transplantation). A simplified version of this protocol is included in figure 1. Although this protocol is knowledge-based[2], the method we present in this paper can be applied to other sorts of protocols as well.

2 Theory

In order to guarantee that a protocol is actually norm-compliant, we want to be able to formalise the protocol and applicable norms and proof the norm compliance by deriving that specific propositions hold for that protocol. For the formalisation of protocols we have chosen to regard them as programs, since there is a clear similarity between protocols and programs, mainly because protocols are, like programs, actually nothing more than a set of actions combined with decisions that determine which actions are performed in what situation.

[1]Of course, combinations of both methods are possible as well, e.g., use violation management in general and constrain the agents on actions that lead to violations that cannot (easily) be detected.

[2]Knowledge-based protocols depend on the knowledge of the agent to decide which action is to be performed next, which results in a change of knowledge. The goal of such protocols is to determine whether something is known by the agent at the end of the protocol's execution.

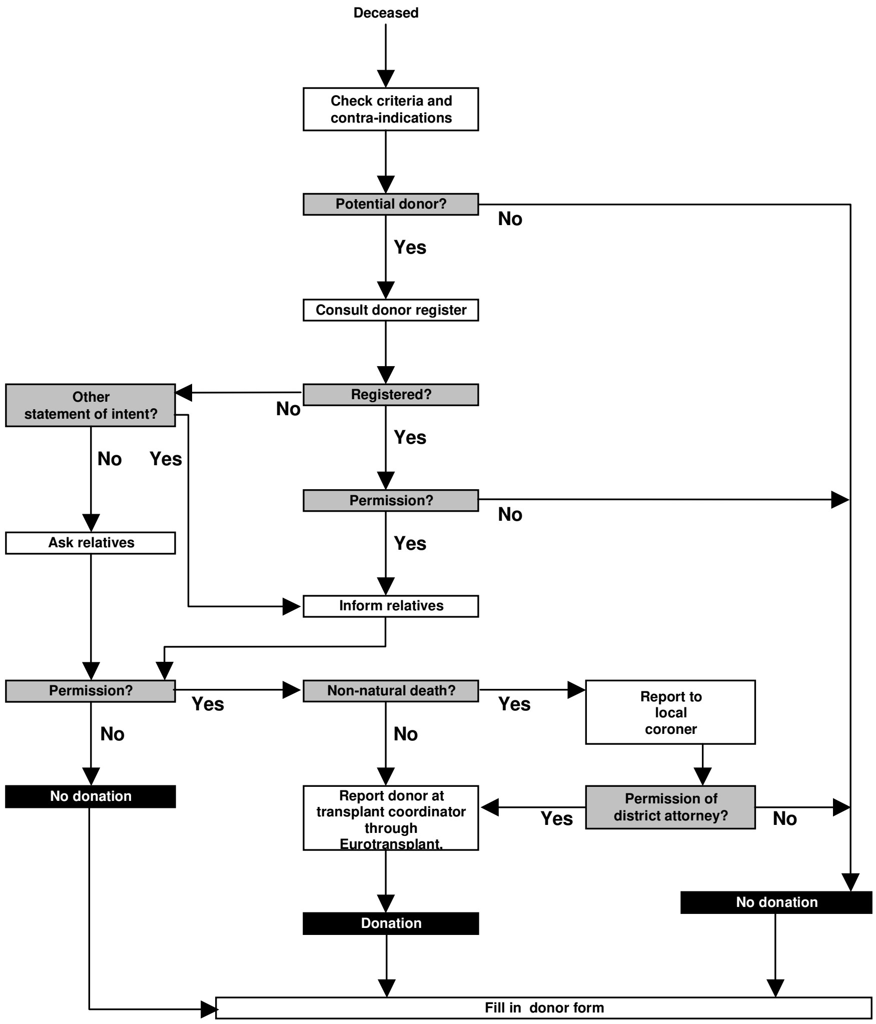

Figure 1: Protocol for organ donation.

It is obvious that for the norms we need a deontic-like logic, or at least a logic that allows the incorporation of deontic operators, and it became clear that we were going to need a temporal logic because of the many temporal relations that exist in law and regulations. Instead of the more complex branching-time logics, like CTL and CTL*, that are for example used in [4] and [6] to express norms, we have chosen the simpler linear-time logics, because the added complexity of branches in time, normally needed for expressing non-determinism, is not required in the domain we are considering. Although protocols themselves may branch this does not imply the need for a branching time logic, since all branches in the protocol are due to different conditions in the world, and not due to non-determinism. We can, therefore, handle the verification of protocol by using a linear-time temporal logic, however, we may need to consider different conditions when checking the protocol, leading to more than one verification process.

Since we consider protocols to be programs and can view the formalised norms as constrains on these programs, the norm compliance of the protocol is reduced to an invariant property holding for this program; that is to say, if we can proof the invariance of non-violation, i.e. no violation is taking place over the entire program run, we can say that the protocol is norm-compliant. This means that, when the protocol is started in a state/world that is violation free, following the protocol to the letter would not make the run include worlds that satisfy a violation. Although the invariance property is enough for checking whether a protocol is norm-compliant, we would rather say a bit more about the protocols we are trying to prove. If, for instance, the protocol is composed of actions that are actually not regulated by the norms in consideration, the protocol is trivially norm-compliant, e.g. being at home all day makes one compliant to the traffic laws all the time, since one is not actually participating in traffic situations. Since such protocols are hardly any interesting for the domain under consideration ($\mathtt{skip}^*$ satisfies almost all violation invariances, though it is not very interesting from an institutional point of view), we want to prove that a protocol is actually goal-oriented. This can be solved by introducing a liveness property that should be satisfied by the protocol.

Clearly, for checking whether a protocol satisfies the properties mentioned above, we need to find out what happens in the states between the start and end of the protocol, which makes traditional (sequential) program verification unfit for this domain. However, a verification method for concurrent programs, such as presented in [15], combined with a temporal logic, does provide the means necessary for checking whether violations do or do not occur during a protocol run. However, the formalism described in [15] is, naturally, limited to checking properties and the likes for concurrent programs, not for checking norm compliance; therefore, we had to change and expand the formalism with the means to express norms and violations. In doing so we encounter some problems, like the need for linking the abstract normative level to the concrete protocol, which we discuss in more detail in the following sections.

2.1 Permissions and Non-Permissions

One of the problems we encountered has to do with permissions. Where obligations and prohibitions have a clear intuitive meaning when it comes to searching for violations, i.e. the process of finding out whether norms are violated, permissions have no meaning in this context at all, since, because of their definition, they cannot be violated. Permissions are norms that only define when violations do not occur. Therefore, in a domain where only permissions exist (no obligations or prohibitions), no violations will occur ever, since none of the holding norms can be violated.

Permissions in law are, however, a bit more useful since they are used to reduce the strict restraints that are posed by the obligations and prohibitions. Consider again a domain without any norms, in such a domain the obligations and prohibitions are used to pose restrictions on people/agents in order to define acceptable and unacceptable behaviour and situations. Since some of these restrictions might be too broad, thus restricting too much situations or too many people/agents which should actually be allowed, permissions can be used to counter these effects and reinstate some of the accepted states and actions as being ‘legal’.

Some domains are even that restricted that permissions are actually needed to perform actions at all. The domain of organ transplantations is such a domain. This domain is actually a part of a larger and more general domain where regulations and laws restrict people to extract organs from people, since cutting in people and operating on the dead is considered a bad thing (desecration of the dead is still registered as a moral offence in the criminal code). In order to make it possible to extract organs to save lives (which is, of course, considered a good thing) in this domain where everything is considered ‘illegal’ and therefore prohibited, permissions had to be introduced. These permissions would then define in which situations which actions are actually ‘legal’, i.e. not raising a violation by breaking the ‘default’ of not being allowed to violate a corpse.

More interesting is, however, that in such a domain not being permitted to do something would actually mean that your are forbidden to do it, because of this general default, thereby making permission necessary in the checking for violations during protocol runs. For the domain of organ transplantations that we are considering here, this needs to be formalised in order to detect violation that arise because of actions (or states) not explicitly permitted. We model this relation between permissions and prohibitions by means of a technique similar to *negation as failure*, as used in logic programming [20] [24]; the inability to derive that you are permitted to do α means that you are forbidden to do α:

$$\sim P\,\alpha \rightarrow F\,\alpha$$

Because of the default assumption that anything is prohibited unless specified otherwise, one can say that the legal system modelled by this rule is *restrictive* in nature, thereby characterising the legal system (see [22] for a discussion on the character of legal systems).

Now, since we add the $\sim P\,\alpha \rightarrow F\,\alpha$ rule to our system to model that permissions are exceptions to 'default' (where this general prohibition might only follow from the characteristic nature of the law), we get into trouble if we don't assume that permissions follow from obligations (i.e., $O\,\alpha \rightarrow P\,\alpha$). This assumption is an axiom in most deontic systems, but we are reluctant to insert it because we feel that in the real world this might not necessarily hold. It is, however, true that a normative system is supposed to uphold this principle, i.e., normative systems should be designed such that obligations to do α can actually be fulfilled, but this is actually the ideal situation. When designing a normative system (thus, when laws are postulated) it should be taken into account that obligations can be fulfilled. However, it is not necessarily the case that this condition is always met in normative systems (due to mistakes in designing the system). In the case presented in sections 4 and 5, however, we can safely assume that this assumption has not been violated by the law-maker.

2.2 Linking Levels

Norms, as mentioned before, are generally specified by means of vague and ambiguous concepts that not always have **one** clear meaning when translated into concrete situations. Norms, such as the law, are specified in this way to ensure that the norm can handle many different concrete situations, even those situations one did not forsee when designing a law. This vagueness makes norms hard to translate into a formal framework while still able to use the formalisation for checking norm compliance of protocols. Since protocols are concrete procedures, the actions taken in such a protocol are not directly mappable to the abstract actions mentioned in the norms. Although the meaning of the abstract action "to discriminate" is quite clear to most people, the problem is that normal protocols do not include a *discriminate* action; protocols can, however, include actions that, in certain cases, can be considered to be "discriminative" of nature.

Of course one can try and iron out all of these discrepancies between this abstract normative level and the concrete protocol when translating the norms (i.e., translate as many abstract actions/concepts into concrete ones), but by doing so one risks losing valuable information. The latter can happen when a vague norm is misinterpreted or situations arise that were not accounted for when translating the norms. This means that situations might arise when a violation occurs (or is absent) according to the interpreted norms, but not according the original norm. In such cases one will have to check the original norms to see whether the occurrence (or absence) of a violation is correct, and since one cannot easily determine in which cases this should be done, all the original norms are to be checked in all cases to determine the correctness of the (non-)violation occurrences.

Instead we rather use a high-level formalisation of the norms which should provide enough room for the formal representation of the norms to keep their vague and ambiguous nature (for a similar approach see [25]. This representation, however, contains, due to the high-level of abstraction, terms and concepts that cannot be clearly related to terms and concepts used in the concrete protocol. Therefore, in order to check whether certain concrete actions and situations contained in the protocol violate a norm we map these concrete actions and situations to the abstract actions and situations described by the norms. These mappings can be of two different forms:

- Either a concrete atomic action a in the protocol is considered to be an instance of an abstract action α mentioned in the norms;

- Or a sequence of concrete atomic actions $a = a_1; a_2; \ldots; a_n$ is considered an implementation of an abstract (complex) actions α mentioned in the norms.

Note that these mappings are one-way only, that is, a concrete action a in a protocol can be considered to be an instance of an abstract action α mentioned in the norms, but since there are many more actions conceivable that can be considered instances of α, we cannot say that a and α are equivalent. These relations between abstract and concrete actions will be formalised by using an *counts as* operator ($DO_x\, a \rightsquigarrow_{viol} DO_x\, \alpha$: meaning *doing a counts as doing α in the case of checking for violations*). In this report we use a simplified version of the *counts as* as defined in [11] and [13]. Since we are only using the $\rightsquigarrow$-operators and the proofs are not dependent on the syntax and semantics of this operator we refrain from giving formal meaning to this operator (interested reader should check the formal aspects of the *counts as* in, for instance, [11] and [13]).

An obvious advantage of this approach is that the interpretation of the abstract concepts is made clear in the counts-as. If the norm compliance proof of a certain protocol seems questionable to someone, one could check the counts-as definitions to see which interpretations where used.

2.3 Knowledge Representation

Although most concepts, such as “removing organs” or “being a doctor”, can be represented easily in a logical framework, there are concepts that have a more intangible, context dependent meaning which is hard to formalise. Please note, however, that the representational complexity of a concept used in a certain context is also determined by the use intended for the concept. While “being a doctor” might be formalised in first-order logic as $doctor(x)$, expressing that element x of the domain has the property of being a doctor, and thus leaving the meaning of being a doctor implicit in specification of the domain, a context might require that one explicitly expresses what it means to be a doctor, e.g. $graduated(x, medicine, university) \wedge specialised\_in(x, surgery) \wedge \ldots \rightarrow doctor(x)$. Although, from a formal point of view, the meaning of $doctor(x)$ is still that element x has the doctor property, the logical formalism now also has the ability to express what it means for elements to have this property.

All this means that trying to represent a domain when verifying the norm compliance of a protocol is actually the process of “fleshing out” the meaning of the concepts used in that domain. Without this formal meaning the verification cannot take place and with the wrong meaning the verification (of a protocol that is actually norm-compliant) could fail. In a sense, the process of verifying norm compliance is actually about defining the logical context of the concepts, and therein very similar to the legal domain where legality/illegality of certain events/actions is determined by the interpretation of the details of the investigation.

Of course these representational choices are debatable, that is one might not agree with a certain representation for this domain, and therefore it also means that the representation can be regarded as an assumption; i.e. given the representation of the concepts presented in section 4.1 we can prove that the protocol of figure 1 is norm-compliant.

3 Logical Formalism

In this section we introduce the formalism we are using for checking the norm compliance of protocols. The basis of this formalism is provided by a method used for the verification of concurrent programs [15], which uses a linear-time temporal logic (LTL) to formalise programs and verify properties of concurrent programs, like mutual exclusion, deadlock freedom, termination, etc.

To use this formalism we consider the protocol to be a program, which we can then formalise by means of the constructs that we discuss in section 3.2. In order to check for norm compliance we need to formalise the norms that are applicable to the protocol which we accomplish by translating the norms into the linear-time temporal logical formulas. In order to be able to do this this we extend the logic of [15] with deontic and past-time operators whose syntax and semantics is discussed in section 3.3. At the end of this section we then look at how we can use this concurrent program verification formalism for verifying that a protocol is norm-complaint.

3.1 Base Formalism

Before we introduce the formal syntax for describing the protocol we introduce the base formalism that is going to be extended in the following sections to provide us with the formal means to verify the norm compliance of protocols. The logic we present here is going to be linked to the protocol syntax, as we describe in 3.2 and extended with operators to allow the formalisation of the norms, which we discuss in section 3.3.

Let $\mathscr{L}_P$ be a classic propositional logic and $\mathscr{A}$ be a set of atomic actions. The classical propositional logic is constructed from atomic propositions and the logical operators $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$, which are used to construct formulas in the normal manner. As semantics for the propositional logic $\mathscr{L}_P$ we use a valuation function $v(p)$ to assign a truth value to the atomic proposition p. We can then define the truth value of a formula P as:

Definition 1 (Semantics of $\mathscr{L}_P$)

- $\mathcal{M} \models p$ *iff* $v(p) = \textbf{true}$*, for all atomic propositions* $p \in \mathscr{L}_P$*;*
- $\mathcal{M} \models \neg P$ *iff* $\mathcal{M} \not\models P$*;*
- $\mathcal{M} \models P \wedge Q$ *iff* $\mathcal{M} \models P$ *and* $\mathcal{M} \models Q$*.*

The semantical definition of the other operators of $\mathscr{L}_P$ can be derived from the combination of these.

Next we extend this base logic to an linear-time temporal logic by including the following temporal operators with the following intuitive meaning:

$\bigcirc A$:	"A holds at the time point immediately after the reference point" (*next-time* operator).
$\Box A$:	"A holds at all time points after the reference point" (*always* or *henceforth* operator).
$\Diamond A$:	"There is a time point after the reference point at which A holds" (*sometime* or *eventually* operator).

$\odot A$:	"A holds at the time point immediately before the reference point" (*just* or *yesterday* operator).
$\diamondsuit A$:	"There is a time point before the reference point at which A holds" (*past-time* operator).
A **atnext** B :	"A will hold at the next time point that B holds" (*first time* or *atnext* operator).
A **until** B :	"A holds at all following time points up to a time point at which B holds" (*until* operator).

Formally, this means that a *temporal propositional logic* $\mathscr{L}_{TP}$ (*with kernel* $\mathscr{L}_P$) is an extension of $\mathscr{L}_P$ as follows. The alphabet of $\mathscr{L}_{TP}$ is that of $\mathscr{L}_P$ with the addition of the operators $\bigcirc$, $\Box$, $\odot$, $\diamondsuit$, **atnext** and **until**. Using this language we come to the following definition of formulas in $\mathscr{L}_{TP}$:

Definition 2 (Formulas)

1. *Every atomic proposition is a formula.*
2. *If A and B are formulas then* $\neg A, (A \wedge B), \bigcirc A, \Box A$, $\odot A$, $\diamondsuit A$, $(A$ **atnext** $B)$ *and* A **until** B *are formulas.*

Next to the operators specified in definition 2 we derive, in the usual manner, the remaining operators as abbreviations of those specified: $A \to B \equiv \neg(A \wedge \neg B)$, $A \vee B \equiv \neg(\neg A \wedge \neg B)$, $A \leftrightarrow B \equiv (A \to B) \wedge (B \to A)$, and $\Diamond A \equiv \neg\Box\neg A$.

We assume that the set atomic propositions of $\mathscr{L}_{TP}$ to be split into two subsets which we will call the set of global and local propositions, respectively. Unlike local propositions, which inherit their truth value from the *temporal state* of the model, global propositions do not change their truth value over time.

The semantics of the temporal propositional logic $\mathscr{L}_{TP}$ is given by a possible worlds, Kripke-like, model. This model include the valuation function v as described earlier, which is extended to handle the temporal changes of local propositions. Next to this valuation function, the model includes a set temporal states, which are used to define the truth value of these local propositions:

Definition 3 (Models of $\mathscr{L}_{TP}$)
A propositional model $\mathcal{M} = \langle v, \mathbf{W} \rangle$ *for the temporal propositional logic* $\mathscr{L}_{TP}$ *consists of:*

- *a valuation function v to define the truth value of the kernel* $\mathscr{L}_P$ *of* $\mathscr{L}_{TP}$*;*
- *an infinite sequence* $\mathbf{W} = \{\eta_1, \eta_2, \eta_3, \ldots\}$ *of states where every* η_i *is a local propositional valuation with respect to v (i.e., an truth assignment for every local proposition).*

Using these models we can now define the value of formulas in $\mathscr{L}_{TP}$. We introduce a combined valuation function v^s, derived from the global valuation function v and local valuation s, to define the truth value of atomic propositions in state s of the model. We can then say that a atomic proposition p is true in a state s of model $\mathcal{M}$ (i.e., $\mathcal{M}, s \models p$) if and only if $v^s(p) =$ **true**. The meaning of this expression, and the semantics of the other formulas of $\mathscr{L}_{TP}$ are given in the following definition:

Definition 4 (Semantics of $\mathscr{L}_{TP}$)
For every model $\mathcal{M} = \langle v, \mathbf{W} \rangle$, formulas A and B, and $s \in \mathbb{N}_0$:

$$
\begin{array}{rcl}
\mathcal{M}, s \vDash A & \Leftrightarrow & v^s(A) = \mathbf{true} \quad \textit{for every atomic proposition} \\
 & & \textit{where } v^s(A) = v(A) \textit{ for every global proposition } A \textit{ and} \\
 & & \quad v^s(A) = \eta_s(A) \textit{ for every local proposition } A \\
\mathcal{M}, s \vDash \neg A & \Leftrightarrow & \mathcal{M}, s \nvDash A \\
\mathcal{M}, s \vDash A \wedge B & \Leftrightarrow & \mathcal{M}, s \vDash A \textit{ and } \mathcal{M}, s \vDash B \\
\mathcal{M}, s \vDash \bigcirc A & \Leftrightarrow & \mathcal{M}, s+1 \vDash A \\
\mathcal{M}, s \vDash \Box A & \Leftrightarrow & \forall t \geq s : \mathcal{M}, t \vDash A \\
\mathcal{M}, s \vDash \odot A & \Leftrightarrow & \mathcal{M}, s-1 \vDash A \quad \textit{for all } s > 0 \textit{ and} \\
 & & \mathbf{false} \textit{ otherwise} \\
\mathcal{M}, s \vDash \diamondsuit A & \Leftrightarrow & \exists 0 \leq t \leq s : \mathcal{M}, t \vDash A \\
\mathcal{M}, s \vDash A \textbf{ atnext } B & \Leftrightarrow & \forall t \geq s : \mathcal{M}, j \nvDash B \quad \textit{or} \\
 & & \mathcal{M}, u \vDash A \quad \textit{for the smallest } u > s \textit{ where } \mathcal{M}, u \vDash B \\
\mathcal{M}, s \vDash A \textbf{ until } B & \Leftrightarrow & \exists t \geq s : \mathcal{M}, t \vDash B \quad \textit{and} \\
 & & \forall s < u < t : \mathcal{M}, u \vDash A
\end{array}
$$

Since the norms as given in Appendix B are stated in a more general approach, we need predicate logic to formalise these abstract norms into logical formulas. This need for a temporal predicate logic for formalising norms mostly stems from the fact that norms are applicable to everyone, which would mean, in the case of human laws as we are using here, that the domain would become enormous (if not infinite). However, in this verification we are looking at the application of these norms to a small part of that infinite domain, since we are only talking about a particular doctor (the one that is following the protocol), a particular patient (namely the one that just died and whose organs are wanted), etc. Moreover, since this particular application of the norms does not require the use of variables (and therefore, quantifiers as well), we are actually using predicate logic in a propositional manner. Thus, by restricting the predicate logic to exclude variables and quantifiers, and making it applicable to a small (finite) domain, we obtain a logic that is very similar to the logic introduced above. In order to use the temporal propositional logic introduced above, we just need to extend it with methods for defining the truth value of the predicates. We do this by extending the model with a (non-empty) domain D and interpretation functions for mapping the predicate, function and constant symbols to relations, functions and elements of the domain, respectively. Thus, the extension of the models for $\mathscr{L}_{TP}$ is the following:

Definition 5 (Extended Models of $\mathscr{L}_{TP}$)
an extended model $\mathcal{M} = \langle D, I, \mathbf{W} \rangle$ for the propositional temporal logic $\mathscr{L}_{TP}$ consists of:

- *a non-empty domain D of objects;*
- *an interpretation function I that:*
  - *assigns an n-ary relation $R \subseteq D^n$ to each global predicate letter in $\mathscr{L}_{TP}$;*
  - *assigns an n-ary function $f^n : D^n \rightarrow D$ to each function letter in $\mathscr{L}_{TP}$;*
  - *assigns a fixed element of D to each constant of $\mathscr{L}_{TP}$;*

- *an infinite sequence* $\mathbf{W} = \{\eta_1, \eta_2, \eta_3, \ldots\}$ *of states where every* η_i *is a local predicate interpretation with respect to v (i.e., assigns a n-ary relation to every local predicate).*

Note that we again make a difference between local and global predicates, by introducing different interpretation functions for each. However, the mapping of functions and constant symbols is the same in every state. Of course the assignment of truth values and the semantics of atomic formulas (as defined in the first item of definition 4) needs to be redefined as well, in order to be able to handle these propositionally used predicates.

Definition 6 (Valuation function v**)**
For every model $\mathcal{M} = \langle D, I, \mathbf{W}$*, n-ary predicate P, terms* $t_1, \ldots, t_n$*, and* $s \in \mathbb{N}$*:*

$$\begin{aligned} \mathcal{M}, s \models P(t_1, \ldots, t_n) \quad \Leftrightarrow \quad & v^s(P(t_1, \ldots, t_n)) \\ & \textit{where } v^s(P(t_1, \ldots, t_n)) = I(P)(I(t_1), \ldots, I(t_n)) \textit{ for every global predicate } P \\ & \textit{and } v^s(P(t_1, \ldots, t_n)) = \eta_s(P)(I(t_1), \ldots, I(t_n)) \textit{ for every local predicate } P \end{aligned}$$

Note that the interpretation of terms (i.e. functions and constants) is fixed throughout the model. This interpretation is resolved recursively by passing the interpretation function down to the terms used as argument of the function. This means that if a term t_i of a global predicate P is a function $f(u_1, \ldots, u_m)$, the interpretation would be the following:

$$\begin{aligned} I(P(t_1, \ldots, t_i, \ldots, t_n)) \quad & \Leftrightarrow \quad I(P)(I(t_1), \ldots, I(t_i), \ldots, I(t_n)) \\ & \Leftrightarrow \quad I(P)(I(t_1), \ldots, I(f(u_1, \ldots, u_m)), \ldots, I(t_n)) \\ & \Leftrightarrow \quad I(P)(I(t_1), \ldots, I(f)(I(u_1), \ldots, I(u_m)), \ldots, I(t_n)) \\ & \qquad \vdots \end{aligned}$$

3.2 Syntax & Semantics of Protocols

As mentioned earlier, we consider protocols to be (a special kind of) programs. In order to be able to express protocols/programs we introduce a formal syntax for the protocols. This syntax, representing the protocol, is then linked to the formal models of the logic in order to make it possible to use the logical formalism, as presented in the previous section, for proving various properties of the protocols, including norm compliance. In this paper we constrain ourselves to the use of single sequential protocols, although more complex protocols, including parallel and cyclic ones, can be expressed an used with the formalism presented after the necessary expansions.

We formally define the structure of a program as the following:

Definition 7 (Formal Protocol)
A protocol Π *is defined over propositional logic* $\mathscr{L}_P$ *and set of atomic actions* $\mathscr{A}$ *as a string of the form*

$$\Pi = \textbf{initial } R \,;\, \Pi_c$$

where R is a formula of $\mathscr{L}_P$ *called the initial condition, and* Π_c *is a* ⟨non-cyclic protocol component⟩

This definition shows that protocols are constructed by means of two components; 1) a set of *initial statements* that are used to initialise global variables at the start of the program-run, and 2) a *sequential program component*. This sequential program component is the formal specification of the protocol and uses atomic actions (including variable assignments) from $\mathscr{A}$, **if-then-else** and **while-do** clauses to specify the protocol. The conditions of these latter two statements are logical formulas (from $\mathscr{L}_P$). For ease of reference all statements in a protocol are labelled, with labels being unique throughout the protocol, i.e., no two labels occurring in a protocol are equal. Definition 8 shows the structure of th sequential component.

Definition 8 (Non-cyclic Protocol Components)

⟨non-cyclic protocol component⟩ ::= ⟨statement sequence⟩; ⟨label⟩ : **stop**
⟨statement sequence⟩ ::= ⟨statement⟩ | ⟨statement⟩; ⟨statementsequence⟩
⟨statement⟩ ::= ⟨label⟩ : ⟨unlabelled statement⟩
⟨unlabelled statement⟩ ::= ⟨element of $\mathscr{A}$⟩ |
if ⟨condition⟩ **then** ⟨statement sequence⟩
else ⟨statement sequence⟩ **fi** |
if ⟨condition⟩ **then** ⟨statement sequence⟩ **fi** |
while ⟨condition⟩ **do** ⟨statement sequence⟩ **od**

Protocols are executed in discrete steps. This means that each step the program component executes a *(indivisible) action*, which is either a statement of $\mathscr{A}$ (i.e. an atomic action or variable assignment), a test of the condition in an **if** statement together with the entry into the respective branch, or a test of the condition of a **while** statement together with the respective entry into or exit from the loop. This means, that only one step of the protocol, labelled by a protocol label, is performed per discrete step, i.e. if a protocol has steps labelled $\lambda_1, \ldots, \lambda_n$, it performs only one of those labelled actions/steps (λ_i) each step. Therefore, we can distinguish the protocol steps by means of the protocol labels attached to the step performed, that is to say, we can determine the "state" of the protocol by referring to the label of the step that the protocol is about to perform.

Now that we have shown the syntax of protocols we determine what the constructs introduced in definition 8 actually mean. To formalise this 'operational semantics' of a program Π, we introduce three concepts, for any statement sequence ψ, $\mathscr{M}_\psi$ the labels occurring in ψ and $\mathscr{F}(\mathscr{L}_P)$ being the set of all formulas of $\mathscr{L}_P$; entry(ψ) $\in \mathscr{M}\psi$ for determining the entry label of ψ, the set trans(ψ) $\subset \mathscr{M}_\psi \times \mathscr{F}(\mathscr{L}_P) \times \mathscr{M}_\psi$ defining the transitions between statements in ψ, and the set exits(ψ) $\subset \mathscr{M}_\psi \times \mathscr{F}(\mathscr{L}_P$ defining the exit conditions of ψ. Using these concepts we can inductively define the meaning of the statements mentioned in definition 8.

Definition 9 (Operational Semantics of Protocol Statements)

1. $\psi \equiv \alpha$: ⟨element of $\mathscr{A}$⟩ :
   entry(ψ) = α,
   trans(ψ) = $\emptyset$,
   exists(ψ) = $\{(\alpha, \top)\}$.
2. $\psi \equiv \alpha$: **if** B **then** ψ_1 **else** ψ_2 **fi** :
   entry(ψ) = α,
   trans(ψ) = trans(ψ_1) $\cup$ trans(ψ_2) $\cup$ $\{(\alpha, B, \text{entry}(\psi_1)), (\alpha, \neg B, \text{entry}(\psi_2))\}$,
   exits(ψ) = exits(ψ_1) $\cup$ exits(ψ_2).
3. $\psi \equiv \alpha$: **if** B **then** ψ_1 **fi** :
   entry(ψ) = α,
   trans(ψ) = trans(ψ_1) $\cup$ $\{(\alpha, B, \text{entry}(\psi_1)\}$,
   exits(ψ) = exits(ψ_1) $\cup$ $\{(\alpha, \neg B)\}$.

4. $\psi \equiv \alpha$: **while** B **do** ψ_1 **od** :
   entry(ψ) = α,
   trans(ψ) = trans(ψ_1) $\cup \{(\alpha, B, \text{entry}(\psi_1))\} \cup \{(\beta, C, \alpha) | (\beta, C) \in \text{exits}(\psi_1)\}$,
   exits(ψ) = $\{(\alpha, \neg B)\}$.
5. $\psi \equiv \alpha$: $\psi_1; \psi_2$ *(ψ_1 unlabelled statement, ψ_2 statement sequence)* :
   entry(ψ) = α,
   trans(ψ) = trans(ψ_1) $\cup$ trans(ψ_2) $\cup \{(\beta, C, \text{entry}(\psi_2)) | (\beta, C) \in \text{exits}(\psi_1)\}$,
   exits(ψ) = exits(ψ_2).

Now, in order to complete the semantics we define the transitions for a protocol part Π_c as mentioned in definitions 7 and 8, i.e. trans(Π_c) $\subset \mathscr{M}_{\Pi_c} \times \mathscr{F}(\mathscr{L}_P) \times \mathscr{M}_{\Pi_c}$, as:

Definition 10 (Operational Semantics of Protocols)
$\Pi_c \equiv \psi; \alpha$: **stop** *(for a statement sequence ψ)* :
trans(Π_c) = trans(ψ) $\cup \{(\beta, C, \alpha) | (\beta, C) \in \text{exits}(\psi)\}$.

Informally $(\alpha, C, \beta) \in \text{trans}(\Pi_c)$ means that: In Π_c, execution can proceed in one step from α to β if C holds. Therefore, trans(Π_c) expresses all possible steps that can be taken in the protocol-part, and thereby describes all possible paths within the protocol.

Let us explain the semantics a bit more by means of a simple example. Consider the following protocol (which is a simplification, but of similar structure to the protocol we use in sections 4 and 5):

$\Pi_c \equiv$ π_1 : **if** $\varphi_1 \wedge \varphi_2$
  then π_2 : $\langle action_1 \rangle$
  else π_3 : **if** $\varphi_1 \wedge \neg\varphi_2$
    then π_4 : $\langle action_2 \rangle$
    fi
  fi;
π_5 : **stop**

After applying the above definition, and decomposing the protocol into the parts and completing the trans and exits for each step we obtain the following:

$$
\begin{aligned}
\text{trans}(\Pi_c) &= \text{trans}(\pi_1) \cup \{(\beta, C, \pi_5) \mid (\beta, C) \in \text{exits}(\pi_1)\} \\
\text{trans}(\pi_1) &= \text{trans}(\pi_2) \cup \text{trans}(\pi_3) \cup \{(\pi_1, \varphi_1 \wedge \varphi_2, \pi_2), (\pi_1, \neg(\varphi_1 \wedge \varphi_2), \pi_3)\} \\
\text{exits}(\pi_1) &= \text{exits}(\pi_2) \cup \text{exits}(\pi_3) \\
\text{trans}(\pi_2) &= \emptyset \\
\text{exits}(\pi_2) &= \{(\pi_2, \top)\} \\
\text{trans}(\pi_3) &= \text{trans}(\pi_4) \cup \{(\pi_3, \varphi_1 \wedge \neg\varphi_2, \pi_4)\} \\
\text{exits}(\pi_3) &= \text{exits}(\pi_4) \cup \{(\pi_3, \neg(\varphi_1 \wedge \neg\varphi_2))\} \\
\text{trans}(\pi_4) &= \emptyset \\
\text{exits}(\pi_4) &= \{(\pi_4, \top)\}
\end{aligned}
$$

Which leads, after combining the trans's and exits's from each step of the protocol-part, to the following transitions for Π_c:

$$
\begin{aligned}
\text{trans}(\Pi_c) = \; & \{(\pi_1, \varphi_1 \wedge \varphi_2, \pi_2),\ (\pi_1, \neg(\varphi_1 \wedge \varphi_2), \pi_3),\ (\pi_2, \top, \pi_5), \\
& (\pi_3, \varphi_1 \wedge \neg\varphi_2, \pi_4),\ (\pi_3, \neg(\varphi_1 \wedge \neg\varphi_2), \pi_5),\ (\pi_4, \top, \pi_5)\}
\end{aligned}
$$

This set of transitions precisely captures the paths occurring in protocol-part Π_c, namely $\pi_1 - \pi_2 - \pi_5$, $\pi_1 - \pi_3 - \pi_4 - \pi_5$ and $\pi_1 - \pi_3 - \pi_5$, and thereby expresses exactly what the protocol means.

Linking Protocols to the Logic

The formalism we just introduced for expressing protocols is an exogenous language, and therefore not a part of our logical framework. However, in order to prove properties and aspects of these protocols we need to connect the protocols to the logical model. To accomplish this we introduce the concepts of protocol states and computations (execution sequences), which we can connect to the states in the model of our logical formalism.

Remember that in section 3.1, at the introduction of the base logic, we assumed a partitioning of the predicates into two different kinds, local predicates and global predicates. This distinction is needed for relating protocols to the logic, where we assume that all predicates whose truth value is changed in the protocol (for example, by means of assignment statements) are local predicates, and all others predicates used in the context of the protocol are global predicates. The idea is then that, while the value of global predicates is still assigned by the interpretation function I of the logical model, the logical states $\eta_i \in \mathbf{W}$ are connected to a state of the protocol to assign a value to the local predicates.

Definition 11 (Protocol States)
*For a program Π = **initial** $R\,;\Pi_c$ over $\mathscr{L}_P$ and $\mathscr{A}$, a program state of Π (with respect to the interpretation I) is a tuple $\eta = (\mu, \lambda_i)$ where:*

- *μ assigns an n-ary relation $R \subseteq D^n$ to every local predicate,*
- *$\lambda_i \in \mathscr{M}_{\Pi_c}$ (the label of the action to be executed next).*

If $\lambda_i = \lambda_e$ (i.e., the label of the **stop**-statement of Π_c), we call η the *terminal state*. The part μ of a program state η is just a (partial) predicate interpretation as used in definition 5. This is, therefore, together with a global interpretation I the mapping (v^μ) that associates every formula of $\mathscr{L}_{TP}$ with a truth value. A run, or computation, of a protocol Π can then be defined as the sequence of protocol states such that: a) the run starts in the (logical) state that satisfies the initial conditions, b) the transition between states are only those included in trans(Π), and c) no changes are made in the model after the protocol has reached its end-label.

Definition 12 (Computations of Π)
An execution *sequence of Π (with respect to $\mathbf{S}, \xi$) is an infinite sequence of $W_\Pi = \{\eta_0, \eta_1, \eta_2, \ldots\}$ of protocol states of Π (with respect to $\mathbf{S}$) with the following properties:*

- *$\eta_0 = (\mu_0, \lambda_0)$ and $\mathbf{S}^{\xi,\mu_0}(R) = \mathbf{t}$;*
- *If $\eta_j = (\mu, \lambda_i)$ then $\eta_{j+1} = (\mu', \lambda_i')$ and* trans(Π) *contains an element $(\lambda_i, C, \lambda_i')$ and $v^\mu(C) =$ **true**. If, moreover, λ_i is the label of a statement not in $\mathscr{A}$ then $\mu' = \mu$;*
- *If $\eta_j = (\mu, \lambda_e)$ then $\eta_{j+1} = \eta_j$.*

Definition 12 effectively captures the above mentioned link between the exogenous protocols and the linear temporal logic. It makes sure that the states of the model correspond to the states of a run of the protocol. Note, however, that, since protocols have choice points because of the if-then and while statements, there can be various different models for a protocol (i.e., different models for every different run of the protocol). Likewise, because of changes in the value of global variables, there can be more than one model for a single run (i.e., different models for the same run, but with other instantiations of the global variables).

Also note that, although an execution sequence is defined as an infinite sequence of program states, time itself is actually semi-finite (no program-states before the start of the protocol). This means that protocols cannot use information about previous runs or next runs (unless explicitly modelled). If protocols Π_1 and Π_2 are run one after another (i.e., $\Pi_1;\Pi_2$), Π_1 cannot use information from or about Π_2 and vice versa; the protocols Π_1 and Π_2 are considered as separate runs. This also means that the value of program variables, truth values of predicates and states and information gathered is restricted to the runtime of a protocol. Propositions and predicates *can* change their truth values during a protocol run, however, but only because of actions that are taken in the protocol.

As explained earlier, the defined protocol is extrageneous to the logic, and although there is a connection between the protocol and the models of $\mathscr{L}_{TP}$ by means of the definition of protocol states, we need some constructs in the logic to represent the actions and their results. To this extent we introduce propositions to denote which action is going to be performed in the state, thereby extending the temporal propositional logic $\mathscr{L}_{TP}$ to the temporal propositional logic of protocols $\mathscr{L}_{TP_\Pi}$:

Definition 13 (Action Label Propositions)
The language $\mathscr{L}_{TP_\Pi}$ is obtained from the extension of $\mathscr{L}_{TP}$ with:

- $\mathrm{at}\,\lambda$ *for every $\lambda \in \mathscr{M}_\Pi$*

So every label (λ) of a step in the protocol is now included as an (atomic) formula, intuitively meaning that “the action λ is executed next”. We use these in the logic to denote the state of the protocol, where $\mathcal{M}, s \models \mathrm{at}\,\lambda$ for some protocol state $\eta_s = (\mu_s, \lambda_s)$ in $\mathcal{M}$, if and only if $\lambda = \lambda_s$.

Next we introduce the abbreviations $DO_x\,\lambda \equiv \mathrm{at}\,\lambda$ and $\mathrm{start}_\Pi \equiv \mathrm{at}\,\lambda_0 \wedge R$ (where λ_0 is the start-label of Π, and R are the initial statements of Π). For ease of reference and understanding we also say that $DONE_d\,\alpha \equiv \odot DO_d\,\alpha$, denoting that α was the label of the previous state (and thus stating that α has just been done). Note that the abstract actions in the norms, as specified in Appendix B, are considered part of some external, vague protocol that are linked to the concrete protocol as given by the execution sequence by means of the counts as operator that we introduced in section 2.2.

3.3 Introducing Deontics

Now that we introduced the logics and the manner of formally expressing the protocols, we need to extend the logic with the means to express the laws and norms. To this extend we semantically introduce the deontic operators for expressing permissions (P), obligations (O) and prohibitions (F) and show that these operators are, in fact, abbreviations of complex temporal definitions. To achieve this we introduce a special set of predicates to denote when violations occur [1] [19], thus, we extend the logic $\mathscr{L}_{TP_\Pi}$ with a special violation predicate $viol(x, \alpha, \delta)$. Semantically this predicate works as any other predicate and intuitively denotes that “a violation has occurred because x did (not) do α with respect to deadline δ”.

To handle the temporal aspects of norms, such as deadlines, we used ideas from [4], [6] and [8] and adapted these to be used with the first-order temporal logic as specified above. Norms including deadlines, which are, in fact, obligations to perform an action before some point in time, can intuitively be split into two cases, one where the action is performed before the deadlines passes (and, therefore, no violation occurring) or one where the deadline has passed while the action has not yet been performed (thereby raising a violation). A straightforward modal semantics of this is the following:

Definition 14 (Deadlines)

$$
\begin{aligned}
\mathcal{M}, s \models O_x(DO_x\, \alpha < \delta) \quad \Leftrightarrow \quad & \exists t \geq s : \mathcal{M}, t \models \delta \textit{ and} \\
& (\forall s \leq u \leq t : \mathcal{M}, u \models \neg viol(x, DO_x\, \alpha, \delta)) \textit{ and} \\
& ((\exists s \leq u \leq t : \mathcal{M}, u \models\ DO_x\, \alpha \textit{ and } \mathcal{M}, u \models \Box\neg viol(x, DO_x\, \alpha, \delta)) \textit{ or} \\
& (\forall s \leq u \leq t : \mathcal{M}, u \not\models\ DO_x\, \alpha \textit{ and } \mathcal{M}, t \models \bigcirc viol(x, DO_x\, \alpha, \delta)))
\end{aligned}
$$

This definition captures the two cases described above, intuitively expressing that a) a deadline always occurs sometime, b) until that moment no violations can occur because of the norm, and c) either the action is done before the deadline and no violations occur ever because of this norm, or the deadline passes while the action has not yet been done and a violation occurs.

Note that this semantic definition is equivalent to the following definition as a reduction to LTL.

Proposition 1 (LTL-reduction of Deadlines)

$$
\begin{aligned}
O_x(DO_x\, \alpha\ <\ \delta) \quad \equiv_{def} \quad & \Diamond\delta \wedge \Big[(\neg\delta \wedge \neg viol(x, DO_x\, \alpha, \delta)\ \wedge\ \neg DONE_x\, \alpha)\ \textbf{until} \\
& ((DO_x\, \alpha \wedge \bigcirc(\Box\neg viol(x, DO_x\, \alpha, \delta))) \vee \\
& (\neg DO_x\, \alpha \wedge \bigcirc(\delta \wedge viol(x, DO_x\, \alpha, \delta))))\Big]
\end{aligned}
$$

In a similar fashion we introduce temporal prohibitions, which are prohibitions that hold up to a certain moment in time. Intuitively this would mean that until a certain moment doing that action would lead to a violation. However, it could be the case that this specified moment never occurs, which would mean that the action is prohibited for ever.

Definition 15 (Temporal Prohibitions)

$$
\begin{aligned}
\mathcal{M}, s \models F_x(DO_x\, \alpha < \delta) \quad \Leftrightarrow \quad & \text{If } \exists t \geq s : \mathcal{M}, t \models \delta \textit{ then} \\
& \quad (((\exists s \leq u \leq t : \mathcal{M}, u \models\ DO_x\, \alpha \textit{ and } \mathcal{M}, u \models \bigcirc\Box viol(x, DO_x\, \alpha, \delta)) \textit{ and} \\
& \quad\quad (\forall s \leq w \leq u : \mathcal{M}, w \not\models\ DO_x\, \alpha \textit{ and } \mathcal{M}, w \models \neg viol(x, DO_x\, \alpha, \delta))) \textit{ or} \\
& \quad (\forall s \leq u \leq t : \mathcal{M}, u \not\models\ DO_x\, \alpha \textit{ and } \mathcal{M}, u \models \neg viol(x, DO_x\, \alpha, \delta) \textit{ and} \\
& \quad\quad \mathcal{M}, t \models \bigcirc\Box\neg viol(x, DO_x\, \alpha, \delta))) \\
& \text{and if } \forall t \geq s : \mathcal{M}, t \not\models \delta \textit{ then if } \mathcal{M}, t \models DO_x\, \alpha \textit{ then } \mathcal{M}, t \models \bigcirc viol(x, DO_x\, \alpha, \delta)
\end{aligned}
$$

This definition captures the following intuitive meaning: a) if at sometime in the future δ occurs, no violation has occurred and will occur until either α is done before δ passes, after which a violation will occur, or α will not be done before δ and no violation will occur, or b) if δ will not occur ever, doing α will lead to a violation. Note that we can use this definition for expressing non-temporal prohibitions as well, since these can be obtained by using $\perp$ as the temporal condition. Again, this semantic definition can be reduced to a complex LTL formula, as shown in the following.

Proposition 2 (LTL-reduction of Temporal Prohibitions)

$$
\begin{aligned}
F_x(DO_x\, \alpha < \delta) \quad \equiv \quad & \Big(\Diamond\delta \rightarrow \Big[(\neg\delta \wedge \neg DONE_x\, \alpha \wedge \neg viol(x, DO_x\, \alpha, \delta))\ \textbf{until} \\
& (\bigcirc(\delta \wedge \Box\neg viol(x, DO_x\, \alpha, \delta)) \vee (DO_x\, \alpha \wedge \bigcirc\Box viol(x, DO_x\, \alpha, \delta)))\Big]\Big) \wedge \\
& \Big(\Box\neg\delta \rightarrow \Big[DO_x\, \alpha \rightarrow \bigcirc\Box viol(x, DO_x\, \alpha, \delta)\Big]\Big)
\end{aligned}
$$

Before we introduce the semantics for permission, let us first look at conditional norms. Conditional norms, in the classical sense, mean that whenever the condition holds, the norm is applicable. This would mean, in the case of deadlines, that whenever the condition holds before the deadline (while α has not yet been done) the obligation (to do α before the deadline) is applicable. In the case of temporal prohibitions it means that the condition being true before the temporal constraint has passed 'activates' the temporal prohibition.

Definition 16 (Conditional Deadlines & Prohibitions)

$$\begin{array}{rcl} O_x(DO_x\,\alpha < \delta \mid \sigma) & \equiv_{def} & \Box((\sigma \rightarrow O_x(DO_x\,\alpha < \delta))\ \mathbf{until}\ (DONE_x\,\alpha \vee \delta)) \\ F_x(DO_x\,\alpha < \delta \mid \sigma) & \equiv_{def} & \Box((\sigma \rightarrow F_x(DO_x\,\alpha < \delta))\ \mathbf{until}\ \delta) \end{array}$$

One can see that these definitions obtain the desired results. The first equivalence denotes, intuitively, that conditional deadlines are considered as deadlines being triggered by the condition, but only if the condition holds before either the deadline has passed or the action has been done. The second equivalence expresses, intuitively, that conditional temporal prohibitions are considered as temporal prohibitions triggered by the condition mentioned, as long as that condition holds before the temporal constraint holds, i.e. one is only prohibited to do α if the conditions of the prohibition occur prior to the temporal constraint of the prohibition.

Although we already discussed the connection between permission and prohibitions in section 2.1 as $\sim P_x(DO_x\,\alpha) \rightarrow F_x(DO_x\,\alpha)$, i.e. not being permitted to do α implies one is forbidden to do α, we also need to semantically formalise what it being permitted would mean. We use the classical interpretation of permissions, as presented in [19], meaning that being permitted to do α expresses that doing α does not lead to a violation. Here, however, we introduce semantically conditional permissions and will show that unconditional permissions can easily be obtained from those.

Definition 17 (Conditional Permissions)

$$\mathcal{M}, s \models P_x(DO_x\,\alpha \mid \sigma) \quad \Leftrightarrow \quad \forall t \geq s : \mathcal{M}, t \models \sigma \textit{ then if } \mathcal{M}, t \models DO_x\,\alpha \textit{ then } \mathcal{M}, t \models \bigcirc\neg viol(x, DO_x\,\alpha)$$

This definition clearly expresses that whenever the condition holds, doing the action permitted will not raise a violation. This definition can be used to express unconditional permissions as well, since if we take σ as $\top$, it would mean that whenever one does α, one will not be in violation. This clearly translates to the following reduction in LTL.

Proposition 3 (LTL-reduction of Permissions)

$$\begin{array}{rcl} P_x(DO_x\,\alpha \mid \sigma) & \equiv & \Box(\sigma \rightarrow (DO_x\,\alpha \rightarrow \bigcirc\neg viol(x, DO_x\,\alpha))) \\ P_x(DO_x\,\alpha) & \equiv & P_x(DO_x\,\alpha \mid \top) \\ & \equiv & \Box(DO_x\,\alpha \rightarrow \bigcirc\neg viol(x, DO_x\,\alpha)) \end{array}$$

While trying to formalise the norms used for the example we discuss in sections 4 and 5 we encountered another kind of norm that needed to be expressed. The norm stated that some action α needed to be done after something had been done (which we model as a proposition δ). Clearly we needed an obligation $O_x(DO_x\,\alpha > \delta)$ expressing that one needs to perform α until δ has passed. But simply expressing this in that manner would mean that one can postpone doing α almost indefinitely, without being in violation. Moreover, since we are trying to see whether a protocol is in violation and protocol runs are defined as infinite sequences of protocol states (see section 3.2), we cannot handle this sort of expressions in trying to determine whether a violation will ever occur, since that would need checking infinite numbers of states to see that α is never actually performed. To this extend we introduce the notion of reaction time, which is a more operational view of the norm in question. Although the law does not say when α should be done after δ, by common sense we understand that it means that α should be done sometime *soon* after δ has occurred. We introduce the operator $\odot^i$ with the following properties: a) $\odot^1\varphi \equiv \odot\varphi$ and b) $\odot^i\varphi \equiv \odot^{i-1} \odot \varphi$; thus, intuitively, defining that φ has taken place i steps in the past.

The idea is then to define $O_x(DO_x\,\alpha > \delta)$ the state in which δ holds to trigger a deadline expressing that α should be done in *r.t.* (the reaction time) steps. The length of this reaction time has to be determined by using the preciseness of the formalisation of the protocol, that is to say, formalising a protocol can introduce many steps between two actions, since many checks need to be made before determining which actions are to be executed.

Definition 18 (Temporal Triggered Obligations)

$$O_x(DO_x\,\alpha > \delta) \quad \equiv_{def} \quad \Box(\delta \rightarrow O_x(DO_x\,\alpha < \odot^{r.t.}\delta))$$

Note that we restricted ourselves here to the deontic operators that we needed in the formalisation of the norms used in the example described in section 4. Of course other kinds of deontic operators can be expressed as well in a similar manner if needed. When formalising other normative domains, one might need to change some of the definitions given above, i.e. simplify or enhance the used definition.

4 Proving Non-Violation Invariance

As mentioned in section 2 we check the protocol on norm compliance by specifying a *safety* property that has to be satisfied by the protocol. This safety property is an invariant, i.e. a formula that should hold during the entire execution of the protocol. The safety property for stating that a protocol should be compliant to the norms is as simple as specifying that non-violation should not change over time (from the start of the protocol on). This can be defined as follows:

Definition 19 (Safety Property of Protocols)

$$\mathrm{start}_\Pi \wedge \Box Norms \quad \rightarrow \quad \Box\neg violation$$

Where $\mathrm{start}_\Pi \equiv$ at α_0 (*the protocol is at its start label*), $Norms$ being the conjunction of all applicable norms, and $violation \equiv \bigvee_{x,\alpha,\delta} viol(x, \alpha, \delta)$ (*violation is the disjunction of all viol-formulas that occur in $Norms$*). This safety property of protocols is defined as the global invariance of $\neg violation$ for the protocol Π under the condition that $Norms$ always holds, i.e., if $\Box Norms$ holds upon the start of running Π, then $\neg violation$ will hold in all states of the run.

To prove that a protocol satisfies this property we will verify that no actions in the protocol change the value of violations, that is to say, no violation will occur during the execution of the protocol on account of an action in the protocol; thus proving that non-violation is an invariance of all actions in the protocol. Combined with the assumption that the protocol is started in a violation-free state we can derive the safety property specified above. We can make this assumption because we are not interested in the situations where this assumption does not hold, such as the situation in which the protocol is started when a violation has already occurred, since starting the protocol in such a situation would say nothing about the norm compliance of the protocol, only that it cannot "repair" the situation it started in. This derivation rule can be defined as:

Theorem 1 (Invariance Rule) *The following rule is valid:*

$$\frac{\begin{array}{l}\mathsf{start}_\Pi \wedge \Box Norms \rightarrow \neg violation \\ \neg violation\ \mathsf{invof}\ \bar{\mathscr{M}}_\Pi\end{array}}{\mathsf{start}_\Pi \wedge \Box Norms \rightarrow \Box\neg violation}$$

Where C invof $\alpha \equiv \alpha \wedge C \rightarrow C$ (*C is an invariant of α*) and C invof $\mathscr{M} \equiv C$ invof $\alpha_1 \wedge \ldots \wedge C$ invof α_m (*C is an invariant of every $\alpha \in \mathscr{M}$*), and $\bar{\mathscr{M}}_\Pi$ is the set of all labels in the program except for the label of stop, the end-statement. This rule is also very close to the intuition one might have about protocols being norm-compliant, namely if there are no steps in the protocol that violate any norm, the protocol will not violate any of the norms as a whole (if no violation existed when the protocol started running).[3]

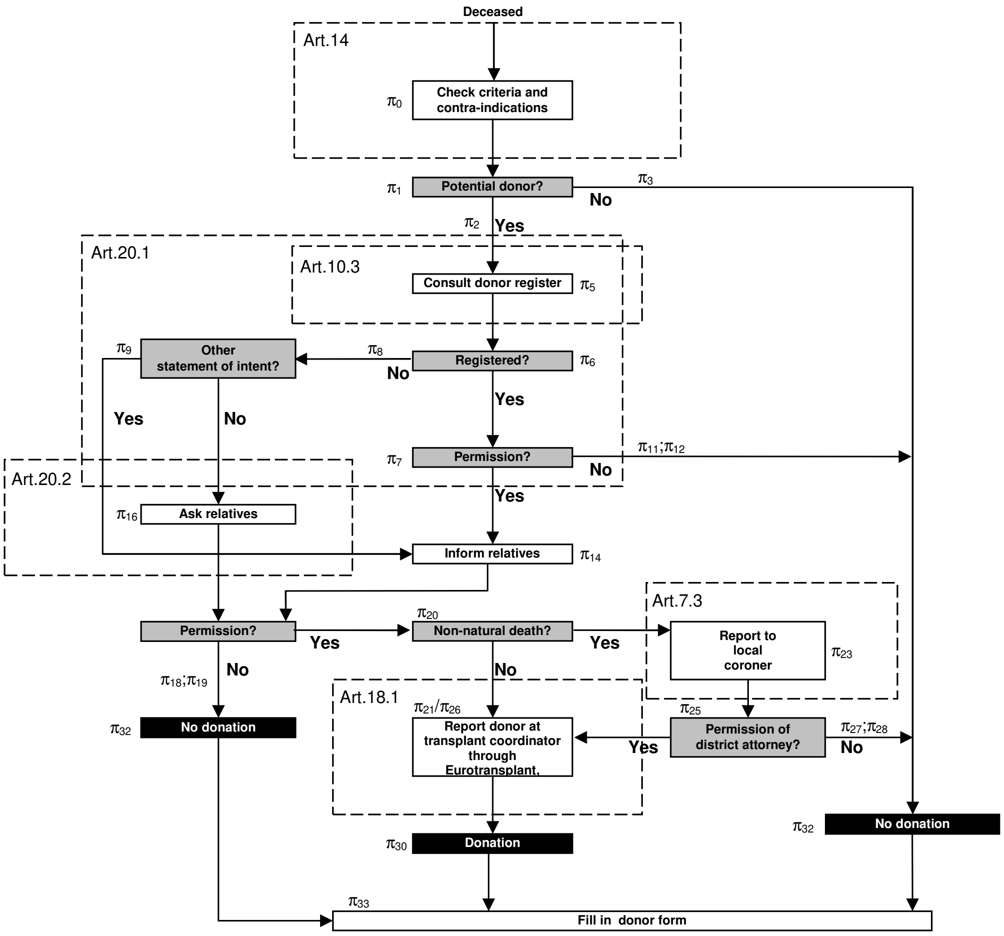

Figure 2: Norms & protocol labels mapped on informal protocol description.

[3]A proof of an invariance rule similar to the one specified in theorem 1 can be found in [15].

4.1 Assumptions

Before we can prove the safety property (and later on the liveness) we need to emphases our assumptions, which we present in this section.

As explained in section 2.3 there have been cases where we had to make representational choices in formalising some of the vague concepts used in the norms. A list of these concepts, introduced as domain knowledge, is the following:

(1) $\mathcal{M} \models necessary\_for(consult(d, register), intended(organ\_removal)) \leftrightarrow$
$\Box(\neg DO_d\, consult(d, register) \rightarrow \neg\Diamond DO_{d'}\ remove\_organ(d', p))$
(2) $\mathcal{M} \models under\_authority(d, consult(d, register))$
(3) $\mathcal{M} \models (\neg registered(p) \wedge \neg other\_statement(z, p)) \rightarrow \neg statement\_of\_intent(w, p)$
(4) $\mathcal{M} \models ((know\_statement\_permission(d, p) \vee know\_other\_statement(d, p) \vee$
$know\_relative\_permission(d, p)) \wedge (\neg non\text{-}natural\_death(p) \vee$
$(non\text{-}natural\_death(p) \wedge know\_DA\_permission(d, p)))) \rightarrow \Box available(organ(p))$
(5) $\mathcal{M} \models known\_non\text{-}natural\_death(d, p) \rightarrow suspicion(d, p, non\text{-}natural\_death)$

The first rule states that "consulting the register is necessary for the intended removal of organs" whenever not consulting the register would lead to not doing the organ removal ever. The second rule expresses that (in our model) d consulting the register always does this under authority of himself. The third item states the a patient not being registered and not having made another statement of intent can be considered as no statement of intent existing for this patient. The fourth item expresses what it means that an organ becomes available. In the context of this work this means that whenever the patient died of natural causes and the doctor knows that he has the permission to remove the organ (either through a registered consent, another statement of intent, or given by the relatives) the organ are available. In the case that the patient did not die of natural causes, but a permission has been obtained from the district attorney, and the doctor knows he has the permission, the organs are available as well. The last item states that if a doctor knows that the patient has died of non-natural causes, then he will also suspect that the patient died of non-natural causes.

The protocol we are examining is used to "change" the knowledge of the doctor about whether or not organs may be extracted. For simplification we made use of propositions to express this knowledge of the doctor. However, some properties of knowledge as expressed in S5 models and epistemic logics in general, see [18] for more details on S5 and epistemic logics, are desired for use in our framework, which we had to add by hand. To this extent we make the following assumption concerning *all* knowledge-proposition, in order to capture the idea that "knowledge is truthful":

$$know\_\varphi \rightarrow \varphi$$

$$know\_not\_\varphi \rightarrow \neg\varphi$$

And to capture that knowing φ means that you do not know $\neg\varphi$:

$$(know\_\varphi \wedge know\_not\_\varphi) \rightarrow \bot$$

Another thing needed before we can derive the norm compliance of the protocol formalised in Appendix A is the meaning of the actions that are taken in the protocol. These actions change the world, that is to say, they change the truth value of proposition from one state to another. In table 1 below, we specified exactly how this is done.

Intuitively this means that an action (α) results in a specified post-condition (ψ_α) if the corresponding pre-condition (φ) held in the model at the time the action was performed. Thus, formally if $\mathcal{M} \models \text{at}\,\pi_i \rightsquigarrow \alpha$ then $\mathcal{M}, s \models (\text{at}\,\pi_i \wedge \varphi) \rightarrow \bigcirc\psi_\alpha$.

Pre-Condition	*Action*	*Post-Condition*
$criteria(p) \wedge$ $contra\text{-}indication(p)$	⟨check_criteria_&_contra-indications⟩	$know\_criteria(d,p) \wedge$ $know\_contra\text{-}indication(d,p)$
$\neg criteria(p) \wedge$ $contra\text{-}indication(p)$	⟨check_criteria_&_contra-indications⟩	$know\_not\_criteria(d,p) \wedge$ $know\_contra\text{-}indication(d,p)$
$criteria(p) \wedge$ $\neg contra\text{-}indication(p)$	⟨check_criteria_&_contra-indications⟩	$know\_criteria(d,p) \wedge$ $know\_not\_contra\text{-}indication(d,p)$
$\neg criteria(p) \wedge$ $\neg contra\text{-}indication(p)$	⟨check_criteria_&_contra-indications⟩	$know\_not\_criteria(d,p) \wedge$ $know\_not\_contra\text{-}indication(d,p)$
$registered(p)$	⟨consult_donor_register⟩	$know\_registered(d,p)$
$\neg registered(p)$	⟨consult_donor_register⟩	$know\_not\_registered(d,p)$
$statement\_permission(p)$	⟨check_permission⟩	$know\_statement\_permission(d,p)$
$\neg statement\_permission(p)$	⟨check_permission⟩	$know\_not\_statement\_permission(d,p)$
$other\_statement(w,p)$	⟨check_other_statement⟩	$know\_other\_statement(d,p)$
$\neg other\_statement(w,p)$	⟨check_other_statement⟩	$know\_not\_other\_statement(d,p)$
$\top$	⟨inform_relatives⟩	$\top$
$\top$	⟨report_to_transplant_coordinator⟩	$\top$
$\top$	⟨report_to_coroner⟩	$\top$
$DA\_permission(DA,p)$	⟨ask_DA_for_permission⟩	$know\_DA\_permission(d,p)$
$\neg DA\_permission(DA,p)$	⟨ask_DA_for_permission⟩	$know\_not\_DA\_permission(d,p)$
$\top$	⟨**fill_donor_form**⟩	$\top$

Table 1: Pre- and Post-conditions of Protocol Actions.

Note that there are actions specified that do not change the world, i.e. pre-condition and post-condition are $\top$. These actions are specified in the real-world protocol, and included in the formalised version, but do not change any knowledge of the doctor that is necessary for the safety or liveness proof of the protocol. The actions themselves, however, might be needed for fulfilling obligations.

4.2 Invariance Proof

Using the formalism described in section 3 we show in this section how the safety property specified earlier is proven for the protocol for obtaining the permission to extract organs for transplantation. This example protocol, introduced in section 1 and shown in figures 1 and 2, as well as the applicable norms, is first formalised (see Appendices A and B). Then, for proving the safety property, i.e. proving that $\neg violation$ is an invariant of the protocol, we make use of the invariance rule as mentioned in theorem 1.

In order to prove that the protocol does not violate any of the norms specified in Appendix B, we need to show that all of the obligations that are stated are actually fulfilled. This means that we need to check whether the obliged action is actually performed at the applicable moment, which is before the deadline has passed:

Theorem 2 (Derivation Rule) *The following rule is valid:*

$$\frac{\begin{array}{c}\mathcal{M} \models O_x(\alpha < \delta) \\ \mathcal{M} \models \text{at}\,\pi_i \rightsquigarrow \alpha \\ \mathcal{M} \models \text{at}\,\pi_i \rightarrow \neg\Diamond\delta\end{array}}{\mathcal{M} \models \text{at}\,\pi_i \rightarrow \bigcirc\neg viol(x,\alpha,\delta)}$$

Intuitively this theorem states that we can derive that the norm $O_x(\alpha < \delta)$ will not be violated by the current action in the protocol (i.e. $\neg viol(x, \alpha, \delta)$ will hold in the next state) if we can show that, at this moment, the obliged action is done and the deadline has not yet passed ($\neg\Diamond\delta$). A proof of this theorem can be found in Appendix C. Note that similar rules could be specified for conditional obligations and (temporal) prohibitions.

Now, let us start the invariance proof by assuming that $\text{start}_\Pi \wedge \Box Norms \rightarrow \neg violation$ holds and try to prove that $\neg violation$ is an invariance of every following step of the protocol, thereby deriving that $\neg violation$ is an invariant of the protocol. Note that we only need to check the actions taken by the protocol, since the "control points" used in the protocol (i.e. protocol labels referring to conditions of **if**-clauses) are trivially norm-compliant since they do not change the value of any *viol*-predicate (actually, the action that is thereafter chosen shows whether the decision made at the control point was correct). This is expressed in step (1).

(1)	$\neg violation$ invof $\bar{\mathcal{M}}_\Pi \backslash \{\pi_0,\pi_5,\pi_7,\pi_9,\pi_{14},\pi_{16},\pi_{21},\pi_{23},\pi_{24},\pi_{26},\pi_{33}\}$

Next we prove that step π_0 of the protocol (checking whether the patient p satisfies the criteria and none of the contra-indications for being a donor) is norm-compliant. The only norm in the law concerning this actions is the fact that doctors are supposed to check whether a patient is brain death before removing any organs (see article 14, included in Appendix B), of which the translation is seen in step (3). Now, since we can derive from the structure of the protocol that $DO_{d'}\ remove\_organ(d', p)$ has not yet occurred, or is occurring now (5), we can derive that the value of V_1 will not be changed by $DO_d\ certify\_death(d, p)$ by applying the theorem introduced above, shown in (6). Finally, after remembering the fact that obligations imply permissions (7) (and therefore do not lead to violations by acting upon the obligation),and adding the fact that no other norms were applicable and thereby cannot be violated (8), we can conclude that $\neg violation$ is an invariant of π_0, see (10).

(2)	at $\pi_0 \rightsquigarrow DO_d\ certify\_death(d,p)$	
(3)	$O_d(DO_d\ certify\_death(d,p) < DO_{d'}\ remove\_organ(d',p))$	Art. 14
(4)	$V_1 = viol(d, DO_d\ certify\_death(d,p), DO_{d'}\ remove\_organ(d',p))$	(Def)
(5)	at $\pi_0 \rightarrow \neg\Diamond DO_d\ remove\_organ(d',p)$	(Π)
(6)	at $\pi_0 \wedge \neg violation \rightarrow \bigcirc\neg V_1$	(DR),(2),(3),(5)
(7)	at $\pi_0 \rightarrow P_d(DO_d\ certify\_death(d,p))$	(3)
(8)	at $\pi_0 \wedge \neg violation \rightarrow \bigcirc\neg viol(d,\alpha,\delta)$ for all viol-predicates other than V_1[4]	(7),(VC)
(9)	at $\pi_0 \wedge \neg violation \rightarrow \bigcirc\neg violation$	(6),(8)
(10)	$\neg violation$ invof π_0	(9)

The next step of the protocol, step π_5, is an instantiation of the $DO_d\ consult(d, register)$ action mentioned in article 10 of the norms (11). The norm, specified in (12) says that only persons authorised by a doctor have the permission to consult the donor register for information on patients, and then only when this is necessary for the removal of organ. Since we introduced the $\sim P_x(DO_x\,\alpha) \rightarrow F_x(DO_x\,\alpha)$ rule in section 2.1, we need to check if the conditions of this permission are met, since, if the conditions do not apply, the above mentioned rule specifies that it is actually forbidden to consult the register. This corresponds to our intuitions of the norm, since people that are not under the authority of a doctor or when it is not necessary for removing organs, one is not permitted to consult the register, actually,

[4] Note that $viol(d, \alpha, \delta)$ is in fact a disjunction of all violation-predicates other than the violation raised by article 14, i.e. $viol(d, DO_d\ certify\_death(d, p), DO_{d'}\ remove\_organ(d', p))$. We will use this formulation instead of the full disjunction for space saving reasons and to make the expression more readable.

one is even forbidden. Next we use the domain knowledge introduced in section 4.1 about what it means to for something to be necessary, shown in (14), i.e. not consulting the register would mean that no organs will ever be removed, which is the case according to the specification to the protocol (the protocol would halt and never terminate, expressed in (15)). Using this we can derive the second condition of article 10, shown in (16). The third condition, i.e. being under the authority of a doctor when consulting the register, is trivially true since, in our case, the doctor is running the protocol and therefore always under the authority of a doctor (17). Having shown that the norm is applicable by combining the derived conditions (16) and (17) we can now derive that article 10 is actually applicable and therefore no violation occurs because of it (19).

(11)	at $\pi_5 \rightsquigarrow DO_d\ consult(d,register)$	
(12)	$P_d((DO_d\ consult(d,register)) \mid$	
	$(under\_authority(d,consult(d,register)) \wedge$	
	$necessary\_for(consult(d,register),intended(organ\_removal)))$	Art. 10.3
(13)	$V_2{=}viol(d,DO_d\ consult(d,register))$	(Def)
(14)	$necessary\_for(consult(d,register),intended(organ\_removal)) \leftrightarrow$	
	$\Box(\neg DO_d\ consult(d,register) \rightarrow \neg\Diamond DO_{d'}\ remove\_organ(d',p))$	(DK)
(15)	$\Box(\neg DO_d\ consult(d,register) \rightarrow \neg\Diamond DO_{d'}\ remove\_organ(d',p))$	(Π)
(16)	$necessary\_for(consult(d,register),intended(organ\_removal))$	(14),(15)
(17)	$under\_authority(d,consult(d,register))$	(DK)
(18)	at $\pi_5 \rightarrow P_d(DO_d\ consult(d,register))$	(12),(16),(17)
(19)	at $\pi_5 \wedge \neg violation \rightarrow \bigcirc\neg V_2$	(11),(18)

This, however, is not the only norm concerning step π_5; there is another law regulating this section of the protocol, namely one that obliges doctors to check for the information in the donor register (see article 20.1 in Appendix B). This norm states that doctors, that have just certified the death of the patient, are supposed to check whether a patient left a statement of intent concerning the removal of organs, which corresponds to the steps of the protocol from π_4 up to π_9 (the actual actions are of course π_5 and π_7 or π_9, but the intermediate steps are required to determine which actions need to be performed), as seen in (20). The norm is specified in (21) and translated by means of the definition 18 into the deadline of (22). Using the fact that step π_0 was considered as doing the $certify\_death$ action (stated in (2)), we can conclude, by looking at the possible transitions of the protocol, that the state of π_4 is always three states after π_0 (either through the transitions $\pi_0;\pi_1;\pi_2;\pi_4$ or $\pi_0;\pi_1;\pi_3;\pi_4$, see the formal protocol in Appendix A for details), shown in (24). Now assuming that the reaction time is at least 4 or 5, as intuition might expect, we can conclude that the deadline has not yet passed, (25), and by applying theorem 2 we obtain (26). By splitting the norm compliance over the individual steps we can conclude that the value of *violation* is not changed in any of these steps ((36) up to (41)). Note that we needed to prove the violation-invariance of doing the consult register action (derived in (11)-(19)) as well as of the statement checking action (derived in (20)-(19)) before we could apply the fact that no other norms are violated as stated in (28) and derive that (31) and (37) hold.

(20)	at $\pi_4;\pi_5;\pi_6;\pi_7;\pi_8;\pi_9 \rightsquigarrow DO_d\ check(d,z)$	
(21)	$O_d(DO_d\ check(d,statement(p)) > \Diamond DONE_d\ certify\_death(d,p))$	Art 20.1
(22)	$\Box(DONE_d\ certify\_death(d,y) \rightarrow O_d(DO_d\ check(d,z) \leq (\odot^{r.t.}\ DONE_d\ certify\_death(d,p))))$	(21)
(23)	$V_3{=}viol(d,DO_d\ check(d,z),DONE_d\ certify\_death(d,p))$	(Def)
(24)	at $\pi_4 \rightarrow \odot^3\ DONE_d\ certify\_death(d,p)$	(2),(Π)
(25)	at $\pi_4 \rightarrow \neg\Diamond\odot^{r.t.}\ DONE_d\ certify\_death(d,p)$	
	For $r.t.$greater or equal to 4	(24)
(26)	at $\pi_4;\pi_5;\pi_6;\pi_7;\pi_8;\pi_9 \wedge \neg violation \rightarrow \bigcirc\neg V_3$	(DR),(21),(20),(25)

(27)	at $\pi_4;\pi_5;\pi_6;\pi_7;\pi_8;\pi_9 \rightarrow P_d(check(d, statement(p)))$	(22),(24)
(28)	at $\pi_5 \wedge \neg violation \rightarrow \bigcirc \neg viol(d,\alpha,\delta)$	
	for all viol-predicates other than V_2 and V_3	(VC)
(29)	at $\pi_4;\pi_6;\pi_7;\pi_8;\pi_9 \wedge \neg violation \rightarrow \bigcirc \neg viol(d,\alpha,\delta)$	
	for all viol-predicates other than V_3	(VC)
(30)	at $\pi_4 \wedge \neg violation \rightarrow \bigcirc \neg violation$	(26),(29)
(31)	at $\pi_5 \wedge \neg violation \rightarrow \bigcirc \neg violation$	(26),(19),(28)
(32)	at $\pi_6 \wedge \neg violation \rightarrow \bigcirc \neg violation$	(26),(29)
(33)	at $\pi_7 \wedge \neg violation \rightarrow \bigcirc \neg violation$	(26),(29)
(34)	at $\pi_8 \wedge \neg violation \rightarrow \bigcirc \neg violation$	(26),(29)
(35)	at $\pi_9 \wedge \neg violation \rightarrow \bigcirc \neg violation$	(26),(29)
(36)	$\neg violation$ invof π_4	(30)
(37)	$\neg violation$ invof π_5	(31)
(38)	$\neg violation$ invof π_6	(32)
(39)	$\neg violation$ invof π_7	(33)
(40)	$\neg violation$ invof π_8	(34)
(41)	$\neg violation$ invof π_9	(35)

Step π_{14} provides us with a strange case, as talking to relatives to inform them what the decisions of the deceased where concerning organ donation does not seem wrong at all, but since we introduced the rule that everything that is not (explicitly) permitted is actually forbidden, this step might actually lead to a violation. There are no norms in the organ donation laws that explicitly, or indirectly, give doctors the permission to inform the relatives. Even so, since an official protocol, which is actually used in the medical domain, includes this action, one would assume that informing the relatives would not violate any of the norms. Therefore, there seems to be something incorrect to just derive that a violation occurs because of this step. Remember, however, that in section 2.1 we explained, when we discussed why we needed the non-permission rule, that this rule is introduced because the general default of the organ donation domain (being included in the domain of cutting in and operating on people) is not to allow anything unless explicitly permitted. Note then that informing relatives is actually not really a part of that domain, and therefore actually not regulated by the laws of that domain. And since the domain of which informing relatives seems part of seems to be more of the 'opposite' default, i.e. anything is allowed if it is not explicitly prohibited, we can safely assume that step π_{14} does not violate any of the concerning norms.

(42)	$\neg violation$ invof π_{14}	*Fact*

The violation-invariance proof of step π_{16} starts in the usual manner; first we introduce the connection between the abstract and concrete levels (43), and then we introduce the norm concerning this step, which, in this case, states that doctors should, after determining that no statement of intent exists, confer with the relatives of the deceased to obtain the permission for extracting the organs (44). Then, after translating the temporal obligation into a deadline by using definition 18 to obtain (46), we start deriving the conditions of theorem 2. As it follows from the protocol that the doctor did check for the statement in the past (namely in steps $\pi_4 - \pi_9$), combined with translating (46) using definition 16 we can obtain (48). Using the fact that the doctor now knows that p is a potential donor, who is not registered and did not leave another statement of intent, and applying the domain knowledge mentioned in section 4.1 that knowledge is truthful, we obtain the desired condition that no statement of intent exists ((49)-(53)). The protocol structure also shows us that the completed *check* action is no more than 3 steps in the past, thus expressing that the deadline of (48) has not yet passed ((54)-(56)). After applying theorem 2 we obtain that V_4 is not violated, (57), and that π_{16} is not affecting the value of *violation*, (61).

(43)	at $\pi_{16} \rightsquigarrow DO_d\ confer\_with(d,x)$	
(44)	$O_d((DO_d\ confer\_with(d,person) > (\Diamond DONE_d\ check(d,z)))\ \vert$	
	$((\neg statement\_of\_intent(w,p) \wedge relative(person) \wedge w{=}z) \wedge \Diamond DONE_d\ certify\_death(d,p)))$	Art 20.2
(45)	$V_4{=}viol(d, DO_d\ confer\_with(d,x))$	(Def)
(46)	$\Box(\Diamond DONE_d\ check(d,z) \rightarrow O_d((DO_d\ confer\_with(d,person) < \odot^{r.t.} \Diamond DONE_d\ check(d,z))\ \vert$	
	$((\neg statement\_of\_intent(w,p) \wedge w{=}z \wedge relative(person)) \wedge \Diamond DONE_d\ certify\_death(d,p))))$	(44)
(47)	at $\pi_{16} \rightarrow \Diamond DONE_d\ check(d,w)$	(Π)
(48)	$\Box(((\neg statement\_of\_intent(w,p) \wedge w{=}z \wedge relative(person)) \wedge \Diamond DONE_d\ certify\_death(d,p)) \rightarrow$	
	$(O_d(DO_d\ confer\_with(d,x) < \odot^{r.t.} \Diamond DONE_d\ check(d,z))$ **until**	
	$(DONE_d\ confer\_with(d,x) \vee \odot^{r.t.} \Diamond DONE_d\ check(d,z)))$	(46),(47)
(49)	at $\pi_{16} \rightarrow know\_potential\_donor(d,p) \wedge know\_not\_registered(d,p) \wedge know\_not\_exist\_other\_statement(d,z,p))$	(Π)
(50)	$know\_potential\_donor(d,p) \wedge know\_not\_registered(d,p) \wedge know\_not\_exist\_other\_statement(d,z,p)) \rightarrow$	
	$(potential\_donor(p) \wedge \neg registered(p) \wedge \neg other\_statement(z,p))$	Prop. of *known*
(51)	$(potential\_donor(p) \wedge \neg registered(p) \wedge \neg other\_statement(z,p)) \rightarrow (\neg registered(p) \wedge \neg other\_statement(z,p))$	$(Tautology)$
(52)	$\neg registered(p) \wedge \neg other\_statement(z,p) \rightarrow \neg statement\_of\_intent(w,p)$	(DK)
(53)	at $\pi_{16} \rightarrow \neg statement\_of\_intent(w,p)$	(49)–(52)
(54)	at $\pi_{10} \rightarrow \Diamond DONE_d\ check(d,z)$	(Π)
(55)	at $\pi_{16} \rightarrow \odot^3 \Diamond DONE_d\ check(d,z)$	(54),(Π)
(56)	at $\pi_{16} \rightarrow \neg \Diamond \odot^{r.t.} \Diamond DONE_d\ check(d,z)$	
	For *r.t.* greater or equal to 4	(55)
(57)	at $\pi_{16} \wedge \neg violation \rightarrow \bigcirc \neg V_4$	(DR),(43),(48),
		(53),(56)
(58)	at $\pi_{16} \rightarrow P_d(confer\_with(d,x))$	(48),(53),(54)
(59)	at $\pi_{16} \wedge \neg violation \rightarrow \bigcirc \neg viol(d,\alpha,\delta)$	
	for all viol-predicates other than V_4	(58),(VC)
(60)	at $\pi_{16} \wedge \neg violation \rightarrow \bigcirc \neg violation$	(57),(59)
(61)	$\neg violation$ invof π_{16}	(60)

At π_{20} the protocol splits two ways, since the doctor has different procedures in case of natural and non-natural deaths. First we will prove that the *announce*-action in π_{21}, which is executed when the patient died of natural causes, is norm-compliant. The law dictates that organs that are supposedly becoming available for transplantation should be announced at an organ centre (*c* in this case), which is formalised in (63). This is supposed to be done by the doctor who certified the death of the patient. Since π_0 was considered as doing $certify\_death$ means this condition of the norm is satisfied. Using the definitions of section 3.3 we obtain the deadline necessary for applying theorem 2, (65). By using the definition of what it means that the organs have become available (introduced in section 4.1), we can show that it follows from the protocol that the deadline of (65) has not yet passed, shown in (71). Again applying theorem 2 we obtain that article 18.1 was not violated in the case the patient died of natural causes, (72).

(62)	at $\pi_{21} \rightsquigarrow DO_d\ announce(d, organ(p), c)$	
(63)	$O_d((DO_d\ announce(d, organ(p), c) > available(organ(p)))\vert\ \Diamond DONE_d\ certify\_death(d,p))$	Art 18.1
(64)	$\Box(available(organ(p)) \rightarrow$	
	$O_d((DO_d\ announce(d, organ(p), c) < \odot^{r.t.} available(organ(p)))\vert\ \Diamond DONE_d\ certify\_death(d,p)))$	(63)
(65)	$\Box(available(organ(p) \rightarrow (\Box(\Diamond DONE_d\ certify\_death(d,p) \rightarrow$	
	$O_d(DO_d\ announce(d,y) < \odot^{r.t.} available(organ(p)))$ **until**	
	$DONE_d\ certify\_death(d,p) \vee \odot^{r.t.} available(organ(p))))))$	(64)

(66)	$V_5{=}viol(d,DO_d\ announce(d,organ(p),c))$	(Def)
(67)	at $\pi_{21}\rightarrow\Diamond DONE_d\ certify\_death(d,p)$	(Π),(2)
(68)	$((know\_statement\_permission(d,p)\vee know\_other\_statement(d,p)\vee know\_relative\_permission(d,p))\wedge$	
	$(\neg non\text{-}natural\_death(p)\vee(non\text{-}natural\_death(p)\wedge know\_DA\_permission(d,p))))\rightarrow$	
	$\Box available(organ(p)))$	(DK)
(69)	at $\pi_{21}\rightarrow((know\_statement\_permission(d,p)\vee know\_other\_statement(d,p)\vee$	
	$know\_relative\_permission(d,p))\wedge\neg non\text{-}natural\_death(p))$	(Π)
(70)	at $\pi_{21}\rightarrow available(organ(p)))$	(68),(69)
(71)	at $\pi_{21}\rightarrow\neg\Diamond\odot^{r.t.} available(organ(p)))$	(Π),(70)
(72)	at $\pi_{21}\wedge\neg violation\rightarrow\bigcirc\neg V_5$	(67),(DR),(62),(63),(71)
(73)	at $\pi_{21}\rightarrow P_d(DO_d\ announce(d,organ(p),c))$	(65),(67),(70)
(74)	at $\pi_{21}\wedge\neg violation\rightarrow\bigcirc\neg viol(d,\alpha.\delta)$	
	For all viol-predicates other than V_5	(73),(VC)
(75)	at $\pi_{21}\wedge\neg violation\rightarrow\bigcirc\neg violation$	(72),(74)
(76)	$\neg violation$ invof π_{21}	(75)

The other 'branch' of the protocol is used when the patient has died of non-natural causes. In such cases organ donation is not necessarily taking place when permission has been granted by the patient himself or by his relatives, because removing the organs may hinder the investigation towards the death of the patient. In such cases the doctor who certified the death of the patient is not allowed to perform the autopsy of the patient. Autopsies are normally performed at the moment of the removal of the organ, and are used to determine the exact causes of death. If, however, the doctor certifying the death of the patient suspects that his patient has not died of natural causes, he is not allowed to perform this autopsy and is obliged to report to the Municipal Autopsist instead (see article 7.3 of the Law on the disposal of the dead included in Appendix B), stated in (78). Following from the structure of the protocol, we can derive, in (81)-(83), that at π_{23} there actually is a suspicion of a non-natural death, therefore activating the obligation for the doctor to report to the autopsist a. Since the doctor *is* reporting to the autopsist (since that is what π_{23} is all about, (77)), we can derive, by application of theorem 2 that no violation will occur because of doing π_{23} at this moment, as stated in (85). Again, remembering that obligations imply permissions, we conclude, since article 7.3 of the Law on the disposal of the dead is the only applicable norm, that π_{23} is non-violation invariant, (89).

(77)	at $\pi_{23}\rightsquigarrow DO_d\ report\_to(d,a)$	
(78)	$O_d(DO_d\ report\_to(d,a){>}suspicion(d,p,non\text{-}natural\_death))$	Art 7.3 LDotD
(79)	$\Box(suspicion(non\text{-}natural\_death)\rightarrow O_d(DO_d\ report\_to(d,a){<}\odot^{r.t} suspicion(d,p,non\text{-}natural\_death)))$	(78)
(80)	$V_6{=}viol(d,DO_d\ report\_to(d,a),suspicion(d,p,non\text{-}natural\_death))$	(Def)
(81)	at $\pi_{23}\rightarrow know\_non\text{-}natural\_death(d,p)$	(Π)
(82)	$know\_non\text{-}natural\_death(d,p)\rightarrow suspicion(d,p,non\text{-}natural\_death)$	(DK)
(83)	at $\pi_{23}\rightarrow suspicion(d,p,non\text{-}natural\_death)$	(81),(82)
(84)	at $\pi_{23}\rightarrow\neg\Diamond\odot^{r.t.} suspicion(d,p,non\text{-}natural\_death)$	(83)
(85)	at $\pi_{23}\wedge\neg violation\rightarrow\bigcirc\neg V_6$	(DR),(78),(77),(83),(84)
(86)	at $\pi_{23}\rightarrow P_d(DO_d\ report\_to(d,a))$	(79),(83)
(87)	at $\pi_{23}\wedge\neg violation\rightarrow\bigcirc\neg viol(d,\alpha,\delta)$	
	For all viol-predicates other than V_6	(86),(VC)
(88)	at $\pi_{23}\wedge\neg violation\rightarrow\bigcirc\neg violation$	(85),(87)
(89)	$\neg violation$ invof π_{23}	(88)

Like earlier in step π_{14} there is no explicit permission or a direct obligation applicable to the next step in the protocol (π_{24}). Similar as before, the action in π_{24} is not really from the domain of organ donations, and, evenmore, is not actually a part of the model protocol as depicted in figures 1 and 2. The protocol in these figures merely states that, in the case of a non-natural death, if the district attorney does not give the permission for organ transplantation the organs may not be extracted, and if the district attorney does give permission, extraction can take place (if all other condition are met, of course). This split in the protocol is justified by article 17 of the law (which is included in section Appendix B), which states that it is prohibited for doctors to extract a patients organs if no permission has been given by the district attorney in the case of a non-natural death. However, from a practical point of view the doctor needs to know what the judgement of the district attorney is, and therefore needs to ask him about it, explaining the explicit formalisation of step π_{24} in the protocol. Since not checking whether the D.A. granted permission for the extraction will ensure that no organs will ever be obtained from this patient, and since it is the organs that the doctor is interested in (his goal is to legally obtain as many organs for transplantation as possible) this step is actually needed (or even obliged) from the Hospitals point of view. Therefore, asking a district attorney for whether this particular permission exists, at this moment, does not seem to violate any of the norms, so we can, again, safely assume that π_{24} in non-violation invariant.

(90)	$\neg \mathit{violation}$ $\mathsf{invof}\ \pi_{24}$	*Assumption*

Earlier we have proven that, in the case of a natural death, the obligation to announce available organs to an organ centre was fulfilled at the right moment, see (62)-(76). However, if the patient has died of non-natural causes, the organs have not yet been announced. This is, as we will show, no problem, since the availability of the organs had not yet been established. Similar to the proof of π_{21} we can derive that the doctor has certified the death of the patient earlier, (96). Now, since the permission obtained from the district attorney determines whether the organs will be available for donation (remember step π_{23}-π_{26} are only performed in case of a non-natural death), we can derive (by using the domain knowledge introduced in section 4.1 which determines what it means that organs are available, stated in (97)) that the organs have become available for transplantation just now, seen in (97)-(99). Thus, the deadline of the norm in (94) has not passed yet in this case either, and we can, similar to the proof of π_{21} derive, by again using theorem 2, that the value of *violation* is not changed by π_{26}.

(91)	$\mathrm{at}\,\pi_{26} \rightsquigarrow DO_d\ \mathit{announce}(d,\mathit{organ}(p),c)$	
(92)	$O_d((DO_d\ \mathit{announce}(d,\mathit{organ}(p),c) > \mathit{available}(\mathit{organ}(p)))\mid\ \diamond DONE_d\ \mathit{certify\_death}(d,p))$	Art 18.1
(93)	$\Box(\mathit{available}(\mathit{organ}(p)) \rightarrow$	
	$O_d((DO_d\ \mathit{announce}(d,\mathit{organ}(p),c) < \odot^{r.t.}\ \mathit{available}(\mathit{organ}(p)))\mid\ \diamond DONE_d\ \mathit{certify\_death}(d,p)))$	(92)
(94)	$\Box(\mathit{available}(\mathit{organ}(p) \rightarrow (\Box(\diamond DONE_d\ \mathit{certify\_death}(d,p) \rightarrow$	
	$O_d(DO_d\ \mathit{announce}(d,p) < \odot^{r.t.}\ \mathit{available}(\mathit{organ}(p)))$ **until**	
	$DONE_d\ \mathit{certify\_death}(d,p) \vee \odot^{r.t.}\ \mathit{available}(\mathit{organ}(p))))))$	(93)
(95)	$V_5 = \mathit{viol}(d, DO_d\ \mathit{announce}(d,\mathit{organ}(p),c))$	(Def)
(96)	$\mathrm{at}\,\pi_{26} \rightarrow \diamond DONE_d\ \mathit{certify\_death}(d,p)$	(Π),(2)
(97)	$((\mathit{know\_statement\_permission}(d,p) \vee \mathit{know\_other\_statement}(d,p) \vee \mathit{know\_relative\_permission}(d,p)) \wedge$	
	$(\neg \mathit{non\text{-}natural\_death}(p) \vee (\mathit{non\text{-}natural\_death}(p) \wedge \mathit{know\_DA\_permission}(d,p)))) \rightarrow$	
	$\Box \mathit{available}(\mathit{organ}(p))$	(DK)
(98)	$\mathrm{at}\,\pi_{26} \rightarrow ((\mathit{know\_statement\_permission}(d,p) \vee \mathit{know\_other\_statement}(d,p) \vee$	
	$\mathit{know\_relative\_permission}(d,p)) \wedge \mathit{non\text{-}natural\_death}(d,p) \wedge \mathit{know\_DA\_permission}(d,p))$	(Π)
(99)	$\mathrm{at}\,\pi_{26} \rightarrow \mathit{available}(\mathit{organ}(p))$	(97),(98)
(100)	$\mathrm{at}\,\pi_{26} \rightarrow \neg \diamond \odot^{r.t.}\ \mathit{available}(\mathit{organ}(p))$	(Π),(99)
(101)	$\mathrm{at}\,\pi_{26} \wedge \neg \mathit{violation} \rightarrow \bigcirc \neg V_7$	(96),(DR),(92),(100),(91)

(102)	at $\pi_{26} \rightarrow P_d(DO_d\, announce(d, organ(p), c))$	(94),(96),(99)
(103)	at $\pi_{26} \wedge \neg violation \rightarrow \bigcirc \neg viol(d, \alpha.\delta)$	
	For all viol-predicates other than V_7	(102),(VC)
(104)	at $\pi_{26} \wedge \neg violation \rightarrow \bigcirc \neg violation$	(101),(103)
(105)	$\neg violation$ invof π_{26}	(104)

The final step of the protocol (π_{33}), filling the donor form, is again an action that is not regulated by the norms on organ donation. There are no norms in the law that oblige doctors to fill the donor form, however, it is clear that this action is required for a good functioning of the organisation. So, although filling donor forms is not explicitly permitted or obliged in the Law on Organ Donations, we can give a explanation similar to the justification of steps π_{14} and π_{24}, thus, π_{33} being an action *outside* the domain regulated by the Law on Organ Donations making the prohibition-as-failure rule ($\sim P\alpha \rightarrow F\alpha$) non-applicable. Moreover, filling donor forms after the process described in the protocol can actually be an organisational norm (which are not considered here), which is done in order to keep some record of steps taken and, if necessary, for proving that nothing was done illegally.

(106)	$\neg violation$ invof π_{33}	*Assumption*

Now, after proving that all actions in the protocol are non-violation invariant (when performed at their specified time), we can derive by applying the invariance rule mentioned in theorem 1 that the protocol is non-violation invariant. This of course means that the protocol is norm-compliant, since no norms where violated by any of the steps from the protocol, or since no violations occurred during a run of the protocol.

(107)	$\neg violation$ invof $\bar{\mathscr{M}}_{\Pi}$	(1),(10),(36),(41),(42), (61),(76),(89),(90), (105),(106)

5 Proving Liveness

As mentioned in section 2, the safety property is not the only property that a protocol needs to be satisfied. Because law is generally applicable to a single context, one who is not participating in the activities of that context is not regulated by these laws; the laws mean nothing to someone not trying to do anything regulated by that particular set of laws. The problem is that laws regulating a specific domain assume that you are trying to do something or otherwise participate in that domain, and only regulates these actions and participations.

While all protocols that satisfy the aforementioned safety property are in fact norm-compliant, it would be nice to show that this protocol is actually goal-oriented as well. To this end, we need to define another property that allows us to determine whether a protocol is, next to being norm-compliant, also trying to achieve something relevant. Norm-compliant protocols that are actually relevant to the domain not only satisfy the violation invariance, but also a liveness property. This sort of properties specifies that a protocol will, at some point, reach a certain (interesting) state. We can use such a property to check whether the protocol achieves a pre-determined goal at the end of its run:

Definition 20 (Liveness Property of Protocols)

$$\text{start}_{\Pi} \wedge \Box Norms \quad \rightarrow \quad \Diamond(\text{at}\ \alpha_e \wedge goal)$$

Where at α_e is the stop-statement of Π and *goal* is the goal that the protocol should reach. In our example this is a complex declarative statement specifying that when the conditions hold (i.e., when the permission for donation is ideally obtained), the agent/doctor running the protocol will know that the donation can take place, and when one of the conditions for the donation fails, the agent/doctor knows that the donation cannot take place.

$$
\begin{array}{rcl}
\text{at}\,\alpha_e & \equiv & \text{at}\,\pi_e \\
\mathit{goal} & \equiv & (\mathit{criteria}(p) \wedge \neg \mathit{contra\text{-}indication}(p) \wedge \\
& & \quad (\mathit{registered}(p) \rightarrow \mathit{statement\_permission}(p)) \wedge \\
& & \quad (\neg \mathit{registered}(p) \rightarrow \mathit{other\_statement}(z, p) \vee \mathit{relative\_permission}(p)) \wedge \\
& & \quad (\neg \mathit{non\text{-}natural\_death}(p) \vee \mathit{DA\_permission}(DA, p)) \\
& & \quad \rightarrow \mathit{know\_permission}(d, \mathit{remove\_organ}(p)) \\
& \bigwedge & \neg(\mathit{criteria}(p) \wedge \neg \mathit{contra\text{-}indication}(p) \wedge \\
& & \quad (\mathit{registered}(p) \rightarrow \mathit{statement\_permission}(p)) \wedge \\
& & \quad (\neg \mathit{registered}(p) \rightarrow \mathit{other\_statement}(z, p) \vee \mathit{relative\_permission}(p)) \wedge \\
& & \quad (\neg \mathit{non\text{-}natural\_death}(p) \vee \mathit{DA\_permission}(DA, p))) \\
& & \quad \rightarrow \mathit{know\_no\_permission}(d, \mathit{remove\_organ}(p))
\end{array}
$$

This *goal* represents that the protocol is supposed to make sure that the agent obtains the knowledge whether it has the permission for the organ transplantation or not, after ending the protocol run. This permission only exists when the conditions specified are met; i.e. when the patient satisfies the criteria and none of the contra-indications for being a donor ($\mathit{criteria}(p) \wedge \neg \mathit{contra\text{-}indication}(p)$), permission for the extraction has been obtained from either the register ($\mathit{statement\_permission}(p)$), another statement of intent ($\mathit{other\_statement}(p)$), or from speaking to the relatives ($\mathit{relative\_permission}(p)$), and, if the cause of death was non-natural ($\mathit{non\text{-}natural\_death}(p)$), permission needs to be obtained from the district attorney as well ($\mathit{DA\_permission}(DA, p)$). When these conditions are met the doctor should know, after completing the protocol, that he has the permission to extract the organs of the patient ($\mathit{know\_permission}(d, \mathit{remove\_organ}(p))$). If, however, one of these conditions has not been met, the doctor should know at the end of the protocol that he does not have the permission to remove the organs ($\mathit{know\_not\_permission}(d, \mathit{remove\_organ}(p))$).

This means that if the protocol is run in a situation where the conditions are met, the result should be, at the end of the protocol, that the doctor knows that he has the permission for the extraction. Likewise, if the protocol is run in a situation where one of the conditions has not been met, the result should be that the doctor knows that he does not have permission. This idea leads us to the following theorem that we will use to derive the liveness of the protocol from Appendix A.

Theorem 3 (Liveness Rule) *The following rule is valid when γ and $\neg\gamma$ are invariants of Π:*

$$
\frac{\begin{array}{c}\text{start}_\Pi \wedge \gamma \rightarrow \Diamond(\text{at}\,\alpha_e \wedge \varphi_1) \\ \text{start}_\Pi \wedge \neg\gamma \rightarrow \Diamond(\text{at}\,\alpha_e \wedge \varphi_2)\end{array}}{\text{start}_\Pi \rightarrow \Diamond(\text{at}\,\alpha_e \wedge (\gamma \rightarrow \varphi_1) \wedge (\neg\gamma \rightarrow \varphi_2))}
$$

Theorem 3 shows us exactly what we discussed above, namely that if starting the protocol (start_Π) in the situation where the conditions are met (γ) makes sure that at the end of the protocol (π_e) the doctor knows that he has the permission ($\mathit{know\_permission}(d, \mathit{remove\_organs}(p))$), and if starting the protocol in the situation where the condition is not met makes sure that at the end of the run the doctor knows that he does not have the permission, we can conclude that the protocol will, in all situations, give the correct output at the end. A proof of this theorem is included in Appendix C.

In order to prove the assumptions of the rule specified in theorem 3, we use the general idea that eventualities like $A \to \Diamond B$ can be broken down into finitely many "smaller steps". The idea would then be to link these smaller steps together into the step required by means of the following rule.

Theorem 4 (Chain Rule) *The following rule is valid:*

$$\frac{\begin{array}{c} A \to \Diamond B \\ B \to \Diamond C \end{array}}{A \to \Diamond C}$$

This rule specifies that expressions of the "smaller steps", e.g. $A \to \Diamond B$ and $B \to \Diamond C$, can be chained together to form the "bigger step", in this case $A \to \Diamond C$. This process can be repeated to obtain the necessary derivations. To derive the "smallest steps" required for the application of this rule, we use the following general rule:

Theorem 5 (Sometime Rule) *The following rule is valid:*

$$\frac{A \to \bigcirc B}{A \to \Diamond B}$$

This rule expresses that if something will happen next, it will happen sometime, which is a general truth following almost directly from the definition of the $\bigcirc$ and $\Diamond$ operators.

5.1 Liveness Proof

As mentioned above we are going to make the proof of the liveness property in two parts, first proving that $\text{start}_\Pi \wedge \gamma \to \Diamond(\text{at}\,\pi_e \wedge \varphi_1)$ holds, with $\varphi_1 \equiv \textit{know\_permission}(d, y)$ and γ being the conjunction of the following three formulas expressing part of the conditions that must:

- $\gamma_1 \equiv \textit{criteria}(p) \wedge \textit{contra-indication}(p)$;
- $\gamma_2 \equiv (\textit{registered}(p) \to \textit{statement\_permission}(p)) \wedge (\neg\textit{registered}(p) \to \textit{other\_statement}(p) \vee \textit{relative\_permission}(p))$;
- $\gamma_3 \equiv \neg\textit{non-natural\_death}(p) \vee \textit{DA\_permission}(DA, p)$.

The first part is concerning the steps to determine whether the patient satisfies the required criteria and none of the contra-indications of being a donor. Assuming that our model satisfies γ, and thereby satisfying γ_1, the result of the action at π_0 would be that the doctor knows that all the criteria are satisfied and that the patient shows none of the contra-indications (2). The protocol then specifies that the doctor knows at this point that the patient is a potential candidate for extracting organs for transplantation, which is specified in the sequence $\pi_1; \pi_2$ in the protocol, and derived in (3)-(4). By applying the Sometime (theorem 5) and Chain rules (theorem 4) on steps (2),(3) and (4) we can derive that, if γ holds, we will eventually reach π_4 while knowing that the patient is a potential donor (5).

(1)	$\text{start}_\Pi \wedge \gamma \to \text{at}\,\pi_0$	(Π)
(2)	$\text{at}\,\pi_0 \wedge \textit{criteria}(p) \wedge \neg\textit{contra-indication}(p) \to \bigcirc(\text{at}\,\pi_1 \wedge \textit{know\_criteria}(d,p) \wedge \textit{know\_no\_contra-indication}(d,p))$	$(PC\ \pi_0)$
(3)	$\text{at}\,\pi_1 \wedge \textit{know\_criteria}(d,p) \wedge \textit{know\_no\_contra-indication}(d,p) \to \bigcirc \text{at}\,\pi_2$	(Π)
(4)	$\text{at}\,\pi_2 \to \bigcirc(\text{at}\,\pi_4 \wedge \textit{know\_potential\_donor}(d,p))$	(Π)
(5)	$\text{start}_\Pi \wedge \gamma \to \Diamond(\text{at}\,\pi_4 \wedge \textit{know\_potential\_donor}(d,p))$	(SR),(CR),(1)–(4)

The next part of the proof is a bit more tricky because of the nature of γ_2. Because γ_2 is a disjunction, it can be true in various different ways, which should all be checked. This means that we need to check whether the protocol gives the right response if:

- The patient is registered and has given his permission;
- The patient is not registered but made another statement giving the permission;
- The patient is not registered, did not make another statement of intent, but the family gives the permission.

First we can derive that we can get to π_5 without much trouble (6), which holds for all instances of γ_2.

(6)	$\text{at}\,\pi_4 \wedge \mathit{know\_potential\_donor}(d,p) \rightarrow \bigcirc \text{at}\,\pi_5$	(Π)

Now, let us start by proving the situation when the patient is registered and has given his permission. From doing the check on the register to see if the patient is registered, the doctor knows, in this situation, that the patient is registered after completing the action in π_5, see (7). Now the doctor is supposed to check the registration to see if the patient has actually given permission for organ extraction, (8)-(9), and after performing this action, the doctor is aware of this fact. This influences certain checks, such that protocol gets from π_{10} to π_{13}, π_{14}, π_{15} and finally π_{20}, which follows directly from the structure of the protocol, (10)-(13).

(7)	$\text{at}\,\pi_5 \wedge \mathit{registered}(p) \rightarrow \bigcirc(\text{at}\,\pi_6 \wedge \mathit{know\_registered}(d,p))$	$(PC\ \pi_5)$
(8)	$\text{at}\,\pi_6 \wedge \mathit{know\_registered}(d,p) \rightarrow \bigcirc \text{at}\,\pi_7$	(Π)
(9)	$\text{at}\,\pi_7 \wedge \mathit{statement\_permission}(d,p) \rightarrow \bigcirc(\text{at}\,\pi_{10} \wedge \mathit{know\_statement\_permission}(d,p))$	$(PC\ \pi_7)$
(10)	$\text{at}\,\pi_{10} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_registered}(d,p) \wedge \mathit{know\_statement\_permission}(d,p) \rightarrow \bigcirc \text{at}\,\pi_{13}$	(Π)
(11)	$\text{at}\,\pi_{13} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_registered}(d,p) \wedge \mathit{know\_statement\_permission}(d,p) \rightarrow \bigcirc \text{at}\,\pi_{14}$	(Π)
(12)	$\text{at}\,\pi_{14} \rightarrow \bigcirc \text{at}\,\pi_{15}$	(Π)
(13)	$\text{at}\,\pi_{15} \wedge \mathit{know\_possible\_donor}(d,p) \wedge \mathit{know\_registered}(d,p) \wedge \mathit{know\_statement\_permission}(d,p) \rightarrow \bigcirc \text{at}\,\pi_{20}$	(Π)

Next, in case the patient is not registered, but made another statement of intent to express that he gives the permission for donating his organs, the action at π_5 will result in the doctor knowing that the patient has not registered (14). The doctor now checks, as the protocol specifies, whether another statement exists, which in this case does exist and therefore results in the doctor knowing this, (15)-(17). Knowing this again leads the protocol from π_{10} to π_{13}, π_{14}, π_{15} and π_{20}, again easily obtainable from the structure of the protocol, (18)-(21).

(14)	$\text{at}\,\pi_5 \wedge \neg \mathit{registered}(p) \rightarrow \bigcirc(\text{at}\,\pi_6 \wedge \mathit{know\_not\_registered}(d,p))$	$(PC\ \pi_5)$
(15)	$\text{at}\,\pi_6 \wedge \mathit{know\_not\_registered}(d,p) \rightarrow \bigcirc \text{at}\,\pi_8$	(Π)
(16)	$\text{at}\,\pi_8 \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,p) \rightarrow \bigcirc \text{at}\,\pi_9$	(Π)
(17)	$\text{at}\,\pi_9 \wedge \mathit{other\_statement}(z,p) \rightarrow \bigcirc(\text{at}\,\pi_{10} \wedge \mathit{know\_other\_statement}(d,p))$	(Π)
(18)	$\text{at}\,\pi_{10} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,p) \rightarrow \bigcirc \text{at}\,\pi_{13}$	(Π)
(19)	$\text{at}\,\pi_{13} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,p) \wedge \mathit{know\_other\_statement}(d,p) \rightarrow \bigcirc \text{at}\,\pi_{14}$	(Π)
(20)	$\text{at}\,\pi_{14} \rightarrow \bigcirc \text{at}\,\pi_{15}$	(Π)
(21)	$\text{at}\,\pi_{15} \wedge \mathit{know\_possible\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,p) \wedge \mathit{know\_other\_statement}(d,p) \rightarrow \bigcirc \text{at}\,\pi_{20}$	(Π)

In the case, however, that the patient did not register, and did not make another statement of intent, the doctor has to speak to the relative to obtain the permission for extracting the organs. Of course the doctor will first check whether the patient is registered, as in (14)-(16), and after checking whether another statement exists, see (22), he has to conclude that no such statement has been made. The protocol then specifies that the doctor is supposed to speak with the relatives for obtaining the permission, see (23)-(25). After talking to the relatives, who give the permission (which follows from the assumption that γ_2 holds), the doctor knows that the permission has been obtained from the relatives, (26). Finally, the protocol also specifies that the doctor will eventually reach π_{20}, while knowing that the permission has been obtained. Since γ_2 follows from γ, which we assumed to be true, and since in all situations listed above the doctor will eventually reach $\pi_2 0$ while knowing that the patient is still a potential donor, we can conclude, again by applying the Sometime and Chain rules, that (28) holds.

(22)	$\text{at}\,\pi_9 \wedge \neg \mathit{other\_statement}(z,p) \rightarrow \bigcirc(\text{at}\,\pi_{10} \wedge \mathit{know\_not\_other\_statement}(d,p))$	(Π)
(23)	$\text{at}\,\pi_{10} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,p) \rightarrow \bigcirc \text{at}\,\pi_{13}$	(Π)
(24)	$\text{at}\,\pi_{13} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,p) \wedge \mathit{know\_not\_other\_statement}(d,p) \rightarrow \bigcirc \text{at}\,\pi_{15}$	(Π)
(25)	$\text{at}\,\pi_{15} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,p) \wedge \mathit{know\_not\_other\_statement}(d,p) \rightarrow \bigcirc \text{at}\,\pi_{16}$	(Π)
(26)	$\text{at}\,\pi_{16} \wedge \mathit{relative\_permission}(x,p) \rightarrow \bigcirc(\text{at}\,\pi_{17} \wedge \mathit{know\_relative\_permission}(d,p))$	$(PC\ \pi_{16})$
(27)	$\text{at}\,\pi_{17} \wedge \mathit{know\_relative\_permission}(d,p) \rightarrow \bigcirc \text{at}\,\pi_{20}$	(Π)
(28)	$\text{start}_\Pi \wedge \gamma \rightarrow \Diamond(\text{at}\,\pi_{20} \wedge \mathit{know\_possible\_donor}(d,p))$	(SR),(CR),(5), (6)−(27)

Now that the doctor has determined whether the patient satisfies the criteria for being a donor, and the doctor knows that he has the permission to extract the organs, he needs, in case of a non-natural death, ask the district attorney for permission to extract the organs. Of course there can be two possibilities; 1) the patient died of natural causes and therefore the doctor can proceed to reporting the organs that are becoming available to the organ centre, see (29)-(30), or 2) the patient did die of non-natural causes and the doctor then reports to the municipal coroner for an autopsy, and asks the district attorney for the permission to continuing with the extraction of the organs, (31)-(34). Since we assumed that γ we know that, if the patient died of non-natural causes, the district attorney will give that permission and the doctor will have to announce the organ to the organ centre, (36). Combining these steps, and applying Sometime and Chain once more, we obtain (37), specifying that, in the case that γ holds, the protocol will eventually reach $\pi_2 9$ while the doctor still knows that the patient is a potential donor.

(29)	$\text{at}\,\pi_{20} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \neg \mathit{non\text{-}natural\_death}(p) \rightarrow \bigcirc \text{at}\,\pi_{21}$	(Π)
(30)	$\text{at}\,\pi_{21} \rightarrow \bigcirc \text{at}\,\pi_{29}$	(Π)
(31)	$\text{at}\,\pi_{20} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{non\text{-}natural\_death}(p) \rightarrow \bigcirc \text{at}\,\pi_{22}$	(Π)
(32)	$\text{at}\,\pi_{22} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{non\text{-}natural\_death}(p) \rightarrow \bigcirc \text{at}\,\pi_{23}$	(Π)
(33)	$\text{at}\,\pi_{23} \rightarrow \bigcirc \text{at}\,\pi_{24}$	(Π)
(34)	$\text{at}\,\pi_{24} \wedge \mathit{DA\_permission}(DA,p) \rightarrow \bigcirc(\text{at}\,\pi_{25} \wedge \mathit{know\_DA\_permission}(d,p))$	$(PC\ \pi_{24})$
(35)	$\text{at}\,\pi_{25} \wedge \mathit{know\_DA\_permission}(d,p) \rightarrow \bigcirc \text{at}\,\pi_{26}$	(Π)
(36)	$\text{at}\,\pi_{26} \rightarrow \bigcirc \text{at}\,\pi_{29}$	(Π)
(37)	$\text{start}_\Pi \wedge \gamma \rightarrow \Diamond(\text{at}\,\pi_{29} \wedge \mathit{know\_potential\_donor}(d,y))$	(SR),(CR),(28),(29)−(36)

In steps π_{29} to $\pi_3 0$, the protocol specifies that the doctor now knows that he has the permission for extracting the organs. Then in step π_{33} the result of the protocol is logged, and the protocol is stopped in π_{34}. This means that we can derive, using previously derived (37) combined with these small steps, that the protocol will eventually reach the end state when γ holds, and that in such a case the doctor will know that he has the permission to extract the organs, (41).

(38)	$\text{at}\,\pi_{29} \wedge \mathit{know\_potential\_donor}(d,p) \rightarrow \bigcirc \text{at}\,\pi_{30}$	(Π)
(39)	$\text{at}\,\pi_{30} \rightarrow \bigcirc(\text{at}\,\pi_{33} \wedge \mathit{know_permission}(d,p))$	$(PC\ \pi_{30})$
(40)	$\text{at}\,\pi_{33} \rightarrow \bigcirc \text{at}\,\pi_{34}$	(Π)
(41)	$\text{start}_\Pi \wedge \gamma \rightarrow \Diamond(\text{at}\,\pi_e \wedge \mathit{know\_permission}(d,p))$	(SR),(CR),(37),(38)−(41)

Having derived the first premise of theorem 3, we will now proceed by deriving the second premise of this theorem. To that extend we assume that $\text{start}_\Pi \wedge \neg\gamma$ holds, with $\neg\gamma$ now equivalent to $\neg\gamma_1 \vee$

$\neg\gamma_2 \vee \neg\gamma_3$, and γ_1, γ_2 and γ_3 as specified above. Of course, $\neg\gamma$ can be true in various different ways, which means that the protocol run can be very different from situation to situation and therefore we need to prove for various different cases that the protocol will eventually reach the end state with the desired result.

We start with proving this eventuality for the case that $\neg\gamma_1$. Note that, in this case, it does not really matter whether γ_2 or $\neg\gamma_2$ holds (and similar for γ_3), since the protocol run is exactly the same for all four combinations (i.e. $\neg\gamma_1 \wedge \gamma_2 \wedge \gamma_3$, $\neg\gamma_1 \wedge \neg\gamma_2 \wedge \gamma_3$, etc). Steps (42)-(46) show what happens in performing π_0 in the case that $\neg\gamma_1$. As mentioned $\neg\gamma_1$ holds if either $\neg criteria(p)$ or $contra\text{-}indication(p)$ holds which means that either the patient does not satisfy the criteria for becoming a donor, or the patient shows contra-indications for being a donor (such as, for instance, being HIV-positive). And, after performing the check in π_0, the doctor knows this, which is derived in (46).

(42)	$\mathrm{start}_\Pi \wedge \neg\gamma \rightarrow \mathrm{at}\,\pi_0$	(Π)
(43)	$\mathrm{at}\,\pi_0 \wedge \neg criteria(p) \wedge \neg contra\text{-}indication(p) \rightarrow \bigcirc(\mathrm{at}\,\pi_1 \wedge know\_not\_criteria(d,p) \wedge know\_no\_contra\text{-}indication(d,p))$	$(PC\ \pi_0)$
(44)	$\mathrm{at}\,\pi_0 \wedge criteria(p) \wedge contra\text{-}indication(p) \rightarrow \bigcirc(\mathrm{at}\,\pi_1 \wedge know\_criteria(d,p) \wedge know\_contra\text{-}indication(d,p))$	$(PC\ \pi_0)$
(45)	$\mathrm{at}\,\pi_0 \wedge \neg criteria(p) \wedge contra\text{-}indication(p) \rightarrow \bigcirc(\mathrm{at}\,\pi_1 \wedge know\_not\_criteria(d,p) \wedge know\_contra\text{-}indication(d,p))$	$(PC\ \pi_0)$
(46)	$\mathrm{start}_\Pi \wedge \neg\gamma_1 \rightarrow \Diamond(\mathrm{at}\,\pi_1 \wedge \neg(know\_criteria(d,p) \wedge know\_no\_contra\text{-}indication(d,p)))$	(SR),(CR), (42)–(45)

With the knowledge that the patient does not satisfy the criteria, or shows one or more of the contra-indications of becoming a suitable donor, the protocol specifies that the doctor will know that the patient is not a possible donor, see (47) and (48). This fact makes the rest of the protocol run fairly simple, as most checks in the protocol, which direct the doctor in doing a certain action, are failed and thereby the actions skipped, see (49)-(58). Therefore, we can easily derive that, after obtaining the knowledge that the patient is not a potential donor, the protocol reaches π_e at which time $know\_no\_permission(d, p)$ will hold.

(47)	$\mathrm{at}\,\pi_1 \wedge \neg(know\_criteria(d,p) \wedge know\_no\_contra\text{-}indication(d,p)) \rightarrow \bigcirc \mathrm{at}\,\pi_3$	(Π)
(48)	$\mathrm{at}\,\pi_3 \rightarrow \bigcirc(\mathrm{at}\,\pi_4 \wedge know\_not\_potential\_donor(d,p))$	$(PC\ \pi_3)$
(49)	$\mathrm{at}\,\pi_4 \wedge know\_not\_potential\_donor(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_6$	(Π)
(50)	$\mathrm{at}\,\pi_6 \wedge know\_not\_potential\_donor(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_8$	(Π)
(51)	$\mathrm{at}\,\pi_8 \wedge know\_not\_potential\_donor(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{10}$	(Π)
(52)	$\mathrm{at}\,\pi_{10} \wedge know\_not\_potential\_donor(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{13}$	(Π)
(53)	$\mathrm{at}\,\pi_{13} \wedge know\_not\_potential\_donor(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{15}$	(Π)
(54)	$\mathrm{at}\,\pi_{15} \wedge know\_not\_potential\_donor(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{20}$	(Π)
(55)	$\mathrm{at}\,\pi_{20} \wedge know\_not\_potential\_donor(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{22}$	(Π)
(56)	$\mathrm{at}\,\pi_{22} \wedge know\_not\_potential\_donor(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{29}$	(Π)
(57)	$\mathrm{at}\,\pi_{29} \wedge know\_not\_potential\_donor(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{31}$	(Π)
(58)	$\mathrm{at}\,\pi_{31} \wedge know\_not\_potential\_donor(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{32}$	(Π)
(59)	$\mathrm{at}\,\pi_{32} \rightarrow \bigcirc(\mathrm{at}\,\pi_{33} \wedge know\_not\_permission(d,p))$	$(PC\ \pi_{32})$
(60)	$\mathrm{at}\,\pi_{33} \rightarrow \bigcirc \mathrm{at}\,\pi_{34}$	(Π)

Proven that the protocol will, in the case that $\neg\gamma_1$ holds, eventually reach π_e, we have only shown a part of the proof for the case that $\neg\gamma$ holds. Since $\neg\gamma$ can hold when $\neg\gamma_1$ does not we need to explore the other possible valuations as well. In the following steps we assume that $\neg\gamma_1$ did not hold, mean γ_1 holds, and will show that, if $\neg\gamma_2$ holds, the protocol will still reach π_e (with the right conclusion concerning the permission for extracting organs). Since γ_1 holds, the first part of this proof is actually already given in (1)-(6), mentioned earlier.

Remember that

$$\begin{aligned}\neg\gamma_2 \;\equiv\;& \neg((registered(p) \rightarrow statement\_permission(p)) \wedge \\ & (\neg registered(p) \rightarrow other\_statement(p) \vee relative\_permission) \\ \equiv\;& registered(p) \wedge \neg statement\_permission(p) \vee \\ & \neg registered(p) \wedge \neg other\_statement(p) \wedge \neg relative\_permission(p)\end{aligned}$$

Like in steps (7)-(28), we spit these into seperate proofs, showing that whenever $registered(p) \wedge \neg statement\_permission(p)$ holds the eventuality can be proven, as well as whenever $\neg registered(p) \wedge \neg other\_statement(p) \wedge \neg relative\_permission(p)$ holds. We start with the former.

The action performed at π_5 (checking whether the patient is registered) is, in this case, successful succesful, and leads to the doctor knowing that the patient is registered, see (61). The protocol now specifies that the doctor needs to check the registration to see if that registration provides him with the permission needed for extracting the organs, (62) and (63). Since, as mentioned above, $registrated(p) \wedge \neg statement\_permission(p)$ holds, the action at π_7 makes sure that the doctor obtains the knowledge that the patient did not want his organs to be used for transplantation, and therefore, the patient is no longer considered a potential donor, (65) and (66). The rest of the run is as derived in (54)-(60). This means, of course, that the protocol will, in the case of a registered patient who denied the organ extraction, eventually reach π_e at which time the doctor knows that he does not have the permission for extracting the organs.

(61)	at $\pi_5 \wedge registered(p) \rightarrow \bigcirc(\text{at } \pi_6 \wedge know\_registered(d,p))$	$(PC\ \pi_5)$
(62)	at $\pi_6 \wedge know\_potential\_donor(d,p) \wedge know\_registered(d,p) \rightarrow \bigcirc \text{at } \pi_7$	(Π)
(63)	at $\pi_7 \wedge \neg statement\_permission(p) \rightarrow \bigcirc(\text{at } \pi_{10} \wedge know\_not\_statement\_permission(d,p))$	$(PC\ \pi_7)$
(64)	at $\pi_{10} \wedge know\_potential\_donor(d,p) \wedge know\_registered(d,p) \wedge know\_not\_statement\_permission(d,p) \rightarrow \bigcirc \text{at } \pi_{11}$	(Π)
(65)	at $\pi_{11} \rightarrow \bigcirc(\text{at } \pi_{12} \wedge \neg know\_potential\_donor(d,p))$	(Π)
(66)	at $\pi_{12} \rightarrow \bigcirc(\text{at } \pi_{15} \wedge know\_not\_potential\_donor(d,p))$	(Π)

In the case that $registered(p) \wedge \neg statement\_permission(p)$ does not hold, it should hold that $\neg registered(p) \wedge \neg other\_statement(p) \wedge \neg relative\_permission(p)$, since we assumed that γ_2 holds. In the following steps we will show that in this case the eventuality will hold as well.

In this case, the action performed at π_5 clearly has the result of the doctor knowing that the patient is not registered, (67). The protocol then specifies that the doctor should check whether another statement of intent exists, (68)-(69). Since this statement does not exists, and the doctor will know this after performing the action in π_9, as shown in (70), the protocol specifies that the doctor should talk to the relatives to see if they provide the permission for the extraction, (71)-(73). Now, since we assumed that the relatives do not give the permission, and the doctor will know this after performing π_{16}, the protocol once again specifies that the patient is not a potential donor. Knowing this, the run continues like before, specified in steps (55)-(60).

(67)	at $\pi_5 \wedge \neg registered(p) \rightarrow \bigcirc(\text{at } \pi_6 \wedge know\_not\_registered(d,p))$	$(PC\ \pi_5)$
(68)	at $\pi_6 \wedge know\_not\_registered(d,p) \rightarrow \bigcirc \text{at } \pi_8$	(Π)
(69)	at $\pi_8 \wedge know\_potential\_donor(d,p) \wedge know\_not\_registered(d,y) \rightarrow \bigcirc \text{at } \pi_9$	(Π)
(70)	at $\pi_9 \wedge \neg other\_statement(p) \rightarrow \bigcirc(\text{at } \pi_{10} \wedge know\_not\_other\_statement(d,p))$	$(PC\ \pi_9)$
(71)	at $\pi_{10} \wedge know\_not\_registered(d,p) \rightarrow \bigcirc \text{at } \pi_{13}$	(Π)
(72)	at $\pi_{13} \wedge know\_not\_registered(d,p) \wedge know\_not\_other\_statement(d,p) \rightarrow \bigcirc \text{at } \pi_{15}$	(Π)

(73)	$\text{at}\,\pi_{15}\wedge know\_not\_registered(d,p)\wedge know\_not\_other\_statement(d,p)\rightarrow\bigcirc\text{at}\,\pi_{16}$	(Π)
(74)	$\text{at}\,\pi_{16}\wedge\neg relative\_permission(d,p)\rightarrow\bigcirc(\text{at}\,\pi_{17}\wedge know\_not\_relative\_permission(d,p))$	$(PC\ \pi_{16})$
(75)	$\text{at}\,\pi_{17}\wedge know\_not\_relative\_permission(d,p)\rightarrow\bigcirc\text{at}\,\pi_{18}$	(Π)
(76)	$\text{at}\,\pi_{18}\rightarrow\bigcirc(\text{at}\,\pi_{19}\wedge\neg know\_potential\_donor(d,p))$	(Π)
(77)	$\text{at}\,\pi_{19}\rightarrow\bigcirc(\text{at}\,\pi_{20}\wedge know\_not\_potential\_donor(d,p))$	(Π)

Having shown that the eventuality holds for the cases where $\neg\gamma_1$ or $\neg\gamma_2$, we only need to show that it also holds if both of $\neg\gamma_1$ and $\neg\gamma_2$ does not hold, i.e. proof the eventuality when $\gamma_1 \wedge \gamma_2$. Since we assumed that $\neg\gamma$, we know that if $\gamma_1 \wedge \gamma_2$ holds, it should hold that $\neg\gamma_3$. In such a case the first steps of the protocol will go as proven above in (1)-(28). Using the fact that $non\text{-}natural\_death(p) \wedge \neg DA\_permission(DA, p)$ combined with the protocol structure, we can easily derive that the run will reach π_{29} sometime, at which time the doctor knows that the patient is not a potential donor, due to the fact that performing π_{24} results in knowing that the district attorney did not give permission to continue with the organ extraction (as was implied by $\neg\gamma_3$). This is shown in the steps (78)-(84). Since the doctor now knows the patient is not a potential donor, the run will end as derived before in (57)-(60).

(78)	$\text{at}\,\pi_{20}\wedge know\_potential\_donor(d,p)\wedge know\_non\_natural\_death(d,p)\rightarrow\bigcirc\text{at}\,\pi_{22}$	(Π)
(79)	$\text{at}\,\pi_{22}\wedge know\_potential\_donor(d,p)\wedge know\_non\_natural\_death(d,p)\rightarrow\bigcirc\text{at}\,\pi_{23}$	(Π)
(80)	$\text{at}\,\pi_{23}\rightarrow\bigcirc\text{at}\,\pi_{24}$	(Π)
(81)	$\text{at}\,\pi_{24}\wedge\neg DA\_permission(DA,p)\rightarrow\bigcirc(\text{at}\,\pi_{25}\wedge know\_not\_DA\_permission(d,p))$	$(PC\ \pi_{24})$
(82)	$\text{at}\,\pi_{25}\wedge know\_not\_DA\_permission\rightarrow\bigcirc\text{at}\,\pi_{27}$	(Π)
(83)	$\text{at}\,\pi_{27}\rightarrow\bigcirc(\text{at}\,\pi_{28}\wedge\neg know\_potential\_donor(d,p))$	(Π)
(84)	$\text{at}\,\pi_{28}\rightarrow\bigcirc(\text{at}\,\pi_{29}\wedge know\_not\_potential\_donor(d,p))$	(Π)

Combining the cases when $\neg\gamma_1$, $\neg\gamma_2$ and $\neg\gamma_3$ to the general case that $\neg\gamma$, shown in (85), (86), (87) and (88), we can apply the liveness rule as specified in theorem 3 to obtain the general liveness property we set out to derive, shown in (89).

(85)	$\text{start}_{\Pi}\wedge\neg\gamma_1\rightarrow\Diamond(\text{at}\,\pi_e\wedge know\_not\_permission(d,remove\_organ))$	$(SR),(CR),(42)-(60)$
(86)	$\text{start}_{\Pi}\wedge\neg\gamma_2\rightarrow\Diamond(\text{at}\,\pi_e\wedge know\_not\_permission(d,remove\_organ))$	$(SR),(CR),(1)-(6),$ $(61)-(66),(67)-(77),$ $(55)-(60)$
(87)	$\text{start}_{\Pi}\wedge\neg\gamma_3\rightarrow\Diamond(\text{at}\,\pi_e\wedge know\_not\_permission(d,remove\_organ))$	$(SR),(CR),(1)-(28),$ $(78)-(84),(57)-(60)$
(88)	$\text{start}_{\Pi}\wedge\neg\gamma\rightarrow\Diamond(\text{at}\,\pi_e\wedge know\_not\_permission(d,remove\_organ))$	$(85),(86),(87)$
(89)	$\text{start}_{\Pi}\rightarrow\Diamond(\text{at}\,\pi_e\wedge(\gamma\rightarrow know\_permission(d,remove\_organ))\wedge$ $(\neg\gamma\rightarrow know\_not\_permission(d,remove\_organ)))$	$(LR),(41),(88)$

6 Related Work

In those situations where agents might deviate from expected behaviour to fulfil their own goals, a multi-agent system needs mechanisms to defend and recommend right and wrong behaviour, along with safe environments to support those mechanisms. As we mentioned in section 1, in eInstitutions expected behaviour is defined by means of *norms*. But providing agents with a set of norms is not enough; an eInstitution should also ensure *norm compliance*.

In literature, there are two approaches for norm compliance from the individual agent perspective:

- agents that always obey norms [3] [23]

- agents that autonomously decide to obey norms [2] [5] [7] [17] [26].

The former ensures norm compliance by default and it is used in those domains where total control of the agent behaviour is needed, but issues on the conflict between the agent goals and the norms should be solved. The latter allows the design of dynamic systems where agents are able to join a society while satisfying their own goals. The conflict between the agents' goals and the norms controlling its behaviour is handled explicitly in the reasoning process of the agent. In [16], *autonomous norm compliance* is divided in two separate processes:

- a process to deliberate about whether to comply with a norm (the *norm deliberation process*),
- a process to update the goals and intentions of agents accordingly (the *norm compliance process*).

In those systems where autonomous norm compliance is allowed from the agent perspective, there is a need to enforce to some extent the compliance of norms from the social perspective. In [16] there is no direct enforcement on norm compliance, but *influenced* norm compliance, where behaviour of other agents against the non-compliance of a norm influences the decision of each agent. In [25] a more direct approach is taken: the agent platform hosting the eInstitution provides time-efficient services to help a special type of agents (the *Police Agents*) to enforce proper behaviour. As Police Agents cannot see or control the internal mental states and the reasoning process of the other agents, norm enforcement is based on the detection of *violation situations* in terms of (public) messages and (visible) actions.

The use of protocols to ease agent interaction (as discussed in section 1) adds an extra level between the norms and the agent behaviour. In this case norm compliance is divided into two different levels:

- *norm compliance of the protocol*: to ensure that a given protocol adheres to the norms defined in a context. If the protocol is norm-compliant, following the protocol ensures that the agent(s) will not violate the norms.
- *protocol compliance by the agent*: to check that the behaviour of an agent complies with the expected behaviour defined by the protocol [27].

The former is the focus of this paper (see section 4), as it is usually overlooked in other works. The latter (protocol compliance) has been studied both in theoretical and practical approaches. In [27] a formal framework for commitment protocols is presented. Verification in this case is an external process and therefore it cannot use the internal knowledge of the agents, only the (observable) behaviour. In [8] protocol compliance is handled by means of interaction scripts that are explicitly accepted by the agents through *interaction contracts*. Each contract includes the interacting agents, the roles they are playing, the contract clauses and the protocol. Verification of protocol compliance is an optional clause in the contract that, if included, specifies who and how will verify the interaction and the actions to take if the protocol is not followed. In [10] interaction protocols are structured in a *performative structure* [21]. Although agents can decide not not follow the protocol (there is no direct control of the agent platform over the agents' beliefs and desires), there is an intermediate actor, the *governor*, that filters any non-allowed message that the agent tries to send to the eInstitution and is not allowed. Therefore protocol compliance is ensured by filtering those messages that, for a given state of the interaction, are not included in the protocol as possible messages to be uttered. However, in none of these works there is a method to ensure that the protocols are norm-compliant.

7 Conclusions & Future Work

In this paper we discuss a formal approach on norm compliance of protocols based on the verification of programs. We give a view of how these techniques can be used, after some adaptation and extension, to verify that a (knowledge-based) protocol is norm-compliant. We also show, as an example, how norm compliance of a knowledge-based protocol (actually used in the medical domain) can be proven.

Currently our formal method is suited for verification of single sequential protocols. We plan to extend our $\mathscr{L}_{TP_\Pi}$ language to prove norm compliance of parallel protocols (such as interaction protocols). We also plan to extend the $\mathscr{L}_{TP}$ language with operators from epistemic logic in order to improve expressiveness of knowledge and beliefs of agents following a protocol. Moreover, we are very interested in seeing how this extended approach can, for instance, be used for the checking of security and authentication protocols.

The framework discussed in this paper uses a theorem proving method to verify the norm compliance of protocols. This is known to be labour-intensive. We are currently considering the use of model-checking, instead.

References

[1] A.R. Anderson. A reduction of deontic logic to alethic modal logic. *Mind*, 67:100–103, 1958.

[2] G. Boella and L. Lesmo. Deliberative normative agents. In C. Dellarocas and R. Conte, editors, *Workshop on Norms and Institutions in Multi-Agent Systems*, pages 15–25. ACM-AAAI, ACM Press, 2000.

[3] M. Boman. Norms in artificial decission making. *Artificial Intelligence and Law*, 7(1):17–35, 1999.

[4] J. Broersen, F. Dignum, V. Dignum, and J.-J. Ch. Meyer. Designing a Deontic Logic of Deadlines. In *7th Int. Workshop on Deontic Logic in Computer Science (DEON'04)*, Portugal, May 2004.

[5] C. Castelfranchi, F. Dignum, C. Jonker, and J. Treur. Deliberative Normative Agents: Principles and architecture. In *Proc. of the 6th Int. Workshop on Agent Theories, Architectures, and Languages (ATAL-99).*

[6] F. Dignum, J. Broersen, V. Dignum, and J.-J. Ch. Meyer. Meeting the Deadline: Why, When and How. In *3rd Goddard Workshop on Formal Approaches to Agent-Based Systems (FAABS)*, Maryland, April 2004.

[7] F. Dignum, D. Morley, and E.A. Sonenberg. Towards socially sophisticated BDI agents. In *DEXA Workshop*, pages 1134–1140, 2000.

[8] V. Dignum. *A Model for Organizational Interaction: based on Agents, founded in Logic*. SIKS Dissertation Series 2004-1. SIKS, 2004. PhD Thesis.

[9] M. Esteva, J. Padget, and C. Sierra. Formalizing a language for institutions and norms. In J.-J.Ch. Meyer and M. Tambe, editors, *Intelligent Agents VIII*, volume 2333 of *LNAI*, pages 348–366. Springer Verlag, 2001.

[10] M. Esteva, J.A. Rodríguez-Aguilar, B. Rosell, and J.L. Arcos. AMELI: An Agent-based Middleware for Electronic Institutions. In *Third International Joint Conference on Autonomous Agents and Multi-agent Systems*, New York, US, July 2004.

[11] D. Grossi, H. Aldewereld, J. Vázquez-Salceda, and F. Dignum. Ontological aspects of the implementation of norms in agent-based electronic institutions. Accepted for the 1st International Symposium on Normative Multiagent Systems (NorMAS2005), 2005.

[12] D. Grossi and F. Dignum. From abstract to concrete norms in agent institutions. In *Proceedings of FAABS III Workshop, Washington, april*, 2004.

[13] D. Grossi, F. Dignum, and J-J. Ch. Meyer. Contextual taxonomies. In J. Leite and P. Toroni, editors, *Proceedings of CLIMA V Workshop, Lisbon, September*, pages 2–17, 2004.

[14] H.L.A. Hart. *The Concept of Law*. Clarendon Press, Oxford, 1961.

[15] Fred Kröger. *Temporal Logic of Programs*, volume 8 of *EACTS monographs on theoretical computer science*. Springer-Verlag, 1987.

[16] F. López y Lopez. *Social Power and NormsL Impact on Agent Behaviour*. PhD thesis, Faculty of Engineering and Applied Science, Univ. of Southampton, 1997.

[17] F. López y Lopez, M. Luck, and M. d'Inverno. A framework for norm-based inter-agent dependence. In *Proceedings of The Third Mexican International Conference on Computer Science*, pages 31–40. SMCC-INEGI, 2001.

[18] J.-J. Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press, New York, USA, 1995.

[19] J.-J. Ch. Meyer and R.J. Wieringa. Deontic logic: A concise overview. In *Deontic Logic in Computer Science: Normative System Specification*, pages 3–16. John Wiley & Sons Ltd., Chichester, UK, 1994.

[20] Henry Prakken and Giovanni Sartor. Argument-based logic programming with defeasible priotities. *Journal of Applied Non-classical Logics*, 7:25–75, 1997.

[21] J.A. Rodriguez. *On the Design and Construction of Agent-mediated Electronic Institutions*. PhD thesis, Inst. d'Investigació en Intel.ligència Artificial, 2001.

[22] M.J. Sergot, F. Sadri, R.A. Kowalski, and F. Kriwaczek. The british nationality act as a logic program. *Communications of the ACM*, 29(5):370–386, May 1986.

[23] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73(1-2):231–252, 1995.

[24] M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM (JACM)*, 23(4):733–742, October 1976.

[25] J. Vázquez-Salceda, H. Aldewereld, and F. Dignum. Implementing norms in multiagent systems. In G. Lindemann, J. Denzinger, I.J. Timm, and R. Unland, editors, *Multiagent System Technologies*, LNAI 3187, pages 313–327. Springer-Verlag, 2004.

[26] J. Vázquez-Salceda and F. Dignum. Modelling electronic organizations. In V. Marik, J. Muller, and M. Pechoucek, editors, *Multi-Agent Systems and Applications III*, LNAI 2691, pages 584–593. Springer-Verlag, 2003.

[27] Mahadevan Venkatraman and Munindar P. Singh. Verifying compliance with commitment protocols. *Autonomous Agents and Multi-Agent Systems*, 2(3):217–236, 1999.

Appendix A Formalised Protocol

The protocol from figure 1 in section 1 can be formalised into the following, using the language mentioned in section 3.2:

Initial $know\_permission(d, remove\_organ) := \texttt{FALSE}$, $know\_potential\_donor(d, y) := \texttt{FALSE}$, $know\_not\_permission(d, remove\_organ) := \texttt{FALSE}$, $know\_not\_potential\_donor(d, y) := \texttt{FALSE}$;
cobegin Π

Π =
π_0 : ⟨check_criteria_&_contra-indications⟩;
π_1 : **if** $know\_criteria(d, y) \wedge know\_no\_contra\text{-}indication(d, y)$
then π_2 : $know\_potential\_donor(d, y) := \texttt{TRUE}$
else π_3 : $know\_not\_potential\_donor(d, y) := \texttt{TRUE}$
fi;
π_4 : **if** $know\_potential\_donor(d, y)$
then π_5 : ⟨consult_donor_register⟩
fi;
π_6 : **if** $know\_potential\_donor(d, y) \wedge know\_registered(d, y)$
then π_7 : ⟨check_permission⟩
else π_8 : **if** $know\_potential\_donor(d, y) \wedge know\_not\_registered(d, y)$
then π_9 : ⟨check_other_statement⟩
fi
fi;
π_{10} : **if** $know\_potential\_donor(d, y) \wedge know\_registered(d, y) \wedge know\_not\_statement\_permission(d, y)$
then π_{11} : $know\_potential\_donor(d, y) := \texttt{FALSE}$;
π_{12} : $know\_not\_potential\_donor(d, y) := \texttt{TRUE}$
else π_{13} : **if** $know\_potential\_donor(d, y) \wedge$
$(know\_registered(d, y) \wedge know\_statement\_permission(d, y)) \vee$
$(know\_not\_registered(d, y) \wedge know\_other\_statement(d, y))$
then π_{14} ⟨inform_relatives⟩
fi
fi;
π_{15} : **if** $(know\_potential\_donor(d, y) \wedge (know\_not\_registered(d, y) \wedge know\_not\_other\_statement(d, y)) \vee$
$(know\_registered(d, y) \wedge know\_includes\_escape\_clause(z)))$
then π_{16} : ⟨ask_relatives⟩;
π_{17} : **if** $know\_not\_relative\_permission(d, y)$
then π_{18} : $know\_potential\_donor(d, y) := \texttt{FALSE}$;
π_{19} : $know\_not\_potential\_donor(d, y) := \texttt{TRUE}$
fi
fi;
π_{20} : **if** $know\_potential\_donor(d, y) \wedge know\_natural\_death(d, y)$
then π_{21} : ⟨report_to_transplant_coordinator⟩
else π_{22} : **if** $know\_potential\_donor(d, y) \wedge know\_non\_natural\_death(d, y))$
then π_{23} : ⟨report_to_coroner⟩;
π_{24} : ⟨ask_DA_for_permission⟩;
π_{25} : **if** $know\_DA\_permission(d, y)$

then π_{26} : ⟨report_to_transplant_coordinator⟩
else π_{27} : $know\_potential\_donor(d, y)$:= FALSE;
π_{28} : $know\_not\_potential\_donor(d, y)$:= TRUE
fi
fi
fi;
π_{29} : **if** $know\_potential\_donor(d, y)$
then π_{30} : $know\_permission(d, remove\_organ)$:= TRUE
else π_{31} : **if** $know\_not\_potential\_donor(d, y)$
then π_{32} : $know\_not\_permission(d, remove\_organ)$:= TRUE
fi
fi;
π_{33} : ⟨fill_donor_form⟩;
π_{34} : **stop**

Appendix B Applicable Norms

Using a typed-predicate logic, where d and d', are doctors; x is a person or relative; y is a patient; u is a district attorney; c is an organ centre; and w and z are statements we can translate the following norms into the logic using a universal quantification over these elements:

Article 10

3. The register can be consulted by or under the authority of a doctor during day and night whenever that is necessary considering the intended removal of an organ.

$$P_x((DO\ consult(x, register)) \mid$$

$$(\exists d\ under\_authority(d, consult(x, register)) \wedge$$

$$necessary\_for(consult(x, register), intended(organ\_removal))))$$

Article 14

1. Before an organ is removed, death is certified by a doctor who may not be involved in the removal or transplantation of the organ. If the intention exists to remove the organ from an insufflated body, death is certified by means of the latest state of medically valid methods and criteria for the determination of brain death by a very knowledgeable doctor. The manner of determining the brain death is recorded in a statement of which a model is included the in article 15, first item, mentioned protocol.

$$O_d((DO\ certify\_death(d, y)) \ < \ (DO\ remove\_organ(d', y)))$$

$$\wedge$$

$$F_d((DO\ remove\_organ(d, y)) \mid (DONE\ certify\_death(d, y)))$$

Article 16

At the removal of an organ from a body, the autopsy as meant by article 3 of the Law on disposal of the dead (Stb. 1991, 133) is not carried out by the doctor who is involved in the removal or implantation of the organ.

$$F_d((DO\ autopsy(d,y)) \mid (DONE\ remove\_organ(d,y)))$$

Article 17

An organ may not be removed when suspicion of a non-natural death exists, until it has been established that the district attorney grants the permission as mentioned in article 76, second item, of the Law on disposal of the dead.

$$F_d((DO\ remove\_organ(d,y)) \ < \ (DONE\ grants(u, permission, organ\_removal)) \mid$$
$$(suspicion(non\_natural\_death, y)))$$

Article 18

1. The doctor who determines the death makes sure that organs supposedly coming available for implantation are announced to an organ centre.

$$O_d((DO_d\ announce(d, organ(p), c)) > available(organ(y)) \mid$$
$$(DONE_d\ certify\_death(d,y)))$$

Article 20

1. The one who determines the death takes care of checking whether a statement of intent as mentioned by article 9 and article 10 is present. If a statement of intent emerging from a register correspond to another existing statement as meant by article 9 the last made statement is applicable.

$$O_d((DO\ check(d,z)) \ > \ (DONE\ certify\_death(d,y)))$$

2. If no statement of intent mentioned by article 9 and article 10 is present the doctor mentioned in the first item takes, in accordance with the protocol, care of consulting the person or persons mentioned by article 11.

$$O_d((DO_d\ confer\_with(d,x) > (\Diamond DONE_d\ check(d,z))) \mid$$
$$(((\neg\exists w\ statement\_of\_intent(w,y) \wedge relative(x) \wedge w = z) \wedge \Diamond DONE_d\ certify\_death(d,y))))$$

Appendix C Proof of Theorems

In sections 4 and 5 several theorems were used to derive the safety and liveness properties of the protocol. Some of these theorems, like theorem 1, theorem 4 and 5 were derived from [15], and proofs of these theorems can be found there. The other theorems are proved in this section.

Derivation Rule

The following theorem was mentioned in section 4 to derive the invariance of a violation predicate:

Theorem 6 (Derivation Rule) *The following rule is valid:*

$$\frac{\begin{array}{c}\mathcal{M} \models O_x(\alpha < \delta)\\ \mathcal{M} \models \text{at}\,\pi_i \rightsquigarrow \alpha\\ \mathcal{M} \models \text{at}\,\pi_i \rightarrow \neg\Diamond\delta\end{array}}{\mathcal{M} \models \text{at}\,\pi_i \rightarrow \bigcirc\neg viol(x, \alpha, \delta)}$$

Proof. We show semantically that theorem 6 holds. First, remember that $O(\alpha < \delta)$ stated in means the following (see definition 14):

$$\begin{array}{rcl}\mathcal{M}, s \models O_x(\alpha < \delta) & \Leftrightarrow & \exists t \geq s : \mathcal{M}, t \models \delta \text{ and}\\ & & (\forall s \leq u < t : \mathcal{M}, u \models \neg V) \text{ and}\\ & & ((\exists s \leq u < t : \mathcal{M}, u \models \alpha \text{ and } \mathcal{M}, u \models \Box\neg V) \text{ or}\\ & & (\forall s \leq u < t : \mathcal{M}, u \not\models \alpha \text{ and } \mathcal{M}, t \models V))\end{array}$$

Now, knowing that $\mathcal{M}, s \models O(\alpha < \delta)$, for s being the state where start_Π holds (remember, we are only interested in those situations where the norms hold and the protocol is started), we also know that there will be a state t such that δ holds. Moreover, all states from s up to t will satisfy $\neg V$, and either there exists a state between s and t (s included) where α holds, in which case $\Box\neg V$ will holds as well, *or* α does not hold in any state between s and t (again, s included), in which case V holds in t. Suppose we have that, for some state $i \geq s$: $\mathcal{M}, i \models \text{at}\,\pi_i$. Now we have to prove that $\mathcal{M}, i \models \bigcirc\neg V$. Since we know that $\text{at}\,\pi_i \rightarrow \neg\Diamond\delta$ is true in all states of the computation we know that $\mathcal{M}, i \models \neg\Diamond\delta$, meaning that the moment that $\text{at}\,\pi_i$ holds the deadline has not yet occurred, we can conclude that $s \leq i < t$. Moreover, since $\text{at}\,\pi_i \rightsquigarrow \alpha$ holds, i.e. $\text{at}\,\pi_i$ counts as α, we know that α happens *before* the deadline. This means that $\exists s \leq i < t : \mathcal{M}, i \models \alpha$ holds, and according to the semantical definition given above we can conclude that $\mathcal{M}, i \models \Box\neg V$ holds as well. Therefore, we can conclude that $\text{at}\,\pi_i \rightarrow \Box\neg V$ and, in particular, $\text{at}\,\pi_i \rightarrow \bigcirc\neg V$ hold.

Liveness Rule

The following theorem was mentioned in section 5 to derive the liveness property of a protocol:

Theorem 7 (Liveness Rule) *The following rule is valid when γ and $\neg\gamma$ are invariants of Π:*

$$\frac{\begin{array}{cr}\text{start}_\Pi \wedge \gamma \rightarrow \Diamond(\text{at}\,\pi_e \wedge \varphi_1) & (1)\\ \text{start}_\Pi \wedge \neg\gamma \rightarrow \Diamond(\text{at}\,\pi_e \wedge \varphi_2) & (2)\end{array}}{\text{start}_\Pi \rightarrow \Diamond(\text{at}\,\pi_e \wedge (\gamma \rightarrow \varphi_1) \wedge (\neg\gamma \rightarrow \varphi_2)) \quad (3)}$$

Proof. We will only consider the cases where start_Π actually holds, since those are the only cases we are interested in and the rule is trivially true those cases where start_Π does not hold.

Let us assume we can derive (1) and (2), and let us assume that γ actually holds. In this case $\mathrm{start}_\Pi \wedge \gamma$ holds as well and we know, because of (1), that $\Diamond(\mathrm{at}\,\pi_e \wedge \varphi_1)$ holds. Since γ is an invariant of Π, γ holds at all steps of the protocol, and thus at the moment that at π_e holds. Therefore, $\Diamond(\mathrm{at}\,\pi_e \wedge \gamma \wedge \varphi_1))$ holds, and we can thus derive that $\Diamond(\mathrm{at}\,\pi_e \wedge (\gamma \rightarrow \varphi_1))$. Since γ holds (and is an invariant of Π), we know that all states satisfies $\neg\gamma \rightarrow \chi$, for arbitrary χ. If we choose $\chi = \varphi$ we thus obtain the desired $\Diamond\pi_e \wedge (\gamma \rightarrow \varphi_1) \wedge (\neg\gamma \rightarrow \varphi_2))$.

Similarly reasoning for $\neg\gamma$ and using (2) instead, we again obtain $\Diamond\pi_e \wedge (\gamma \rightarrow \varphi_1) \wedge (\neg\gamma \rightarrow \varphi_2))$. We can therefore conclude that, whenever start_Π holds, $\Diamond\pi_e \wedge (\gamma \rightarrow \varphi_1) \wedge (\neg\gamma \rightarrow \varphi_2))$ holds, and thus $start_\Pi \rightarrow \Diamond(at\,\alpha_e \wedge (\gamma \rightarrow \varphi_1) \wedge (\neg\gamma \rightarrow \varphi_2))$.