Università di Pisa
Dipartimento di Informatica

Technical Report: TR-02-05

Distributed States Logic

Carlo Montangero   Laura Semini

April 19, 2002

ADDRESS: Corso Italia 40, 56125 Pisa, Italy.   TEL: +39 050 2212700   FAX: +39 050 2212726

Distributed States Logic*

Carlo Montangero   Laura Semini

Dipartimento di Informatica, Università di Pisa.
{monta,semini}@di.unipi.it

Abstract. We introduce a temporal logic to reason on global applications. First, we define a modal logic for localities that embeds the local theories of each component into a theory of the distributed states of the system. We provide the logic with a sound and complete axiomatization. The contribution is that it is possible to reason about properties that involve several components, even in the absence of a global clock, which is a meaningless notion in an asynchronous setting. Then, we extend the logic with a temporal operator. We support our proposal by working out an example, a simple secure communication system.

1 Introduction

The current trend towards global computing needs software that works in an open, concurrent, distributed, high–latency, security–sensitive environment. Besides, this software must be reliable, scalable, and "shipped today". Several trends are emerging in response to the challenges involved in the development of software with so demanding requirements.

On one side, there is an increasing interest in the seamless integration of asynchronous communication in programming, coordination, and specification languages, since message–passing, event–based programming, call–backs, continuations, dataflow models, workflow models etc. are ubiquitous in global computing. Notable examples in this direction can be found in the context of the Microsoft .NET initiative, like the introduction of support for the delegate–based asynchronous calling model in the libraries of the Common Language Runtime [19], and the proposal of *chords* in Polyphonic C# to accommodate asynchronous methods in C# [25]. Our previous work with Oikos_tl [22] is an example of coverage of asynchronous communication in coordination and specification languages.

Another significant trend is represented by Component–Oriented Programming, that aims at producing software components for a software market and for late composition. Composers are third parties, possibly the end user, who are not able or willing to modify components. This trend emphasizes the need for high quality specifications that put the composer into the position to decide what can be composed under which conditions. In a previous work with Oikos–*adtl* [18, 17], a specification language for distributed systems based on asynchronous communications, we showed how to accommodate asynchronous communication in the composition of distributed systems specifications.

A notable example of component programming in the context of global computing is offered by the Web Services [12], which leverage the standard representation of data provided by XML to foster the construction of new components (services) by the coordination of other services. Since the cooperation is based on asynchronous protocols, this is also an example of the convergence of asynchronous communications and component programming.

Formal methods can play a major role in global computing. Precisely because the actors are programmatically independent, they need to have reliable ways to share precise knowledge

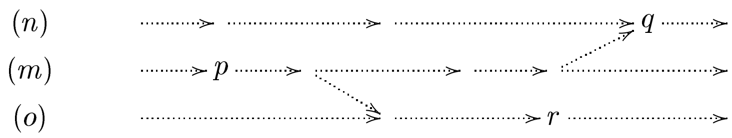
* An Extended Abstract of this work will appear in the Proceedings of the 9^{th} International Symposium on Temporal Representation and Reasoning (TIME 2002), Manchester, 7–9 July 2002, IEEE Computer Society Press.

of the artifacts they use or produce, independently of the particular technology (programming languages, middleware, ...) they rely on. Formal methods offer exactly this kind of independence and precision, since they provide abstract models to share when operating or developing with components. They can provide ways to make precise the specifications of the components and of their contextual dependencies, and to prove in advance global properties, i.e. that a composition will meet the specifications it addresses.

In this paper we define YALL, an extension of temporal logic to deal with distributed systems. We introduce new operators to be able to name the components of the system and causally relate properties which might hold in distinguished components, although in an asynchronous setting. A typical YALL formula might be:

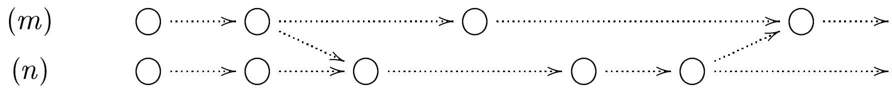
$$\mathsf{m}\, p \ \ \text{LEADS\_TO} \ \ \mathsf{n}\, q \wedge \mathsf{o}\, r \tag{1}$$

where the operator LEADS_TO is similar to Unity's $\mapsto$ (leads to) [2], and m, n and o express locality. Formula (1) says that a property p holding in component m, causes properties q and r to hold in future states of components n and o, respectively. An example is the computation below. Horizontal arrows denote the sequence of states of a component, oblique arrows denote the communications.



The usual choices to build a Kripke model for formulae like (1) are to consider the set of worlds W to be:

– the set of the states of a computation, i.e. the union of all the states of the system components, like the circles in the following figure.

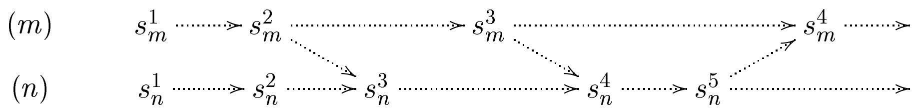


This choice was adopted in Oikos–*adtl* and has shown some problems. For instance, consequence weakening, or, more in general, the possibility of reasoning on logical relations between formulae like the premises or the consequences of (1), is not part of the logic. In particular, a formula like

$$(\mathsf{n}\, q \wedge \mathsf{o}\, r) \rightarrow \mathsf{n}\, q \tag{2}$$

which would permit to weaken the consequences of (1) would not be a legal formula, since no world can satisfy the conjunction $\mathsf{n}\, q \wedge \mathsf{o}\, r$.

– the set of global states, or snapshots, of the system, where each world is a set of states, one for each component. These sets must satisfy some constraints to be coherent with the communications between the subsystems. Examples of worlds are $\{s^i_m,\ s^j_n\}^{i=1,2}_{1\leq j\leq 3}$, while $\{s^3_m,\ s^2_n\}$ would not be a legal world.



This choice, adopted in many logics for distributed systems (see Section 5) is not applicable in the case of asynchronous communication. Think of the case of property p holding *only* in state s_m^2 and q holding *only* in states s_n^j, for $1 \leq j \leq 5$. The following formula would be valid in the model

$$\mathsf{m}\, p \rightarrow \mathsf{n}\, q \tag{3}$$

inferring a remote knowledge which is meaningless in an asynchronous setting. Moreover, it would be natural to say that world $\{s_m^3, s_n^4\}$ follows $\{s_m^2, s_n^3\}$. In this case, one could assert that $\mathsf{n}\, p$ LEADS_TO $\mathsf{m}\, q$ holds, if p and q hold in s_n^3 and s_m^3, respectively, even though no causal relationship exists between these two states.

– a third possibility would be to consider all the pairs of states as worlds. But then, formula (3) would be valid in the model above if q holds in all the states of component n. Even if this is philosophically more acceptable, we claim that a better solution can be found. Besides, the problem of the wrong causal dependencies would persist.

The purpose of this paper is to introduce the distributed state logic DSL, that carries over all meaningful propositional rules, like *and* simplification, so that they can be exploited orthogonally to any temporal operator. A major consequence of the introduction of DSL is that the exploitation of the local theories in the proofs of the distributed properties becomes smooth and robust. The contribution here is that it is then possible to reason about properties that involve several components, even in the absence of a global clock, which is a meaningless notion in an asynchronous setting.

The semantic domain of DSL, the power–set of the set of all system states, even if chosen for technical reasons, makes YALL a very expressive language, that meets the pragmatic expectations of a designer fully (see Section 5 for a discussion).

Section 2 introduces the modal logic DSL, and its sound and complete axiomatization. Section 3 extends DSL with the temporal operator LEADS_TO, as the most interesting fragment of YALL. Section 4 works out an example, namely a simple secure communication system. The last sections cover a discussion of the main design issues, related work and future perspectives.

2 DSL

We assume a countable set of propositional letters P, with $p, q, \ldots$ ranging over P. The DSL well–formed formulae over a finite set of components $\Sigma = \{m_1, m_2, \ldots, m_k\}$ are defined by:

$$F \ ::= \ p \mid \bot \mid \ \sim F \mid F \wedge F' \mid \mathsf{m_i} F$$

where $\bot$ is the propositional constant *false*, and $\mathsf{m_i}$ for $i = 1 \ldots k$ are unary location operators. With $\bar{\mathsf{m}}_\mathsf{i}$ we denote the dual of $\mathsf{m_i}$, i.e., $\bar{\mathsf{m}}_\mathsf{i} F \equiv \sim \mathsf{m_i} \sim F$. With $\top$ we denote *true*, i.e. $\top \equiv \sim \bot$.

Semantics. A model $\mathcal{M}$ for DSL formulae is a tuple $(W, R_1, \ldots, R_k, V)$, with u, v, w ranging over W. The reachability relations R_i satisfy the following conditions:

$$(u,\, v) \in R_i \quad \rightarrow \quad (v,\, v) \in R_i \tag{4}$$

$$(u,\, v) \in R_i \quad \rightarrow \quad (v,\, w) \in R_i \rightarrow v = w \tag{5}$$

$$(u,\, v) \in R_i \quad \rightarrow \quad \nexists w.\ (v,\, w) \in R_j \text{ for } j \neq i \tag{6}$$

The semantics of DSL formule is given by:

$$(\mathcal{M},\, u) \models \top$$

$$(\mathcal{M},\, u) \models p \ \text{ iff } \ p \in V(u)$$

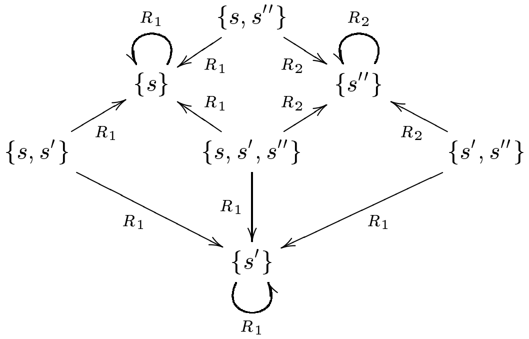
$$(\mathcal{M},\, u) \models \sim F \ \text{ iff } \ \text{not } (\mathcal{M},\, u) \models F$$

$$(\mathcal{M},\, u) \models F \wedge F' \ \text{ iff } \ (\mathcal{M},\, u) \models F \text{ and } (\mathcal{M},\, u) \models F'$$

$$(\mathcal{M},\, u) \models \mathsf{m_i} F \ \text{ iff } \ \exists v.\ (u, v) \in R_i \text{ and } (\mathcal{M},\, v) \models F$$

Let S_i be the set of states of component m_i, with $S_i \cap S_j = \emptyset$ for $i \neq j$, $S = \bigcup_{k}^{i=1,} S_i$, $DS = 2^S$, and $ds, ds' \in DS$. The frame $(DS, R_1, \ldots, R_k)$, where $(ds, ds') \in R_i$ if and only if ds' is a singleton set $\{s\}$, with $s \in S_i \cap ds$, satisfies the conditions (4)–(6) above. We call these frames, frames on DS, and call DS the set of *distributed states*, from which the name of the logic DSL. The frames on DS play a central role in the paper, since they are used to build the models for YALL formulae. Some examples follow.

Example 1. Let the set DS be built on $S_1 = \{s, s'\}$ and $S_2 = \{s''\}$, then the frame on DS can be represented as:



Note 2. For the sake of readability, we often let m, n range over Σ, with $m \neq n$. In these cases, we use also S_m, S_n, m and n .

Example 3. If we take $s \in S_m$, $s' \in S_n$, $V(\{s\}) = \{p\}$, $V(\{s'\}) = \{q\}$, then the distributed state $\{s, s'\}$ satisfies $\mathsf{m}p \wedge \mathsf{n}q$.

Example 4. The implication $\mathsf{m}F \wedge F' \rightarrow \mathsf{m}F \wedge \mathsf{m}F'$ holds, while the converse does not. Indeed, for $ds = \{s, s'\} \subseteq S_m$, and $V(\{s\}) = \{p\}$, $V(\{s'\}) = \{q\}$, we have $ds \models \mathsf{m}p \wedge \mathsf{m}q$, but not $ds \models \mathsf{m}\,(p \wedge q)$. With an eye to the full logic YALL, this non–equivalence is useful to specify that an event can have different future effects, without constraining them to occur in the same state (see Section 5). Finally, $\mathsf{m}F \vee F' \leftrightarrow \mathsf{m}F \vee \mathsf{m}F'$.

Example 5. The formula $\mathsf{mn}F$ is false. In fact, $ds \models \mathsf{mn}F$ if and only if there exists an $s \in S_n \cap S_m \cap ds$ such that $\{s\} \models F$, but no such s can exist since S_m and S_n are disjoint. Conversely, $\mathsf{mm}F$ is satisfiable, and it is equivalent to $\mathsf{m}F$.

Example 6. The formula $\mathsf{m}\top$ is satisfied by all the distributed states ds such that $ds \cap S_m \neq \emptyset$.

Axiom system. We propose the following axiomatization for DSL.

PC	axioms of the propositional calculus
K	$\bar{\mathsf{m}}(F \rightarrow F') \;\rightarrow\; (\bar{\mathsf{m}}F \rightarrow \bar{\mathsf{m}}F')$
DSL1	$\bar{\mathsf{m}}(\bar{\mathsf{m}}F \leftrightarrow F)$
DSL2	$\bar{\mathsf{m}}\bar{\mathsf{n}}\bot$
MP	$\dfrac{F \quad F \rightarrow F'}{F'}$ **Nec** $\dfrac{F}{\bar{\mathsf{m}}F}$

Example 7. The following formulae can be derived. Formulae are followed by the list of axioms or rules used in the proof. Proofs are in the appendix.

axiom 4	$\bar{\mathsf{m}}F \;\rightarrow\; \bar{\mathsf{m}}\bar{\mathsf{m}}F$	$[DSL1, K]$
D1	$\mathsf{mm}F \leftrightarrow \mathsf{m}F$	$[DSL1, K, PC]$

D2	$\mathsf{m}(F \wedge F') \rightarrow (\mathsf{m}F \wedge \mathsf{m}F')$	$[PC, Nec, K]$
D3	$\bar{\mathsf{m}}(F \rightarrow F') \ \rightarrow \ (\mathsf{m}F \rightarrow \mathsf{m}F')$	$[Nec, K, MP, PC]$
D4	$\bar{\mathsf{m}}F \rightarrow (\mathsf{m}\top \rightarrow \mathsf{m}F)$	$[D3]$
D5	$\bar{\mathsf{m}}(\mathsf{m}F \leftrightarrow F)$	$[DSL1, PC]$
D6	$(\bar{\mathsf{m}}(F \rightarrow F') \wedge \bar{\mathsf{m}}(F' \rightarrow F'')) \rightarrow \bar{\mathsf{m}}(F \rightarrow F'')$	$[Nec, K]$
D7	$\mathsf{m}(F \vee F') \leftrightarrow (\mathsf{m}F \vee \mathsf{m}F')$	$[D3, Nec, K, PC]$
D8	$\bar{\mathsf{m}}((\mathsf{m}F \wedge \mathsf{m}F') \rightarrow \mathsf{m}(F \wedge F'))$	$[D5, D6, D7, Nec, K]$

The soundness of the axioms is easy to see. We prove completeness.

Completeness. Let $(W^{DSL}, R_1^{DSL}, \ldots, R_k^{DSL}, V^{DSL})$ be the canonical model for DSL. We recall that worlds in W^{DSL} are maximal consistent sets of DSL formulae (DSL–MCS in the following), and that $(u, v) \in R_i^{DSL}$ if and only if $\bar{\mathsf{m}}_\mathsf{i}F \in u \ \rightarrow \ F \in v$. We need to show that, for all i, R_i^{DSL} satisfies conditions (4)–(6).

Cond. (4): we prove that $(u, v) \in R_i^{DSL} \ \rightarrow \ (v, v) \in R_i^{DSL}$

Suppose $\bar{\mathsf{m}}_\mathsf{i}F \in v$. u is a DSL–MCS and hence (see DSL1) $\bar{\mathsf{m}}_\mathsf{i}(\bar{\mathsf{m}}_\mathsf{i}F \rightarrow F) \in u$. But $(u, v) \in R_i^{DSL}$, hence $\bar{\mathsf{m}}_\mathsf{i}F \rightarrow F \in v$. Thus, by modus ponens, $F \in v$.

Cond. (5): we prove that $(u, v) \in R_i^{DSL}$ and $(v, w) \in R_i^{DSL}$ imply $v = w$

It is sufficient to prove that $v \subseteq w$. In fact, v and w are DSL–MCS and it is not the case that $v \subset w$, thus $v = w$. Let $F \in v$. u is a DSL–MCS and hence (see DSL1) it includes $\bar{\mathsf{m}}_\mathsf{i}(F \rightarrow \bar{\mathsf{m}}_\mathsf{i}F)$. But $(u, v) \in R_i^{DSL}$, hence $F \rightarrow \bar{\mathsf{m}}_\mathsf{i}F \in v$. Thus, by modus ponens, $\bar{\mathsf{m}}_\mathsf{i}F \in v$. As $(v, w) \in R_i^{DSL}$, we conclude that $F \in w$.

Cond. (6): we prove that $(u, v) \in R_i^{DSL}$ implies $\nexists w. \ (v, w) \in R_j^{DSL}$, for $j \neq i$

Assume $(v, w) \in R_j^{DSL}$. As u is a DSL–MCS, it includes $\bar{\mathsf{m}}_\mathsf{i}\bar{\mathsf{m}}_\mathsf{j}\bot$ (DSL2). As $(u, v) \in R_i^{DSL}$, then $\bar{\mathsf{m}}_\mathsf{j}\bot \in v$. As $(v, w) \in R_j^{DSL}$, then $\bot \in w$, which is an absurd. □

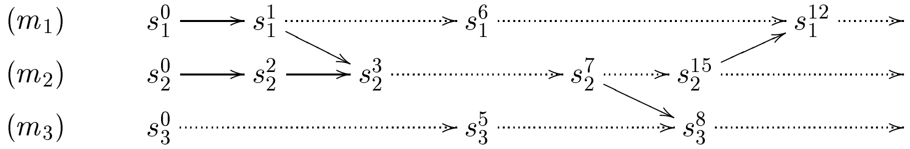
3 Adding time: a fragment of YALL

YALL extends DSL adding temporal operators. Since the focus of this work is on DSL, we consider only a fragment of YALL. The full logic adds pragmatically useful operators, but almost no more technical issue. Formulae are built as follows:

$$\phi \ ::= \ F \mid F \ \textsc{leads\_to} \ F' \mid F \ \textsc{because} \ F'$$

where F, $F' \in DSL$. Operator LEADS_TO expresses a liveness condition, and is similar to Unity's $\mapsto$ (leads to). Operator BECAUSE expresses a safety condition, and says that F must be preceded by F'.

Semantics. The models for YALL are built on structures like the one in the following figure, which describes the computation of a system with three components (m_1, m_2, m_3). We call s_i^j the j^{th} state of component m_i. Plain arrows denote single transitions and communications, dotted arrows denote a sequence of them, and define a causal dependency relationship.



The causal dependencies between the states of a computation define a partial order relation R^*, where $(s, s') \in R^*$ if and only if s' causally depends on s. For example, in the computation above, (s_1^0, s_1^1), (s_1^0, s_2^3), (s_1^1, s_2^3), $(s_2^2, s_1^{12}) \in R^*$.

A model $\mathcal{M}$ for YALL formulae is a tuple $(DS, R_1, \ldots, R_k, \leq, V)$, where:

$$ds \leq ds' \text{ iff } \forall s \in ds, \exists s' \in ds'. (s, s') \in R^* \text{ and } \forall s' \in ds', \exists s \in ds. (s, s') \in R^*$$

Let $\mathcal{M}$ be a model, and ds_0 the set of the initial states of $\mathcal{M}$, we define:

$$\begin{array}{lll} \mathcal{M} \models_Y F & \text{iff} & \forall ds \geq ds_0.\, ds \models F \\ \mathcal{M} \models_Y F \text{ LEADS\_TO } F' & \text{iff} & \forall ds \geq ds_0.\, ds \models F \text{ implies } \exists\, ds' \geq ds.\, ds' \models F' \\ \mathcal{M} \models_Y F \text{ BECAUSE } F' & \text{iff} & \forall ds \geq ds_0.\, ds \models F \text{ implies } \exists\, ds' \leq ds.\, ds' \models F' \end{array}$$

where $\models$ is the DSL satisfiability relation.

Rules. We present the most useful rules of the logic. Among them, LEADS_TO and BECAUSE introduction, strengthening of premises and weakening of consequences of LEADS_TO and BECAUSE, transitivity.

First, we observe that the definition for $\mathcal{M} \models_Y F$ entails that a necessitation rule holds (we use $\vdash_{DSF}, \vdash_{YALL}$ for the sake of comprehension). Rule **LI** permits LEADS_TO introduction. **LTR** is the rule for transitivity, **LWS** permits the strengthening of the premise, and the weakening of the consequences, and **LPD** and **LCC** stay for premise disjunction and consequence conjunction, respectively.

The rules for the BECAUSE operator (**BI**, **BTR**, **BSW**, **BPD**, and **BCC**) are obtained by substituting LEADS_TO with BECAUSE in the corresponding rules for LEADS_TO.

$$\frac{(\vdash_{DSF})\ F}{(\vdash_{YALL})\ F}\,\textbf{Nec} \qquad \frac{F \to G}{F \text{ LEADS\_TO } G}\,\textbf{LI} \qquad \frac{F \text{ LEADS\_TO } F' \quad F' \text{ LEADS\_TO } F''}{F \text{ LEADS\_TO } F''}\,\textbf{LTR}$$

$$\frac{G \to F \quad F \text{ LEADS\_TO } F' \quad F' \to G'}{G \text{ LEADS\_TO } G'}\,\textbf{LSW}$$

$$\frac{F' \text{ LEADS\_TO } F \quad F'' \text{ LEADS\_TO } F}{F' \vee F'' \text{ LEADS\_TO } F}\,\textbf{LPD} \qquad \frac{F \text{ LEADS\_TO } F' \quad F \text{ LEADS\_TO } F''}{F \text{ LEADS\_TO } F' \wedge F''}\,\textbf{LCC}$$

4 An Example: Private Keys

Consider the system $\{b, t, u\}$, where b is a component that broadcasts the encrypted version of a message to all the other components in the system, i.e. t (trusted) and u (untrusted). We assume that these components try to decrypt the message. We represent with predicate p the fact that the message is readable, and with predicate *dep* the fact that a decryption has been attempted. However, the decryption yields p if and only if the key is held. Predicate *key* represents the property of holding the key.

4.1 Reasoning on distributed states: DSL

The properties of the distributed states of the system are described by the following DSL formulae:

$$(\sim \mathsf{b}\top) \to ((key \wedge dep) \leftrightarrow p) \tag{7}$$

$$\bar{\mathsf{t}}\, key \tag{8}$$

$$\bar{\mathsf{u}} \sim key \tag{9}$$

Formula (7) tells that in all components, with the exception of b, p and dep are equivalent only if the key is held. Indeed, if (7) holds, as required, in all $ds \in DS$, it holds in particular in all ds which are singleton sets. So, it holds for all $\{s_t\}$ and $\{s_u\}$. Since in these states the premise of (7) is satisfied, so it is the conclusion, i.e. in all states of t and u: $(key \wedge dep) \leftrightarrow p$. We derive the property for t:

$$\dfrac{\dfrac{(\sim \mathsf{b}\top) \to ((key \wedge dep) \leftrightarrow p)}{\bar{\mathsf{t}}(\sim \mathsf{b}\top) \to \bar{\mathsf{t}}((key \wedge dep) \leftrightarrow p)}\,\text{Nec, K} \qquad \dfrac{\bar{\mathsf{t}}\,\bar{\mathsf{b}}\bot}{\bar{\mathsf{t}}(\sim \mathsf{b}\top)}}{\bar{\mathsf{t}}((key \wedge dep) \leftrightarrow p)}\,\text{MP}$$

Component t holds the key (8), while component u does not (9). We derive that t is able to correctly decrypt the message. We concentrate in one of the two implication, i.e. $(key \wedge dep) \to p$ and prove that $\bar{\mathsf{t}}\,(dep \to p)$ (the top line is a tautology of the propositional calculus):

$$\dfrac{\dfrac{\dfrac{\dfrac{((key \wedge dep) \to p) \to (key \to (dep \to p))}{\bar{\mathsf{t}}((key \wedge dep) \to p) \to \bar{\mathsf{t}}(key \to (dep \to p))}\,\text{Nec, K} \qquad \bar{\mathsf{t}}((key \wedge dep) \leftrightarrow p)}{\bar{\mathsf{t}}\,(key \to (dep \to p))}\,\text{MP}}{\bar{\mathsf{t}}\,key \to \bar{\mathsf{t}}\,(dep \to p)}\,\text{K} \qquad \bar{\mathsf{t}}\,key}{\bar{\mathsf{t}}\,(dep \to p)}\,\text{MP}$$

We now consider component u and prove that $\bar{\mathsf{u}} \sim p$ holds, i.e. that the untrusted component is not able to read the message. We consider the implication $p \to (key \wedge dep)$ (the top line is a tautology of the propositional calculus):

$$\dfrac{\dfrac{\dfrac{(p \to (key \wedge dep)) \to (\sim key \to \sim p)}{\bar{\mathsf{u}}\,(p \to (key \wedge dep)) \to \bar{\mathsf{u}}\,(\sim key \to \sim p)}\,\text{Nec, K} \qquad \bar{\mathsf{u}}\,((key \wedge dep) \leftrightarrow p)}{\bar{\mathsf{u}}\,(\sim key \to \sim p)}\,\text{MP} \qquad \bar{\mathsf{u}} \sim key}{\bar{\mathsf{u}} \sim p}\,\text{K,MP}$$

4.2 Reasoning on distributed computations: YALL

We now add some constraints on the temporal behaviour of the private keys system: as soon as the message is readable in b, b broadcasts its encrypted version (10); t and u try to decrypt the message (11, 12).

$$\mathsf{b}\,p \;\; \textsc{leads\_to} \;\; \mathsf{t}\,ep \wedge \mathsf{u}\,ep \tag{10}$$
$$\mathsf{t}\,ep \;\; \textsc{leads\_to} \;\; \mathsf{t}\,dep \tag{11}$$
$$\mathsf{u}\,ep \;\; \textsc{leads\_to} \;\; \mathsf{u}\,dep \tag{12}$$

Note that $\bar{\mathsf{u}} \sim p$ still holds, by Nec. We prove that $\mathsf{b}\,p$ $\textsc{leads\_to}$ $\mathsf{t}\,p$. We exploit the conclusion of Section 4.1: $\bar{\mathsf{t}}\,(dep \to p)$

$$\dfrac{\dfrac{\dfrac{\mathsf{b}\,p \;\textsc{leads\_to}\; \mathsf{t}\,ep \wedge \mathsf{u}\,ep \quad (\mathsf{t}\,ep \wedge \mathsf{u}\,ep) \to \mathsf{t}\,ep}{\mathsf{b}\,p \;\textsc{leads\_to}\; \mathsf{t}\,ep}\,\text{LSW} \qquad \mathsf{t}\,ep \;\textsc{leads\_to}\; \mathsf{t}\,dep}{\mathsf{b}\,p \;\textsc{leads\_to}\; \mathsf{t}\,dep}\,\text{LTR} \qquad \dfrac{\bar{\mathsf{t}}\,(dep \to p)}{\mathsf{t}\,dep \to \mathsf{t}\,p}\,\text{D3}}{\mathsf{b}\,p \;\textsc{leads\_to}\; \mathsf{t}\,p}\,\text{LSW}$$

5 Discussion and Related Work

The semantic domain of DSL. The choice of using 2^S as a semantic domain for the distributed state logic formulae, and the non–equivalence between $\mathsf{m}\,(F \wedge F)$ and $\mathsf{m}\,F \wedge \mathsf{m}\,F'$ are useful to specify that a given condition can have different future effects, without constraining them to occur in the same state. Similarly, we can express complex preconditions in a temporal formula. For instance, assume we want to specify and reason on the delivery of credit cards to customers. The bank, for security reasons, sends separately the card and the code. Once the customer has got both of them, he is allowed to withdraw money from an ATM machine:

$$\mathsf{bank}\,\mathit{new\_card}\ \ \textsc{leads\_to}\ \ \mathsf{user}\,\mathit{receive\_card} \wedge \mathsf{user}\,\mathit{receive\_code} \tag{13}$$
$$\mathsf{user}\,\mathit{can\_withdraw}\ \ \textsc{because}\ \ \mathsf{user}\,\mathit{receive\_card} \wedge \mathsf{user}\,\mathit{receive\_code} \tag{14}$$

The equivalence between $\mathsf{m}\,(F \wedge F)$ and $\mathsf{m}\,F \wedge \mathsf{m}\,F'$ would have required the following specification:

$$\mathsf{bank}\,\mathit{new\_card}\ \ \textsc{leads\_to}\ \ \mathsf{user}\,\mathit{receive\_card} \tag{15}$$
$$\mathsf{bank}\,\mathit{new\_card}\ \ \textsc{leads\_to}\ \ \mathsf{user}\,\mathit{receive\_code} \tag{16}$$
$$\mathsf{user}\,\mathit{can\_withdraw}\ \ \textsc{because}\ \ \mathsf{user}\,\mathit{receive\_card} \tag{17}$$
$$\mathsf{user}\,\mathit{can\_withdraw}\ \ \textsc{because}\ \ \mathsf{user}\,\mathit{receive\_code} \tag{18}$$

since (13), (14) would be too restrictive, asking for card and code to be received at the same time.

Last, but not least, with an eye to a 1^{st} order extension, a formula like (13) makes it easier to bind variables in *card* and *code* than it would be with the distinguished formulae (15), (16).

Classical Logic. Another point of discussion is why we need a modality (m) rather than a distinguished propositional symbol $here_m$, to replace systematically each sub–formula $\mathsf{m}\,F$ with $here_m \wedge F$. The motivation is that we do not want the equivalence between $\mathsf{m}\,(F \wedge F')$ and $\mathsf{m}\,F \wedge \mathsf{m}\,F'$, as discussed previously. On the contrary, the two translations $here_m \wedge F \wedge F'$ and $here_m \wedge F \wedge here_m \wedge F'$ would be equivalent.

Hybrid Logic. Hybrid logic allows the specifier to directly refer to specific points (states) in the model, through the use of *nominals* [1]. A nominal i is an atom which is true at exactly one point in any model. The operator $@_i$ permits to jump to the point named by nominal i. We might consider defining an hybrid signature including distinguished sets of state variables, one for each component, and translate $\mathsf{m}\,F$ in $\exists x.\ @_x F$, where x is a state variable in the appropriate set. However, we claim that the resulting setting would be more complex than the one offered by DSL.

Metric and Layered Temporal Logic. Some similarities can be found between our location operator and the $MLTL$ operators defined in [15], that make it possible to compose formulae associated with different time granularities and to switch from one granularity to another. Time instants are organized in temporal domains, and the set of temporal domains is totally ordered with respect to the coarseness of the domain elements. To look for an embedding of DSL, we can consider three domains: *system*, with a unique element; *components*, whose elements are the components $m_1, \ldots, m_k$; *states*, the domain of the states. Then the formulae are translated using an appropriate combination of $MLTL$ operators. For instance, the translation of $\mathsf{m}\,F$ should be $\diamond\Delta_m^{components}\diamond\exists\alpha\Delta_\alpha^{states}F$. Also in this case, we claim that, since the full expressive power of $MLTL$ is not needed, the simpler framework of DSL is of pragmatical interest.

Other logics for distributed systems. Various extensions of temporal logic have been defined in the literature to deal with distributed systems.

In TTL [14], for each local state of the system, a *visibility* function specifies which remote information is accessible. The visibility function is defined on the basis of a relation among states which is *symmetric* in the case of states belonging to distinguished components.

A trace based extension of linear time temporal logic, called $TrPTL$, has been defined in [23] (see also [24]). The logic has been designed to be interpreted over infinite traces, i.e., labelled partial orders of actions, which respect some dependence relations associated to the alphabet of actions.

In [11], a temporal logic, StepTL, is defined and interpreted over multistep transition systems. These are a well known extension of transition systems, permitting to describe as concurrent the steps of computation that can actually be executed in parallel. A multistep transition system thus contains transitions of the form $s\,A\,s'$, where A is a set of actions, instead of a single one.

Three distinguished logics are presented in [21] to describe systems composed of sets of communicating *agents*. The logics differentiate on the amount of information each agent can have on the other agents running on the system, but share a common setting: agents communicate via common actions. The models for these logics are runs of networks of synchronizing automata. The logics $\mathbf{D}_0$ and $\mathbf{D}_1$ presented in [3] are based on a similar approach.

In all these proposals, components communicate via some form of synchronization, and logic formulae are interpreted on models shaping:

$$\begin{tikzcd}
(m) & {} \arrow[rr, dotted] & & a \arrow[r, dotted] & b \arrow[rr, dotted] \arrow[d, leftrightarrow] & & c \arrow[r, dotted] & {} \\
(n) & {} \arrow[r, dotted] & d \arrow[rr, dotted] & & e \arrow[r, dotted] & f \arrow[rr, dotted] & & {}
\end{tikzcd}$$

Therefore, in any logic defined over these models, it is not possible to express the asymmetric nature of causality we are interested in when modelling the behaviour of agents communicating asynchronously by message passing. Indeed, in the previous model we can both assert that a causes f and that d causes c.

A logic closer to YALL is proposed in [13], where a branching time temporal logic for asynchronously communicating sequential agents (ACSAs) is defined. ACSAs communicate asynchronously via message passing. The logic contains temporal modalities indexed with a local point of view of one agent and allows an agent “i” to refer to local properties of another agent “j” according to the latest message received: an agent can gain information about another agent by receiving messages but not by sending them. We allow agents to make remote future assertions: therefore it is easier to express global liveness properties.

Knowledge Logic. A logic to reason on asynchronous message passing systems is proposed in [4]. The language used, $\mathcal{L}_n^U$, is obtained by extending their language of knowledge with the modal operators U and $\bigcirc$. Formulae in $\mathcal{L}_n^U$ permit to express how the n *agents* in a system gain knowledge over time. A set of characteristic formulae valid in the logic are presented, but a sound and complete axiom system is not defined. The authors focus their attention on systems based un-reliable communications, while only state that properties of reliable communications can be expressed. A major difference with our work relies on the models used to interpret $\mathcal{L}_n^U$ formulae. Even if the knowledge of the agents is limited to thier current local histories, i.e. sequences of messages sent or received and of internal actions, interpretation structures are based on the notions of global time and global state.

Partial Order Temporal Logics.

Partial Order Temporal Logics (POTL) [20] permits to deal with the causal relationships between the events of a set of processes executing concurrenlty. The Interleaving Set Temporal Logic (ISTL) [10] extends POTL with features form linear temporal logic and braching temporal logic. The Kripke structures for both logics are very defiierent from the one defined in this paper.

We are addressing a specific class of systems that we consider very relevant nowadays, that is distributed systems with asynchronous message passing. These systems have a few notable characteristics: there is no global state, and interactions among components occur only via messages. As a consequence, a specification is essentially devoted to describing the causal relationships among the components. We think that these characterististics are so important that the designer working on a specification will greatly benefit if they are naturally embedded already in the basic model he is using. Hence, the investigation in Kripke's structures presented in this paper.

DSL vs Oikos–*adtl*. The work reported here stems from our experience with Oikos–*adtl*, a specification language for distributed systems based on asynchronous communications, designed to support the composition of specifications [18]. Oikos–*adtl* is intended to give designers a language to express the properties of interest in a natural way, and it is associated with a refinement method which supports the gradual introduction of details, as design proceeds. It has been used to specify software architectures and patterns [16, 9] and to analyse security issues in mobile systems [8, 6, 5]. It is supported by a proof assistant, Mark [7], that deploys a number of proof strategies that partially automate property verification.

Coming back to our motivating example in the introduction, in Oikos–*adtl* it is possible to weaken the consequences of a formula like (1) including operator LEADS_TO, but the rule shapes

$$\frac{\mathsf{m}\,p \;\; \text{LEADS\_TO} \;\; \mathsf{n}\,q \wedge \mathsf{o}\,r}{\mathsf{m}\,p \;\; \text{LEADS\_TO} \;\; \mathsf{n}\,q}$$

since a formula like (2) is not part of the logic. So, the price is writing one rule for each possible weakening relation.

6 Conclusions

To reason on global applications, we have introduced the temporal logic YALL. Models for YALL are space–time diagrams describing the behaviour of a set of components communicating asynchronously. The logic has been introduced in two steps. First, we have defined DSL, a modal logic for localities that embeds the theories describing the local states of each component into a theory of the distributed states of the system. No notion of time or state transition is present at this stage. To support reasoning in the logic, we have presented a sound and complete axiom system. Then, we have added a temporal operator, and the corresponding derivation rules. The contribution is that it is possible to reason about properties that involve several components, even in the absence of a global clock, which is a meaningless notion in an asynchronous setting. The logic has been used to reason on the properties of a simple secure communication system.

Future work includes the extension of YALL to predicate logic, the introduction of an *event* operator, the study of compositionality results, and a revision of the theorem prover Mark. We foresee that formulae in the 1^{st} order extension will shape $\mathsf{m}\, p(x)$ LEADS_TO $\mathsf{n}\, q(x, y)$, and be interpreted as $\forall x.[\mathsf{m}\, p(x)$ LEADS_TO $\exists y.\, \mathsf{n}\, q(x, y)]$. This way, the semantics should smoothly extend that of YALL. Compositionality results will permit to derive the properties satisfied by a system from the properties satisfied by its components when executed in isolation. This requires reasoning on the possible interferences due to communications from the added components.

Acknowledgments

We gratefully thank Massimo Franceschet, Angelo Montanari, and Francesca Scozzari for interesting discussions on a draft of the paper. The work was supported by Projects Sahara and Degas.

References

1. C. Areces. *Logic Engineering. The Case of Description and Hybrid Logics.* PhD thesis, ILLC, University of Amsterdam, The Netherlands., 2000.
2. K.M. Chandy and J. Misra. *Parallel Program Design: A Foundation.* Addison-Wesley, Reading Mass., 1988.
3. H.-D. Ehrich, C. Caleiro, A. Sernadas, and G. Denker. Logics for specifying concurrent information systems. In J. Chomicki and G. Saake, editors, *Logic for Databases and Information Systems*, pages 167–198. Kluver Academic Publishers, 1998.
4. R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge.* MIT Press, 1995.
5. G. Ferrari, C. Montangero, L. Semini, and S. Semprini. Refinement Calculus for Mobility: expressing security policies. In J. Vitek, editor, *Workshop on Distributed Object Security*, Denver, Co, 1999.
6. G. Ferrari, C. Montangero, L. Semini, and S. Semprini. Ad Hoc Network Applications: Specification, Design, and Verification in mob_{adtl}. In G.-C. Roman and G.P. Picco, editors, *Proc. ICSE'01 Workshop on Software Engineering and Mobility*, Toronto, CA, 13,14 May 2001.
7. G. Ferrari, C. Montangero, L. Semini, and S. Semprini. Mark, a reasoning kit for mobility. *Automated Software Engineering*, 2002. To appear.
8. G.Ferrari, C.Montangero, L.Semini, and S.Semprini. Multiple Security Policies in Mob_{adtl}. In *Proc. Work. on Issues in the Theory of Security*, Geneva, 2000.
9. A. Hudaib and C. Montangero. Formal Patterns for Secure Software Architectures in Oikos_adtl. In *WSDAAL '01*, Como, It., Sept. 2001.
10. S. Katz and D. Peled. Interleaving set temporal logic. *Theoretical Computer Science*, 75(3):263–287, 1990.
11. K.Lodaya, R.Parikh, R.Ramanujam, and P.S.Thiagarajan. A Logical Study of Dis- tributed Transition Systems. *Information and Computation*, 119(1):91–118, '95.
12. H. Kreger. Web Services Conceptual Architecture. IBM Software Group, May 2001. www-4.ibm.com/software/solutions/webservices/pdf/WSCA.pdf.
13. Lodaya, Ramanujam, and Thiagarajan. Temporal logics for communicating sequential agents: I. *Int. Journal of Found. of Computer Science*, 3(2):117–159, '92.
14. A. Masini and A. Maggiolo-Schettini. TTL: A formalism to describe local and global properties of distributed systems. *Informatique théorique et Applications/Theoretical Informatics and Applications*, 26(2):115–149, 1992.
15. A. Montanari. *Metric and Layered Temporal Logic for Time Granularity.* PhD thesis, University of Amsterdam, 1996. Chapter 3.
16. C. Montangero and L. Semini. Refining by Architectural Styles or Architecting by Refinements. In *2nd Int. Software Architecture Workshop, Proc. of the SIGSOFT '96 Workshops*, pages 76–79, San Francisco, CA, Oct 1996. ACM Press.

17. C. Montangero and L. Semini. Software specification and design: from formal methods to standard middleware. In *Proc. 6th ERCIM Inter. Workshop on Formal Methods for Industrial Critical Systems (FMICS'01)*, 2001.
18. C. Montangero and L. Semini. Composing Specifications for Coordination. In P. Ciancarini and A. Wolf, editors, *Proc. COORDINATION 99*, volume 1594 of *LNCS*, pages 118–133, Amsterdam, April 99. Springer-Verlag.
19. Microsoft .NET. at http://research.microsoft.com/project7.net/project_7.htm.
20. Pinter and Wolper. A temporal logic for reasoning about partially ordered specifications. In *Proc.* 3^{th} *ACM Principles of Distributed Computing*, pages 28–37, Vancouver, B.C., 1984.
21. R. Ramanujam. Locally linear time temporal logic. In *Proc.* 11^{th} *IEEE Symp. on Logic In Computer Science*, pages 118–127. IEEE Computer Society, 1996.
22. L. Semini and C. Montangero. A Refinement Calculus for Tuple Spaces. *Science of Computer Programming*, 34:79–140, 1999.
23. P. S. Thiagarajan. A trace based extension of linear time temporal logic. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 438–447, Paris, Jul 1994. IEEE Computer Society Press.
24. P. S. Thiagarajan and J. G. Henriksen. Distributed versions of linear time temporal logic: A trace perspective. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 643–681. Springer-Verlag, 1998.
25. C. Wille. *Presenting C#*. SAMS Publishing, 2000.

Appendix: Proofs from Example 7

Axiom 4

$$\dfrac{\bar{\mathsf{m}}(\bar{\mathsf{m}}F \leftarrow F)}{\bar{\mathsf{m}}\bar{\mathsf{m}}F \leftarrow \bar{\mathsf{m}}F}\,K$$

D1

$$\dfrac{\dfrac{\dfrac{\dfrac{\bar{\mathsf{m}}(\bar{\mathsf{m}} \sim F \leftrightarrow \sim F)}{\bar{\mathsf{m}}\bar{\mathsf{m}} \sim F \leftrightarrow \bar{\mathsf{m}} \sim F}\,K}{\sim \mathsf{m} \sim\sim \mathsf{m} \sim\sim F \leftrightarrow \sim \mathsf{m} \sim\sim F}}{\sim \mathsf{m}\mathsf{m}F \leftrightarrow \sim \mathsf{m}F}}{\mathsf{m}F \leftrightarrow \mathsf{m}\mathsf{m}F}$$

D2 We show that $\mathsf{m}(F \wedge F') \rightarrow \mathsf{m}F$, $\mathsf{m}(F \wedge F') \rightarrow \mathsf{m}F'$ is proved analogously.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{F \wedge F' \rightarrow F}{\sim (F \wedge F') \leftarrow \sim F}}{\bar{\mathsf{m}}(\sim (F \wedge F') \leftarrow \sim F)}\,Nec}{\bar{\mathsf{m}} \sim (F \wedge F') \leftarrow \bar{\mathsf{m}} \sim F}\,K}{\sim \mathsf{m} \sim\sim (F \wedge F') \leftarrow \sim \mathsf{m} \sim\sim F}}{\mathsf{m}(F \wedge F') \rightarrow \mathsf{m}F}$$

D3

$$\dfrac{\dfrac{\dfrac{\dfrac{\bar{\mathsf{m}}(F \rightarrow F') \qquad \dfrac{(F \rightarrow F') \rightarrow (\sim F' \rightarrow \sim F)}{\bar{\mathsf{m}}(F \rightarrow F') \rightarrow \bar{\mathsf{m}}(\sim F' \rightarrow \sim F)}\,\text{Nec, K}}{\bar{\mathsf{m}}(\sim F' \rightarrow \sim F)}\,\text{MP}}{\bar{\mathsf{m}} \sim F' \rightarrow \bar{\mathsf{m}} \sim F}\,\text{K}}{\sim \mathsf{m}F' \rightarrow \sim \mathsf{m}F}\,\text{Def}}{\mathsf{m}F \rightarrow \mathsf{m}F'}\,\text{PC}$$

D5

$$\dfrac{\dfrac{\bar{\mathsf{m}}(\bar{\mathsf{m}} \sim F \leftrightarrow \sim F)}{\bar{\mathsf{m}}(\sim \mathsf{m} \sim\sim F \leftrightarrow \sim F)}}{\bar{\mathsf{m}}(\mathsf{m}F \leftrightarrow F)}$$

D6 For $A = F \to F'$, $B = F' \to F''$, and $C = F \to F''$

$$
\begin{array}{ll}
A \wedge B \to C & \\ \hline
\sim A \vee \sim B \vee C & \\ \hline
A \to (B \to C) & \\ \hline
\bar{\mathsf{m}} A \to \bar{\mathsf{m}}(B \to C) & Nec, K \\ \hline
\bar{\mathsf{m}} A \to (\bar{\mathsf{m}} B \to \bar{\mathsf{m}} C) & K \\ \hline
\sim \bar{\mathsf{m}} A \vee \sim \bar{\mathsf{m}} B \vee \bar{\mathsf{m}} C & \\ \hline
\sim (\bar{\mathsf{m}} A \wedge \bar{\mathsf{m}} B) \vee \bar{\mathsf{m}} C & \\ \hline
(\bar{\mathsf{m}} A \wedge \bar{\mathsf{m}} B) \to \bar{\mathsf{m}} C &
\end{array}
$$

D7 We prove $\mathsf{m}(F \vee F') \to (\mathsf{m}F \vee \mathsf{m}F')$ on the left, and $\mathsf{m}(F \vee F') \leftarrow (\mathsf{m}F \vee \mathsf{m}F')$ on the right.

$$
\begin{array}{ll}
(A \wedge B) \to (A \wedge B) & \\ \hline
\sim A \vee \sim B \vee (A \wedge B) & \\ \hline
A \to (B \to (A \wedge B)) & \\ \hline
\bar{\mathsf{m}} A \to \bar{\mathsf{m}}(B \to (A \wedge B)) & Nec, K \\ \hline
\bar{\mathsf{m}} A \to (\bar{\mathsf{m}} B \to \bar{\mathsf{m}}(A \wedge B)) & K \\ \hline
\sim \bar{\mathsf{m}} A \vee \sim \bar{\mathsf{m}} B \vee \bar{\mathsf{m}}(A \wedge B) & \\ \hline
\sim (\bar{\mathsf{m}} A \wedge \bar{\mathsf{m}} B) \vee \bar{\mathsf{m}}(A \wedge B) & \\ \hline
(\bar{\mathsf{m}} A \wedge \bar{\mathsf{m}} B) \to \bar{\mathsf{m}}(A \wedge B) & \\ \hline
(\sim \mathsf{m} \sim A \wedge \sim \mathsf{m} \sim B) \to \sim \mathsf{m} \sim (A \wedge B) & \\ \hline
\sim (\mathsf{m} \sim A \vee \mathsf{m} \sim B) \to \sim \mathsf{m}(\sim A \vee \sim B) & \\ \hline
(\mathsf{m} \sim A \vee \mathsf{m} \sim B) \leftarrow \mathsf{m}(\sim A \vee \sim B) & \\ \hline
\mathsf{m}(F \vee F') \to (\mathsf{m}F \vee \mathsf{m}F') & F \equiv \sim A, \; F' \equiv \sim B
\end{array}
$$

$$
\dfrac{
\dfrac{\dfrac{F \to (F \vee F')}{\bar{\mathsf{m}}(F \to (F \vee F'))}\,Nec \qquad D3}{\mathsf{m}F \to \mathsf{m}(F \vee F')}\,MP
\qquad
\dfrac{\dfrac{F' \to (F \vee F')}{\bar{\mathsf{m}}(F' \to (F \vee F'))}\,Nec \qquad D3}{\mathsf{m}F' \to \mathsf{m}(F \vee F')}\,MP
}{(\mathsf{m}F \vee \mathsf{m}F') \to \mathsf{m}(F \vee F')}
$$

D8 If we prove $\bar{\mathsf{m}}((\mathsf{m}F \wedge \mathsf{m}F') \to (F \wedge F'))$ and $\bar{\mathsf{m}}((F \wedge F') \to \mathsf{m}(F \wedge F'))$ then we can apply D6 and conclude. The second formula is an instance of D5, we prove the first one:

$$
\dfrac{(\mathsf{m}F \to F \;\wedge\; \mathsf{m}F' \to F') \to (\mathsf{m}F \wedge \mathsf{m}F' \;\to\; F \wedge F')}{\bar{\mathsf{m}}(\mathsf{m}F \to F \;\wedge\; \mathsf{m}F' \to F') \to \bar{\mathsf{m}}(\mathsf{m}F \wedge \mathsf{m}F' \;\to\; F \wedge F')}\,Nec, K
$$

$$
\dfrac{\dfrac{\dfrac{D5}{\bar{\mathsf{m}}(\mathsf{m}F \to F) \;\wedge\; \bar{\mathsf{m}}(\mathsf{m}F' \to F')} \qquad \dfrac{D7}{(\bar{\mathsf{m}}F \wedge \bar{\mathsf{m}}F') \to \bar{\mathsf{m}}(F \wedge F')}}{\bar{\mathsf{m}}(\mathsf{m}F \to F \;\wedge\; \mathsf{m}F' \to F')}\,MP}{\bar{\mathsf{m}}(\mathsf{m}F \wedge \mathsf{m}F' \;\to F \wedge F')}\,MP, \text{with the conclusions of the previous derivation}
$$