# Modal Specifications

*Kim Guldstrand Larsen*
Aalborg University Center, DENMARK

**Abstract**

We present a theory of *Modal Specifications* which has been specifically designed in order to allow *loose* specifications to be expressed. Modal Specifications extends Process Algebra in the sense that specifications may be combined using process constructs. Moreover, Modal Specifications is given an *operational interpretation* imposing restrictions on the transitions of possible implementations by telling which transitions are *necessary* and which are *admissable*. This allows a *refinement ordering* between Modal Specifications to be defined extending the well–established notion of bisimulation. In the paper we present a logical characterization of the refinement–ordering and derive characteristic logical formulas from any given Modal Specifications. Also, we explore the possibility of combining Modal Specifications themselves logically, and we briefly comment on the automation of refinement.

## 1 Motivation

We think of *program development* as being logically divided into three phases: a *specification* phase, in which a collection of desired properties of the final program is described; the *implementation* phase, in which the program or process with the desired properties is being developed; and finally, the *verification* phase, which consists of a formal justification of the correctness of the designed implementation with respect to the initial specification.

Obviously, when developing large systems, we do not expect to be able to derive an implementation immediately from the initial specification $S_0$. Rather, we expect the implementation phase to consist of a series of small and successive refinements of the initial specification until eventually an implementation $I$ can be extracted directly (so–called *stepwise refinement*). In order to guarantee the correctness of the extracted implementation with respect to the initial specification each individual refinement step must *preserve* correctness; i.e. if $S_{i+1}$ is the refinement of $S_i$, then any implementation correct with respect to $S_{i+1}$ must also be correct with respect to $S_i$. Identifying a specification with its set of correct implementations, the implementation phase can thus be illustrated as a decreasing chain of sets:

$$S_0 \supseteq S_1 \supseteq \cdots\cdots \supseteq S_n \qquad (1)$$

with the final implementation $I$ being a member of $S_n$.

Now, consider any intermediate specification as a *partial implementation.* That is, certain implementation decisions may have been made, but there still remains to be implemented a number of subcomponents according to given subspecifications. A refinement–step (see figure 1) will then simply consist of a refinement of one of the components. Moreover, the correctness of the global refinement–step ought to be immediately implied by the correctness of the refinement of the component, as this obviously will greatly simplify the task of verification. In this case, the combined theory of the specification and implementation language is said to be *compositional.*

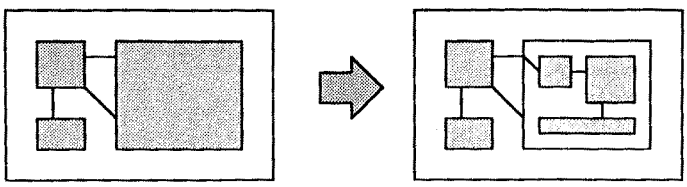


Figure 1: A refinement–step

Through the study of examples it has become evident that the analyses involved in the verification of a refinement–step is often tedious and delicate even for moderate size examples. Thus, it is rather clear that *computer assistance* is essential; not just to ensure the correctness of the verification but even to make the analyses feasible. However, it is important that the tools provided do more than just check and report the correctness (or lack of correctness) of a given refinement–step. The tool must also provide *explanations* of the answers they give: a correct refinement–step has most likely been preceded by several incorrect attempts; knowing *why* these attempts where erroneous would obviously be of great help in the pursuit of a correct one.

Within the framework of Process Algebra — CCS being a prominent example — the languages for specification and implementation coincide. The only difference (if any) between specifications and implementations will be their computational feasibility (in whatever model of computation that is used). The notion of correctness of an implementation with respect to a given specification may be defined as an *abstracting equivalence* between processes. Especially, the equivalence induced by the notion of *bisimulation* [Park 80, Mil80] enjoys many pleasant properties: it admits a very elegant proof technique based on fixed point induction [Park 80]; it has an alternative modal characterization [HenMil 85]; it is preserved by all natural process constructions (including those of CCS) [Lar 86, Sim85, GroVan89] and thus supports compositionality; inequivalent processes can be distinguished by (probabilistic) testing [Abr87, LarSkou89]; and the equivalence of finite–state processes can be automatically decided in polynomial time [KanSmo83, PaigeTar]. In fact a fair

number of systems has recently emerged that will (among other things) support verification of bisimulation equivalence [CW, KanSmo83, Auto, Tau], though only the prototype system of [Tau] offers some explanation in case processes are *in*equivalent.

So, what more can we ask for? Well, practical experience [LarMil 87, LT88b] has lead us to believe that Process Algebra is expressively too poor in order to provide a convenient specification language. Obviously, any specification you may care to write down in Process Algebra will limit the possible implementations to a single equivalence class. Thus, the inclusions in equation (1) simply reduces to equalities:

$$S_0 = S_1 = \cdots\cdots = S_n \tag{2}$$

We consider this undesirable, as it forces all decisions as to the behaviour of the implementation to be made as early as in the initial specification. What seems to be needed is *loose specifications* — i.e. specifications which are possibly satisfied by more than a single equivalence class of processes — so that certain behavioural aspects can be left to be decided later.

Also, in any compositional proof method, it is essential that one can express the behavioural constraints which is imposed upon each subsystem by the others, since it may be difficult to obtain a sufficiently simple subspecification for the subsystem's behaviour in the absence of the constraint. Again, loose specifications seems to be called for (see also [LarMil 87, HutLar89]).

In [LT88b] a theory of *Modal Specifications* was put forward with the specific intention of allowing *loose* specifications to be expressed. Modal Specifications extends Process Algebra in the sense that specifications may be combined using process constructs. Moreover, Modal Specifications is given an *operational interpretation* imposing restrictions on the transitions of possible implementations by telling which transitions are *necessary* and which are *admissable*. This allows a *refinement ordering* between Modal Specifications to be defined extending in a natural way the notion of bisimulation.

In section 2 we give a short review of Modal Specifications and refinement. In section 3 we present a logical characterization of the refinement ordering, hereby providing the theoretical foundation for explaining incorrect refinement-steps. In section 4 we derive logical formulas characterizing a given Modal Specification. This allows the reasoning about refinements between Modal Specifications to be addressed in a more conventional logical framework. In section 5 we explore the possibility of combining Modal Specifications logically; and finally in section 6 we comment on automating the reasoning about refinements between Modal Specifications.

## 2 Modal Specifications and Refinements

Modal Specifications is given an operational description in the style of [Pl81], using labelled transition systems. Modal specifications impose restrictions on the transitions of possible

implementations by telling which transitions are *necessary* and which are *admissible*. The transition systems for Modal Specifications therefore have *two* transition relations: $\longrightarrow_\Box$ describing the *required* transitions and $\longrightarrow_\Diamond$ describing the *allowed* transitions.

**Definition 2.1** *A* modal transition system *is a structure* $\mathcal{S} = (S, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$, *where* $S$ *is a set of specifications,* $A$ *is a set of actions and* $\longrightarrow_\Box$, $\longrightarrow_\Diamond \subseteq S \times A \times S$, *satisfying the condition* $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$. *Also, we shall impose the restriction of* image–finiteness *on* $\longrightarrow_\Diamond$, *i.e. for all* $T \in S$ *and* $a \in A$ *we assume the set* $\{T' \mid T \xrightarrow{a}_\Diamond T'\}$ *to be finite.*

The condition $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$ says that anything required is also allowed, ensuring that any modal specification is consistent. Thus, our use of modalities is a deontic one. The behaviour of processes themselves is assumed to be given in terms of a standard labelled transition system $\mathcal{P} = (P, A, \longrightarrow)$. We will view processes as specifications where all requirements are necessary ones, as reflected in the derived modal transition system $\mathcal{S} = (P, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$, with $\longrightarrow_\Box = \longrightarrow_\Diamond = \longrightarrow$.

Now, when is a specification $S$ a refinement of another specification $T$, in the sense that it allows fewer implementations? Intuitively, we would expect that any behavioural aspect *allowed* by $S$ should also be allowed by $T$; and dually, that the behavioural aspects which are already guaranteed by the weaker specification $T$ must also be guaranteed by $S$. Using the derivation relations $\longrightarrow_\Box$ and $\longrightarrow_\Diamond$ this may be formalized by the following notion of *refinement.*

**Definition 2.2** *A* refinement $R$ *is a binary relation on* $S$ *such that whenever* $SRT$ *and* $a \in A$ *then the following holds:*

1. *Whenever* $S \xrightarrow{a}_\Diamond S'$, *then* $T \xrightarrow{a}_\Diamond T'$ *for some* $T'$ *with* $S'RT'$,
2. *Whenever* $T \xrightarrow{a}_\Box T'$, *then* $S \xrightarrow{a}_\Box S'$ *for some* $S'$ *with* $S'RT'$.

$S$ *is said to be a refinement of* $T$ *in case* $(S,T)$ *is contained in some refinement* $R$. *We write* $S \lhd T$ *in this case.*

A straightforward generalization allows us to compare specifications from different modal transition systems (essentially by applying the above definition to disjoint sums of modal transition systems). In particular, if $p \lhd S$, where $p$ is a process (viewed as a specification through the derived modal transition system) and $S$ is a specification, we will say that $p$ is an *implementation* of $S$.

For $R$ a binary relation on $S$ let $\mathcal{R}(R)$ be the binary relation on $S$ induced by the above definition. That is, $\mathcal{R}(R)$ is the set of pairs $(S,T)$ satisfying for all actions $a$ the clauses 1 and 2. Then, define $\lhd^n$ inductively by $\lhd^0 = S \times S$ and $\lhd^{n+1} = \mathcal{R}(\lhd^n)$. Then — due to the assumption of image–finiteness – it can be shown that $\lhd = \bigcap_{n \geq 0} \lhd^n$.

Now, the defined refinement relation $\lhd$ enjoys many pleasant properties: $\lhd$ is itself a refinement; in fact the maximal one. Also, $\lhd$ is a *preorder* (reflexive and transitive), supporting design through stepwise refinement as illustrated in (1). However, $\lhd$ is *not* in general an equivalence but allows looseness in specifications. As an example the weakest specification $\mathcal{U}$ is one which constantly allows any action, but never requires that any action must be performed. Operationally, $\mathcal{U}$ is completely defined by $\mathcal{U} \xrightarrow{a}_{\diamond} \mathcal{U}$ for all actions $a$. It is easily verified that $S \lhd \mathcal{U}$ for any modal specification $S$. If $\longrightarrow_{\Box} = \longrightarrow_{\diamond}$ (e.g. when the modal transition system is derived from a process system) the notion of refinement specializes to that of bisimulation [Park 80, Mil80].

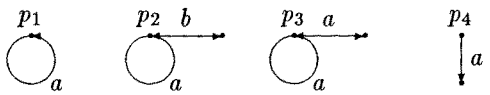


Figure 2: Potential implementations


Figure 3: Specifications of $a$–sender and a–transmitter

**Example 2.3** An $a$–sender is a process that will never refuse to perform the action $a$ as long as the observer asks for nothing else. Examples of $a$–senders are the processes $p_1$ and $p_2$ in figure 2, whereas $p_3$ and $p_4$ are examples of processes not being $a$–senders (they may both refuse to do $a$ after one $a$ action). The notion of an $a$–sender may be specified operationally as $S$ in figure 3. The transition $S \xrightarrow{a}_{\Box} S$ guarantees that any implementation can perform $a$ and become an $a$–sender. The transitions $S \xrightarrow{b}_{\diamond} \mathcal{U}$, $b \neq a$, allows further an implementation to perform any action different from $a$, after which no restrictions is imposed (indicated by $\mathcal{U}$). Now, it is perfectly simple to construct refinements proving $p_1 \lhd S$ and $p_2 \lhd S$. Also, it may easily be argued that $\neg(p_3, p_4 \lhd S)$. A similar but slightly wider class of processes is that of $a$–transmitters, consisting of all processes possessing an infinite $a$–computation. Thus, besides $p_1$ and $p_2$, also $p_3$ is an $a$–transmitter, whereas $p_4$ is not. Operationally, we may specify the concept of $a$–transmitter as $T$ in figure 3. Clearly $T$ is more liberal than $S$, since it has the additional $a$–transition $T \xrightarrow{a}_{\diamond} \mathcal{U}$, allowing an implementation to perform $a$ without necessarily becoming an $a$–transmitter again. It is easily proved that $\{(S,T),(\mathcal{U},\mathcal{U})\}$ is a refinement, and hence that $S \lhd T$. □

Modal Specifications is an extension of Process Algebra in the sense that specifications may be combined using the process constructs from Process Algebra. In particular, a

*single generic static construct* on specifications is introduced in [LT88b] and investigated through examples in [HutLar89]. Here, we shall only assume that specifications are closed under $|$ (parallel composition without synchronizations) and $+$ (summation), with the operational semantics satisfying the following constraints: $S \mid T \xrightarrow{a}_m V$ if and only if either $V = S' \mid T$ with $S \xrightarrow{a}_m S'$ or $V = S \mid T'$ with $T \xrightarrow{a}_m T'$ where $m$ ranges over $\Box, \Diamond$; $S + T \xrightarrow{a}_m V$ if and only if either $S \xrightarrow{a}_m V$ or $T \xrightarrow{a}_m V$ again with $m$ ranging over $\Box, \Diamond$. It is easily proven that parallel composition as well as summation preserves $\lhd$.

**Example 2.4** Recall the modal specification of an $a$–sender $S$ in figure 3. Now, consider combining an $a$–sender with some other process. We may then wonder whether the combined system itself will be an $a$–sender. Provided the combination is that of parallel composition respectively summation, the question is simply whether $\mathcal{U} \mid S$ respectively $\mathcal{U} + S$ is a refinement of $S$. In fact $\mathcal{U} \mid S \lhd S$ holds as $\{(\mathcal{U} \mid S, S), (\mathcal{U} \mid S, \mathcal{U}), (\mathcal{U} \mid \mathcal{U}, \mathcal{U})\}$ is a refinement, showing that a parallel composition of an $a$–sender with *any* process is again an $a$–sender. In the case of summation $\mathcal{U} + S \lhd S$ does *not* hold as $\mathcal{U} + S \xrightarrow{a}_\Diamond \mathcal{U}$ can only be matched by $S \xrightarrow{a}_\Diamond S$, but $\mathcal{U} \not\lhd S$ as $\mathcal{U}$ can not match $S \xrightarrow{a}_\Box S$. To ensure that the combined process is again an $a$–sender it seems necessary that both processes in the summation are $a$–senders. If this is the case we are certain to obtain a new $a$–sender as $S + S \lhd S$ is easily seen to hold. If, however, one of the processes is only an $a$–transmitter we cannot be certain to obtain an $a$–sender as $T + S \not\lhd S$. □

# 3 Logical Characterization of Refinement

In example 2.4 we may consider $\mathcal{U} + S$ as an attempted refinement of the Modal Specification $S$. However, as $\mathcal{U} + S \not\lhd S$, obviously the attempt is an erroneous one. In order to guide towards a correct refinement, we should *explain* why the attempt was erroneous.

Below we shall present a logical characterization of refinement which will provide us with the theoretical foundation for such explanations. The characterization extends the modal characterization of bisimulation by Hennessy and Milner [HenMil 85] and was independently discovered by Gérard Boudol [Bou89]. The characterization identifies a specification with it's properties in such a way, that the more refined (or concrete) a specification is the more properties it enjoys. Thus, an explanation of $\mathcal{U} + S \not\lhd S$ simply consists in exhibiting a property enjoyed by $S$ but not by $\mathcal{U} + S$.

The formulas of *Hennessy–Milner Logic*, $\mathcal{M}$ is given by the following abstract syntax:

$$F ::= \mathrm{tt} \mid \mathrm{ff} \mid F \wedge G \mid F \vee G \mid \langle a \rangle F \mid [a] F$$

where $a \in Act$.

Our interpretation of Hennessy–Milner Logic is an intuitionistic one relative to a Modal Transition System $\mathcal{S} = (S, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$. We define the *satisfaction relation*, $\models \subseteq S \times \mathcal{M}$, inductively as follows:

1. $S \models \mathrm{tt} \quad \Leftrightarrow \mathrm{tt}$
2. $S \models \mathrm{ff} \quad \Leftrightarrow \mathrm{ff}$
3. $S \models F \vee G \Leftrightarrow (S \models F)$ or $(S \models G)$
4. $S \models F \wedge G \Leftrightarrow (S \models F)$ and $(S \models G)$
5. $S \models \langle a \rangle F \quad \Leftrightarrow \exists S'.S \xrightarrow{a}_{\Box} S' \wedge S' \models F$
6. $S \models [a]F \quad \Leftrightarrow \forall S'.S \xrightarrow{a}_{\Diamond} S' \Rightarrow S' \models F$

Note, that for processes our interpretation of Hennessy–Milner Logic coincides with the standard interpretation in [HenMil 85]. The intension of the above definition is the following: whenever a modal specification $S$ satisfies a property $\langle a \rangle F$, any implementation of $S$ must have an $a$–transition leading to a state satisfying $F$; dually, whenever $S$ satisfies a property $[a]F$ any $a$–transition of any implementation must lead to states satisfying $F$. For a modal specification $S$ let $\mathcal{M}(S)$ be the set of properties enjoyed by $S$. Then the following characterization result shows that our intension is met.

**Theorem 3.1** *For any modal specification $S$ and $T$, $S \lhd T$ if and only if $\mathcal{M}(T) \subseteq \mathcal{M}(S)$.*

**Proof** *Only–if:* Assume $S \lhd T$ and $T \models F$. We show that $S \models F$ by induction on the structure of $F$. We only consider the cases involving modalities leaving the more trivial cases to the reader.

$F = \langle a \rangle G$: Then $T \xrightarrow{a}_{\Box} T'$ and $T' \models G$. Since $S \lhd T$, $S \xrightarrow{a}_{\Box} S'$ with $S' \lhd T'$ for some $S'$. By Induction Hypothesis, $S' \models G$ and hence $S \models \langle a \rangle G$.

$F = [a]G$: Now let $S \xrightarrow{a}_{\Diamond} S'$. Then, since $S \lhd T$, $T \xrightarrow{a}_{\Diamond} T'$ with $S' \lhd T'$ for some $T'$. By definition of the satisfaction relation $T' \models G$, and therefore $S' \models G$ by the Induction Hypothesis. Since this argument holds whenever $S \xrightarrow{a}_{\Diamond} S'$ we conclude $S \models [a]G$.

*If:* We show that $\mathcal{M}(T) \not\subseteq \mathcal{M}(S)$ whenever $S \not\lhd T$. We proceed by induction in the smallest $n$ such that $S \not\lhd^n T$. The basis $n = 0$ is trivial. For the Induction step assume the validity of the statement for $1, \ldots, n-1$, and assume $S \not\lhd^n T$. Then one of the following two situations must occur:

i) For some $S'$, $S \xrightarrow{a}_{\Diamond} S'$ but whenever $T \xrightarrow{a}_{\Diamond} T'$, then $S' \not\lhd^{n-1} T'$. Let $\{T_1, \ldots, T_m\}$ be the set of $a$–derivatives of $T$ under modality $\Diamond$. Then, by Induction Hypothesis there exists properties $G_1, \ldots, G_m$ such that $T_i \models G_i$, but $S' \not\models G_i$ for $i = 1 \ldots m$. It is then obvious that $F = [a](G_1 \vee \ldots \vee G_m)$ will be a property enjoyed by $T$ but not by $S$.

ii) For some $T'$, $T \xrightarrow{a}_{\Box} T'$ but whenever $S \xrightarrow{a}_{\Box} S'$ then $S' \not\lhd^{n-1} T'$. Let $\{S_1, \ldots, S_m\}$ be the set of $a$–derivatives of $S$ under modality $\Box$. Then, by Induction Hypothesis there exists properties $G_1, \ldots, G_m$ such that $T' \models G_i$ but $S_i \not\models G_i$ for $i = 1 \ldots m$. It is then obvious that $F = \langle a \rangle (G_1 \wedge \ldots \wedge G_m)$ is a property enjoyed by $T$ not by $S$. $\Box$

**Example 3.2** We can now explain why the attempted refinement $\mathcal{U} + S$ of $S$ in example 2.4 is erroneous by exhibiting a property enjoyed by $S$ but not by $\mathcal{U} + S$. The constructive proof of the Characterization Theorem yields the property $[a]\langle a\rangle \mathrm{tt}$. That is, any implementation of $S$ will have the ability to perform $a$ after any initial $a$–transition (in fact we know that any $a$–sender will have this property invariantly), whereas implementations of $\mathcal{U} + S$ will not necessarily have this property. The process $p_3$ of figure 2 is a perfect implementation of $\mathcal{U} + S$, but does clearly not satisfy the property $[a]\langle a\rangle \mathrm{tt}$. □

# 4 Characteristic Properties

The Characterization result of the previous section shows that any Modal Specifications not being related under refinement may be distinguished by some property of Hennessy–Milner Logic. In this section we will strengthen this result by showing that any Modal Specification $S$ may be fully characterized by a *single* (recursively specified) property $X_S$, allowing questions of refinements between Modal Specifications to be translated into (more standard) questions about implications between logical formulas. The idea of deriving logical formulas from operational behaviour was first introduced by Graf and Sifakis [GrafSif 86] for finite *processes* and has later and recently been extended to finite–state processes by [IngGodZee, Stef89]. Here, we pursue the same idea for Modal Specifications.

Let $\mathcal{S} = (S, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$ be a given Modal Transition System. Then we may identify any property $F$ of Hennessy–Milner Logic with the set of Modal Specifications satisfying it. Under this view the connectives of Hennessy–Milner Logic ($\wedge$, $\vee$, $\langle a\rangle$ and $[a]$) become monotonic operations on sets of Modal Specifications. Thus, using standard fixedpoint theory [Tarski], we can define properties recursively (see also [Lar87]). In particular, let $\{X_T \mid T \in S\}$ be the *maximal* properties satisfying the following set of (simultaneous recursive) equations:

$$X_T = \bigwedge_{T \xrightarrow{a}_\Box T'} \langle a\rangle X_{T'} \wedge \bigwedge_a [a]\Big(\bigvee_{T \xrightarrow{a}_\Diamond T'} X_{T'}\Big) \tag{3}$$

where $\bigvee_\emptyset = \mathrm{ff}$ and $\bigwedge_\emptyset = \mathrm{tt}$. Then $X_T$ characterizes $T$ in the following way:

**Theorem 4.1** *For any Modal Specifications $S$ and $T$, $S \lhd T$ if and only if $S \models X_T$.*

**Proof** *If:* We show that the relation:

$$R = \{(S,T) \mid S \models X_T\}$$

is a refinement. So let $(S,T) \in R$. Assume $S \xrightarrow{a}_\Diamond S'$, As $S \models X_T$ it follows from equation (3) that $S' \models \bigvee\{X_{T'} \mid T \xrightarrow{a}_\Diamond T'\}$, and hence $S' \models X_{T'}$ for some $T'$ with $T \xrightarrow{a}_\Diamond T'$. Assume $T \xrightarrow{a}_\Box T'$. As $S \models X_T$ it follows from equation (3) that $S \models \langle a\rangle X_{T'}$ and hence $S' \models X_{T'}$ for some $S'$ with $S \xrightarrow{a}_\Box S'$.

*Only if:* We show that the family of sets $A_T = \{S \mid S \lhd T\}$ constitutes a post–fixed point to the transformation on sets of Modal Specifications associated with equation (3). Now, let $S \in A_T$, i.e. $S \lhd T$. We must show that:

1. $S \in \langle a \rangle A_{T'}$ whenever $T \xrightarrow{a}_{\Box} T'$, and
2. $S \in [a](\bigvee\{A_{T'} \mid T \xrightarrow{a}_{\Diamond} T'\})$ for all $a$.

For 1, assume $T \xrightarrow{a}_{\Box} T'$. As $S \lhd T$, $S \xrightarrow{a}_{\Box} S'$ with $S' \lhd T'$ for some $S'$. Thus, $S \in \langle a \rangle A_{T'}$.

For 2, assume $S \xrightarrow{a}_{\Diamond} S'$. As $S \lhd T$, $T \xrightarrow{a}_{\Diamond} T'$ with $S' \lhd T'$ for some $T'$. Thus, $S' \in A_{T'} \subseteq \bigvee\{A_{T'} \mid T \xrightarrow{a}_{\Diamond} T'\}$, and hence $S \in [a] \bigvee\{A_{T'} \mid T \xrightarrow{a}_{\Diamond} T'\}$.

□

In particular, viewed as specifications of processes, $X_T$ and $T$ identifies exactly the same set of implementations. Also, it follows as an easy corollary, that $S \lhd T$ if and only if the implication $X_S \Rightarrow X_T$ is logically valid.

**Example 4.2** Consider once more the $a$–sender and $a$–transmitter of example 2.4. The Modal Transition System defined by the $a$–sender induces the following system of equations (assuming that $a$ and $b$ coinstitutes all actions):

$$X_S = \langle a \rangle X_S \wedge [a] X_S \wedge [b] X_U$$
$$X_U = [a] X_U \wedge [b] X_U$$

As $[a]\mathrm{tt} \equiv \mathrm{tt}$, clearly $X_U \equiv \mathrm{tt}$, and the above system of equations can be simplified to the following:

$$X_S = \langle a \rangle X_S \wedge [a] X_S$$

which exactly matches the recursive specification given in [Lar87] of the class of processes that will not deadlock on $a$. The Modal Transition System for the $a$–transmitter induces the following system of equations:

$$X_T = \langle a \rangle X_T \wedge [a](X_T \vee X_U) \wedge [b] X_U$$
$$X_U = [a] X_U \wedge [b] X_U$$

Simplifying this yields

$$X_T = \langle a \rangle X_T$$

which clearly describes the class of processes with an infinite $a$–computation. □

# 5 Logical Combinations of Modal Specifications

The Characteristic Property Theorem of the previous section shows that any Modal Specification may alternatively be expressed as the *maximal* solution to a recursively specified property of Hennessy–Milner Logic. This suggests to us that Modal Specifications may be useful for describing *safety properties*, whereas *liveness properties* seem not to be expressible within Modal Specifications as they are characterized as *minimal* solutions to recursive equations (see [Lar87]). So — from a *theoretical* view — Modal Specifications is less expressive than Hennessy–Milner Logic (with recursion added). However, in [HutLar89], it is indicated through a number of examples that — from a *practical* view — Modal Specifications suffices. Even so, we will in this section suggest ways of extending the expressive power of Modal Specifications.

We first note that any Modal Specification $T$ is *consistent*, in the sense that there exists some process (implementation) $p_T$ refining it: for any Modal Transition System $\mathcal{S} = (S, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$ define the derived process system $\mathcal{P}_\mathcal{S} = (P, A, \longrightarrow)$ where $P = \{p_T \,|\, T \in S\}$ and $p_T \xrightarrow{a} p_{T'}$ if and only if $T \xrightarrow{a}_\Box T'$. Then it is easy to show that $p_T \lhd T$.

Some would claim that any sufficiently expressive specification language necessarily must contain inconsistent specifications. We can very easily make room for inconsistent specification simply by dropping the condition that $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$. Then inconsistency will arise when transitions of implementations are required without being allowed. As a prime example the strongest specification $\mathbf{O}$ is one which constantly requires a transition for any action, but never allows any action. Operationally, $\mathbf{O}$ is completely defined by $\mathbf{O} \xrightarrow{a}_\Box \mathbf{O}$ for any action $a$. It is easily verified that $\mathbf{O} \lhd S$ for any modal specification $S$. The Characterization Theorem 3.1 and the Characteristic Property Theorem 4.1 do *not* depend on the condition $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$ and are therefore also valid for the new larger class of Modal Transition Systems. As an example, the Characteristic Property for $\mathbf{O}$ is defined by the following equation:

$$X_\mathbf{O} = \langle a \rangle X_\mathbf{O} \wedge [a]\mathrm{ff}$$

which clearly has $X_\mathbf{O} = \mathrm{ff}$ as maximal solution.

Allowing inconsistencies, also makes it possible to combine Modal Specifications "logically". If $S$ and $T$ are two Modal Specifications, then $S \wedge T$ is a specification that will require transitions in case either $S$ *or* $T$ makes the requirement, and will allow a transition only in case both $S$ *and* $T$ allows the transition. Dually, $S \vee T$ will allow any transition allowed by just one of $S$ and $T$, and will only require a transitions in case it is required by both $S$ and $T$. This intuition is reflected by the inference-rules below:

$$\frac{S \xrightarrow{a}_\Diamond S'}{S \vee T \xrightarrow{a}_\Diamond S'} \qquad \frac{T \xrightarrow{a}_\Diamond T'}{S \vee T \xrightarrow{a}_\Diamond T'} \qquad \frac{S \xrightarrow{a}_\Box S' \quad T \xrightarrow{a}_\Box T'}{S \vee T \xrightarrow{a}_\Box S' \vee T'}$$

$$\frac{S \xrightarrow{a}_\Box S'}{S \wedge T \xrightarrow{a}_\Box S'} \qquad \frac{T \xrightarrow{a}_\Box T'}{S \wedge T \xrightarrow{a}_\Box T'} \qquad \frac{S \xrightarrow{a}_\Diamond S' \quad T \xrightarrow{a}_\Diamond T'}{S \wedge T \xrightarrow{a}_\Diamond S' \wedge T'}$$

The significance of $\wedge$ and $\vee$ as constructions on Modal Specifications is clear from the following theorem:

**Theorem 5.1** *Let $S$ and $T$ be Modal Specification. Then $S \vee T$ respectively $S \wedge T$ is the least upper bound respectively the greatest lower bound of $S$ and $T$ with respect to $\lhd$.*

Using the Characteristic Property Theorem the following Corollary is immediate:

**Corollary 5.2** *Let $S$ and $T$ be Modal Specifications. Then $(X_S \vee X_T) \Rightarrow X_{S \vee T}$ and $X_{S \wedge T} \Rightarrow (X_S \wedge X_T)$.*

For $\wedge$ and $\vee$ to be truely logical connectives we would have hoped that the implications in the above corollary turned out to be equivalences. However, this is *not* the case in general as will be demonstrated by examples below. Currently Gérard Boudol is searching for conditions on $S$ and $T$ under which the above implications are guaranteed to be equivalences.

**Example 5.3** Consider the Modal Specifications $A$ and $B$ operationally specified as follows:

$$A \xrightarrow{a}_m U \qquad A \xrightarrow{b}_\Diamond U$$
$$B \xrightarrow{b}_m U \qquad B \xrightarrow{a}_\Diamond U$$

where $m$ ranges over $\Diamond$ and $\Box$. Thus, $A$ ($B$) is intended to specify the class of processes that can perform the action $a$ ($b$). This is confirmed by the Characteristic Properties for $A$ (assuming $a$ and $b$ constitutes the set of actions):

$$X_A = \langle a \rangle X_U \wedge [a] X_U \wedge [b] X_U$$
$$X_U = [a] X_U \wedge [b] X_U$$

which simplifies to $X_A = \langle a \rangle \mathrm{tt}$. Now, the "conjunction" of $A$ and $B$ is completely described by the following modal transitions:

$$A \wedge B \xrightarrow{a}_\Box U \qquad A \wedge B \xrightarrow{b}_\Box U \qquad A \wedge B \xrightarrow{a}_\Diamond U \wedge U \qquad A \wedge B \xrightarrow{b}_\Diamond U \wedge U$$

with Characteristic Property $X_{A \wedge B} = \langle a \rangle \mathrm{tt} \wedge \langle b \rangle \mathrm{tt}$. Thus, in this case the conjunction $A \wedge B$ is a truely logical one. Similarly, the "disjunction" of $A$ and $B$ is described by the following modal transitions:

$$A \vee B \xrightarrow{a}_\Diamond U \qquad A \vee B \xrightarrow{b}_\Diamond U$$

with Characteristic Property $X_{A \vee B} = \mathrm{tt}$. Thus, in this case the implementations of $A \vee B$ is *more* than the union of the implementations of $A$ and $B$, being just the processes satisfying $\langle a \rangle \mathrm{tt} \vee \langle b \rangle \mathrm{tt}$. $\Box$

# 6 Implementations

Refinements between finite–state Modal Specifications may be decided in polynomial time. The existing PROLOG–system for *arguing* about bisimulation between processes in [Hil87] is easily modified to argue about refinements between Modal Specifications.

In contrast to the algorithms of [PaigeTar, KanSmo83] the algorithm used in [Hil87] tries to construct a *minimal* refinement containing a given pair of Modal Specifications. The algorithm follows the recursive definition of refinement very closely, but by "memorizing" a pair of Modal Specifications once it has been determined outside the refinement–ordering $\lhd$, backtracking is avoided and the overall running time becomes polynomial. Furthermore, the algorithm used in the system of [Hil87] has the additional feature that it generates *small* distinguishing properties for Modal Specifications not in the refinement–ordering. A description of the general algorithm underlying [Hil87] is the subject of a paper to come [Lar?].

Here, we demonstrate the use of the system on some examples. First we define the Modal Specifications $\mathcal{U}, S$ and $T$ (the syntax used should be obvious):

```
?- u :=: may(a);u + may(b);u.
?- s :=: must(a);s + may(b);u.
?- t :=: must(a);t + may(a);u + may(b);u.
```

We ask whether $\mathcal{U} \mid S$ is a refinement of $S$:

```
?- refine(u/s,s).

Searching for a refinement...
Here is a listing of a refinement between the specifications
     u/s     and     s

0
u/s
                s
                          LHS--> b ==> 2
                          LHS--> b ==> 1
                          LHS--> a ==> 0
                          RHS--> a ==> 0
1
u/s
                u
                          LHS--> b ==> 2
                          LHS--> b ==> 1
                          LHS--> a ==> 1
2
u/u
                u
```

```
LHS--> b ==> 2
LHS--> a ==> 2
```

The system correctly returns a set of three pairs of Modal Specifications (containing the given pair) together with a (coded) argument for why the set constitutes a refinement.

Asking whether $\mathcal{U} + S$ is a refinement of $S$ yields the following respons:

```
?- refine(u+s,s).

Searching for a refinement...
The specification:
     s
enjoys some properties which the specification
     u+s
doesn't. One property is:
     [a]<a>tt
```

## Acknowledgement

This work has been carried out as part of the TAU–project [Tau], a project supported by the FTU–program under the danish research council. I would like to thank Liu Xinxin for many helpful discussions on the subject of this paper, and also thanks to Gérard Boudol for his independent work on Modal Specifications.

## References

[Abr87] S. Abramsky: *Observation Equivalence as a Testing Equivalence*, TCS, 1987.

[Auto] Lecompte, Madelaine, Vergamini: *AUTO: A Verification System for Parallel and Communicating Processes*, INRIA, Sophia–Antipolis, 1988.

[Bou89] G. Boudol: *Grafical Specifications*, unpublished note, 1989.

[CW] Cleaveland, Parrow, Steffen: *The Concurrency Workbench*, University of Edinburgh.

[GrafSif 86] S.Graf and J.Sifakis: *A Logic for the Description of Non–deterministic Programs and Their Properties*, Information and Control, vol 68, no 1–3, 1986.

[GroVan89] Groote, Vaandrager: *Structured Operational Semantics and Bisimulation as a Congruence*, 1989.

[HenMil 85] M.Hennessy and R.Milner: *Algebraic Laws for Nondeterminism and Concurrency*, Journal of the Association for computing Machinery, pp. 137–161, 1985.

[Hil87] M. Hillerstr"om: *Verification of CCS–processes*, Master–Thesis, Aalborg University Center, 1987.

[Holmstr 87] S. Holmström: *Reasoing about CCS agents using Hennessy–Milner logic extended with fixed points*, unpublished paper, 1987.

[HutLar89] Hüttel, Larsen: *The use of Static Constructs in A Modal Process Logic*, to be presented at Logic at Botik'89, USSR.

[IngGodZee] Ingolfsdottir, Godskesen, Zeeberg: Master Thesis, Aalborg University, 1987.

[KanSmo83] Kannellakis, Smolka: *CCS Expressions, finite state processes, and three problems of equivalence*, 1983. To appear in Information and Computation.

[LT88b] Larsen, Kim. G and Bent Thomsen: *A Modal Process Logic*, in Proceedings of Third Annual symposium on Logic in Computer Science, Edinburgh, 1988.

[Lar 86] K.G.Larsen: *A Context Dependent Bisimulation between Processes*, Ph.D Thesis, Edinburgh University, 1986.

[Lar 87a] K.G.Larsen: *A Context Dependent Bisimulation between Processes*, Theoretical Computer Science 49, 1987.

[Lar87] Larsen, Kim G.: *Proof Systems for Hennessy-Milner Logic with Recursion*, in: *CAAP 88*, Springer Lecture Notes in Computer Science 299, 1988. (Extended version to appear in Theoretical Computer Science, North-Holland).

[Lar?] Larsen: *Arguing about Membership of Maximal Fixedpoints*, future paper.

[LarMil 87] K.G.Larsen and R.Milner: *Verifying a Protocol Using Relativized Bisimulation*, in Proceedings of ICALP'87, LNCS 267.

[LarSkou89] Larsen, Skou: *Bisimulation Through Probabilistic Testing*, Proceedings of ACM POPL'89.

[Mil80] R. Milner: *Calculus of Communicating Systems*, LNCS 92.

[Mil 83] R. Milner: *Calculi for Synchrony and Asynchrony*, Theoretical Computer Science 25, 1983.

[PaigeTar] Paige, Tarjan: *Three Partition Refinement Algorithms*, SIAM J. Comput., vol. 16, no. 6, 1987.

[Park 80] D.Park: *Concurrency and automata on infinite sequences*, Proc. 5th GI Conf., LNCS 104, 1981.

[Pl81] G.Plotkin: *A Structural Approach to Operational Semantics*, Tech. Rep., DAIMI FN–19, Computer Sc., Aarhus University, Denmark, 1981.

[Sim85] R. de Simone: *Higher–level synchronising devices in MEIJE–CCS*, TCS 37, 1985.

[Stef89] B. STeffen: *Characteristic Formulae*, Edinburgh University, 1989.

[Tarski] A. Taski: *A Lattice–Theoretical Fixpoint Theorem and Its applications*, Pacific Journal of Math. 5, 1955.

[Tau] Larsen, Skou: *TAU: Theories for Parallel Systems, their Automation and Usage*, Aalborg University Center, 1987.