SQRL Report 39
Torwards a Deontic Logic for Fault Tolerance

Pablo F.Castro
castropf@mcmaster.ca
Department of Computing & Software
McMaster University
Hamilton, Canada

T.S.E.Maibaum
tom@maibaum.org
Department of Computing & Software
McMaster University
Hamilton, Canada

Acknowledgements

We wish to thank Javier Blanco for his useful comments about the philosophers example. We also want to express our gratitude to Marcelo Frias and Nazareno Aguirre for their useful critics.

1 Introduction

In recent years *deontic logics* (see [12] for an historical introduction) have generated great interest among computer scientists. This is mainly because this variant of standard logic allows us to deal with the notions of *obligation*, *permission* and *violation*, which arise naturally in, almost, all the fields of computing.

In this document we define a new *deontic logic*, with the purpose in mind of using it for specifying (and reasoning about) *fault-tolerant* programs or systems. Because of the nature of our field of research (fault tolerant systems), we need certain characteristics in our logic. Particularly, we need to be able to express temporal assertions, recovery actions, permission and obligation predicates on actions. The logic defined in the following sections has been influenced by various past ideas; for example, the obligation operator (and its properties) are similar to those defined in [15] and [16]. On the other hand, the temporal extension of the logic takes some concepts from [15], in particular the given semantics using *traces*. Finally, the weak permission operator (see the next section) is exactly that defined in [10]; we shall compare our work with this framework several times in what follows. We can remark that the new elements introduced here are several; some novel properties about the deontic operators will be given (e.g., axiom A13 below), and the definition of obligation given in section 2 is slightly different that those in the literature.

Finally, the soundness and completeness of a basic propositional version of the logic defined will be proved; this will give us some confidence about the definitions introduced. In the following definitions we follow the notation (and some concepts) introduced in [15].

2 A Propositional Deontic Logic

As usual, we start by defining a propositional version of deontic logic (we will call it PDL) by introducing its syntax and semantics. PDL is a *modal action logic*, which uses boolean operators for combining actions terms. Here, we follow the approach proposed by S.Kent ([14]) in the sense that actions are interpreted as a set of events.

After defining the key components of PDL, we present an axiomatic system which has some similarities to those given for Dynamic Propositional Logic ([10]) and Modal Boolean Logic ([4]). Finally, we prove the soundness and completeness of the resulting system.

Definition 1 (Language)**.** *A language (or vocabulary) for PDL is a a tuple:* $\langle \Phi_0, \Delta_0 \rangle$*, where:*

- Φ_0 *is a (finite) set of atomic propositional letters, we will denote them:* $p_1, ..., p_n$*.*
- Δ_0 *is a (finite) set of atomic actions, denoted by:* $\alpha_1, ..., \alpha_m$*.*

Using the sets Φ_0 and Δ_0, we can define the set of formulas and action terms of a given language.

Definition 2 (action terms)**.** *Given a vocabulary* $\langle \Phi_0, \Delta_0 \rangle$*, we define the set of action terms (called* Δ*) as follows:*

- $\Delta_0 \subseteq \Delta$.
- $\emptyset, \mathbf{U} \in \Delta$.
- *if* $\alpha, \beta \in \Delta$*, then* $\alpha \sqcup \beta \in \Delta$ *and* $\alpha \sqcap \beta \in \Delta$.
- *if* $\alpha \in \Delta$*, then* $\overline{\alpha} \in \Delta$.
- *No other expression belongs to* Δ.

■

We use the greek letters: $\alpha, \beta, \gamma, ...$ for variables over Δ. In a similar way we define the set of well-formed formulas.

Definition 3 (Formulas)**.** *Given a vocabulary:* $\langle \Phi_0, \Delta_0 \rangle$ *we define the set of well-formed formulas (*Φ*) as follows:*

- $\Phi_0 \subseteq \Phi$.
- $\top, \bot \in \Phi$.
- *if* $\alpha, \beta \in \Delta$ *then* $\alpha =_{act} \beta \in \Phi$
- *if* $\varphi_1, \varphi_2 \in \Phi$ *then* $\varphi_1 \rightarrow \varphi_2 \in \Phi$.
- *if* $\varphi \in \Phi$ *then* $\neg\varphi \in \Phi$.
- *if* $\varphi \in \Phi$ *and* $\alpha \in \Delta$ *then* $\langle\alpha\rangle\phi \in \Phi$.
- *if* $\alpha \in \Delta$ *then* $\mathsf{P}(\alpha) \in \Phi$ *and* $\mathsf{P}_{\mathsf{w}}(\alpha) \in \Phi$.
- *No other expression belongs to* Φ.

■

As usual, we can define some derived operators:

- $\phi \vee \psi \overset{\mathsf{def}}{\Longleftrightarrow} (\neg\phi) \rightarrow \psi$.
- $\phi \wedge \psi \overset{\mathsf{def}}{\Longleftrightarrow} \neg(\neg\phi \vee \neg\psi)$.

- $[\alpha]\phi \overset{\mathsf{def}}{\Longleftrightarrow} \neg\langle\alpha\rangle\neg\phi$.

We call $\mathsf{P}(-)$ permission or *strong permission*, whereas $\mathsf{P}_{\mathsf{w}}(-)$ is *weak permission*; the differences between the two will become evident with their semantics definitions.

Note that we have not yet defined the obligation operator $\mathsf{O}(-)$; here we do not take the usual definition: $\mathsf{O}(\alpha) \equiv \neg\mathsf{P}(\overline{\alpha})$, instead we define:

$$\mathsf{O}(\alpha) \overset{\mathsf{def}}{\Longleftrightarrow} \mathsf{P}(\alpha) \wedge \neg\mathsf{P}_{\mathsf{w}}(\overline{\alpha})$$

We will explain this definition later. Before this, we need to introduce the concept of semantic structures.

Definition 4 (models)**.** *Given a language $L = \langle\Phi_0, \Delta_0\rangle$, an L-Structure is a tuple: $M = \langle\mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{I}, \mathcal{P}\rangle$ where:*

- *$\mathcal{W}$, is a set of worlds.*
- *$\mathcal{R}$, is a $\mathcal{E}$-labeled relation between worlds. We require that, if $(w, w', e) \in \mathcal{R}$ and $(w, w'', e) \in \mathcal{R}$, then $w' = w''$, i.e., $\mathcal{R}$ is functional.*
- *$\mathcal{E}$, is a not-empty set of (names of) events.*
- *$\mathcal{I}$, is a function:*
  - *For every $p \in \Phi_0 : \mathcal{I}(p) \subseteq \mathcal{W}$*
  - *For every $\alpha \in \Delta_0 : \mathcal{I}(\alpha) \subseteq \mathcal{E}$.*

  In addition, the interpretation $\mathcal{I}$ has to satisfy the following properties:

  I.1 *For every $\alpha_i \in \Delta_0$: $|\mathcal{I}(\alpha_i) - \bigcup\{\mathcal{I}(\alpha_j) \mid \alpha_j \in (\Delta_0 - \{\alpha_i\})\}| \leq 1$.*

  I.2 *For every $e \in \mathcal{E}$: if $e \in \mathcal{I}(\alpha_i) \cap \mathcal{I}(\alpha_j)$, where $\alpha_i \neq \alpha_j \in \Delta_0$, then: $\cap\{\mathcal{I}(\alpha_k) \mid \alpha_k \in \Delta_0 \wedge e \in \mathcal{I}(\alpha_k)\} = \{e\}$.*

  I.3 *$\mathcal{E} = \bigcup_{\alpha_i \in \Delta_0} \mathcal{I}(\alpha_i)$.*
- *$\mathcal{P} \subseteq \mathcal{W} \times \mathcal{E}$, is a relation which indicates which event is permitted in which world.*

■

We can extend the function $\mathcal{I}$ to well-formed action terms and formulas, as follows:

- $\mathcal{I}(\neg\varphi) \overset{\mathsf{def}}{=} \mathcal{W} - \mathcal{I}(\varphi)$.
- $\mathcal{I}(\varphi \to \psi) \overset{\mathsf{def}}{=} \mathcal{I}(\neg\varphi) \cup \mathcal{I}(\psi)$.
- $\mathcal{I}(\alpha \sqcup \beta) \overset{\mathsf{def}}{=} \mathcal{I}(\alpha) \cup \mathcal{I}(\beta)$.
- $\mathcal{I}(\alpha \sqcap \beta) \overset{\mathsf{def}}{=} \mathcal{I}(\alpha) \cap \mathcal{I}(\beta)$.
- $\mathcal{I}(\overline{\alpha}) \overset{\mathsf{def}}{=} \mathcal{E} - I(\alpha)$.
- $\mathcal{I}(\emptyset) \overset{\mathsf{def}}{=} \emptyset$.
- $\mathcal{I}(\mathbf{U}) \overset{\mathsf{def}}{=} \mathcal{E}$.

Conditions **I.1** and **I.2** in definition 4 express topological requirements on the possible interpretations of atomic actions. **I.1** says that *the isolated application of an action always generates at most one event*; otherwise we will have an undesired nondeterminism in our models, as the different ways of executing an atomic action arise because you can execute it together with other actions (perhaps environmental actions). **I.2** establishes that *if an event is a result of the execution of two of more actions, then the concurrent execution of all the actions which generate it will give us only this event.* This condition also ensures that there will not occur a weird nondeterminism.

Some notation is needed for dealing with the relational part of the structure: we will use the notation $w \xrightarrow{e} w'$ when $(w, w', e) \in \mathcal{R}$. For a given $e \in \mathcal{E}$ we define the relation $\mathcal{R}_e = \{(w, w') \mid (w, w', e) \in \mathcal{R}\}$. Also, given a $w \in \mathcal{W}$ we define: $\mathcal{P}_w = \{e \in \mathcal{E} \mid (w, e) \in \mathcal{P}\}$. These definitions will be useful in the following sections. Let us introduce the relation $\vDash$ between models and formulae.

Definition 5 ($\vDash$)**.** *Given a vocabulary $L = \langle \Phi_0, \Delta_0 \rangle$ and a L-structure $M = \langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{I}, \mathcal{P} \rangle$, we define the relation $\vDash$ between worlds and formulas as follows:*

- $w, M \vDash p \overset{\text{def}}{\Longleftrightarrow} w \in \mathcal{I}(p)$
- $w, M \vDash \alpha =_{act} \beta \overset{\text{def}}{\Longleftrightarrow} \mathcal{I}(\alpha) = \mathcal{I}(\beta)$
- $w, M \vDash \neg\varphi \overset{\text{def}}{\Longleftrightarrow}$ *not* $w \vDash \varphi$.
- $w, M \vDash \varphi \to \psi \overset{\text{def}}{\Longleftrightarrow} w \vDash \neg\varphi$ *or* $w \vDash \psi$ *or both.*
- $w, M \vDash \langle\alpha\rangle\phi \overset{\text{def}}{\Longleftrightarrow}$ *there exists some* $w' \in \mathcal{W}$ *and* $e \in \mathcal{I}(\alpha)$ *such that* $w \xrightarrow{e} w'$ *and* $w', M \vDash \phi$.
- $w, M \vDash \mathsf{P}(\alpha) \overset{\text{def}}{\Longleftrightarrow}$ *for all* $e \in \mathcal{I}(\alpha)$, $\mathcal{P}(w, e)$ *holds.*
- $w, M \vDash \mathsf{P}_{\mathsf{w}}(\alpha) \overset{\text{def}}{\Longleftrightarrow}$ *there exists some* $e \in \mathcal{I}(\alpha)$ *such that* $\mathcal{P}(w, e)$

■

As usual, we say that $M \vDash \varphi$ iff for all worlds $w \in \mathcal{W}$ we have: $w, M \vDash \varphi$. And we say: $\vDash \varphi$, if $M \vDash \varphi$ for all models M. Some intuition about each operator is useful:

- $[\alpha]\phi$, *after executing* α *in any possible way* ϕ *will hold.*
- $[\alpha_1 \sqcup \alpha_2]\phi$, *every way of executing* α_1 *or* α_2 *leads to* ϕ.
- $[\overline{\alpha}]\phi$, *after executing an action different from* α ϕ *holds.*
- $[\alpha_1 \sqcap \alpha_2]$, *every way of executing both* α_1 *and* α_2 *leads to* ϕ.
- $\mathsf{P}(\alpha)$, *all the different ways of executing* α *are allowed.*
- $\mathsf{P}_{\mathsf{w}}(\alpha)$, *there is at least one way of executing* α *which is allowed.*

Note that boolean operators are interpreted on the set of events. Another approach is to interpret each atomic action as a relation, and then each boolean operator as a relation operator. Here, it is more useful for us to have a set of events interpreting an action, in particular for the semantics of the permission predicate.

If we follow the semantics we can see why the formula: $\neg\mathsf{P}(\overline{\alpha})$ does not give us the correct obligation operator. Since the semantics of permission is defined using a "for all" quantifier over

events, its negation can be defined using an "existential" quantifier; then the intuitive meaning of the expression $\mathsf{O}(\alpha) \overset{\mathsf{def}}{\Longleftrightarrow} \neg\mathsf{P}(\overline{\alpha})$ is: *something is obligated iff there exists an event which is not related to α which not permitted*. We can prove that this obligation operator is too weak, in the sense that it results in *Ross's paradox* ([24]). The introduction of the obligation operator using the weak permission is stronger (it does not result in Ross's paradox), and intuitively more acceptable for us.

A comment may be useful about the empty action ($\emptyset$); it seems that this is a weird action in some sense. In section 4 we will add temporal semantics to our logics, and it will be possible to prove in all possible models the action never happens. In some sense, it is a impossible action. Then, why predicate about whether it is permitted or not? In [10], $\neg\mathsf{P}(\emptyset)$ is valid, but in our logic we have $\mathsf{P}(\emptyset)$ and $\neg\mathsf{P}_{\mathsf{w}}(\emptyset)$; which is correct? From the common sense point of view, this discussion is immaterial; thus, we choose to allow impossible actions since it gives us completeness of our logic. Additionally, we will show that this fact is important for simplifying proofs in the logic.

3 A Deductive System

In this section we present a deductive system, this is a normal modal system (in the sense that the **k**-axiom can be deduced from it), the axioms for the box modalities are similar to those given at [4] and [7]. We assume the standard definition of the relation $\vdash \subseteq \wp(\Phi) \times \Phi$. And we will say: $\vdash_{DPL} \varphi$, if φ is a theorem of the following axiomatic system (we shall omit the subscript DPL when there are no confusions).

An important characteristic in our set of axioms (which, up to the authors knowledge, it is not shared with other works) is that it establishes a deep connection between the weak version of permission and the strong version of it. Actually, one of these axioms can be seen as a kind of "compactness" property that our models satisfy, this property is implied by the restrictions assumed about them. This is a key fact exploited in the completeness proof. Before going into details, we need to introduce the notions of *canonical action terms* and *boolean algebra of action terms*. For the next definitions we consider the following fixed vocabulary:

$$\Phi_0 = \{p_1, ..., p_m\} \quad \Delta_0 = \{\alpha_1, ..., \alpha_n\}$$

This language induces the set Δ of boolean terms (see definition 2), we call Φ_{BA} to some axiomatization of boolean algebras (note that there exist complete axiomatizations, see [26]). Then, the set Δ/Φ_{BA} is the quotient set of the boolean terms by $=_{act}$, the point is that using this set we can define the (atomic) boolean algebra $\langle \Delta/\Phi_{BA}, \sqcup_{[]}, \sqcap_{[]}, -_{[]}, [\emptyset]_{BA}, [U]_{BA} \rangle$ as follows:

- $-_{[]}[\alpha]_{BA} = [\overline{\alpha}]_{BA}$
- $[\alpha]_{BA} \sqcup_{[]} [\beta]_{BA} = [\alpha \sqcup \beta]_{BA}$
- $[\alpha]_{BA} \sqcap_{[]} [\beta]_{BA} = [\alpha \sqcap \beta]_{BA}$

It is straightforward to prove that this is a boolean algebra. Furthermore, since the terms in Δ are generated by a finite set Δ_0 of atomic actions, the quotient boolean algebra is finite, and therefore atomic. We call $at(\Delta/\Phi_{BA})$ (or $at(\Delta)$ when no confusions arise) to the set of atoms of the quotient boolean algebra of terms. Note also that we can define $\sqsubseteq_{[]}$ in the usual way.

It will be useful to remember the following theorems about atomic boolean algebras.

Theorem 1. *For every finite boolean algebra $\langle A, \cup, \cap, -, 0, 1 \rangle$, the following holds: for all $x \in A$, there exist atoms $a_1, ..., a_n$ such that: $x = a_1 \cup ... \cup a_n$.*

Proof. *See [20].*
■

Theorem 2. *For every finite boolean algebra* $B = \langle A, \cup, \cap, -, 0, 1\rangle$*, and considering* $A = \{a \mid a \text{ is an atom of } B\}$*, there exists an isomorphism between* B *and the boolean algebra* $\langle \wp(A), \cup, \cap, -, \emptyset, A\rangle$*. The isomorphism is defined by* $f(x) = \{a \mid a \leq x\}$*.*
Proof. *See [20].*
■

An useful corollary of this theorem is:

Corollary 1. *Given a boolean algebra* B*, and element* x *of* B*, then* x *is an atom iff the cardinality of* $\{a \mid a \leq x \wedge a \text{ is an atom}\}$ *is 1.*
Proof. *One direction is trivial. The other is straightforward using the fact that* f *is bijective.*
■

At this point we are ready to present our axiomatic system. over formulas.

Definition 6 (Axioms for DPL)**.** *Given a vocabulary* $\langle \Phi_0, \Delta_0\rangle$*, where* $\Delta_0 = \{\alpha_1, ..., \alpha_n\}$*. The axiomatic system is composed by the following axioms:*

1. *The sets of propositional tautologies.*
2. *A set of axioms for boolean algebras for action terms (a complete one), including standard axioms for equality.*
3. *The following set of axioms*

   A1. $\langle\alpha\rangle\bot \leftrightarrow \bot$
   A2. $[\emptyset]\varphi$
   A3. $\langle\alpha\rangle\varphi \wedge [\alpha]\psi \rightarrow \langle\alpha\rangle(\varphi \wedge \psi)$
   A4. $[\alpha \sqcup \alpha']\varphi \leftrightarrow [\alpha]\varphi \wedge [\alpha']\varphi$
   A5. $[\alpha]\varphi \rightarrow [\alpha \sqcap \alpha']\varphi$
   A6. $\mathsf{P}(\emptyset)$
   A7. $\mathsf{P}(\alpha \sqcup \beta) \leftrightarrow \mathsf{P}(\alpha) \wedge \mathsf{P}(\beta)$
   A8. $\mathsf{P}(\alpha) \vee \mathsf{P}(\beta) \rightarrow \mathsf{P}(\alpha \sqcap \beta)$
   A9. $\neg\mathsf{P}_\mathsf{w}(\emptyset)$
   A10. $\mathsf{P}_\mathsf{w}(\alpha \sqcup \beta) \leftrightarrow \mathsf{P}_\mathsf{w}(\alpha) \vee \mathsf{P}_\mathsf{w}(\beta)$
   A11. $\mathsf{P}_\mathsf{w}(\alpha \sqcap \beta) \leftrightarrow \mathsf{P}_\mathsf{w}(\alpha) \wedge \mathsf{P}_\mathsf{w}(\beta)$
   A12. $\mathsf{P}(\alpha) \wedge \alpha \neq \emptyset \rightarrow \mathsf{P}_\mathsf{w}(\alpha)$
   A13. $\bigwedge_{[\alpha]_{BA} \wedge \alpha \sqsubseteq \alpha'}(P_w(\alpha) \vee (\alpha =_{act} \emptyset)) \rightarrow \mathsf{P}(\alpha')$
   A14. $\mathsf{P}(\alpha) \wedge \neg\mathsf{P}_\mathsf{w}(\overline{\alpha}) \leftrightarrow \mathsf{O}(\alpha)$
   A15. $\langle\alpha\rangle\varphi \leftrightarrow \neg[\alpha]\neg\varphi$
   A16. $(\alpha_1 \sqcup ... \sqcup \alpha_n) =_{act} \mathbf{U}$
   A17. $\alpha =_{act} \alpha' \leftrightarrow [\beta](\alpha =_{act} \alpha')$

And the following deduction rules:

$$\mathsf{MP}: \frac{\varphi \quad \varphi \to \psi}{\psi} \qquad \mathsf{GN}: \frac{\varphi}{[\alpha]\phi} \qquad \mathsf{BA}: \frac{\alpha =_{act} \alpha' \quad \varphi[\alpha]}{\varphi[\alpha/\alpha']}$$

■

Some explanation is needed for axiom A13; this axiom expresses an intuitive connection between strong and week versions of permission: "if every subset of an action α is weekly permitted then α is strongly permitted". Or, by the contrapositive, "if an action is not strongly allowed, then some subset is not weekly permitted". As the reader can deduce, this is a kind of compactness property which relates both version of permission. Note that the given formula is finite, because the underlying canonical term algebra is finite; in this way we can avoid second order quantifiers. Unfortunately, in a first order extension of this axiomatic system we cannot use the same trick.

Two essential requirements for propositional logics are the soundness and completeness properties; we shall show that the given axiomatic system has both properties. Although one future extensions of the logic (the first order tense version) are not complete (but sound); these two theorems give us enough confidence about the adequacy of the basic system, whose axioms will remain in future versions.

Note that the deduction rule BA uses substitution on formulae: the notation $\varphi[\alpha]$ means that the formula (which the metavariable denotes) φ has an occurrence of the boolean term (which the metavariable denotes) α, and we write $\varphi[\alpha/\alpha']$ to mean that the term α is replaced in some of its occurrences by the boolean term α'.

In [4] and [3], two different complete, and sound, systems are given for the modal part of the logic (that is, action terms and box modality), but both systems do not have deontic concepts, and the complement described in those papers is the absolute one. Note that the one described here is a kind of relative complement.

Theorem 3 (soundness)**.** *The axiomatic system defined in definition 6 is sound with respect to the models defined in definition 4, that is:*

$$\vdash \varphi \Rightarrow \vDash \varphi$$

Proof. *We have to prove that each axiom is valid, and that the deduction rules preserve validity. Axioms 1-3 and the deduction rules MP and GN are very standard and their soundness proofs can be found in the literature. On the other hand, it is clear that boolean algebra axioms are valid, since the interpretation of action operators are given by means of set operators. We prove the validity of axioms 4-17 and that the deduction rule BA preserves validity.*
Axiom 4: Straightforward by first order properties. See axiom 7's proof.
Axiom 5: Direct using subset properties and "for all" properties.
Axiom 6: Straightforward by definition of $\vDash$ and vacuous domain.
Axiom 7: Suppose $w, W \vDash \mathsf{P}(\alpha \sqcup \beta)$, for arbitrary model M and world w. This means that: $\forall e \in \mathcal{I}(\alpha \sqcup \beta) : \mathcal{P}(w, e)$, using first order logic we get: $(\forall e \in \mathcal{I}(\alpha) : \mathcal{P}(w, e)) \wedge (\forall e \in \mathcal{I}(\alpha) : \mathcal{P}(w, e))$, and this implies: $w, M \vDash \mathsf{P}(\alpha) \wedge \mathsf{P}(\beta)$.
Axiom 8: Similar reasoning as before, but using the fact that: $\mathcal{I}(\alpha \sqcap \beta) = \mathcal{I}(\alpha) \cap \mathcal{I}(\beta)$.
Axiom 9: For every model M and world w, by logic we have: $\neg(\exists e \in I(\emptyset) : \mathcal{P}(w, e))$, and this means: $\vDash \neg \mathsf{P_w}(\emptyset)$.
Axiom 10: Suppose $w, M \vDash \mathsf{P_w}(\alpha \sqcup \beta)$; by definition we obtain: $\exists e \in \mathcal{I}(\alpha \sqcup \beta) : \mathcal{P}(w, e)$ and the using definition of $\mathcal{I}$ and properties of $\exists$ we get: $w, M \vDash \mathsf{P_w}(\alpha) \vee \mathsf{P_w}(\beta)$.
Axiom 11: Similar to Axiom 10.

Axiom 12: Suppose that $w, M \vDash \mathsf{P}(\alpha) \wedge \alpha \neq \emptyset$*; this means:* $\forall e \in I(\alpha) : e \in \mathcal{P}_w$ *and* $I(\alpha) \neq \emptyset$*; by basic first order reasoning we get:* $\exists e \in I(\alpha) : \mathcal{P}(w, e)$*, but this implies* $w, M \vDash \mathsf{P_w}(\alpha)$*.*
Axiom A13: Suppose that for some w*:*

$$w, M \vDash \bigwedge_{[\alpha]_{BA} \wedge \alpha \sqsubseteq \alpha'} (\mathsf{P_w}(\alpha) \vee \alpha =_{act} \emptyset)$$

and not:

$$w, M \nvDash \mathsf{P}(\alpha')$$

The last formula implies: $\exists e \in \mathcal{I}(\alpha') : \neg\mathcal{P}(w, e)$*. Now, using condition* ***I.1*** *and* ***I.2****, we can reason by cases:*

- *If* $e \in \mathcal{I}(\alpha_i) - \bigcup_{j \neq i}(\mathcal{I}(\alpha_j))$*, for some* α_i*. Therefore by condition* ***I.1*** *we get:* $\mathcal{I}(\alpha_i \sqcap (\bigsqcup_{j \neq i} \overline{\alpha_j})) = \{e\}$*.. And then* $w, M \nvDash \mathsf{P_w}(\alpha_i \sqcap (\bigsqcap_{j \neq i} \overline{\alpha_j}))$*. This giving us a contradiction.*
- *If* $e \in \mathcal{I}(\alpha_i) \cap \mathcal{I}(\alpha_j)$*, for some* $i \neq j$*. Then, let* $\alpha_1^1, ..., \alpha_m^1$ *all the atomic action such that:* $e \in \mathcal{I}(\alpha_k^1)$*, for* $1 \leq k \leq m$*. Then by condition I2, we get:* $\bigcap_{1 \leq k \leq m} \mathcal{I}(\alpha_k^1) = \{e\}$*. And then* $w, M \nvDash \mathsf{P_w}(\alpha_1^1 \sqcap ... \sqcap \alpha_m^1)$*, giving a contradiction, the result follows.*

Axiom A14, Axiom A15, Axiom16: Straightforward.
Axiom A17: The result follows from the fact that action interpretations are fixed, and they do not depend on states.
BA: The result is straightforward from the fact that if we have $\alpha =_{act} \alpha'$ *then* $I(\alpha) = I(\alpha')$*, here using the Leibniz equality property deduce that if* $\vDash \varphi[\alpha]$ *then* $\vDash \varphi[\alpha']$*.*
■

Note that axioms A1, A3 and rule GN implies that we have a normal modal logic; in [5] a number of useful standard modal logics can be found; when we use some of these properties during the proofs, we will use the word ML to indicate this.

As for most modal logics, the deduction theorem does not hold in our formalism, but we can prove a very weak version of it: it holds for boolean formulas, as the following theorem shows.

Theorem 4 (Deduction Theorem for BA)**.** $\{\alpha' =_{act} \alpha\} \vdash \varphi$ *if only if* $\vdash ((\alpha' =_{act} \alpha) \rightarrow \varphi)$*.*
Proof. *By induction on the length of the proof. It is similar to the standard proof, changing the case when we have a proof* $P_1, ..., P_n$ *and* $P_n = \varphi$*, and we obtain it using a generalization of some* P_i*, that is:* $[\beta]P_i = \varphi$*, for some* i*. Then by the inductive hypothesis:*

$$\vdash (\alpha =_{act} \alpha') \rightarrow P_i$$

Using GN

$$\vdash [\beta]((\alpha =_{act} \alpha') \rightarrow P_i)$$

Then using modal logic properties:

$$\vdash [\beta](\alpha =_{act} \alpha') \rightarrow [\beta]P_i$$

By axiom A17:

$$\vdash (\alpha =_{act} \alpha') \rightarrow [\beta]P_i$$

■

The following theorems of the axiomatic system defined are used in the completeness proof, actually in [4] theorem T3 is used for axiomatizing the modal part of boolean logic, and it should be enough for the modal part of our logic. Because we are looking an algebraic view of the logic, in our axiomatic system we focused on operational properties.

Theorem 5. *The following are theorems of DPL,*

T1. $\mathsf{P}(\alpha) \wedge \alpha' \sqsubseteq \alpha \rightarrow \mathsf{P}(\alpha')$

T2. $\mathsf{P}_{\mathsf{w}}(\alpha') \wedge \alpha' \sqsubseteq \alpha \rightarrow \mathsf{P}_{\mathsf{w}}(\alpha)$

T3. $[\alpha]\varphi \wedge (\alpha' \sqsubseteq \alpha) \rightarrow [\alpha']\varphi$

T4. $[\alpha]\varphi \wedge [\alpha']\psi \rightarrow [\alpha \sqcup \alpha'](\varphi \vee \psi)$

T5. $[\alpha]\varphi \wedge [\alpha']\psi \rightarrow [\alpha \sqcap \alpha'](\varphi \wedge \psi)$

Proof.
T1) We can prove $\{\alpha' =_{act} \alpha \sqcap \alpha'\} \vdash \mathsf{P}(\alpha) \rightarrow \mathsf{P}(\alpha')$*, and the theorem follows by theorem 4 and by definition of* $\sqsubseteq$*.*

1. $\alpha' =_{act} \alpha \sqcap \alpha'$	*Hyp.*
2. $\mathsf{P}(\alpha) \rightarrow \mathsf{P}(\alpha \sqcap \alpha')$	*Axiom A8*
3. $\mathsf{P}(\alpha) \rightarrow \mathsf{P}(\alpha')$	*BA, 1, 2*

□
T2) Similar to T1 but using axiom A10.
□
T3) Similar to T1 but using axiom A5.
T4)

1. $[\alpha]\varphi \rightarrow [\alpha](\varphi \vee \psi)$	*ML*
2. $[\alpha']\psi \rightarrow [\alpha'](\varphi \vee \psi)$	*ML*
3. $[\alpha]\varphi \wedge [\varphi']\psi \rightarrow [\alpha](\varphi \vee \psi) \wedge [\alpha'](\varphi \vee \psi)$	*PL,1 ,2*
4. $[\alpha](\varphi \vee \psi) \wedge [\alpha'](\varphi \vee \psi) \rightarrow [\alpha \sqcup \alpha'](\varphi \vee \psi)$	*A4*
5. $[\alpha]\varphi \wedge [\alpha']\psi \rightarrow [\alpha \sqcup \alpha'](\varphi \vee \psi)$	*PL, 3, 4*

□
T5)

1. $[\alpha]\varphi \wedge \alpha \sqcap \alpha' \sqsubseteq \alpha \rightarrow [\alpha \sqcap \alpha']\varphi$	*T3*
2. $[\alpha]\varphi \wedge \top \rightarrow [\alpha \sqcap \alpha']\varphi$	*Definition of* $\sqsubseteq$
3. $[\alpha]\varphi \rightarrow [\alpha \sqcap \alpha']\varphi$	*PL, 2*
4. $[\alpha]\varphi \wedge [\alpha']\psi \rightarrow [\alpha \sqcap \alpha'](\varphi \wedge \psi)$	*PL & ML, 4.*

□
■

This theorem says that $\mathsf{P}(-)$ is antitone, and $\mathsf{P}_{\mathsf{w}}(-)$ is monotone. An interesting remark is that the strong permission defines an ideal on the event structure, and the weak permission defines a filter. Although this fact seems very interesting, we do not pursue any further this issue here.

Now, we can introduce the following (canonical) model:

Definition 7 (canonical model)**.** $\mathcal{C} = \langle \mathcal{E}_{\mathcal{C}}, \mathcal{W}_{\mathcal{C}}, \mathcal{R}_{\mathcal{C}}, \mathcal{P}_{\mathcal{C}}, \mathcal{I}_{\mathcal{C}} \rangle$ *where:*

- $\mathcal{E}_{\mathcal{C}} \stackrel{\mathsf{def}}{=} at(\Delta)$
- $\mathcal{W}_{\mathcal{C}} \stackrel{\mathsf{def}}{=} \{\Gamma \mid \Gamma \textit{ is a maximal consistent set of } formulae\}$

- $\mathcal{R}_{\mathcal{C}} \stackrel{\text{def}}{=} \bigcup\{\mathcal{R}_{\alpha,w,w'} \mid w,w' \in \mathcal{W}_{\mathcal{C}} \wedge \alpha \in \Delta \wedge (\forall \varphi \in \Phi : [\alpha]\varphi \in w \Rightarrow \varphi \in w')\}$, *where* $R_{\alpha,w,w'} \stackrel{\text{def}}{=} \{w \stackrel{[\alpha']_{BA}}{\rightarrow} w' \mid \forall [\alpha']_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha)\}$
- $\mathcal{P}_{\mathcal{C}} \stackrel{\text{def}}{=} \bigcup\{\mathcal{P}_{w,\alpha} \mid w \in \mathcal{W}_{\mathcal{C}} \wedge \mathsf{P}(\alpha) \in w\}$, *where:* $\mathcal{P}_{w,\alpha} \stackrel{\text{def}}{=} \{(w,[\alpha']_{BA}) \mid [\alpha']_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha)\}$
- $\mathcal{I}_{\mathcal{C}}(\alpha_i) \stackrel{\text{def}}{=} \{[\alpha']_{BA} \in \mathcal{E}_{\mathcal{C}} \mid\vdash_{\Phi_{BA}} \alpha' \sqsubseteq \alpha\}$
- $\mathcal{I}_{\mathcal{C}}(p_i) \stackrel{\text{def}}{=} \{w \in \mathcal{W}_{\mathcal{C}} \mid p_i \in w\}$

■

We will use this model to show the completeness of the logic; the usual way to do this is to prove an equivalent result: *each consistent set of formulas has a model*. First, we have to establish a number of useful lemmas:

Lemma 1. $\forall \alpha \in \Delta, \forall [\alpha']_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha) : \vdash_{\Phi_{BA}} \alpha' \sqsubseteq \alpha$
Proof. *The proof is by induction on the term* α.
Base Case)

- *If* $\alpha = \emptyset$, *then by definition* $\mathcal{I}_{\mathcal{C}}(\emptyset) = \emptyset$ *and the result follows by empty domain. Note that we use the symbol* $\emptyset$ *in two different ways: the first one as an action term, and the second one as the empty set.*
- *If* $\alpha = \alpha_i$, *then the result is straightforward by definition of* $\mathcal{I}_{\mathcal{C}}$.

Inductive Case)

- *case* $(\alpha = \alpha' \sqcup \alpha'')$*: let* $[\gamma]_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha' \sqcup \alpha'')$ *be an event; then by definition 7 we have* $[\gamma]_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha') \cup \mathcal{I}_{\mathcal{C}}(\alpha'')$, *by hypothesis we obtain:* $\vdash_{\Phi_{BA}} \gamma \sqsubseteq \alpha'$ *or* $\vdash_{\Phi_{BA}} \gamma \sqsubseteq \alpha''$, *and therefore by properties of boolean algebras we obtain:* $\vdash_{\Phi_{BA}} \gamma \sqsubseteq \alpha' \sqcup \alpha''$.
- *case* $(\alpha = \alpha' \sqcap \alpha'')$*: similar argument to the last step.*
- *case* $(\alpha = \overline{\alpha'})$*: suppose* $[\gamma]_{BA} \in \mathcal{E} - \mathcal{I}_{\mathcal{C}}(\alpha)$, *then* $\nvdash_{BA} \gamma \sqsubseteq \alpha$*; since* γ *is an atom, by boolean algebra properties we get:* $\vdash_{\Phi_{BA}} \gamma \sqsubseteq \overline{\alpha}$.

■

Now, we can prove a fundamental lemma.

Lemma 2 (truth lemma)**.** $w, \mathcal{C} \vDash \varphi \Leftrightarrow \varphi \in w$.
Proof. *The proof is by induction on* φ.
Base Case. Using the definition we get

$$w, \mathcal{C} \vDash p_i \Leftrightarrow w \in \mathcal{I}_{\mathcal{C}}(p_i) \Leftrightarrow p_i \in w$$

Inductive Case. We have several cases:

CASE I. we have to prove $w, \mathcal{C} \vDash [\alpha]\varphi \Leftrightarrow [\alpha]\varphi \in w$.
$\Rightarrow$) *Suppose* $w, \mathcal{C} \vDash [\alpha]\varphi$, *this means:*

$$\begin{array}{ll} \forall [\gamma]_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha), \forall w' \in \mathcal{W}_{\mathcal{C}} : w \stackrel{[\gamma]_{BA}}{\rightarrow} w' \Rightarrow w', \mathcal{C} \vDash \varphi & \\ \equiv & \textit{[inductive hypothesis]} \\ \forall [\gamma]_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha), \forall w' \in \mathcal{W}_{\mathcal{C}} : w \stackrel{[\gamma]_{BA}}{\rightarrow} w' \Rightarrow \varphi \in w' & \textit{(*)} \end{array}$$

On the other hand, suppose $[\alpha]\varphi \notin w$*, then (recalling properties of maximal consistent sets)* $\langle\alpha\rangle\neg\varphi \in w$*. Now, consider the set:* $\Gamma = \{\neg\varphi\} \cup \{\psi \mid [\alpha]\psi \in w\}$*. We claim that this set is consistent, if not:*

$$\exists\psi_1, ..., \psi_n \in \Gamma : \{\psi_1, ..., \psi_n, \varphi\} \vdash \bot$$

by definition of contradiction. But using this we can deduce:

$\langle\alpha\rangle\neg\varphi \wedge [\alpha]\psi_1 \wedge ... \wedge [\alpha]\psi_n \in w$	
$\Rightarrow$	*[axiom 3 and maximal consistent set properties]*
$\langle\alpha\rangle(\neg\varphi \wedge \psi_1 \wedge ... \wedge \psi_n) \in w$	
$\Leftrightarrow$	*[hypothesis]*
$\langle\alpha\rangle\bot \in w$	
$\Leftrightarrow$	*[axiom 1]*
$\bot \in w$ *!!*	

Then Γ *has to be consistent, and therefore it has a maximal consistent extension (by Zorn's lemma)* Γ^**. But by definition of* $\mathcal{R}_{\mathcal{C}}$*:*

$$\forall[\gamma]_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha) : w \overset{[\gamma]_{BA}}{\rightarrow} \Gamma^* \wedge \Gamma^* \vDash \neg\varphi$$

which contradicts (*) and therefore $[\alpha]\varphi \in w$*.*
$\Leftarrow$*) Suppose* $[\alpha]\varphi \in w$*; we have to prove* $w \vDash [\alpha]\varphi$*. Suppose that* $w \nvDash [\alpha]\varphi$ *then this means:*

$$\exists[\gamma]_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha), \exists w' \in \mathcal{W}_{\mathcal{C}} : w \overset{[\gamma]_{BA}}{\rightarrow} w' \wedge w', \mathcal{C} \nvDash \varphi$$

which is equivalent to (by ind.hyp.):

$$\exists[\gamma]_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha), \exists w' \in \mathcal{W}_{\mathcal{C}} : w \overset{[\gamma]_{BA}}{\rightarrow} w' \wedge \varphi \notin w' \quad (**)$$

But, by definition of $\mathcal{R}_{\mathcal{C}}$*, this means:*

$\exists w' \in \mathcal{W}_{\mathcal{C}} : (\forall\psi : [\gamma]\psi \in w \Rightarrow \psi \in w') \wedge \varphi \notin w'$	
$\Rightarrow$	*[logic]*
$\neg([\gamma]\varphi) \in w$	
$\Leftrightarrow$	*[max.cons.set properties]*
$[\gamma]\varphi \notin w$	*(***)*

But we know by lemma 1 that $\gamma \sqsubseteq \alpha$*. From here and using hypothesis (*$[\alpha]\varphi \in w$*) and using theorem T3, we obtain:* $[\gamma]\varphi \in w$*!!. And therefore:* $w, \mathcal{C} \vDash [\alpha]\varphi$*.*

<u>CASE II.</u> *We have to prove:* $w, \mathcal{C} \vDash \mathsf{P}(\alpha) \Leftrightarrow \mathsf{P}(\alpha) \in w$*.*
$\Rightarrow$*) Suppose* $w, \mathcal{C} \vDash \mathsf{P}(\alpha)$*, this means:*

$$\forall[\gamma]_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha) : \mathcal{P}_{\mathcal{C}}(w, [\gamma]_{BA})$$

Because of lemma 1, this implies (using definition of $\mathcal{P}_{\mathcal{C}}$*) that either* $\mathsf{P}(\alpha)$ *or* $\mathsf{P}(\beta)$ *where* $\vdash_{\Phi_{BA}} \alpha \sqsubseteq \beta$*, since there is no other way to introduce this relation in the canonical model. In both cases the result follows, in the first trivially, in the second one by using T1.*
$\Leftarrow$*) Suppose that* $\mathsf{P}(\alpha) \in w$*, by definition of* $\mathcal{P}_{\mathcal{C}}$ *this means:*

$$\forall[\gamma]_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha) : \mathcal{P}_{\mathcal{C}}(w, [\gamma]_{BA})$$

But using the definition of $\vDash$ *we get:* $w, \mathcal{C} \vDash P(\alpha)$.

CASE III. $w, \mathcal{C} \vDash \mathsf{P_w}(\alpha) \Leftrightarrow \mathsf{P_w}(\alpha) \in w$.
For the case $\alpha =_{act} \emptyset$ *the equivalence is trivial; let us prove the other case* ($\alpha \neq_{act} \emptyset$).
$\Rightarrow$) *Suppose* $w, \mathcal{C} \vDash \mathsf{P_w}(\alpha)$ *that means:*

$$\exists [\gamma]_{BA} \in \mathcal{I}_{\mathcal{C}} : \mathcal{P}_{\mathcal{C}}(w, [\gamma]_{BA})$$

By definition of $\mathcal{P}_{\mathcal{C}}$ *this only happens if for some* β: $\gamma \sqsubseteq \beta$ *and* $\mathsf{P}(\beta) \in w$. *Then by theorem T1 this implies* $\mathsf{P}(\gamma) \in w$, *and therefore, using axiom A12, we get:* $\mathsf{P_w}(\gamma) \in w$; *from this, by theorem T2, we obtain* $\mathsf{P_w}(\alpha) \in w$.
$\Leftarrow$) *Suppose* $\mathsf{P_w}(\alpha) \in w$. *We know by properties of atomic boolean algebras that:*

$$\begin{array}{ll} [\alpha]_{BA} = [\gamma_1]_{BA} \sqcup_{[]} ... \sqcup_{[]} [\gamma_n]_{BA} & \textit{for some } [\gamma_1]_{BA}, ..., [\gamma_n]_{BA} \textit{ atoms in } \Delta/\Phi_{BA} \\ \Leftrightarrow & \textit{[def. of } \Delta/\Phi_{BA}\textit{]} \\ [\alpha]_{BA} = [\gamma_1 \sqcup ... \sqcup \gamma_n]_{BA} & \end{array}$$

But this implies by deduction rule BA that $\mathsf{P_w}(\gamma_1 \sqcup ... \sqcup \gamma_n) \in w$. *By axiom A10, this implies:*

$$\mathsf{P_w}(\gamma_1) \vee ... \vee \mathsf{P_w}(\gamma_n) \in w$$

Let γ_i *be some of these action terms such that* $\mathsf{P_w}(\gamma_i) \in w$; *since* $\gamma_i \in at(\Delta)$, *we have:*

$$\bigwedge_{[\alpha]_{BA} \wedge \alpha \sqsubseteq \gamma_i} (\mathsf{P_w}(\alpha) \vee (\alpha =_{act} \emptyset)) \in w$$

Then by MP and A13 we get $\mathsf{P}(\gamma_i) \in w$. *By definition of* $\mathcal{P}_{\mathcal{C}}$, *this implies that:*

$$\exists [\gamma]_{BA} \in \mathcal{I}_{\mathcal{C}}(\alpha) : \mathcal{P}_{\mathcal{C}}([\gamma]_{BA}, w)$$

and this is just the definition of $w \vDash \mathsf{P_w}(\alpha)$.
■

Note that we have to prove that the defined interpretation $\mathcal{I}_{\mathcal{C}}$ holds with the restrictions **I.1** and **I.2** (**I.3** is satisfied by definition). The following theorems do this,

Theorem 6. *The function* $\mathcal{I}_{\mathcal{C}}$ *satisfies the conditions* ***I.1,I.2****.*
Proof. *First note that all the atoms of the boolean algebra* Δ/Φ_{BA} *(the Lindenbaum-Tarski algebra [26]) have the following form (or are equivalent to it):*

$$\alpha_1^1 \sqcap ... \sqcap \alpha_m^1 \sqcap \alpha_1^2 \sqcap ... \sqcap \alpha_k^2$$

Where for all $\alpha_i \in \Delta_0$: $\alpha_i = \alpha_j^1$ *or* $\overline{\alpha_i} = \alpha_j^2$, *for some* j. *That is, the atoms in the Lindenbaum algebra can be represented by terms which are composed of "intersections" of atomic actions or their negations. It is for this reason that the atoms of the Lindenbaum algebra are quite suitable for representing labels in the model: each of them point out which atomic actions are executed and which are not.*

That $\mathcal{I}_{\mathcal{C}}$ *satisfies conditions* ***I.1,I.2*** *is implied by the underlying structure of the generated Lindenbaum Algebra:*
I.1*: If* $[\gamma] \in \mathcal{I}_{\mathcal{C}}(\alpha_i) - \bigcup_{j \neq i}(\mathcal{I}_{\mathcal{C}}(\alpha_j))$, *then* $\gamma =_{act} \alpha_i \sqcap (\bigsqcap_{j \neq i}(\overline{\alpha_j}))$, *where* $\bigsqcap$ *is used to denote the application of* $\sqcap$ *to a finite sequence of boolean terms.*
I.2*: We have to show that if* $[\gamma] \in \mathcal{I}(\alpha_i) \cap \mathcal{I}(\alpha_j)$, *for some* $i \neq j$. *Then:*

$$\bigcap \{\mathcal{I}(\alpha_k) \mid [\gamma] \in \mathcal{I}(\alpha_k)\} = \{[\gamma]\} \tag{1}$$

In this case it is easy to see that:

$$\gamma =_{act} \alpha_1^1 \sqcap ... \sqcap \alpha_m^1 \sqcap \overline{\alpha_1^2} \sqcap ... \sqcap \overline{\alpha_{m'}^2} \tag{2}$$

where α_i^1 are the atomic actions which have the equivalence class $[\gamma]$ in its interpretation, and α_i^2 are the rest. Since the right term in equation 2 is an atom, every other $[\gamma']$ that satisfies condition 1 also satisfies: $[\gamma'] = [\gamma]$. The theorem follows.
■

We have proved that the canonical model has the correct behavior; the completeness follows:

Corollary 2. *For every consistent set Γ of DPL, there is a model which satisfies it.*
Proof. *If Γ is consistent, then there exists a maximal extension of it which is a maximal consistent set, and therefore this set is a world w in the canonical model. By the definition of canonical model we know $w, \mathcal{C} \vDash \Gamma$; this finalizes the proof.*
■

From it we obtain:

Corollary 3. *If every finite subset of a set Γ of formulae is satisfable, then Γ is satisfable.*

The following theorems of the axiomatic system defined give us the first flavour of it.

Theorem 7. *The following sentences are theorems of PDL.*

T6. $\neg(\mathsf{O}(\alpha) \wedge \mathsf{O}(\overline{\alpha}))$

T7. $\mathsf{O}(\alpha) \wedge \mathsf{O}(\beta) \rightarrow \mathsf{O}(\alpha \sqcap \beta)$

T8. $\mathsf{P}(\mathbf{U}) \rightarrow \mathsf{P}(\alpha)$ *for every action* α

T9. $\mathsf{P_w}(\alpha) \rightarrow \mathsf{P_w}(\mathbf{U})$ *for every action* α

T10. $\mathsf{O}(\mathbf{U}) \leftrightarrow \mathsf{P}(\mathbf{U})$

T11. $\mathsf{O}(\emptyset) \leftrightarrow \neg\mathsf{P_w}(\mathbf{U})$

T12. $\mathsf{O}(\alpha) \rightarrow \mathsf{P}(\alpha)$

Proof.
T6. $\mathsf{O}(\alpha) \wedge \mathsf{O}(\overline{\alpha}) \leftrightarrow \mathsf{P}(\alpha) \wedge \mathsf{P}(\overline{\alpha}) \wedge \neg\mathsf{P_w}(\overline{\alpha}) \wedge \neg\mathsf{P_w}(\alpha) \rightarrow \mathsf{P_w}(\alpha) \wedge \mathsf{P_w}(\overline{\alpha}) \wedge \neg\mathsf{P_w}(\overline{\alpha}) \wedge \neg\mathsf{P_w}(\alpha) \leftrightarrow false.$
Then $\neg(\mathsf{O}(\alpha) \wedge \mathsf{O}(\overline{\alpha}))$

T7. $\mathsf{O}(\alpha) \wedge \mathsf{O}(\beta) \rightarrow \mathsf{P}(\alpha) \wedge \neg\mathsf{P_w}(\overline{\alpha}) \wedge \mathsf{P}(\beta) \wedge \neg\mathsf{P_w}(\overline{\beta}) \rightarrow \mathsf{P}(\alpha \sqcup \beta) \wedge \neg\mathsf{P_w}(\overline{\alpha} \sqcup \alpha\beta) \leftrightarrow \mathsf{P}(\alpha \sqcup \beta) \wedge \neg\mathsf{P_w}(\overline{\alpha \sqcap \beta})$

T8. we have $\mathsf{P}(\mathbf{U})$ *and as* $\alpha \sqsubseteq \mathbf{U}$*, using theorem* **??***, we get* $\mathsf{P}(\alpha)$

T9. similar to T6.

T10. $\mathsf{O}(\mathbf{U}) \leftrightarrow \mathsf{P}(\mathbf{U}) \wedge \neg\mathsf{P_w}(\overline{\mathbf{U}}) \leftrightarrow \mathsf{P}(\mathbf{U}) \wedge \neg\mathsf{P_w}(\emptyset) \leftrightarrow \mathsf{P}(\mathbf{U})$.

T11. $\mathsf{O}(\emptyset) \leftrightarrow \mathsf{P}(\emptyset) \wedge \neg\mathsf{P_w}(\overline{\emptyset}) \leftrightarrow \mathsf{P}(\emptyset) \wedge \neg\mathsf{P_w}(\mathbf{U}) \leftrightarrow \neg\mathsf{P_w}(\mathbf{U})$

T12. straightforward by definition.
■

Properties T6 and T8 are desirable from an intuitive point of view: T6 says if two actions are inconsistent, then they cannot both be obligated. T8 says that if an action is obligated, then it is permitted. Note that we do not have that obligation implies enabledness that is: $\mathsf{O}(\alpha) \rightarrow \langle\alpha\rangle\top$. This property is correct from some philosophical point of view (e.g., the Kantonian principle that ought implies can), but we do not believe that this is the case in computing systems: an action can be obligated, but it cannot be executed since it is waiting for some resource. In this aspect our definition of obligation differs from the one given at [10]. in that framework the Kantonian principle is followed. We do not have neither $\mathsf{P_w}(\alpha) \rightarrow \langle\alpha\rangle\top$ nor $\mathsf{P}(\alpha) \rightarrow \langle\alpha\rangle\top$.

Property T11 is strange at first sight; it says that it is obligated to do an impossible action if no action is allowed. Actually, it only says that in this case any action will give us problems. Note that we can add restrictions on models to avoid undesirable properties.

3.1 Some Considerations about the Resulting System

Several deontic logics have been studied, and developed, in the past few decades. It is usual to classify them into two groups: *ought-to-be* and *ought-to-do* logics. The first ones are logics where the deontic predicates are applied over facts, or situations, for example: *it is allowed that this wall be white*. Philosophers and lawyers have developed these kind of systems for several purposes, in particular for reasoning about *moral systems*. A severe drawback of *ought-to-be* systems is the existence of many paradoxes, contrary-to-duty paradoxes being among the most problematic (see [25] for a introduction). Perhaps the reader will have noted that in these logics we can nest deontic predicates, for example $\mathsf{O}(\mathsf{O}(\mathsf{P}(A)))$: *it is obligated that it is obligated that A is allowed*. These are hard to reason about and often counterintuitive.

When deontic logics began to be used in computer science, several authors pointed out that *ought-to-be* logics are not very useful for formalizing computing systems, and they focused on *ought-to-do* logics. In this case, the deontic predicates are on actions, and, as shown in several papers (see [10], [15], [3]), this approach can help us to avoid several paradoxes. In particular, nested deontic predicates disappear, and several *contrary-to-duty* paradoxes are not problematic in *ought-to-do* logics. Of course, we are not saying here that *ought-to-do* logics are better than *ought-to-be* logics; we are only pointing out the fact the former kinds of system are more appropriate for computer science.

The logic defined in this document is an *ought-to-do* logic, and therefore several paradoxes are avoided. Also we can remark that other dangerous paradoxes, like Ross's paradoxes, are not valid in our version of deontic logic. Ross's paradox can be enunciated as follows,

$$\mathsf{O}(\alpha) \rightarrow \mathsf{O}(\alpha \sqcup \beta)$$

Intuitively, *if you are obligated to send a letter, then you are obligated to send it or to burn it*. This paradox also has a version with the permission predicate:

$$\mathsf{P}(\alpha) \rightarrow \mathsf{P}(\alpha \sqcup \beta)$$

Both are not desirable in a computing system. We do not want that from: *you are allowed to write to a disk*, we can deduce: *you are allowed to write to a disk or to format it*!. It is not hard to see that both paradoxes are not valid in the logic defined.

Perhaps some words on *contrary-to-duty* paradoxes (or CTD structures for short) are useful. These kinds of structures are one of the most controversial in deontic logic; we are interested in them because the notion of recovery in fault-tolerant systems is embedded inside these structures. Consider the well-known example of *contrary-to-duty* structure:

1. It is forbidden for Smith to kill his mother.

2. If Smith kills his mother, he ought to kill her gently.

3. Smith kills her mother.

Usually, the problem in classic deontic logics is that we can deduce from this situation (using some kind of implication) that a derived obligation of 1 is *it is forbidden for Smith to kill his mother gently*, and this contradicts the conclusion that we can obtain from 2. Several solutions have been given for this case (see [3]), and we can solve it pointing out that different kinds of violations are involved in the situation. However, a lot of CTD structures are been proposed that do not involve actions ([25] proposes some of them). Note that a notion of recovery is embedded here; if we do not respect the primary obligation, then we have to *kill gently*. We can think of this action as a *recovery action*. We shall come back to CTD structures later, when we have more logical machinery at hand.

In addition, we can classify our action logic as *exogenous*; these kinds of logics express in an explicit way the structure of actions, as opposed to *endogenous* logics (for example, temporal logics) which assume implicit actions. It is not hard to see that *endogenous* logics are more suitable for specifying systems at a high level of abstraction, while *exogenous* logics are suitable for program verification.

4 Spicing up DPL with Time

We have defined the basic logic, but if we want a good logic for specifying computing systems, the dimension of time is needed. Several types of temporal logics have been used by computer scientists in recent decades. In this section we shall introduce a *Branching Time Logic*; this logic is very similar to CTL logic (see [8]), that is, we allow temporal predicates combined with quantifiers on paths. Although, CTL* logics are more expressive, they are much harder to axiomatize, as is shown in [23], we leave for further work a CTL* version of the logic presented here.

On the other hand, the temporization shown below is an Okhamist logic since formulae are evaluated with respect to an history and an instant, that is, we evaluate predicates using fixed traces. This semantics allows us to introduce the peculiar predicate $\mathsf{Done}()$, which can be used to predicate on the immediate past; this operator is mentioned in [10] and [16], but here we offer an axiomatization with some new axioms, and we show that mixing it with temporal notions allow us to express interesting properties. Other past operators are not described here, though the extension of the logic to support these is immediate.

First, we will present some changes to the definitions given before to be able to introduce time in the language. Let us define the modal formulae.

Definition 8 (Temporal Formulae)**.** *Given a DPL vocabulary $\langle \Phi_0, \Delta_0 \rangle$, the set of temporal deontic formulae (Φ_T) is defined as follows:*

- $\Phi_0 \subseteq \Phi_T$.
- $\top, \bot \in \Phi_T$.
- *if* $\alpha, \beta \in \Delta$, *then* $\alpha =_{act} \beta \in \Phi_T$
- *if* $\varphi_1, \varphi_2 \in \Phi_T$, *then* $\varphi_1 \rightarrow \varphi_2 \in \Phi_T$.
- *if* $\varphi \in \Phi_T$, *then* $\neg\varphi \in \Phi_T$.
- *if* $\varphi \in \Phi_T$ *with no temporal operators and* $\alpha \in \Delta$, *then* $\langle\alpha\rangle\phi \in \Phi_T$.

- *if* $\alpha \in \Delta$ *then* $\mathsf{P}(\alpha) \in \Phi_T$ *and* $\mathsf{P_w}(\alpha) \in \Phi_T$.
- *If* $\alpha \in \Delta$ *, then* $\mathsf{Done}(\alpha) \in \Phi_T$.
- *If* $\varphi_1, \varphi_2 \in \Phi$*, then* $\mathsf{AN}\varphi$, $\mathsf{AG}\varphi$, $\mathsf{A}(\varphi_1 \,\mathcal{U}\, \varphi_2)$, $\mathsf{E}(\varphi_1 \,\mathcal{U}\, \varphi_2) \in \Phi_T$.

The temporal operators are the classic ones in CTL logics; intuitively, the predicate $\mathsf{AN}\varphi$ means *in all possible executions* φ *is true at the next moment*, $\mathsf{AG}\varphi$ means *in all executions* φ *is always true*, $\mathsf{A}(\varphi_1 \,\mathcal{U}\, \varphi_2)$ means *for every possible execution* φ_1 *is true until* φ_2 *becomes true* and $\mathsf{E}(\varphi_1 \,\mathcal{U}\, \varphi_2)$ says there *exists some execution where* φ_1 *is true until* φ_2 *becomes true*. As usual, using these operators we can define the dual versions of them:

- $\mathsf{AF}\varphi \overset{\mathsf{def}}{\Longleftrightarrow} \mathsf{A}(\top \,\mathcal{U}\, \varphi)$.
- $\mathsf{EF}\varphi \overset{\mathsf{def}}{\Longleftrightarrow} \neg\mathsf{AG}\neg\varphi$.
- $\mathsf{EN}\varphi \overset{\mathsf{def}}{\Longleftrightarrow} \neg\mathsf{AN}\neg\varphi$.

For defining the semantics of the temporal version of the logic, we need some changes to the structures considered, as well needing to introduce the definition of traces. Firstly, we define the notion of initial states, then all the possibles traces of execution will be defined with respect to an initial state. Although, we have to remark that this is not an anchored logic as defined in [2].

Definition 9 (Temporal Model)**.** *Given a language* $L = \langle \Phi_0, \Delta_0 \rangle$*,* $M = \langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{I}, \mathcal{P}, w \rangle$ *is called a temporal structure, where:*

- $\langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{I}, \mathcal{P} \rangle$ *is a model as defined in definition 4.*
- $w \in W$ *is the initial state.*

■

Using the initial state we can consider all the traces (or paths) which start in this state. Note that some path could be finite; in this case we will transform it into an infinite path by adding a transition in the last state to itself, the label of this added transition being an event which is not in the model. Intuitively, this means that though the actual component cannot do any action, the environment continues running. With this purpose in mind, we define an universal set of events $\mathcal{E}_\mathsf{U}$, such that for every model $M = \langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{I}, \mathcal{P}, w \rangle$, $\mathcal{E} \subsetneq \mathcal{E}_\mathsf{U}$ holds.

Definition 10 (Traces)**.** *Given a model* $M = \langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{I}, \mathcal{P}, w \rangle$ *a trace is a (labelled) path* $s_0 \overset{e_0}{\rightarrow} s_1 \overset{e_1}{\rightarrow} s_2 \overset{e_2}{\rightarrow} ...$*, where for every* i*:* $s_i \overset{e_i}{\rightarrow} s_{i+1} \in \mathcal{R}$*, and* $s_0 = w$*. The set of all traces on* w *is* $\Sigma(w)$*.*

■

If $\pi = s_0 \overset{e_0}{\rightarrow} ... \overset{e_{n-1}}{\rightarrow} s_n$ is a finite trace, then we can transform this into an infinite one using an event $e' \notin \mathcal{E}$, as follows:

$$\pi = s_0 \overset{e_0}{\rightarrow} ... \overset{e_{n-1}}{\rightarrow} s_n \overset{e'}{\rightarrow} s_n \overset{e'}{\rightarrow} s_n \overset{e'}{\rightarrow} ...$$

That is, some external action is executed but it does not affect the local state. Actually, the selected external action does not matter, and we will see that a different selection of an external action does not affect the semantics. We will call the infinite trace the *completion* of the finite one.

Definition 11 (Infinite and Finite Traces)**.** *Given a model $M = \langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{I}, \mathcal{P}, w \rangle$, we call $\Sigma(w)^*$ the set of finite traces on w; the set of infinite traces is called $\Sigma^\infty(w)$, which consists of all infinite traces in the model plus the infinite traces that we can obtain from $\Sigma(w)^*$ by completion.*

We need some additional notation; given an infinite trace (or path) $\pi = s_0 \stackrel{e_0}{\rightarrow} s_1 \stackrel{e_1}{\rightarrow} s_2 \stackrel{e_2}{\rightarrow} ...$, we call $\pi^i = s_i \stackrel{e_i}{\rightarrow} s_{i+1} \stackrel{e_{i+1}}{\rightarrow} ...$ the subpath of π starting at position i. The notation $\pi_i = s_i$ is used to denote the i-th element in the path, and we write $\pi[i, j]$ (where $i \leq j$) for the subpath $s_i \stackrel{e_{i+1}}{\rightarrow} ... \stackrel{e_j}{\rightarrow} s_j$. Finally, given a finite path $\pi' = s'_0 \stackrel{e'_0}{\rightarrow} .... \stackrel{e'_n}{\rightarrow} s_n$, we say $\pi' \preceq \pi$ if π' is an initial subpath of π, that is: $s_i = s'_i$ and $e_i = e'_i$ for $0 \leq i \leq n$.

The relation $\vDash_{DTL}$ is defined using paths and models, and an interesting point to note is that we also use a given instant to evaluate formulae. It is needed since the predicate $\mathsf{Done}(-)$ allows us to predicate about the immediate past. Note that in the following definition we use the relation $\vDash$ defined in definition 5.

Definition 12 ($\vDash_{DTL}$)**.** *Given a model $M = \langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{I}, \mathcal{P}, w \rangle$, a trace $\pi = s_0 \stackrel{e_0}{\rightarrow} s_1 \stackrel{e_1}{\rightarrow} s_2 \stackrel{e_2}{\rightarrow} ... \in \Sigma^\infty(w)$, we define the relation $\vDash_{DTL}$ as follows:*

- $\pi, i, M \vDash_{DTL} \varphi \stackrel{\mathsf{def}}{\Longleftrightarrow} \pi_i, \langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{I}, \mathcal{P} \rangle \vDash \varphi$, *if φ does not contain any temporal predicates.*
- $\pi, i, M \vDash_{DTL} \neg\varphi \stackrel{\mathsf{def}}{\Longleftrightarrow}$ *not* $\pi, i, M \vDash_{DTL} \varphi$.
- $\pi, i, M \vDash_{DTL} \varphi_1 \rightarrow \varphi_2 \stackrel{\mathsf{def}}{\Longleftrightarrow}$ *either not* $\pi, i, M \vDash_{DTL} \varphi_1$ *or* $\pi, i, M \vDash_{DTL} \varphi_2$.
- $\pi, i, M \vDash_{DTL} \mathsf{Done}(\alpha) \stackrel{\mathsf{def}}{\Longleftrightarrow} i > 0$ *and* $e_{i-1} \in \mathcal{I}(\alpha)$.
- $\pi, i, M \vDash_{DTL} \mathsf{AN}\varphi \stackrel{\mathsf{def}}{\Longleftrightarrow} \forall \pi' \in \Sigma^\infty(w)$ *such that* $\pi[0, i] \preceq \pi'$ *we have that* $\pi', i + 1, M \vDash_{DTL} \varphi$.
- $\pi, i, M \vDash_{DTL} \mathsf{AG}\varphi \stackrel{\mathsf{def}}{\Longleftrightarrow} \forall \pi' \in \Sigma^\infty(w)$ *such that* $\pi[0, i] \preceq \pi'$ *we have that* $\forall j \geq i : \pi', j, M \vDash_{DTL} \varphi$.
- $\pi, i, M \vDash_{DTL} \mathsf{A}(\varphi_1 \,\mathcal{U}\, \varphi_2) \stackrel{\mathsf{def}}{\Longleftrightarrow} \forall \pi' \in \Sigma^\infty(w)$ *such that* $\pi[0, i] \preceq \pi'$ *we have that* $\exists j \geq i : \pi', j, M \vDash_{DTL} \varphi_2$ *and* $\forall i \leq k < j : \pi', k, M \vDash_{DTL} \varphi_1$
- $\pi, i, M \vDash_{DTL} \mathsf{E}(\varphi_1 \,\mathcal{U}\, \varphi_2) \stackrel{\mathsf{def}}{\Longleftrightarrow} \exists \pi' \in \Sigma^\infty(w)$ *such that* $\pi[0, i] \preceq \pi'$, *we have that* $\exists j \geq i : \pi', j, M \vDash_{DTL} \varphi_2$ *and* $\forall i \leq k < j : \pi', k, M \vDash_{DTL} \varphi_1$

■

We say that $M \vDash_{DTL} \varphi$ if $\pi, i, M \vDash_{DTL} \varphi$ for all paths π and instants i. And we say $\vDash_{DTL} \varphi$ if φ holds for all models M. Note that this notion of validity is not anchored (as defined in [2]), in other words, we consider all the instants of a trace to say that a given formula is satisfied by a path. Anchored temporal logics have some advantages and some drawbacks; we feel that, for a simpler presentation of our ideas, a non anchored logic is more convenient. Next we present an axiomatic system for the semantics just described; some axioms are the classic ones for CTL and the others allow us to exploit the relationship between modal and temporal operators.

Definition 13 (DTL Axioms)**.** *Given a vocabulary $\langle \Phi_0, \Delta_0 \rangle$, the axiomatic system is composed of the (substitutions of) the following axioms:*

- *All the axioms given in definition 6.*

TempAx1. $\langle \mathbf{U} \rangle \top \rightarrow (\mathsf{AN}\varphi \leftrightarrow [\mathbf{U}]\varphi)$

TempAx2. $[\mathbf{U}]\bot \rightarrow (\mathsf{AN}\varphi \leftrightarrow \varphi)$

TempAx3. $\mathsf{EN}(\varphi \vee \psi) \leftrightarrow (\mathsf{EN}\varphi) \vee (\mathsf{EN}\psi)$

TempAx4. $\mathsf{AN}\varphi \leftrightarrow \neg\mathsf{EN}\neg\varphi$

TempAx5. $\mathsf{E}(\varphi\, \mathcal{U}\, \psi) \leftrightarrow \psi \vee (\varphi \wedge \mathsf{ENE}(\varphi\, \mathcal{U}\, \psi))$

TempAx6. $\mathsf{A}(\varphi\, \mathcal{U}\, \psi) \leftrightarrow \psi \vee (\varphi \wedge \mathsf{ANA}(\varphi\, \mathcal{U}\, \psi))$

TempAx7. $[\alpha]\mathsf{Done}(\alpha)$

TempAx8. $[\overline{\alpha}]\neg\mathsf{Done}(\alpha)$

TempAx9. $\neg\mathsf{Done}(\emptyset)$

TempAx10. $\neg\mathsf{Done}(\mathbf{U}) \rightarrow \neg\mathsf{Done}(\alpha)$

And the following deduction rules:

- *Rules given in definition* 6.

TempRule1. $\{\neg\mathsf{Done}(\mathbf{U}) \rightarrow \varphi, \varphi \rightarrow \mathsf{AN}\varphi\} \vdash \varphi$

TempRule2. $\{\varphi\} \vdash \mathsf{AG}\varphi$

TempRule3. $\{\varphi \rightarrow (\neg\psi \wedge \mathsf{EN}\varphi)\} \vdash \varphi \rightarrow \neg\mathsf{A}(\vartheta\, \mathcal{U}\, \psi)$

TempRule4. $\{\varphi \rightarrow (\neg\psi \wedge \mathsf{AN}(\varphi \vee \neg\mathsf{E}(\vartheta\, \mathcal{U}\, \psi)))\} \vdash \varphi \rightarrow \neg\mathsf{E}(\vartheta\, \mathcal{U}\, \psi)$

■

Some comments will be useful; note that the formula $\neg\mathsf{Done}(\mathbf{U})$ holds only at the beginning of each trace, and therefore we can think of this as asserting that the actual instant is the beginning. Axioms **TempAx1** and **TempAx2** relate the box modal operator with the temporal operators, reflecting the semantics introduce above. On the other hand, **TempAx3 - TempAx6** are classic axioms for CTL logic (given at [9]). Axioms **TempAx7-TempAx10** define the $\mathsf{Done}(-)$ operator, mainly using the box modality. The first inference rules is a kind of induction rule, it says: *if something is true at the beginning, and it is preserved by every action, we can deduce that it holds everywhere*. The other rules are standard for temporal logics.

We prove some useful theorems of this system. Note that since the CTL system is embedded in the axiomatic system given in definition 13, we can derive all the CTL theorems; we only focus on the new ones.

Theorem 8. $\{\neg\mathsf{Done}(\mathbf{U}) \rightarrow \varphi, \mathsf{AN}\varphi\} \vdash \varphi$
Proof.

1- $\mathsf{AN}\varphi$	*Hyp.*
2- $\varphi \rightarrow \mathsf{AN}\varphi$	*PL, 1*
3- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow \varphi$	*Hyp.*
4- φ	***TempRule1****, 2, 3*

■

Corollary 4. $\{\neg\mathsf{Done}(\mathbf{U}) \rightarrow \varphi, [\mathbf{U}]\varphi\} \vdash \varphi$
Proof. *The proof is by cases:*
Case 1: $\langle\mathbf{U}\rangle\top$

1- $\langle U\rangle\top$	*Hyp.*
2- $\mathsf{AN}\varphi \leftrightarrow [\mathbf{U}]\varphi$	*MP,* ***TempAx1****, 1*
3- $[\mathbf{U}]\varphi$	*Hyp.*
4- $\mathsf{AN}\varphi$	*MP, 3, 2*
5- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow \varphi$	*Hyp.*
6- φ	*Theorem 8, 4, 5*

Case 2: $[\mathbf{U}]\bot$

1- $[\mathbf{U}]\bot$	*Hyp.*
2- $\mathsf{AN}\varphi \leftrightarrow \varphi$	*MP,* ***TempAx2****, 1*
3- $\varphi \rightarrow \varphi$	*PL*
4- $\varphi \rightarrow \mathsf{AN}\varphi$	*PL, 3, 2*
5- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow \varphi$	*Hyp.*
6- φ	***TempRule1****, 4, 5*

■

Both theorem 8 and corollary 4 are two different formulations of the induction principle **TempRule1**. The next theorem allows us to characterize deadlock: *when no action is enabled, we stay in this state forever.* This property is established in [13] as an axiom. During the proof we will use some theorems of CTL; the interested reader can consult them in [8].

Theorem 9. . $[\mathbf{U}]\bot \wedge \varphi \rightarrow \mathsf{AN}([\mathbf{U}]\bot \wedge \varphi)$
Proof.

1- $[\mathbf{U}]\bot \rightarrow (\mathsf{AN}\varphi \leftrightarrow \varphi)$	***TempAx2***
2- $[\mathbf{U}]\bot \wedge \varphi \rightarrow \mathsf{AN}\varphi$	*PL, 1*
3- $[\mathbf{U}]\bot \rightarrow ((\mathsf{AN}[\mathbf{U}]\bot) \leftrightarrow [\mathbf{U}]\bot)$	***TempAx2***
4- $[\mathbf{U}]\bot \rightarrow \mathsf{AN}[\mathbf{U}]\bot$	*PL, 3*
5- $[\mathbf{U}]\bot \wedge \varphi \rightarrow (\mathsf{AN}[\mathbf{U}]\varphi \wedge \mathsf{AN}\varphi)$	*PL,2,4*
6- $[\mathbf{U}]\bot \wedge \varphi \rightarrow \mathsf{AN}([\mathbf{U}]\bot \wedge \varphi)$	*PL, CTL property.*

■

Now, we can prove some important properties about the $\mathsf{Done}(-)$ operator.

Theorem 10 ($\mathsf{Done}(-)$ properties)**.** *The following are theorems of the axiomatic system defined above.*

T13. $\mathsf{Done}(\alpha) \wedge \alpha \sqsubseteq \alpha' \rightarrow \mathsf{Done}(\alpha')$

T14. $\mathsf{Done}(\alpha \sqcup \beta) \rightarrow \mathsf{Done}(\alpha) \vee \mathsf{Done}(\beta)$

T15. $\mathsf{Done}(\alpha \sqcap \beta) \leftrightarrow \mathsf{Done}(\alpha) \wedge \mathsf{Done}(\beta)$

T16. $\mathsf{Done}(\alpha \sqcup \beta) \wedge \mathsf{Done}(\overline{\alpha}) \rightarrow \mathsf{Done}(\beta)$

T17. $[\alpha]\varphi \wedge [\beta]\mathsf{Done}(\alpha) \rightarrow [\beta]\varphi$

Proof.

T13.*: We use the induction theorem for proving this property.*

Base Case:

1- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow (\neg\mathsf{Done}(\alpha) \wedge \neg\mathsf{Done}(\alpha'))$	***TempAx10.***
2- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow (\neg\mathsf{Done}(\alpha') \rightarrow \neg\mathsf{Done}(\alpha))$	*PL, 1*
3- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow (\mathsf{Done}(\alpha) \rightarrow \mathsf{Done}(\alpha'))$	*PL, 2*
4- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow (\mathsf{Done}(\alpha) \wedge \alpha \sqsubseteq \alpha' \rightarrow \mathsf{Done}(\alpha'))$	*PL, 3*

Ind.Case. We use th BA deduction theorem and suppose: $\alpha \sqsubseteq \alpha'$.

1- $[\alpha]\mathsf{Done}(\alpha)$	***TempAx7***
2- $[\alpha']\mathsf{Done}(\alpha')$	***TempAx7***
3- $[\alpha]\mathsf{Done}(\alpha) \wedge [\alpha']\mathsf{Done}(\alpha') \rightarrow [\alpha \sqcap \alpha'](\mathsf{Done}(\alpha) \wedge \mathsf{Done}(\alpha'))$	*T5*
4- $[\alpha \sqcap \alpha'](\mathsf{Done}(\alpha) \wedge \mathsf{Done}(\alpha'))$	*MP, 1, 2, 3*
5- $[\alpha]\mathsf{Done}(\alpha') \wedge \alpha \sqsubseteq \alpha' \rightarrow [\alpha]\mathsf{Done}(\alpha')$	*T3*
6- $\alpha \sqsubseteq \alpha'$	*Hyp.*
7- $[\alpha]\mathsf{Done}(\alpha')$	*PL, 2, 5, 6*
8- $[\alpha]\mathsf{Done}(\alpha) \rightarrow [\alpha]\mathsf{Done}(\alpha')$	*PL, 1, 7*
9- $[\alpha](\mathsf{Done}(\alpha) \rightarrow \mathsf{Done}(\alpha'))$	*ML, 8*
10-$[\overline{\alpha}]\neg\mathsf{Done}(\alpha)$	***TempAx8***
11-$[\overline{\alpha}](\neg\mathsf{Done}(\alpha) \vee \mathsf{Done}(\alpha'))$	*PL, 10*
12-$[\overline{\alpha}](\mathsf{Done}(\alpha) \rightarrow \mathsf{Done}(\alpha'))$	*PL,11*
13-$[\alpha]((\mathsf{Done}(\alpha) \rightarrow \mathsf{Done}(\alpha')) \wedge ([\overline{\alpha}](\mathsf{Done}(\alpha) \rightarrow \mathsf{Done}(\alpha'))) \rightarrow [\alpha \sqcup \alpha'](\mathsf{Done}(\alpha) \rightarrow \mathsf{Done}(\alpha'))$	*T4*
14-$[\alpha \sqcup \overline{\alpha}](\mathsf{Done}(\alpha) \rightarrow \mathsf{Done}(\alpha'))$	*PL, 9, 12, 13*
15-$[\mathbf{U}](\mathsf{Done}(\alpha) \rightarrow \mathsf{Done}(\alpha'))$	*BA, 14*
16-$(\mathsf{Done}(\alpha) \rightarrow \mathsf{Done}(\alpha')) \rightarrow [\mathbf{U}](\mathsf{Done}(\alpha) \rightarrow \mathsf{Done}(\alpha'))$	*PL, 15*

□

T14.: By induction:

Base Case:

1- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow \neg\mathsf{Done}(\alpha \sqcup \beta)$	***TempAx10***
2- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow \neg\mathsf{Done}(\alpha)$	***TempAx10***
3- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow \neg\mathsf{Done}(\beta)$	***TempAx10***
4- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow ((\neg\mathsf{Done}(\alpha) \wedge \neg\mathsf{Done}(\beta)) \rightarrow \neg\mathsf{Done}(\alpha \sqcup \beta))$	*PL, 1, 2, 3*
5- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow (\mathsf{Done}(\alpha \sqcup \beta) \rightarrow \mathsf{Done}(\alpha) \vee \mathsf{Done}(\beta))$	*PL, 4*

Ind.Case:

1- $[\alpha \sqcup \beta]\mathsf{Done}(\alpha \sqcup \beta)$	***TempAx10***
2- $[\alpha]\mathsf{Done}(\alpha)$	***TempAx10***
3- $[\beta]\mathsf{Done}(\beta)$	***TempAx10***
4- $[\alpha]\mathsf{Done}(\alpha) \wedge [\beta]\mathsf{Done}(\beta) \rightarrow [\alpha \sqcup \beta](\mathsf{Done}(\alpha) \vee \mathsf{Done}(\beta))$	*T4*
5- $[\alpha \sqcup \beta](\mathsf{Done}(\alpha) \vee \mathsf{Done}(\beta))$	*PL, 2, 3, 4*
6- $[\alpha \sqcup \beta](\mathsf{Done}(\alpha \sqcup \beta)) \rightarrow [\alpha \sqcup \beta](\mathsf{Done}(\alpha) \vee \mathsf{Done}(\beta))$	*PL, 1, 5*
7- $[\alpha \sqcup \beta](\mathsf{Done}(\alpha \sqcup \beta) \rightarrow \mathsf{Done}(\alpha) \vee \mathsf{Done}(\beta))$	*ML, 6*
8- $[\overline{\alpha}]\neg\mathsf{Done}(\alpha) \rightarrow [\overline{\alpha} \sqcap \overline{\beta}]\neg\mathsf{Done}(\alpha)$	*A5*
9- $[\overline{\beta}]\neg\mathsf{Done}(\beta) \rightarrow [\overline{\alpha} \sqcap \overline{\beta}]\neg\mathsf{Done}(\beta)$	*A5*
10-$[\overline{\alpha} \sqcap \overline{\beta}]\neg\mathsf{Done}(\alpha) \wedge [\overline{\alpha} \sqcap \overline{\beta}]\neg\mathsf{Done}(\beta) \rightarrow [\overline{\alpha} \sqcap \overline{\beta}]\neg\mathsf{Done}(\alpha \sqcup \beta)$	*PL, 8, 9, 1*
11-$[\overline{\alpha} \sqcap \overline{\beta}](\neg\mathsf{Done}(\alpha) \wedge \neg\mathsf{Done}(\beta) \rightarrow \neg\mathsf{Done}(\alpha \sqcup \beta))$	*ML, 10*
12-$[\overline{\alpha \sqcup \beta}](\mathsf{Done}(\alpha \sqcup \beta) \rightarrow \mathsf{Done}(\alpha) \vee \mathsf{Done}(\beta))$	*PL, 11*
13-$[\mathbf{U}](\mathsf{Done}(\alpha \sqcup \beta) \rightarrow \mathsf{Done}(\alpha) \vee \mathsf{Done}(\beta))$	*BA, 12, 7*

□ ***T15.***$\rightarrow$):

1- $\mathsf{Done}(\alpha \sqcap \beta) \wedge \alpha \sqcap \beta \sqsubseteq \alpha \rightarrow \mathsf{Done}(\alpha)$	*T13*
2- $\alpha \sqcap \beta \sqsubseteq \alpha$	*Def.*$\sqsubseteq$
3- $\mathsf{Done}(\alpha \sqcap \beta) \rightarrow \mathsf{Done}(\alpha)$	*PL, 1, 2*
4- $\mathsf{Done}(\alpha \sqcap \beta) \wedge \alpha \sqcap \beta \sqsubseteq \beta \rightarrow \mathsf{Done}(\beta)$	*T3*
5- $\alpha \sqcap \beta \sqsubseteq \beta$	*Def.*$\sqsubseteq$
6- $\mathsf{Done}(\alpha \sqcap \beta) \rightarrow \mathsf{Done}(\beta)$	*PL, 4, 5*
7- $\mathsf{Done}(\alpha \sqcap \beta) \rightarrow \mathsf{Done}(\alpha) \wedge \mathsf{Done}(\beta)$	*PL, 3, 6*

$\leftarrow$): *By induction:*
Base Case:

1- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow \neg\mathsf{Done}(\alpha)$	***TempAx10***
2- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow \neg\mathsf{Done}(\beta)$	***TempAx10***
3- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow \neg\mathsf{Done}(\alpha \sqcap \beta)$	***TempAx10***
4- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow (\mathsf{Done}(\alpha) \wedge \mathsf{Done}(\beta) \rightarrow \mathsf{Done}(\alpha \sqcap \beta))$	*PL, 2, 3*

Ind. Case:

1- $[\alpha]\mathsf{Done}(\alpha) \wedge \alpha \sqcap \beta \sqsubseteq \alpha \rightarrow [\alpha \sqcap \beta]\mathsf{Done}(\alpha)$	*T3*
2- $[\alpha]\mathsf{Done}(\alpha)$	***TempAx7***
3- $\alpha \sqcap \beta \sqsubseteq \alpha$	*Def.* $\sqsubseteq$
4- $[\alpha \sqcap \beta]\mathsf{Done}(\alpha)$	*PL, 1, 2 ,3*
5- $[\beta]\mathsf{Done}(\beta)$	*T3*
6- $[\beta]\mathsf{Done}(\beta) \wedge \alpha \sqcap \beta \sqsubseteq \beta \rightarrow [\alpha \sqcap \beta]\mathsf{Done}(\beta)$	***TempAx7***
7- $\alpha \sqcap \beta \sqsubseteq \beta$	*Def.*$\sqsubseteq$
8- $[\alpha \sqcap \beta]\mathsf{Done}(\beta)$	*PL,5, 6, 7*
9- $[\alpha \sqcap \beta]\mathsf{Done}(\alpha \sqcap \beta)$	***TempAx7***
10-$[\alpha \sqcap \beta]\mathsf{Done}(\alpha) \wedge [\alpha \sqcap \beta]\mathsf{Done}(\beta) \rightarrow [\alpha \sqcap \beta]\mathsf{Done}(\alpha \sqcap \beta)$	*PL, 4, 8, 9*
11-$[\alpha \sqcap \beta](\mathsf{Done}(\alpha) \wedge \mathsf{Done}(\beta) \rightarrow \mathsf{Done}(\alpha \sqcap \beta))$	*ML, 10*
12-$[\overline{\alpha}]\neg\mathsf{Done}(\alpha)$	***TempAx8***
13-$[\overline{\beta}]\neg\mathsf{Done}(\beta)$	***TempAx8***
14-$[\overline{\alpha} \sqcup \overline{\beta}](\neg\mathsf{Done}(\alpha) \vee \neg\mathsf{Done}(\beta))$	*PL&T4, 12, 13*
15-$[\overline{\alpha \sqcap \beta}]\neg(\mathsf{Done}(\alpha) \wedge \mathsf{Done}(\beta))$	*PL & BA*
16-$[\overline{\alpha \sqcap \beta}](\mathsf{Done}(\alpha) \wedge \mathsf{Done}(\beta) \rightarrow \mathsf{Done}(\alpha \sqcap \beta))$	*PL, 15*
17-$[\mathbf{U}](\mathsf{Done}(\alpha) \wedge \mathsf{Done}(\beta) \rightarrow \mathsf{Done}(\alpha \sqcap \beta))$	*PL& A4, 11, 16*

□
T16:

1- $\mathsf{Done}(\alpha \sqcup \beta) \wedge \mathsf{Done}(\overline{\alpha}) \rightarrow \mathsf{Done}((\alpha \sqcup \beta) \sqcap \overline{\alpha})$	*T15*
2- $\mathsf{Done}(\alpha \sqcup \beta) \wedge \mathsf{Done}(\overline{\alpha}) \rightarrow \mathsf{Done}(\beta)$	*BA*

T17:

1- $[\alpha]\mathsf{Done}(\alpha)$	***TempAx10***
2- $([\alpha]\varphi) \leftrightarrow ([\alpha](\mathsf{Done}(\alpha) \rightarrow \varphi))$	*ML, 1*
3- $([\alpha](\mathsf{Done}(\alpha) \rightarrow \varphi) \wedge [\beta]\mathsf{Done}(\alpha)) \rightarrow [\alpha \sqcup \beta]((\mathsf{Done}(\alpha) \rightarrow \varphi) \vee \mathsf{Done}(\alpha))$	*T4*
4- $([\alpha](\mathsf{Done}(\alpha) \rightarrow \varphi) \wedge [\beta]\mathsf{Done}(\alpha)) \rightarrow [\alpha \sqcup \beta]\varphi$	*PL, 3*
5- $[\alpha \sqcup \beta]\varphi \rightarrow [\beta]\varphi$	*A4*
6- $([\alpha](\mathsf{Done}(\alpha) \rightarrow \varphi) \wedge [\beta]\mathsf{Done}(\alpha)) \rightarrow [\beta]\varphi$	*PL, 4, 5*
7- $([\alpha]\varphi \wedge [\beta]\mathsf{Done}(\alpha)) \rightarrow [\beta]\varphi$	*PL, 2, 6*

■

5 An Example: Diarrheic Philosophers

Here we take the classical example of dining philosophers of Dijkstra ([6]), but we add some features to it, such that every process has the possibility of crashing for an indefinite amount of time. The example allows us to show how the logic described in earlier sections can be used to specify a system, and moreover to predicate about fault behavior, and fault tolerance. With this goal in mind, we exhibit some properties of the described model, for example, that certain types of faults are more benign than other ones, in the sense that the first kind of faults enable some type of progress, while the latter ones do not.

Example 1. *In some college, a fixed number of philosophers are dedicated to thinking about different problems. Because each philosopher must eat to survive, the college has a (circular) table which contains a big bowl of pasta. Each philosopher has a seat and two forks, one for each hand. But because of budgetary reasons neighbouring philosophers have to share forks. In addition, the pasta could be contaminated (usually philosophers think for a long time and so the pasta may develop some dangerous bacteria, probably because they should share forks!) and therefore philosophers could get a stomach ache and then they must goes to the bathroom. The problem, of course, is when a philosopher goes to the bathroom with some forks in his hands. A philosopher may come back, or not, from the bathroom (the details are left to the reader's imagination).*

Solution:

The main point to make about the following specification is the way in which the possible faults (i.e., when a given philosopher goes to the bathroom) are modeled. We model this scenario as a system violation; we will see that two possible violations can be defined. First, if a philosopher goes to the bath with two forks in his hands, this is obviously the worst situation that could happen (this situation will be modeled with the predicate $v_1 \wedge \neg v_2$). The other is when a philosopher only takes one fork with him (we use the predicate v_2 to model this); in this case, we will prove that undesirable blocking is not possible. Additionally, we will see that when violation v_1 upgrades to a violation v_2 the system can avoid suffering undesired blocking. Using this specification, a very interesting set of properties can be proved, some of them will be shown later.

Of course, we intend that that this specification to be used in a broader context, where a given fault-tolerant program can be checked against the given specification and, therefore, we can deduce some program properties.

Firstly, the atomic predicates in our example are the following:

$$\Phi_0 = \{i.thk, i.eating, i.hungry, i.bathroom, i.has_L, i.has_R, fork_i.up, fork_i.down\} \quad 0 \leq i \leq n$$

Where $i.eating$ tells us if philosopher i is eating, $i.hungry$ if the philosopher is hungry, $i.bathroom$ will be true when i is in the bathroom, $i.has_L$ and $i.has_R$ allow us to know if i has the left or right fork in his hand. And $fork_i.up$, $fork_i.down$ will be true if fork i is up or down, respectively.

The atomic actions are the following:

$$\Delta_0 = \{i.up_L, i.up_R, i.down_R, i.down_L, i.getbad, i.getthk, i.gethungry\}$$

Recall that $0 \leq i \leq n$, for some n. Intuitively, $i.up_L$ ($i.up_R$) is the action of philosopher i picking up the left (right) fork; $i.down_L$ and $i.down_R$ are the inverses. On the other hand, $i.getbad, i.getthk$ and $i.gethungry$ are executed when philosopher i gets sick, gets thinking or gets hungry, respectively.

We have several axioms, but note that many of them are frame axioms (i.e., they express usually implicit restrictions); these axioms can be avoided if a more abstract language of specification is used, and then translated to our logic.

Let us establish the initial conditions:

Phil1. $\neg\mathsf{Done}(\mathbf{U}) \rightarrow (\bigwedge_{0\leq i\leq n} i.thk) \wedge (\bigwedge_{0\leq i\leq n} fork_i.down) \wedge (\bigwedge_{0\leq i\leq n} \neg i.v_1 \wedge \neg i.v_2)$

That is, in the initial state all the philosophers are thinking, all the forks are down and there are no violations. We also need to express that some states are disjoint:

Phil2. $fork_i.down \veebar fork_i.up$
Phil3. $i.eating \veebar i.thk \veebar i.hungry \veebar i.bathroom$
Phil4. $(i.thk \vee i.bathroom) \rightarrow [i.up_L \sqcup i.up_R]\bot$

Here the symbol $\veebar$ denotes the strict version of $\vee$. Axiom **Phil2** says that each fork can either be up or down but not both. **Phil3** is similar but expressing a disjoint condition on philosopher states. **Phil4** says that a thinking philosopher cannot take any fork.

Phil5. $([i.up_R]i.has_R) \wedge ([i.up_L]i.has_L)$
$\wedge([i.down_L]\neg i.has_R) \wedge ([i.down_L]\neg i.has_L)$
Phil6. $(\neg i.has_R \rightarrow [\overline{i.up_R}]\neg i.has_R)$
$\wedge(\neg i.has_L \rightarrow [\overline{i.up_L}]\neg i.has_L)$
$\wedge(\neg i.has_L \rightarrow [i.down_L]\bot)$
$\wedge(\neg i.has_R \rightarrow [i.down_R]\bot)$
Phil7. $(i.has_L \rightarrow [i.getthk]\mathsf{Done}(i.down_L))$
$\wedge(i.has_R \rightarrow [i.getthk]\mathsf{Done}(down_R))$
Phil8. $([i.getthk]i.thk) \wedge (\neg i.thk \rightarrow [\overline{i.getthk}]\neg i.thk)$

The first axiom models the behavior of the $i.up$ and $i.down$ actions. On the other hand, axiom **Phil6** is a frame axiom; it says that only the actions $i.down_L$ and $i.down_R$ can modify the predicates $i.has_L$ and $i.has_R$. **Phil7** and **Phil8** are similar but they model the behavior of the $i.getthk$ action.

Phil9. $fork_i.up \rightarrow [i.up_L \sqcup (i+1).up_R]\bot$
Phil10. $fork_i.up \leftrightarrow ((i+1).has_R \vee i.has_L)$

Axiom **Phil9** expresses that if a fork is up then none of the corresponding philosophers can take it (here "+" denotes the addition module $n+1$). The other formula establishes that a fork i is up if and only if the philosopher i, or $i+1$, has it in his hands.

We also need to predicate that two neighboring philosophers cannot take the same fork at the same time; this is expressed by the following axiom:

Phil11. $i.up_L \sqcap (i+1).up_R = \emptyset$

In an implementation we should use a semaphore to ensure this requirement.

The next set of axioms models the behavior of the $i.gethungry$ action.

Phil12. $(i.gethungry \sqcap (i+1).gethungry) = \emptyset$
Phil13. $i.thk \wedge \neg(i-1).hungry \wedge \neg(i+1).hungry \wedge$
$fork_i.down \wedge fork_{i+1}.down \rightarrow \langle i.gethungry\rangle\top$
Phil14. $(i.thk \wedge (((i-1).hungry \vee (i+1).hungry) \vee$
$((i-1).v_1 \wedge i.has_L) \vee ((i+1).v_1 \wedge i.has_R))) \rightarrow [i.gethungry]\bot$
Phil15. $i.hungry \wedge (fork_i.down \vee i.has_L) \wedge (fork_{i+1}.down \vee i.has_R) \rightarrow \mathsf{AN}i.eating$
Phil16 $i.hungry \wedge ((i+1).v_1 \wedge (i+1).has_R) \wedge ((i-1).v_1 \wedge (i-1).has_L) \rightarrow \mathsf{AN}i.thk$
Phil17. $([i.gethungry]i.hungry) \wedge (\neg i.hungry \rightarrow [\overline{i.gethungry}]\neg i.gethungry)$

The first formulae establish that no two neighbor philosophers can get hungry at the same time; if we allow concurrency here, it will give us some problems. Again some mechanism for mutual exclusion is needed in the implementation. **Phil14** expresses that if some philosopher is getting hungry and some neighbour is already in that state, the philosopher has to wait. Obviously, this specification exhibits an starvation problem; to avoid this a priority queue is needed. For simplicity we do not deal with this problem in this example.

Phil13 tell us when a philosopher will have the possibility of getting hungry. Axiom **Phil15** says that if a philosopher is hungry and he can take both forks, then he will start to eat. The last axioms in this set specify the behavior of $i.gethungry$ and some frame conditions.

The following set of sentences specify when a philosopher is eating,

Phil18. $i.eating \leftrightarrow (i.has_L \wedge i.has_R \wedge \neg i.bathroom)$
Phil19. $i.eating \rightarrow [i.up_L \sqcup i.up_R \sqcup i.getbetter \sqcup i.gethungry]\bot$
Phil20. $i.eating \rightarrow [\mathbf{U}]\mathsf{Done}(i.getthk \sqcup i.getbad)$
Phil21. $i.eating \leftrightarrow \mathsf{O}(i.down_L \sqcap i.down_R)$

Axiom **Phil18** says that a philosopher is eating iff he has both forks and he is not in the bathroom. Axiom **Phil19** restricts the actions that can be done when a philosopher is eating. **Phil20** says if a philosopher is eating, then he can only start to thinking again or getting sick. And the last axiom establishes an obligation about the release of the forks. Some explanation is needed about this obligation; note that the deontic predicate here (**Phil21**) says what must happen in an ideal world, but perhaps it can be violated. The important point here is that the obligation predicate allow us to differentiate an ideal scenario from one that is not, and this is a strong benefit of deontic logic.

The $i.getbad$ action can be modeled as follows:

Phil22. $([i.getbad]i.bathroom) \wedge (\neg i.bathroom \rightarrow [\overline{i.getbad}]\neg i.bathroom)$
Phil23. $(i.bathroom \rightarrow [i.getthk]i.thk) \wedge i.bathroom \rightarrow [\overline{i.getthk}]i.bathroom$

Here we have the basic axioms which define the behavior of $i.getbad$. We note axiom **Phil23** which says that if a philosopher gets better then he has to free those forks that he has in his hands. After he coming better, a philosopher goes to the thinking state.

Finally, we present a collection of axioms for modeling the notion of violation. The predicates $i.v_1, i.v_2$ are used with this purpose; $i.v_1$ becomes true when philosopher i (after eating) does not release both forks. $i.v_2$ is a refinement of $i.v_1$; it is true when philosopher i only releases one fork, but he holds onto the remaining fork. These variables allow us to reason about which situations an undesirable blocking becomes possible and when we can avoid it. Interesting, we can predicate about recovery from bad scenarios (for example, when $i.v_1$ "upgrades" to $i.v_2$). The axioms are:

V1. $\neg i.v_1 \wedge \mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow ([\overline{i.down_L \sqcap i.down_R}]i.v_1) \wedge ([i.down_L \sqcap i.down_R]\neg i.v_1)$
V2. $\neg i.v_1 \wedge \neg\mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow [\mathbf{U}]\neg i.v_1$
V3. $i.v_2 \leftrightarrow i.v_1 \wedge (\neg i.has_L \veebar \neg i.has_R)$
V4. $(i.v_1 \rightarrow [i.down_L \sqcap i.down_R]\neg i.v_1) \wedge (i.v_1 \wedge \neg i.v_2 \rightarrow [\overline{i.down_L \sqcap i.down_R}]i.v_1)$
V5. $i.v_2 \rightarrow [\overline{i.down_L \sqcup i.down_R}]i.v_2$
V6. $((i.v_2 \wedge \neg i.has_L) \rightarrow [i.down_R]\neg(\neg i.v_2 \wedge \neg i.v_1)) \wedge ((i.v_2 \wedge \neg i.has_R) \rightarrow [i.down_R](\neg i.v_2 \wedge \neg i.v_1))$

The first axiom defines when $i.v_1$ can become true, that is, when philosopher i is obligated to put down both forks but he does not do that; otherwise $i.v_1$ is false. **V2** is needed to say that the other actions do not affect the variable $i.v_1$. **V3** defines $i.v_2$; it is true if only if the philosopher has only one fork and violation $i.v_1$ is true. Intuitively, this occur when a philosopher only

takes one fork with him to the bathroom, or perhaps if he puts down one fork after getting in a violation state. **V4** establishes that down both forks is a recovery action for $i.v_2$, and, if we are not in a violation of type $i.v_2$, then doing something different will leave us in the same violation. This has the important implication that when we are in a violation of type $i.v_2$, we can recovery from $i.v_1$ and $i.v_2$; exactly this is specified by axiom **V6**.

5.1 Some Properties

In this section we will prove some important properties about the specification given above. We show some important lemmas, which allow us to modularize the proofs and we leave their (long) proofs for the appendix. After of that, we shall prove a property which tells us that if some philosopher i is always in a violation $i.v_1$, then none of his neighbors will eat.

Our first lemma establishes that no two neighbouring philosophers can have the same fork at the same time.

Lemma 3. $\mathbf{Phil} \vdash \neg((i+1).has_R \wedge i.has_L)$
□

The second lemma says that if a philosopher is in a violation, then he has some fork in his hands:

Lemma 4. $\mathbf{Phil} \vdash i.v_1 \rightarrow (i.has_R \vee i.has_L)$
□

The next lemma tells us that if it is not the case of philosopher i being in a violation then in the next state either he will not be in a violation or he will not be eating. That is, if a philosopher goes into a violation, then he cannot be eating.

Lemma 5. $\mathbf{Phil} \vdash \neg i.v_1 \rightarrow [\mathbf{U}](\neg v_1 \vee \neg i.eating)$
□

The following lemma characterizes, in some way, the relationship between violations $i.v_1$ and $i.v_2$: if a philosopher is in violation $i.v_1$ and not in $i.v_2$ (that is, he has both forks), then no neighbor philosopher of him can be eating.

Lemma 6. $\mathbf{Phil} \vdash i.v_1 \wedge \neg i.v_2 \rightarrow \neg(i+1).eating \wedge \neg(i-1).eating$
□

The next lemma says that if a philosopher is not eating, then in the next state he is not eating or he is not in a violation.

Lemma 7. $\mathbf{Phil} \vdash \neg i.eating \rightarrow [\mathbf{U}](\neg i.v_1 \vee \neg i.eating)$
□

The following lemma tell us that if a philosopher is in a violation, then he cannot be eating.

Lemma 8. $\mathbf{Phil} \vdash i.v_1 \rightarrow \neg i.eating$
□

It seems obvious that if a philosopher is in a violation then he is in the bathroom, the following lemma formalizes that intuition.

Lemma 9. $\mathbf{Phil} \vdash i.v_1 \rightarrow i.bathroom$
□

If a philosopher is thinking then he has neither the right fork nor the left fork.

Lemma 10. Phil $\vdash i.thk \rightarrow \neg i.has_L \wedge \neg i.has_R$

Lastly, no two neighbour can be hungry at the same time.

Lemma 11. Phil $\vdash \neg i.hungry \wedge \neg(i+1).hungry$

At this point we are ready to prove the first important property: if philosopher i is always in violation $i.v_1$, but not in violation $i.v_2$, then none of his neighbors can eat.

Property 1. $\vdash \mathsf{AG}(i.v_1 \wedge \neg i.v_2) \rightarrow \mathsf{AG}(\neg i.eating \wedge \neg(i+1).eating \wedge \neg(i-1.eating))$
Proof.

1- $i.v_1 \rightarrow \neg i.eating$	*lemma 8*
2- $\mathsf{AG}(i.v_1) \rightarrow \mathsf{AG}(\neg i.eating)$	*CTL, 1*
3- $i.v_1 \wedge \neg i.v_2 \rightarrow$ $\neg(i+1).eating \wedge \neg(i-1).eating$	*lemma 6*
4- $\mathsf{AG}(i.v_1 \wedge \neg i.v_2) \rightarrow \mathsf{AG}(\neg(i+1).eating \wedge \neg(i-1).eating)$	*CTL, 3*
5- $\mathsf{AG}(i.v_1 \wedge \neg i.v_2) \rightarrow \mathsf{AG}(\neg i.eating \wedge \neg(i+1).eating \wedge \neg(i-1).eating)$	*CTL, 2, 4*

■

Now, we will prove that a violation $i.v_2$ is less dangerous than a $i.v_1 \wedge \neg i.v_2$ violation; in the sense that the first type of violation allows neighbors to progress in some cases. Suppose that a philosopher goes to the bathroom with the left fork, then his right neighbour is free to take his left fork (the right fork of the philosopher who has gone to the bathroom); moreover, he will be lucky in the sense that his left neighbour does not compete anymore for that resource. In addition, if the right neighbour of the lucky philosopher will not go to the bathroom, then we can ensure that the lucky philosopher will have the possibility of eating in the future.

Property 2.

$$\vdash \mathsf{AG}(((i+1).v_2 \wedge \neg(i+1).has_R) \wedge \neg i.bathroom \wedge \neg(i-1).bath) \rightarrow \mathsf{EF}i.eating$$

Proof.

1-$\mathsf{AG}(((i+1).v_2 \wedge \neg(i+1).has_R) \wedge \neg i.bathroom \wedge \neg(i-1).bathroom)$ $\wedge i.thinking \wedge (i-1).thinking \rightarrow \mathsf{EF}i.eating$	*Lemma 13*
2-$\mathsf{AG}(((i+1).v_2 \wedge \neg(i+1).has_R) \wedge \neg i.bathroom \wedge \neg(i-1).bathroom)$ $\wedge i.hungry \wedge (i-1).thinking \rightarrow \mathsf{EF}i.eating$	*Lemma 12*
3-$\mathsf{AG}(((i+1).v_2 \wedge \neg(i+1).has_R) \wedge \neg i.bathroom \wedge \neg(i-1).bathroom)$ $\wedge i.thk \wedge (i-1).hungry \rightarrow \mathsf{EF}i.eating$	*Lemma 15*
4-$\mathsf{AG}(((i+1).v_2 \wedge \neg(i+1).has_R) \wedge \neg i.bathroom \wedge \neg(i-1).bathroom)$ $\wedge i.hungry \wedge (i-1).eating \rightarrow \mathsf{EF}i.eating$	*PL, Lemma 14*
5-$\mathsf{AG}(((i+1).v_2 \wedge \neg(i+1).has_R) \wedge \neg i.bathroom \wedge \neg(i-1).bathroom)$ $\wedge i.thk \wedge (i-1).eating \rightarrow \mathsf{EF}i.eating$	*PL, Lemma 14*
6-$\mathsf{AG}(((i+1).v_2 \wedge \neg(i+1).has_R) \wedge \neg i.bathroom \wedge \neg(i-1).bathroom)$ $\wedge i.eating \rightarrow \mathsf{EF}i.eating$	*CTL*
7-$i.eating \veebar i.bathroom \veebar i.hungry \veebar i.thk$	*Phil3*
8-$(i-1).eating \veebar (i-1).bathroom \veebar (i-1).hungry \veebar (i-1).thk$	*Phil3*
9-$\neg(i.hungry \wedge (i-1).hungry) \wedge \neg(i.eating \wedge (i-1).eating)$	*lemma 3 & lemma 11*
10-$\vdash \mathsf{AG}((i.v_2 \wedge \neg i.has_R) \wedge \neg(i-1).bathroom \wedge \neg(i+1).bathroom)$ $\rightarrow \mathsf{AF}(i-1).eating$	*CTL, 1-9*

■

Note that we cannot ensure that philosopher i will always eat, because this depends on fair scheduling. If we add some kind of fairness restriction on our specification then we are able to prove it.

Obviously, we can prove the symmetric case.

Property 3. $\vdash \mathsf{AG}(((i-1).v_2 \wedge \neg(i-1).has_L) \wedge \neg i.bathroom \wedge \neg(i+1).bathroom) \rightarrow \mathsf{EF}i.eating$
■

Using both property 2 and property 3 we can obtain the following corollary.

Corollary 5. $\vdash \mathsf{AG}(i.v_2 \wedge \neg(i+1).bathroom \wedge \neg(i-1).bathroom \neg(i+2).bathroom \wedge (i-2).bathroom) \rightarrow \mathsf{EF}((i-1).eating \vee (i+1).eating)$
Proof.

1- $i.v_2 \rightarrow \neg i.has_L \vee \neg i.has_R$	***V3***
2- $\mathsf{AG}(i.v_2 \wedge \neg i.has_L \wedge \neg(i+1).bathroom \wedge \neg(i+2).bathroom) \rightarrow \mathsf{EF}((i+1).eating)$	*Property 3*
3- $\mathsf{AG}(i.v_2 \wedge \neg i.has_R \wedge \neg(i-1).bathroom \wedge \neg(i-2).bathroom) \rightarrow \mathsf{EF}((i-1).eating)$	*Property 2*
4- $\mathsf{AG}(i.v_2 \wedge \neg(i.has_R \wedge i.has_L) \wedge \neg(i-1).bathroom \wedge$ $\neg(i+1).bathroom \wedge \neg(i+2).bathroom \wedge \neg(i-2).bathroom) \rightarrow$ $\mathsf{EF}((i+1).eating) \vee \mathsf{EF}((i-1).eating)$	*CTL, 2, 3*
5- $\mathsf{AG}(i.v_2 \wedge \neg i.bathroom \wedge \neg(i-1).bathroom \wedge \neg(i+2).bathroom \wedge \neg(i-1).bathroom)$ $\rightarrow \mathsf{EF}((i+1).eating \vee (i-1).eating)$	*CTL, 1, 4*

■

Imposing fairness restrictions we should be able to probe the following property:

$\vdash \mathsf{AG}(i.v_2 \wedge \neg(i+1).bathroom \wedge \neg(i-1).bathroom \wedge \neg(i+2).bathroom \wedge (i-2).bathroom) \rightarrow \mathsf{AF}((i-1).eating \vee (i+1).eating)$

that is, in case of a v_2 violation one of the two neighbours will eventually eat. We leave this property for future work.

Finally, if we consider the CTL property:

$$(\mathsf{AG}p \rightarrow \mathsf{EF}q) \rightarrow (\mathsf{A}(r\ \mathcal{U}\ \mathsf{AG}p) \rightarrow \mathsf{EF}q)$$

then we obtain the following corollary from theorem 5:

Corollary 6. $\vdash \mathsf{A}(i.v_1\ \mathcal{U}\ \mathsf{AG}i.v_2) \wedge \mathsf{AG}(\neg(i+1).bathroom \wedge \neg(i-1).bathroom \wedge \neg(i+2).bathroom \wedge (i-2).bathroom) \rightarrow \mathsf{EF}((i-1).eating \vee (i+1).eating)$

Which says that if a violation v_1 upgrades to a violation of type v_2 (for example; if the philosopher puts down a fork) then at least one of the two neighbours will have the possibility of eating in the future.

6 Conclusions and Further Work

In this document we have presented the main characteristics of a new deontic logic, which can be used for reasoning about fault-tolerance. We have illustrated this framework using a variation

of the dinning philosophers problem, by means of this example we have shown how concepts like fault-recovery can be formalized using the formalism introduced above.

The logic described in earlier sections has interesting metaproperties, e.g., soundness, completeness and compactness. The fact that we use boolean operators on actions gives us an appealing expressive power. In addition, a strong connection between two different versions of permission was shown; this fact allows us to define the obligation predicate in a natural way, avoiding several paradoxes. It seems more or less direct to prove a version of the small model theorem for our logic, which give us a decision algorithm. Related to this, probably a tableaux method is possible, even for the temporal version of the logic, which enable us to implement a counterexample finder.

Modularizing specifications is a successful technique to reduce the complexity of logical specifications, an interesting work seems to extend the formalism described above to be able to deal with modules and relationships between them. How the deontic operators are affected by modularity seems to be a key problem here. We may extend the ideas of [17] where category theory is used to define the connection between components, but yet we need to distribute the deontic operators, e.g.; perhaps a component is in a violation but it does not implies that there is a global violation, and perhaps the recovery action should be local. To differentiate between local and global permissions, likely, is a promising future work.

The extension to first-order is needed to deal with more complex problems, probably some properties (e.g., compactness) will be loosed in such an extensions.

A Additional Proofs and Lemmas

A.1 Additional Lemmas

The proofs of the following lemmas can be found in the next subsection.

Lemma 12.

$\vdash \mathsf{AG}(((i{+}1).v_2 \wedge \neg(i{+}1).has_R) \wedge \neg i.bathroom \wedge \neg(i{-}1).bathroom) \wedge i.hungry \wedge (i{-}1).thinking \rightarrow \mathsf{EF}i.eating$

□

The next one is variation of the above lemma.

Lemma 13.

$\vdash \mathsf{AG}(((i{+}1).v_2 \wedge \neg(i{+}1).has_R) \wedge \neg i.bathroom \wedge \neg(i{-}1).bathroom) \wedge i.thinking \wedge (i{-}1).thinking \rightarrow \mathsf{EF}i.eating$

□

Lemma 14. $\vdash \mathsf{AG}(((i{+}1).v_2 \wedge \neg(i{+}1).has_R) \wedge \neg i.bathroom \wedge \neg(i{-}1).bathroom) \wedge (i.thinking \vee i.hungry) \wedge (i-1).eating \rightarrow \mathsf{EF}i.eating$
□

The next one is variation of the above lemma.

Lemma 15.

$\vdash \mathsf{AG}(((i{+}1).v_2 \wedge \neg(i{+}1).has_R) \wedge \neg i.bathroom \wedge \neg(i{-}1).bathroom) \wedge i.thk \wedge (i{-}1).hungry \rightarrow \mathsf{EF}i.eating$

□

We need one more lemma, the next one says if a philosopher cannot be in the bathroom, and a neighbour is hungry, then he will be thinking in the next state.

Lemma 16.

$$i.thk \wedge (i-1).hungry \wedge \mathsf{AN}\neg i.bathroom \rightarrow \mathsf{EN}i.thk$$

□

A.2 Additional Proofs

Proof of lemma 3:
We use induction to prove it.
Base Case:

1- $\neg\mathsf{Done}(univ) \rightarrow (\neg fork_i.down)$	PL, Phil1
2- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow (\neg i.has_L \wedge \neg(i+1).has_R)$	PL, 1, Phil2, Phil10

Ind. Case:

1- $\neg(i+1).has_R \wedge \neg i.has_L \rightarrow [(i+1).up_R](i+1).has_R$	PL, Phil5
2- $i.up_L \sqcap (i+1).up_R = \emptyset$	Phil11
3- $(i+1).up_R \sqsubseteq \overline{i.up_L}$	BA, 2
4- $\neg i.has_L \rightarrow [\overline{i.up_L}]\neg i.has_L$	PL, Phil5
5- $[\overline{i.up_L}]\neg i.has_L \rightarrow [i+1.up_R]\neg i.has_L$	PL, T3, 3
6- $\neg i.has_L \rightarrow [i+1.up_R]\neg i.has_L$	PL, 4, 5
7- $\neg(i+1).has_R \wedge \neg i.has_L \rightarrow [(i+1).up_R]((i+1).has_R \wedge \neg i.has_L)$	ML, 6, 1
8- $\neg i.has_L \rightarrow [i.up_L]i.has_L$	PL, Phil5
9- $i.up_L \sqsubseteq \overline{i+1.up_R}$	BA, 2
10-$\neg(i+1).has_R \rightarrow [\overline{(i+1).up_R}]\neg(i+1).has_L$	PL, Phil5
11-$[\overline{(i+1).up_R}]\neg(i+1).has_R \rightarrow [i.up_L]\neg(i+1).has_L$	PL, T3, 9
12-$\neg(i+1).has_R \rightarrow [i.up_L]\neg(i+1).has_R$	PL, 10, 11
13-$\neg(i+1).has_R \wedge \neg i.has_L \rightarrow [i.up_L](\neg(i+1).has_R \wedge i.has_L)$	PL, 12, 8
14-$\neg(i+1).has_R \wedge \neg i.has_L \rightarrow [i.up_L \sqcup i.up_R](\neg(i+1).has_R \vee \neg i.has_L)$	PL, T4, 7, 13
15-$\neg i.has_L \rightarrow [\overline{i.up_L}]\neg i.has_L$	PL, Phil5
16-$\neg(i+1).has_R \rightarrow [\overline{(i+1).up_R}]\neg(i+1).has_R$	PL, Phil5
17-$\neg i.has_L \wedge \neg(i+1).has_R \rightarrow [\overline{i.up_L} \sqcup \overline{(i+1).up_R}](\neg i.has_L \vee \neg(i+1).has_R)$	PL, T4, 15, 16
18-$\neg i.has_L \wedge \neg(i+1).has_R \rightarrow [\mathbf{U}](\neg i.has_L \vee \neg(i+1).has_R)$	PL, BA, T4, 7, 17
19-$i.has_L \rightarrow fork_i.up$	PL, Phil9
20-$fork_i.up \rightarrow [(i+1).up_R]\bot$	PL, Phil10
21-$i.has_L \rightarrow [(i+1).up_R]\bot$	PL, Phil9
22-$fork_i.up \rightarrow [i.up_L]\bot$	PL, Phil9
23-$i.has_L \rightarrow [i.up_L]\bot$	PL, Phil9
24-$i.has_L \wedge \neg(i+1).has_R \rightarrow [(i+1).up_R]\bot$	PL, 19, 22
25-$\neg(i+1).has_R \rightarrow [\overline{(i+1).up_R}]\neg(i+1).has_R$	PL, Phil6
26-$\bot \rightarrow \neg(i+1).has_R$	PL
27-$i.has_L \wedge \neg(i+1).has_R \rightarrow [(i+1).up_R]\neg(i+1).has_R$	ML, 24, 26
28-$i.has_L \wedge \neg(i+1).has_R \rightarrow [\overline{(i+1).up_R}]\neg(i+1).has_R$	PL, 25
29-$\neg(i+1).has_R \rightarrow \neg(i+1).has_R \vee \neg i.has_L$	PL
30-$i.has_L \wedge \neg(i+1).has_R \rightarrow [\mathbf{U}]\neg(i+1).has_R$	PL, BA. 28, 29
31-$i.has_L \wedge \neg(i+1).has_R \rightarrow [\mathbf{U}](\neg(i+1).has_R \vee \neg i.has_L)$	PL, 29, 30
32-$(i+1).has_R \rightarrow fork_i.up$	PL, Phil10
33-$(i+1).has_R \rightarrow [i.up_L]\bot$	PL, Phil9
34-$(i+1).has_R \wedge \neg i.has_L \rightarrow [i.up_L]\bot$	PL, 22, 32
35-$\neg i.has_L \rightarrow [\overline{i.up_L}]\neg i.has_L$	PL, Phil6
36-$(i+1).has_R \wedge \neg i.has_L \rightarrow [\overline{i.up_L}]\neg i.has_L$	PL, 35
37-$(i+1).has_R \wedge \neg i.has_L \rightarrow [\overline{i.up_L}](\neg i.has_L \vee \neg(i+1).has_R)$	PL, 36
38-$(i+1).has_R \wedge \neg i.has_L \rightarrow [i.up_L](\neg i.has_L \vee (i+1).has_R)$	PL, 34
39-$(i+1).has_R \wedge \neg i.has_L \rightarrow [\mathbf{U}](\neg i.has_L \vee \neg(i+1), has_R)$	BA, PL, 38, 37
40-$(\neg(i+1).has_R \vee \neg i.has_L) \rightarrow [\mathbf{U}](\neg i.has_L \vee \neg(i+1).has_R)$	PL, 18, 31, 39

■

Proof of lemma 4:

By induction:

Base Case:

1- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow \neg i.v_1$	PL, Phil1
2- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow (i.v_1 \rightarrow (i.has_L \vee i.has_R))$	PL, 1, Phi10

Ind. Case:

1- $\neg i.has_L \wedge \neg i.has_R \rightarrow \neg i.eating$	PL, Phil18
2- $\neg i.eating \rightarrow \mathsf{O}(i.down_L \sqcap i.down_R)$	PL, Phil21
3- $\neg i.has_L \wedge \neg i.has_R \rightarrow \neg\mathsf{O}(i.down_L \sqcap i.down_R)$	PL, 1, 2
4- $\neg i.v_1 \wedge \neg\mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow [\mathbf{U}]\neg i.v_1$	PL, V2
5- $\neg i.v_1 \wedge \neg(i.has_L \vee i.has_R) \rightarrow [\mathbf{U}]\neg i.v_1$	PL, 3, 4
6- $\imath.v_1 \wedge \neg(i.has_L \vee i.has_R) \rightarrow [\mathbf{U}](\neg i.v_1 \vee (i.has_L \vee i.has_R))$	PL, 5
7- $i.v_1 \wedge (i.has_L \vee i.has_R) \rightarrow [i.down_R \sqcap i.down_L]\neg i.v_1$	PL, V4
8- $i.v_1 \wedge (i.has_L \vee i.has_R) \rightarrow [i.down_R \sqcap i.down_L](\neg i.v_1 \vee (i.has_L \vee i.has_R))$	PL, 7
9- $i.has_L \rightarrow [\overline{i.down_L}]i.has_L$	PL, Phil6
10- $i.has_R \rightarrow [\overline{i.down_R}]i.has_R$	PL, Phil6
11- $i.has_L \vee i.has_R \rightarrow [\overline{i.down_L} \sqcup \overline{i.down_R}](i.has_L \vee i.has_R)$	PL, T4, 9, 10
12- $\neg i.v_1 \wedge (i.has_L \vee i.has_R) \rightarrow [\overline{i.down_L \sqcap i.down_R}](\neg i.v_1 \vee (i.has_L \vee i.has_r))$	PL, 11
13- $\neg i.v_1 \wedge \mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow [i.down_L \sqcap i.down_R]\neg i.v_1$	V1
14- $\neg i.v_1 \wedge \neg\mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow [\mathbf{U}]\neg i.v_1$	V3
15- $[\mathbf{U}]\neg i.v_1 \wedge i.down_L \sqcap i.down_R \sqsubseteq \mathbf{U} \rightarrow [i.down_L \sqcap i.down_R]\neg i.v_1$	T3
16- $[\mathbf{U}]\neg i.v_1 \rightarrow [i.down_L \sqcap i.down_R]\neg i.v_1$	PL, 15
17- $\neg i.v_1 \wedge \neg\mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow [i.down_L \sqcap i.down_R]\neg i.v_1$	PL, 14, 16
18- $\neg i.v_1 \rightarrow [i.down_L \sqcap i.down_R]\neg i.v_1$	PL, 13, 17
19- $\neg i.v_1 \wedge (i.has_L \vee i.has_R) \rightarrow [i.down_L \sqcap i.down_R]\neg i.v_1 \vee (i.has_L \vee i.has_R)$	PL, 18
20- $\neg i.v_1 \wedge (i.has_L \vee i.has_R) \rightarrow [\mathbf{U}](\neg i.v_1 \vee (i.has_L \vee i.has_R))$	PL, 12, 19
21- $i.v_1 \wedge i.has_L \wedge \neg i.has_R \rightarrow i.v_2$	PL, V3
22- $i.v_1 \wedge \neg i.has_R \rightarrow [i.down_L]\neg i.v_1$	PL, V6
23- $i.v_1 \wedge i.has_L \wedge \neg i.has_R \rightarrow [i.down_L]\neg i.v_1$	PL, 21, 22
24- $i.v_1 \wedge i.has_R \wedge \neg i.has_L \rightarrow i.v_2$	PL, V3
25- $i.v_2 \wedge \neg i.has_R \rightarrow [i.down_R]\neg i.v_1$	PL, V6
26- $i.v_1 \wedge i.has_R \wedge \neg i.has_L \rightarrow [i.down_R]\neg i.v_1$	PL, 25
27- $i.has_L \rightarrow [\overline{i.down_L}]i.has_L$	Phil6
28- $i.v_1 \wedge i.has_L \wedge \neg i.has_R \rightarrow [i.down_L \sqcup \overline{i.down_L}](\neg i.v_1 \vee i.has_L)$	ML, , T4, 23, 26 27
29- $i.v_1 \wedge i.has_L \wedge \neg i.has_R \rightarrow [\mathbf{U}](\neg i.v_1 \vee (i.has_L \vee i.has_R))$	PL, 28
30- $i.has_R \rightarrow [\overline{i.down_R}]i.has_R$	PL, Phil6
31- $i.v_1 \wedge i.has_R \wedge \neg i.has_L \rightarrow [\overline{i.down_R}](\neg i.v_1 \vee (i.has_R \vee i.has_L))$	PL, 26, 30
32- $i.v_1 \wedge i.has_R \wedge \neg i.has_L \rightarrow [\mathbf{U}](\neg i.v_1 \vee i.has_R \vee i.has_L)$	PL, 26, 31, T4
33- $i.v_1 \wedge i.has_R \wedge i.has_L \rightarrow \neg i.v_2$	PL, V3
34- $i.has_R \rightarrow [\overline{i.down_R}]i.has_R$	PL, Phil6
35- $i.has_L \rightarrow [\overline{i.down_L}]i.has_L$	PL, Phil6
36- $i.has_R \wedge i.has_L \rightarrow [\overline{i.down_R} \sqcup \overline{i.down_L}](i.has_R \vee i.has_L)$	BA, 34, 35
37- $i.v_1 \rightarrow [i.down_L \sqcap i.down_R]\neg i.v_1$	PL, V7
38- $i.has_R \wedge i.has_L \wedge i.v_1 \rightarrow [\mathbf{U}](\neg i.v_1 \vee (i.has_r \vee i.has_L))$	PL, T4, 36, 35
39- $\neg i.v_1 \vee (i.has_R \vee i.has_R) \rightarrow [\mathbf{U}](\neg i.v_1 \vee (i.has_R \vee i.has_L))$	PL, 20, 29 32, 38

■

Proof of Lemma 5:

1- $\neg i.v_1 \wedge \neg \mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow [\mathbf{U}]\neg i.v_1$	V2
2- $\neg i.v_1 \wedge \neg i.eating \rightarrow [\mathbf{U}]\neg i.v_1$	PL, Phil21, 1
3- $\neg i.v_1 \wedge \neg i.eating \rightarrow [\mathbf{U}]\neg i.v_1$	PL, V1
4- $\neg i.v_1 \wedge \mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow [i.down_L \sqcap i.down_R]\neg i.v_1$	PL, V1
5- $\neg i.v_1 \wedge i.eating \rightarrow [\overline{i.down_L sqcap i.down_R}]i.v_1$	PL, Phil21, V2
6- $\neg i.v_1 \wedge i.eating \rightarrow [i.down_L \sqcap o.down_R]\neg i.v_1$	PL, V1
7- $[i.down_L]\neg i.has_L \wedge [i.down_R]\neg i.has_R$	Phil6
8- $[i.down_L \sqcap i.down_R](\neg i.has_L \wedge \neg i.has_R)$	PL, T5, 7
9- $[i.down_L \sqcap i.down_R]\neg i.eating$	PL, 8, Phil18
10-$\neg i.v_1 \wedge i.eating \rightarrow [i.down_L \sqcap i.down_R]\neg i.eating$	PL, 9
11-$i.eating \rightarrow [\mathbf{U}]\mathsf{Done}((i.down_L \sqcap i.down_R) \sqcup i.getbad)$	Phil20
12-$i.eating \rightarrow [\overline{i.down_L \sqcap i.down_R}]\mathsf{Done}(i.getbad)$	PL,T16, T17, 11
13-$([i.getbad]i.bathroom) \wedge (i.bathroom \rightarrow \neg i.eating)$	PL, Phil3, Phil18, Phil22
14-$i.eating \rightarrow [\overline{i.down_L sqcap i.down_R}]\neg i.eating$	PL, T17, 13, 14
15-$\neg i.v_1 \wedge i.eating \rightarrow [\overline{i.down_L \sqcap i.down_R}]\neg i.eating$	PL, 14
16-$\neg i.v_1 \wedge i.eating \rightarrow [\mathbf{U}]\neg i.eating$	PL, 10, T4, 15
17-$\neg i.v_1 \wedge (\neg i.eating \vee i.eating) \rightarrow [\mathbf{U}](\neg i.eating \vee \neg i.v_1)$	PL, 2, 16
18-$\neg i.v_1 \rightarrow [\mathbf{U}](\neg i.v_1 \vee \neg i.eating)$	PL, 17

■

Proof of lemma 6:

By induction. We prove $i.v_1 \wedge \neg i.v_2 \rightarrow \neg(i+1).eating$, the another is similar.

Base Case:

1- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow \neg i.v_1$	PL, Phil1
2- $\neg\mathsf{Done}(\mathbf{U}) \rightarrow (i.v_1 \wedge \neg i.v_2 \rightarrow \neg(i+1).eating)$	PL, 1

Ind.Case:

1-$(i+1).eating \rightarrow (i+1).has_L \wedge (i+1).has_R$	PL, Phil18
2-$(i+1).has_R \rightarrow \neg i.has_L$	Lemma 3
3-$(i+1).eating \rightarrow \neg i.has_L$	PL, 10, 11
4-$(i+1).eating \rightarrow \neg(\neg i.v_2 \wedge i.v_1)$	PL, V3, 1
5-$[\mathbf{U}](\neg(i+1).eating \vee \neg(\neg i.v_2 \wedge i.v_1))$	GN, 4
6-$\neg(i.v_1 \wedge \neg i.v_2) \vee (\neg(i+1).eating) \rightarrow [\mathbf{U}](\neg(i+1).eating \vee \neg(\neg i.v_2 \wedge i.v_1))$	PL, 14

■

Proof of Lemma 7:

1- $\neg i.eating \leftrightarrow \neg i.has_L \vee \neg i.has_R \vee i.bathroom$	Phil18
2- $\neg i.v_1 \wedge \neg \mathsf{O}(\imath.down_L \sqcap i.down_R) \rightarrow [\mathbf{U}]\neg i.v_1$	V2
3- $\neg i.eating \rightarrow \neg \mathsf{O}(i.down_L \sqcap i.down_R)$	Phil21
4- $\neg i.v_1 \wedge \neg i.eating \rightarrow [\mathbf{U}]\neg i.v_1$	PL, 2, 3
5- $\neg i.v_1 \wedge \neg i.eating \rightarrow [\mathbf{U}](\neg i.v_1 \vee \neg i.eating)$	PL, 4
6- $i.v_1 \rightarrow [i.down_L \sqcap i.down_R]\neg i.v_1$	V4
7- $i.v_1 \rightarrow i.has_L \vee i.has_R$	Lemma 3
8- $i.v_1 \rightarrow i.has_L \vee i.has_R$	Lemma 4
9- $i.bathroom \wedge i.has_L \rightarrow [\overline{i.getbetter}]i.bathroom$	Phil23
10-$i.bathroom \wedge i.has_L \rightarrow [i.getbetter]\mathsf{Done}(i.down_L)$	Phil19
11-$[i.down_L]\neg i.has_L$	Phil4
12-$i.bathroom \wedge i.has_L \rightarrow [i.getbetter]\neg i.has_L$	PL, T17, 10, 11
13-$i.bathroom \wedge i.has_R \rightarrow [i.getbetter]donei.down_R$	Phil19
14-$[i.down_R]\neg i.has_R$	Phil4
15-$i.bathroom \wedge i.has_L \rightarrow [i.getbetter]\neg i.has_L$	PL, 13, 14
16-$i.bathroom \wedge (i.has_R \vee i.has_L) \rightarrow [i.getbetter](\neg i.bathroom \wedge (\neg i.has_R \vee \neg i.has_L))$	PL, 12, 15
17-$i.bathroom \wedge (i.has_R \vee i.has_L) \rightarrow$ $[i.getbetter \sqcup \overline{i.getbetter}](\neg i.bathroom \wedge (\neg i.has_R \vee \neg i.has_L))$	PL, 9, 16
18-$i.bathroom \wedge (i.has_R \vee i.has_L) \rightarrow [\mathbf{U}]\neg i.eating$	PL, 1, 17
19-$i.v_1 \wedge \neg i.eating \rightarrow [\mathbf{U}](\neg i.v_1 \vee \neg i.eating)$	PL, 1, 8, 18
20-$\neg i.eating \rightarrow [\mathbf{U}](\neg i.v_1 \vee \neg i.eating)$	PL, 19, 5

■

Proof of Lemma 8: By induction:
Base Case:

1-$\neg \mathsf{Done}(\mathbf{U}) \rightarrow \neg i.v_1 \wedge \neg i.eating$	PL, Phil1
2-$\neg \mathsf{Done}(\mathbf{U}) \rightarrow (i.v_1 \rightarrow \neg i.eating)$	PL, 1

Ind.Case:

1-$\neg i.v_1 \rightarrow [\mathbf{U}]\neg(i.v_1 \wedge i.eating)$	lemma 5
2-$\neg i.eating] \rightarrow [\mathbf{U}]\neg(i.v_1 \wedge i.eating)$	lemma 7
3-$\neg i.v_1 \vee \neg i.eating \rightarrow [\mathbf{U}]\neg(i.v_1 \wedge i.eating)$	PL, 1, 2

■

Proof of Lemma 9 By Induction: base case straightforward using **Phil1**, the inductive case is as follows:

1-$\neg i.v_1 \wedge \mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow [i.down_L \sqcap i.down_R]\neg i.v_1$	V1
2-$\neg i.v_1 \wedge \mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow [\overline{i.down_L \sqcap i.down_R}]i.v_1$	V1
3-$\mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow i.eating$	Phil21
4-$i.eating \rightarrow [\mathbf{U}]\mathsf{Done}(i.getthk \sqcup i.getbad)$	Phil20
5-$i.eating \rightarrow i.has_L \wedge i.has_R$	Phil18
6-$i.has_L \wedge i.has_R \rightarrow [i.getthk](\mathsf{Done}(i.down_L) \wedge \mathsf{Done}(i.down_R))$	Phil7
7-$i.eating \rightarrow [\mathbf{U}]\mathsf{Done}(i.gethtk) \vee [\mathbf{U}]\mathsf{Done}(i.getbad)$	ML, T14, 4
8-$i.eating \rightarrow [\mathbf{U}](\mathsf{Done}(i.down_L \sqcap \mathsf{Done}(i.down_R))) \vee [\mathbf{U}](\mathsf{Done}(i.getbad))$	T17, 6, 7
9-$i.eating \rightarrow [\overline{i.down_L \sqcap i.down_R}]\neg\mathsf{Done}(i.down_L \sqcap i.down_R)$	PL, TempAx8
10-$i.eating \rightarrow [\overline{i.down_L \sqcap i.down_R}]\mathsf{Done}(i.getbad)$	T4, T16, 8, 9
11-$[i.getbad]i.bathroom$	Phil22
12-$i.eating \rightarrow [\overline{i.down_L \sqcap i.down_R}]i.bathroom$	T17, 10, 11
13-$\neg i.v_1 \wedge \mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow [\overline{i.down_L \sqcap i.down_R}](i.bathroom \vee \neg i.v_1)$	ML, 12
14-$\neg i.v_1 \wedge \mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow [\mathbf{U}](i.bathroom \vee \neg i.v_1)$	BA, PL, 1, 3
15-$\neg i.v_1 \wedge \neg\mathsf{O}(i.down_L \sqcap i.down_R) \rightarrow [\mathbf{U}](i.bathroom \vee \neg i.v_1)$	V2
16-$\neg i.v_1 \rightarrow [\mathbf{U}](i.bathroom \vee \neg i.v_1)$	BA, PL, 14, 15
17-$i.bathroom \rightarrow [\overline{i.getthk}]i.bathroom$	Phil22
18-$i.v_1 \wedge i.has_L \rightarrow [i.down_L]\neg i.v_1$	Phil5
19-$i.v_1 \wedge \imath.has_R \rightarrow [i.down_R]\neg i.v_1$	Phil5
20-$i.v_1 \rightarrow i.has_R \vee i.has_L$	lemma 4
21-$i.bathroom \wedge i.has_L \rightarrow [i.getthk]\neg i.v_1$	T17, Phil7, V6
22-$i.bathroom \wedge i.has_R \rightarrow [i.getthk]\neg i.v_1$	T17, Phil7, V6
23-$i.v_1 \wedge i.bathroom \rightarrow [i.getthk]\neg i.v_1$	ML, 20, 21, 22
24-$i.bathroom \wedge i.v_1 \rightarrow [\mathbf{U}](\neg i.v_1 \vee i.bathroom)$	BA, PL, 16, 17, 23

■

Proof of Lemma 10: By induction, base case is direct using **Phil1**, we prove the inductive case:

1-$i.thk \rightarrow [\overline{i.getthk}]\neg i.thk$	Phil8
2-$i.has_L \rightarrow [i.getthk]\mathsf{Done}(i.down_L)$	Phil7
3-$[i.down_L]\neg i.has_L \wedge [i.getthk]\mathsf{Done}(i.down_L) \rightarrow [i.getthk]\neg i.has_L$	T17
4-$i.has_R \rightarrow [i.getthk]\mathsf{Done}(i.down_R)$	Phil7
5-$[i.down_R]\neg i.has_R \wedge [i.getthk]\mathsf{Done}(i.down_R) \rightarrow [i.getthk]\neg i.has_R$	T17
6-$\neg i.thk \wedge (i.has_L \vee i.has_R) \rightarrow [i.getthk](i.thk \wedge (\neg i.has_L \wedge \neg i.has_R))$	PL, 3, 5, Phil5
7-$(\neg i.has_L \rightarrow [\overline{i.up_L}]\neg i.has_L) \wedge (\neg i.has_R \rightarrow [\overline{i.up_R}]\neg i.has_R)$	Phil7
8-$(i.getthk \sqsubseteq \overline{i.up_L}) \wedge (i.getthk \sqsubseteq \overline{i.up_R})$	BA, Phil7
9-$i.getthk \sqsubseteq \overline{i.up_L \sqcup i.up_R}$	BA, 8
10-$\neg i.has_L \wedge \neg i.has_R \rightarrow [i.getthk](\neg i.has_L \wedge \neg i.has_R)$	T3, 7, 8
11-$\neg i.thk \wedge (\neg i.has_R \wedge \neg i.has_L) \rightarrow [i.getthk](i.thk \wedge (\neg i.has_L \wedge \neg i.has_R))$	ML, 10, Phil8
12-$\neg i.thk \rightarrow [\mathbf{U}](\neg i.thk \vee (\neg i.has_L \wedge i.has_R))$	BA, PL, 1, 6, 11
13-$i.thk \wedge (i.has_L \wedge i.has_R) \rightarrow [i.up_L \sqcup i.up_R]\bot$	Phil9
14-$i.thk \wedge (\neg i.has_L \wedge \neg i.has_R) \rightarrow [i.up_L \sqcup i.up_R](\neg i.thk \vee (\neg i.has_R \wedge \neg i.has_L))$	ML, 13
15-$i.thk \wedge (\neg i.has_L \wedge \neg i.has_R) \rightarrow [\overline{i.up_L \sqcup i.up_R}](\neg i.thk \vee (\neg i.has_L \wedge \neg i.has_R))$	ML, Phil6
16-$\neg i.thk \vee (\neg i.has_L \wedge \neg i.has_R) \rightarrow [\mathbf{U}](\neg i.thk \vee (\neg i.has_L \wedge \neg i.has_R))$	BA, PL, 1, 12, 15

■

Proof of Lemma 11 By induction, the base case is, as usual, using Phil1, the inductive case is as follows:

1-$\neg i.hungry \wedge (i+1).hungry \rightarrow [i.gethungry]\bot$	Phil14
2-$\neg i.hungry \rightarrow [\overline{i.gethungry}]\neg i.hungry$	Phil17
3-$\neg i.hungry \wedge (i+1).hungry \rightarrow [\mathbf{U}](\neg i.hungry \vee \neg(i+1).hungry)$	BA, PL, 1, 2
4-$i.gethungry \sqcap (i+1).gethungry = \emptyset$	Phil12
5-$i.gethungry \sqsubseteq \overline{i+1.gethungry}$	BA, 4
6-$\neg(i+1).hungry \rightarrow [\overline{(i+1).gethungry}]\neg(i+1).gethungry$	Phil17
7-$(i+1).hungry \rightarrow [i.gethungry](i+1.hungry)$	PL, T4, 5, 6
8-$\neg i.hungry \wedge \neg(i+1).hungry \rightarrow [i.gethungry](\neg i.hungry \vee \neg(i+1).hungry)$	T4, 6, 7
9-$\neg i.hungry \rightarrow [\overline{i.gethungry}]\neg i.hungry$	Phil17
10-$\neg i.hungry \wedge \neg(i+1).hungry \rightarrow [\overline{i.gethungry}](\neg i.hungry \vee \neg(i+1).hungry)$	ML, 9
11-$i.hungry \wedge \neg(i+1).hungry \rightarrow [\mathbf{U}](\neg i.hungry \vee \neg(i+1).hungry)$	Symmetrically from 3
12-$\neg i.hungry \vee \neg(i+1).hungry \rightarrow [\mathbf{U}](\neg i.hungry \vee \neg(i+1).hungry)$	BA, PL, 3, 10, 11

■

Proof of lemma 12:

1-$(i.hungry) \wedge (fork_i.down \vee i.has_R) \wedge (fork_{i+1}.down \vee i.has_L)$ $\rightarrow \mathsf{AN} i.eating$	CTL, Phil15
2-$(i-1).thk \rightarrow \neg(i-1).has_L \wedge \neg(i-1).has_R$	lemma 10
3-$\neg(i-1).has_L \rightarrow fork_i.down \vee i.has_R$	PL, Phil10
4-$\neg(i+1).has_R \rightarrow fork_{i+1}.down \vee i.has_L$	PL, Phil10
5-$\mathsf{AG}((i+1).v_2 \wedge \neg(i+1).has_R \wedge \neg i.bathroom \wedge \neg(i-1).bathroom)$ $\wedge i.hungry \wedge (i-1).thk \rightarrow \mathsf{EN}(i.eating)$	CTL, 1, 2, 3 ,4

■

Proof of lemma 13:

1-$(i-1).thk \rightarrow \neg(i-1).has_L \wedge \neg(i-1).has_R$	lemma 10
2-$i.thk \rightarrow \neg i.has_L \wedge \neg i.has_R$	lemma 10
3-$\neg(i-1).has_L \wedge \neg i.has_R \rightarrow fork_i.down$	Phil10
4-$\neg(i+1).v_2 \wedge (i+1).has_L \rightarrow \neg(i+1).has_R$	PL, V3
5-$i.thk \rightarrow \neg i.bathroom$	PL, Phil3
6-$\neg(i+1).has_R \wedge \neg i.has_L \rightarrow fork_{i+1}.down$	Phil10
7-$(i+1).v_2 \rightarrow (i+1).bathroom$	lemma 9
8-$(i+1).bathroom \rightarrow \neg(i+1).hungry$	PL, Phil3
9-$(i+1).v_2 \rightarrow \neg(i+1).hungry$	PL, 7, 8
10-$i.thk \wedge \neg(i-1).hungry \wedge \neg(i+1).hungry \wedge fork_i.down \wedge fork_{i+1}.down$ $\rightarrow \langle i.gethungry\rangle\top$	PL, Phil13
11-$\langle i.gethungry\rangle i.hungry \rightarrow \langle \mathbf{U}\rangle i.hungry$	T3
12-$\langle \mathbf{U}\rangle i.hungry \rightarrow \mathsf{EN} i.hungry$	PL, TempAx1
13-$(i, gethungry \sqcap \overline{(i-1).gethungry}) = \emptyset$	Phil12
14-$i.gethungry \sqsubseteq \overline{(i-1).gethungry}$	BA, 13
15-$(i-1).thk \wedge \mathsf{AN}(\neg(i-1).bathroom) \rightarrow [\overline{(i-1).gethungry}](i-1).thk$	lemma 16
16-$(i-1).thk \wedge \mathsf{AN}(\neg(i-1).bathroom) \rightarrow [i.gethungry](i-1).thk$	PL, T3, 16
17-$[i.gethungry](i-1).thk \wedge \langle i.gethungry\rangle i.hungry \rightarrow$ $\langle i.gethungry\rangle((i-1).thk \wedge i.hungry)$	ML
18-$\mathsf{AN}((i+1).v_2 \wedge \neg(i+1).has_R \wedge \neg i.bathroom \wedge$ $\neg(i-1).bathroom) \wedge i.thk \wedge (i-1).thk \rightarrow \langle i.gethungry\rangle((i-1).thk \wedge i.hungry)$	CTL, 10, 17
19-$\mathsf{AG}((i+1).v_2 \wedge \neg(i+1).has_R \wedge \neg i.bathroom \wedge \neg(i-1).bathroom) \wedge$ $i.thk \wedge (i-1).thk \rightarrow \mathsf{EN} i.eating$	CTL, 18, lemma 12

■

Proof of lemma 14:

1-$(i-1).eating \rightarrow \mathsf{ANDone}(i.getthk \sqcup i.getbad)$	Phil20
2-$(i-1).eating \rightarrow \mathsf{AN}(\mathsf{Done}(i.getthk) \vee \mathsf{Done}(i.getbad))$	T14
3-$[(i-1).getbad](i-1).bad$	Phil22
4-$\mathsf{ANDone}((i-1).getbad) \wedge [(i-1).getbad](i-1).bathroom \rightarrow \mathsf{AN}(i-1).bathroom$	T17, TempAx1, TempAx2
5-$(i-1).eating \rightarrow \mathsf{ANDone}((i-1).getthk) \vee \mathsf{ANDone}((i-1).getbad)$	CTL, 2
6-$(i-1).eating \rightarrow \mathsf{ANDone}((i-1).getthk) \vee \mathsf{AN}(i-1).bathroom$	PL, 3, 4, 5
7-$\mathsf{AG}\neg(i-1).bathroom \rightarrow \mathsf{AN}\neg(i-1).path$	CTL
8-$(i-1).eating \wedge \mathsf{AG}\neg(i-1).bathroom \rightarrow \mathsf{ANDone}(i.getthk) \vee \mathsf{AN}\bot$	CTL, 6, 7
9-$(i-1).eating \wedge \mathsf{AG}\neg(i-1).bathroom \rightarrow \mathsf{ANDone}(i.getthk)$	CTL, 8
10-$[(i-1).getthk](i-1).thk \wedge [\mathbf{U}]\mathsf{Done}((i-1).getthk) \rightarrow (i-1).thk$	T17
11-$(i-1).eating \wedge \mathsf{AN}\neg(i-1).bathroom \rightarrow \mathsf{AN}(i-1).thk$	CTL, 9 ,10
12-$\mathsf{AN}\neg i.bathroom \rightarrow \mathsf{A}(i.hungry \vee i.thk \vee i.eating)$	PL, Phil3
13-$\mathsf{AG}((i+1).v_2 \wedge \neg(i+1).has_R \wedge \neg i.bathroom \neg(i-1).bathroom) \wedge (i.hungry \vee i.thk \vee i.eating) \wedge (i-1).eating \rightarrow \mathsf{EF}i.eating$	CTL 11, 12 Lemma 16, Lemma 12

■

Proof of lemma 15

1- $(i-1).hungry \wedge (fork_{i-1}.down \vee (i-1).has_L) \wedge (fork_i.down \wedge (i-1).has_R) \rightarrow \mathsf{AN}(i-1).eating$	Phil15
2- $i.thk \rightarrow \neg i.has_L \wedge \neg i.has_R$	Lemma 10
3- $\neg i.has_R \rightarrow (i-1).has_L \vee fork_{i-1}.down$	Phil10
4- $i.thk \rightarrow (i-1).has_L \vee fork_{(i-1)}.down$	PL, 2, 3
5- $i.thk \wedge \mathsf{AN}\neg i.bathroom \wedge (i-1).hungry \rightarrow \mathsf{EN}i.thk$	lemma 16
6-$i.thk \wedge \mathsf{AN}\neg i.btah \wedge (i-1).hungry \rightarrow \mathsf{EN}(i.thk \wedge (i-1).eating)$	CTL, 1, 4, 5
7-$\mathsf{AG}((i+1).v_2 \wedge \neg(i+1).has_R \wedge \neg i.bathroom \wedge \neg(i-1).bathroom) \wedge i.thk \wedge (i-1).hungry \rightarrow \mathsf{EN}(i.thk \wedge (i-1).eating)$	CTL, 6
8-$\mathsf{AG}((i+1).v_2 \wedge \neg(i+1).has_R \wedge \neg i.bathroom \wedge \neg(i-1).bathroom) \wedge i.thk \wedge (i-1).hungry \rightarrow \mathsf{EF}(i.eating)$	Lemma 14

■

Proof of lemma 16:

1-$\neg((i-1).hungry \wedge (i+1).hungry)$	lemma 11
2-$(i+1).hungry \rightarrow [(i-1).gethungry]\bot$	PL, Phil14
3-$\mathsf{AN}\neg(i-1).bathroom \rightarrow [\mathbf{U}]\neg(i-1).bathroom$	PL, TempAx1
4-$[\mathbf{U}]\neg(i-1).bathroom \wedge [i.getbad]i.bathroom \rightarrow [i.getbad]\bot$	BA, $\mathsf{T5}$
5-$[i.getbad]i.bathroom$	Phil22
6-$\mathsf{AN}\neg(i-1).bathroom \rightarrow [i.getbad]\bot$	PL, 3, 4, 5
7-$[(i-1).gethungry]\bot \rightarrow [(i-1).gethungry]\neg(i-1), gethungry$	ML
8-$\neg(i-1).hungry \rightarrow \overline{[i.gethungry]}\neg(i-1).hungry$	Phil17
9-$\neg(i-1).hungry \wedge \mathsf{AN}(\neg(i-1).bathroom) \rightarrow [\mathbf{U}]\neg(i-1).hungry$	BA, PL, 6, 8
10-$(i-1).thk \rightarrow \neg(i-1).hungry$	PL, Phil3
11-$(i-1).thk \wedge (i+1).hungry \wedge \mathsf{AN}\neg(i-1).bathroom \rightarrow \mathsf{AN}\neg(i-1).hungry$	PL, 9, 10
12-$(i-1).thk \rightarrow [(i-1).up_L \sqcup (i+1).up_L]\bot$	Phil4
13-$(i-1).thk \rightarrow \overline{[(i-1).up_L \sqcup (i-1).up_L]}(\neg(i-1).has_L \wedge (i-1).has_R)$	PL, $\mathsf{T5}$, Phil6
14-$(i-1).thk \rightarrow [\mathbf{U}]\neg i.eating$	PL, BA, 12, 13
15-$(i-1).thk \rightarrow \neg i.eating$	PL, Phil4
16-$(i-1).thk \rightarrow \mathsf{AN}\neg i.eating$	PL, TempAx1, 14, 15
17-$\neg(i+1).eating \wedge \neg(i-1).hungry \wedge \neg(i-1).bathroom \rightarrow (i-1).thk$	PL, Phil4
18-$(i-1).thk \wedge (i+1).hungry \wedge \mathsf{AN}\neg(i-1).bathroom \rightarrow \mathsf{AN}((i-1).thk)$	CTL, 11, 16, 17

■

References

[1] M.Abadi. *The Power of Temporal Proofs.* Theorical Computer Science, Volume 65, Issue 1, 1989.

[2] Zohar Manna, Amir Pnueli. *The Anchored Version of the Temporal Framework.* Lectures Notes in Computer Science, Vol.354. Springer-Verlag, 1988.

[3] J.Broersen. *Modal Action Logics for Reasoning about Reactive Systems.* PhD. Thesis. Vrije University. 2003.

[4] G.Gargov and S.Passy. *A Note on Boolean Logic.* In P.P.Petkov, editor, Mathematical Logic. Proceedings of the Heyting Summerschool. Plenum Press. 1990.

[5] Brian. F.Chellas. *Modal Logic:An Introduction.* Cambridge University Press, 1980.

[6] Dijkstra. E. W. *Hierarchical Ordering of Sequential Processes.* Acta Informatica 1, 115-138.

[7] D.Harel, D.Kozen and J.Tiuryn. *Dynamic Logic.* MIT Press, 2000.

[8] E.A.Emerson. *Temporal and Modal Logic.* J.van Leeuwen (ed.). Handbook of Theorical Computer Science.

[9] E.A.Emerson. *Decision Procedures and Expressiveness in the Temporal Logic of Branching Time.* Proceedings of STOC, 1982. Vol.B. MIT Press. 1990.

[10] J.J.Meyer. *A Different Approach to Deontic Logic: Deontic Logic Viewed as a Variant of Dynamic Logic* Notre Dame Journal of Formal Logic, Volume 29, Number 1, 1988.

[11] J.J.Meyer. *Dynamic Logic for Reasoning about Actions and Agents.* 2003

[12] J.J.Meyer, R.J.Wieringa. *Deontic Logic, a Concise Overview.* Deontic Logic in Computer Science. Normative System Specification. Chichester, John Wiley & Sons, 1993.

[13] F.Kroger. *Temporal Logic of Programming.* Springer-Verlag, 1987.

[14] S.Kent, T.S.E.Maibaum and W.J.Quirk. *Formally Specifying Temporal Constraints and Error Recovery.* Proceedings of IEEE First International Symposium on requirements Engineering (RE93). San Diego, US, 1993.

[15] T.Maibaum *Temporal Reasoning over Deontic Specifications.* Deontic Logic in Computer Science. John Wiley & Sons. 1993.

[16] T.Maibaum, S.Khosla. *The Prescription and Description of State-Based Systems.* In B.Banieqbal, H.Barringer and A.Pnueli (eds). Temporal Logic in Computation, LNCS, Springer-Verlag. 1985.

[17] J.Fiadeiro and T.Maibaum. *Temporal Theories as Modularization Units for Concurrent System Specification.* Formal Aspects of Computing.

[18] J.Fiadeiro and T.Maibaum. *Towards Object Calculi.* In G.Saake and A.Sernadas, editors, Information Systems.Correctness and Reusability, Workshop IS-CORE 91.

[19] Zohar Manna, Amir Pnueli. *The Temporal Logic of reactive and Concurrent Systems.* Springer-Verlag New York Inc. 1992.

[20] J.Donald Monk. *Mathematical Logic.* Springer-Verlag, Graduate Texts in Mathematics.1976.

[21] P.BlackBurn, M.de Rijke and Y. de Venema. *Modal Logic.* Cambridge Tracts in Theorical Computer Science 53, 2001.

[22] S.Kent, T.Maibaum and B.Quirk. *Specifying Deontic Behavior in Modal Action Logic.* Report, Forest Research Project. 1991.

[23] M.Reynolds. *An Axiomatization of full Computation Tree Logic.* Journal of Symbolic Logic. 2003

[24] A.Ross. *Imperatives and Logic.* Philosophy of Science 11. 1944.

[25] M.Sergot and H.Prakken. *Contrary-to-duty obligations.* Proc. Second International Workshop on Deontic Logic in Computer Science (DEON 94). Oslo, January 1994.

[26] R.Sikorski. *Boolean Algebras.* 3rd ed. New York, Springer-Verlag, 1969.