# Hennessy-Milner Logic with Greatest Fixed Points as a Complete Behavioural Specification Theory

Nikola Beneš[1⋆], Benoît Delahaye[2], Uli Fahrenberg[2], Jan Křetínský[1,3⋆⋆], and Axel Legay[2]

[1] Masaryk University, Brno, Czech Republic
[2] Irisa / INRIA Rennes, France
[3] Technische Universität München, Germany

**Abstract.** There are two fundamentally different approaches to specifying and verifying properties of systems. The *logical* approach makes use of specifications given as formulae of temporal or modal logics and relies on efficient model checking algorithms; the *behavioural* approach exploits various equivalence or refinement checking methods, provided the specifications are given in the same formalism as implementations. In this paper we provide translations between the logical formalism of Hennessy-Milner logic with greatest fixed points and the behavioural formalism of disjunctive modal transition systems. We also introduce a new operation of quotient for the above equivalent formalisms, which is adjoint to structural composition and allows synthesis of missing specifications from partial implementations. This is a substantial generalisation of the quotient for deterministic modal transition systems defined in earlier papers.

## 1 Introduction

There are two fundamentally different approaches to specifying and verifying properties of systems. Firstly, the *logical* approach makes use of specifications given as formulae of temporal or modal logics and relies on efficient model checking algorithms. Secondly, the *behavioural* approach exploits various equivalence or refinement checking methods, provided the specifications are given in the same formalism as implementations.

In this paper, we discuss different formalisms and their relationship. As an example, let us consider labelled transition systems and the property that *"at all time points after executing* request*, no* idle *nor further requests but only* work *is allowed until* grant *is executed"*. The property can be written in *e.g.* CTL [13] as

$$\mathrm{AG}(\mathsf{request} \Rightarrow \mathrm{AX}(\mathsf{work}\ \mathrm{AW}\ \mathsf{grant}))$$

⋆ The author has been supported by the Czech Science Foundation grant No. GAP202/11/0312.
⋆⋆ The author is partially supported by the Czech Science Foundation, project No. P202/10/1469

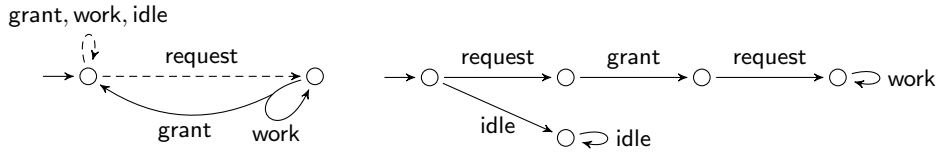



**Fig. 1.** DMTS specification corresponding to AG(request ⇒ AX(work AW grant)), and its implementation

or as a recursive system of equations in Hennessy-Milner logic [28] as

$$X = [\mathsf{grant}, \mathsf{idle}, \mathsf{work}]X \wedge [\mathsf{request}]Y$$
$$Y = (\langle \mathsf{work} \rangle Y \vee \langle \mathsf{grant} \rangle X) \wedge [\mathsf{idle}, \mathsf{request}]\mathbf{ff}$$

where the solution is given by the greatest fixed point.

As formulae of modal logics can be difficult to read, some people prefer automata-based behavioural specifications to logical ones. One such behavioural specification formalism is the one of disjunctive modal transition systems (DMTS) [25]. Fig. 1 (left) displays a specification of our example property as a DMTS. Here the dashed arrows indicate that the transitions *may or may not* be present, while branching of the solid arrow indicates that at least one of the branches *must* be present. An example of a labelled transition system that *satisfies* our logical specifications and *implements* the behavioural one is also given in Fig. 1.

The alternative between logical and behavioural specifications is not only a question of preference. Logical specification formalisms put a powerful logical language at the disposal of the user, and the logical approach to model checking [13, 33] has seen a lot of success and tool implementations. Automata-based specifications [11, 26], on the other hand, have a focus on *compositional* and *incremental* design in which logical specifications are somewhat lacking, with the trade-off of generally being less expressive than logics.

To be more precise, automata-based specifications are, by design, compositional in the sense that they support structural *composition* of specifications and, in most cases, its adjoint, *quotient*. This is useful, even necessary, in practical verification, as it means that (1) it is possible to infer properties of a system from the specifications of its components, and (2) the problem of correctness for a system can be decomposed into verification problems for its components. We refer to [27] for a detailed account on composition and decomposition.

It is thus desirable to be able to translate specifications from the logical realm into behavioural formalisms, and *vice versa* from behavioural formalisms to logic-based specifications. This is, then, the first contribution of this paper: we show that Hennessy-Milner logic with greatest fixed points ($\nu$HML) and DMTS (with several initial states) are equally expressive, and we provide translations forth and back. For doing this, we introduce an auxiliary intermediate formalism NAA (a nondeterministic extension of acceptance automata [21, 34]) which is equivalent in expressiveness to both $\nu$HML and DMTS.

We also discuss other desirable features of specification formalisms, namely structural composition and quotient. As an example, consider a specification $S$

of the final system to be constructed and $T$ either an already implemented component or a specification of a service to be used. The task is to construct the most general specification of the rest of the system to be implemented, in such a way that when composed with any implementation of $T$, it conforms with the specification $S$. This specification is exactly the quotient $S/T$.

**Contribution** Firstly, we show that the formalisms of $\nu$HML, NAA and DMTS have the same expressive power, and provide the respective translations. As a result, the established connection allows for a graphical representation of $\nu$HML as DMTS. This extends the graphical representability of HML without fixed points as modal transition systems [9, 26]. In some sense this is optimal, as due to the alternation of least and greatest fixed points, there seems to be no hope that the whole $\mu$-calculus could be drawn in a similarly simple way.

Secondly, we show that there are natural operations of conjunction and disjunction for NAA which mimic the ones of $\nu$HML. As we work with multiple initial states, disjunction is readily defined, and conjunction extends the one for DMTS [5]. Thirdly, we introduce structural composition on NAA. For simplicity we assume CSP-style synchronisation of labels, but the construction can easily be generalised to other types of label synchronisation.

Finally, we provide a solution to the open problem of the general quotient. We extend the quotient constructions for deterministic modal transition systems (MTS) and acceptance automata [34] to define the quotient for the full class of (possibly nondeterministic) NAA. We also provide a more efficient procedure for (possibly nondeterministic) MTS. These constructions are the technically most demanding parts of the paper.

With the operations of structural composition and quotient, NAA, and hence also DMTS and $\nu$HML, are fully compositional behavioural specification theories and form a *commutative residuated lattice* [20, 38] up to equivalence. This makes a rich algebraic theory available for compositional reasoning about specifications. Most of the constructions we introduce are implemented in a prototype tool [7]. Due to space constraints, some of the proofs had to be omitted from the paper.

**Related work** Hennessy-Milner logic with recursion [28] is a popular logical specification formalism which has the same expressive power as $\mu$-calculus [24]. It is obtained from Hennessy-Milner logic (HML) [22] by introducing variables and greatest and least fixed points. Hennessy-Milner logic with *greatest* fixed points ($\nu$HML) is equivalent to $\nu$-calculus, *i.e.* $\mu$-calculus with greatest fixed points only.

DMTS have been proposed as solutions to algebraic process equations in [25] and further investigated also as a specification formalism [5, 27]. The DMTS formalism is a member of the modal transition systems (MTS) family and as such has also received attention recently. The MTS formalisms have proven to be useful in practice. Industrial applications started as early as [10] where MTS have been used for an air-traffic system at Heathrow airport. Besides, MTS classes are advocated as an appropriate base for interface theories in [35] and for product line theories in [30]. Further, an MTS based software engineering methodology

for design via merging partial descriptions of behaviour has been established in [37] and methods for supervisory control of MTS shown in [14]. Tool support is quite extensive, *e.g.* [2, 5, 8, 15].

Over the years, many extensions of MTS have been proposed. While MTS can only specify whether or not a particular transition is required, some extensions equip MTS with more general abilities to describe what *combinations* of transitions are possible. These include DMTS [25], 1-MTS [16] allowing to express exclusive disjunction, OTS [3] capable of expressing positive Boolean combinations, and Boolean MTS [4] covering all Boolean combinations. The last one is closely related to our NAA, the acceptance automata of [21, 34], as well as hybrid modal logic [6, 32].

Larsen has shown in [26] that any finite MTS is equivalent to a HML formula (without recursion or fixed points), the *characteristic formula* of the given MTS. Conversely, Boudol and Larsen show in [9] that any consistent and *prime* HML formula is equivalent to a MTS. Here we extend these results to $\nu$HML formulae, and show that any such formula is equivalent to a DMTS, solving a problem left open in [25]. Hence $\nu$HML supports full compositionality and decomposition in the sense of [27]. This finishes some of the work started in [9, 26, 27].

Quotients are related to *decomposition* of processes and properties, an issue which has received considerable attention through the years. In [25], a solution to bisimulation $C(X) \sim P$ for a given process $P$ and context $C$ is provided (as a DMTS). This solves the quotienting problem $P/C$ for the special case where both $P$ and $C$ are processes. This is extended in [29] to the setting where the context $C$ can have several holes and $C(X_1, \ldots, X_n)$ must satisfy a property $Q$ of $\nu$HML. However, $C$ remains to be a process context, not a specification context. Our *specification* context allows for arbitrary specifications, representing infinite sets of processes and process equations. Another extension uses infinite conjunctions [18], but similarly to the other approaches, generates partial specifications from an overall specification and a given set of processes. This is subsumed by a general quotient.

Quotient operators, or *guarantee* or *multiplicative implication* as they are called there, are also well-known from various logical formalisms. Indeed, the algebraic properties of our parallel composition $\parallel$ and quotient / resemble closely those of multiplicative conjunction & and implication $\multimap$ in *linear logic* [19], and of spatial conjunction and implication in *spatial logic* [12] and *separation logic* [31, 36]. For these and other logics, proof systems have been developed which allow one to reason about expressions containing these operators.

In spatial and separation logic, & and $\multimap$ (or the operators corresponding to these linear-logic symbols) are first-class operators on par with the other logical operators, and their semantics are defined as certain sets of processes. In contrast, for NAA and hence, via the translations, also for $\nu$HML, $\parallel$ and / are *derived* operators, and we provide constructions to reduce any expression which contains them, to one which does not. This is important from the perspective of reuse of components and useful in industrial applications.

## 2 Specification Formalisms

In this section, we define the specification formalisms $\nu$HML, DMTS and NAA and show that they are equivalent.

For the rest of the paper, we fix a finite alphabet $\Sigma$. In each of the formalisms, the semantics of a specification is a set of implementations, in our case always a set of *labelled transition systems* (LTS) over $\Sigma$, *i.e.* structures $(S, s^0, \longrightarrow)$ consisting of a set $S$ of *states*, an initial state $s^0 \in S$, and a *transition relation* $\longrightarrow \subseteq S \times \Sigma \times S$. We assume that the transition relation of LTS is always *image-finite*, i.e. that for every $a \in \Sigma$ and $s \in S$ the set $\{s' \in S \mid s \xrightarrow{a} s'\}$ is finite.

### 2.1 Hennessy-Milner Logic with Greatest Fixed Points

We recap the syntax and semantics of HML with variables developed in [28]. A *HML formula* $\phi$ over a set $X$ of variables is given by the abstract syntax $\phi ::= \mathbf{tt} \mid \mathbf{ff} \mid x \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a \rangle \phi \mid [a]\phi$, where $x$ ranges over $X$ and $a$ over $\Sigma$. The set of such formulae is denoted $\mathcal{H}(X)$. Notice that instead of including fixed point operators in the logic, we choose to use declarations with a greatest fixed point semantics, as explained below.

A *declaration* is a mapping $\Delta : X \to \mathcal{H}(X)$. We shall give a greatest fixed point semantics to declarations. Let $(S, s^0, \longrightarrow)$ be an LTS, then an *assignment* is a mapping $\sigma : X \to 2^S$. The set of assignments forms a complete lattice with $\sigma_1 \sqsubseteq \sigma_2$ iff $\sigma_1(x) \subseteq \sigma_2(x)$ for all $x \in X$ and $\big(\bigsqcup_{i \in I} \sigma_i\big)(x) = \bigcup_{i \in I} \sigma_i(x)$.

The semantics of a formula is a subset of $S$, given relative to an assignment $\sigma$, defined as follows: $(\!|\mathbf{tt}|\!)\sigma = S$, $(\!|\mathbf{ff}|\!)\sigma = \emptyset$, $(\!|x|\!)\sigma = \sigma(x)$, $(\!|\phi \wedge \psi|\!)\sigma = (\!|\phi|\!)\sigma \cap (\!|\psi|\!)\sigma$, $(\!|\phi \vee \psi|\!)\sigma = (\!|\phi|\!)\sigma \cup (\!|\psi|\!)\sigma$, $(\!|\langle a \rangle \phi|\!)\sigma = \{s \in S \mid \exists s \xrightarrow{a} s' : s' \in (\!|\phi|\!)\sigma\}$, and $(\!|[a]\phi|\!)\sigma = \{s \in S \mid \forall s \xrightarrow{a} s' : s' \in (\!|\phi|\!)\sigma\}$. The semantics of a declaration $\Delta$ is then the assignment defined by $(\!|\Delta|\!) = \bigsqcup\{\sigma : X \to 2^S \mid \forall x \in X : \sigma(x) \subseteq (\!|\Delta(x)|\!)\sigma\}$: the greatest (pre)fixed point of $\Delta$.

An *initialised* HML declaration, or *$\nu$HML formula*, is a structure $(X, X^0, \Delta)$, with $X^0 \subseteq X$ finite sets of variables and $\Delta : X \to \mathcal{H}(X)$ a declaration. We say that an LTS $(S, s^0, \longrightarrow)$ *implements* (or models) the formula, and write $S \models \Delta$, if it holds that there is $x^0 \in X^0$ such that $s^0 \in (\!|\Delta|\!)(x^0)$. We write $[\![\Delta]\!]$ for the set of implementations (models) of a $\nu$HML formula $\Delta$.

### 2.2 Disjunctive Modal Transition Systems

A DMTS is essentially a labelled transition system (LTS) with two types of transitions, *may* transitions which indicate that implementations are permitted to implement the specified behaviour, and *must* transitions which proclaim that any implementation is required to implement the specified behaviour. Additionally, *must* transitions may be *disjunctive*, in the sense that they can require that *at least one* out of a number of specified behaviours must be implemented. We now recall the syntax and semantics of DMTS as introduced in [25]. We modify

the syntax slightly to permit multiple initial states and, in the spirit of later work [5,17], ensure that all required behaviour is also allowed:

A *disjunctive modal transition system* (DMTS) over the alphabet $\Sigma$ is a structure $(S, S^0, \dashrightarrow, \longrightarrow)$ consisting of a set of *states* $S$, a finite subset $S^0 \subseteq S$ of *initial states*, a *may*-transition relation $\dashrightarrow \subseteq S \times \Sigma \times S$, and a *disjunctive must*-transition relation $\longrightarrow \subseteq S \times 2^{\Sigma \times S}$. It is assumed that for all $(s, N) \in \longrightarrow$ and all $(a, t) \in N$, $(s, a, t) \in \dashrightarrow$. We usually write $s \overset{a}{\dashrightarrow} t$ instead of $(s, a, t) \in \dashrightarrow$ and $s \longrightarrow N$ instead of $(s, N) \in \longrightarrow$. We also assume that the may transition relation is image-finite. Note that the two assumptions imply that $\longrightarrow \subseteq S \times 2^{\Sigma \times S}_{\mathrm{Fin}}$ where $2^X_{\mathrm{Fin}}$ denotes the set of all finite subsets of $X$.

A DMTS $(S, S^0, \dashrightarrow, \longrightarrow)$ is an *implementation* if $S^0 = \{s^0\}$ is a singleton and $\longrightarrow = \{(s, \{(a, t)\} \mid s \overset{a}{\dashrightarrow} t\}$, hence if $N$ is a singleton for each $s \longrightarrow N$ and there are no superfluous may-transitions. Thus DMTS implementations are precisely LTS.

We proceed to define the semantics of DMTS. First, a relation $R \subseteq S_1 \times S_2$ is a *modal refinement* between DMTS $(S_1, S_1^0, \dashrightarrow_1, \longrightarrow_1)$ and $(S_2, S_2^0, \dashrightarrow_2, \longrightarrow_2)$ if it holds for all $(s_1, s_2) \in R$ that

- for all $s_1 \overset{a}{\dashrightarrow} t_1$ there is $s_2 \overset{a}{\dashrightarrow} t_2$ for some $t_2 \in S_2$ with $(t_1, t_2) \in R$, and
- for all $s_2 \longrightarrow N_2$ there is $s_1 \longrightarrow N_1$ such that for each $(a, t_1) \in N_1$ there is $(a, t_2) \in N_2$ with $(t_1, t_2) \in R$.

Such a modal refinement is *initialised* if it is the case that, for each $s_1^0 \in S_1^0$, there is $s_2^0 \in S_2^0$ for which $(s_1^0, s_2^0) \in R$. In that case, we say that $S_1$ refines $S_2$ and write $S_1 \leq_{\mathrm{m}} S_2$. We write $S_1 \equiv_{\mathrm{m}} S_2$ if $S_1 \leq_{\mathrm{m}} S_2$ and $S_2 \leq_{\mathrm{m}} S_1$.

We say that an LTS $I$ *implements* a DMTS $S$ if $I \leq_{\mathrm{m}} S$ and write $[\![S]\!]$ for the set of implementations of $S$. Notice that the notions of implementation and modal refinement agree, capturing the essence of DMTS as a *specification theory*: A DMTS may be *gradually* refined, until an LTS, in which all behaviour is fully specified, is obtained.

For DMTS $S_1$, $S_2$ we say that $S_1$ *thoroughly* refines $S_2$, and write $S_1 \leq_{\mathrm{t}} S_2$, if $[\![S_1]\!] \subseteq [\![S_2]\!]$. We write $S_1 \equiv_{\mathrm{t}} S_2$ if $S_1 \leq_{\mathrm{t}} S_2$ and $S_2 \leq_{\mathrm{t}} S_1$. By transitivity, $S_1 \leq_{\mathrm{m}} S_2$ implies $S_1 \leq_{\mathrm{t}} S_2$.

*Example 1.* Figs. 2 and 3 show examples of important basic properties expressed both as $\nu$HML formulae, NAA (see below) and DMTS. For DMTS, may transitions are drawn as dashed arrows and disjunctive must transitions as branching arrows. States with a short incoming arrow are initial (the DMTS in Fig. 3 has *two* initial states).

$$X = \langle a \rangle \mathrm{tt} \wedge [a]X \wedge [b]X$$

$$(\{s_0\}, \{s_0\}, \mathrm{Tran})$$
$$\mathrm{Tran}(s_0) = \left\{ \{(a, s_0)\}, \{(a, s_0), (b, s_0)\} \right\}$$

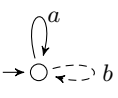



**Fig. 2.** $\nu$HML formula, NAA and DMTS for the invariance property "*there is always an 'a' transition available*", with $\Sigma = \{a, b\}$

$$X = \langle b \rangle \mathbf{tt} \vee \big( \langle a \rangle \mathbf{tt} \wedge [a]X \wedge [b]X \wedge [c]X \big)$$

$$(\{s_0, s_1\}, \{s_0\}, \mathrm{Tran})$$
$$\mathrm{Tran}(s_0) = \Big\{ \{(b, s_1)\}, \{(b, s_1), (a, s_1)\}, \{(b, s_1), (c, s_1)\}, \{(b, s_1), (a, s_1), (c, s_1))\}, \{(a, s_0)\}, \{(a, s_0), (c, s_0)\} \Big\}$$
$$\mathrm{Tran}(s_1) = 2^{\{s_1\} \times \{a,b,c\}}$$

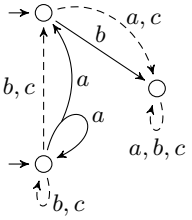
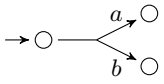



**Fig. 3.** $\nu$HML formula, NAA and DMTS for the ("weak until") property "*there is always an 'a' transition available, until a 'b' transition becomes enabled*", with $\Sigma = \{a, b, c\}$

**Modal Transition Systems** An interesting subclass of DMTS are *modal transition systems* (MTS) [26]. A DMTS $(S, S^0, \dashrightarrow, \longrightarrow)$ is said to be a MTS if (1) $S^0 = \{s^0\}$ is a singleton, (2) for every $s \longrightarrow N$, the set $N$ is a singleton. Hence, for each transition, we specify whether it must, may, or must not be present; no disjunctions can be expressed. It is easy to see that MTS are less expressive than DMTS, *i.e.* there are DMTS $S$ for which no MTS $S'$ exists so that $[\![S]\!] = [\![S']\!]$. One example is provided on the right. Here any implementation must have an $a$ or a $b$ transition from the initial state, but then any MTS which permits all such implementations will also allow implementations without any transition from the initial state.




### 2.3 NAA

We now define NAA, the nondeterministic extension to the formalism of acceptance automata [34]. We shall use this formalism to bridge the gap between $\nu$HML and DMTS. A *nondeterministic acceptance automaton* over the alphabet $\Sigma$ is a structure $(S, S^0, \mathrm{Tran})$ where $S$ and $S^0$ are the states and initial states as previously, and $\mathrm{Tran} : S \to 2^{2^{\Sigma \times S}_{\mathrm{Fin}}}$ assigns admissible transition sets.

A NAA $(S, S^0, \mathrm{Tran})$ is an *implementation* if $S^0 = \{s^0\}$ is a singleton and $\mathrm{Tran}(s) = \{M\}$ is a singleton for every $s \in S$; clearly, NAA implementations are precisely LTS. We also define the *inconsistent NAA* to be $\bot = (\emptyset, \emptyset, \emptyset)$ and the *universal NAA* by $\top = (\{s\}, \{s\}, 2^{2^{\Sigma \times \{s\}}})$.

A relation $R \subseteq S_1 \times S_2$ is a *modal refinement* between NAA $(S_1, S_1^0, \mathrm{Tran}_1)$, $(S_2, S_2^0, \mathrm{Tran}_2)$ if it holds for all $(s_1, s_2) \in R$ and all $M_1 \in \mathrm{Tran}_1(s_1)$ that there exists $M_2 \in \mathrm{Tran}_2(s_2)$ such that

- $\forall (a, t_1) \in M_1 : \exists (a, t_2) \in M_2 : (t_1, t_2) \in R$,
- $\forall (a, t_2) \in M_2 : \exists (a, t_1) \in M_1 : (t_1, t_2) \in R$.

We define and use the notions of initialised modal refinement, $\leq_{\mathrm{m}}$, $\equiv_{\mathrm{m}}$, implementation, $\leq_{\mathrm{t}}$, and $\equiv_{\mathrm{t}}$ the same way as for DMTS.

**Proposition 2.** *The class of NAA is preordered by modal refinement $\leq_{\mathrm{m}}$, with bottom element $\bot$ and top element $\top$.*

Note that as implementations of all our three formalisms $\nu$HML, DMTS and NAA are LTS, it makes sense to use thorough refinement $\leq_{\mathrm{t}}$ and equivalence $\equiv_{\mathrm{t}}$ *across* formalisms, so that we *e.g.* can write $S \leq_{\mathrm{t}} \Delta$ for a NAA $S$ and a $\nu$HML formula $\Delta$.

### 2.4 Equivalences

We proceed to show that $\nu$HML, DMTS and NAA are equally expressive:

**Theorem 3.** *For any set $\mathcal{S}$ of LTS, the following are equivalent:*

1. *There exists a $\nu$HML formula $\Delta$ with $[\![\Delta]\!] = \mathcal{S}$.*
2. *There exists a finite NAA $S$ with $[\![S]\!] = \mathcal{S}$.*
3. *There exists a finite DMTS $S$ with $[\![S]\!] = \mathcal{S}$.*

*Furthermore, the latter two statements are equivalent even if we drop the finiteness constraints.*

Note that we could drop the finiteness assumption about the set of variables of $\nu$HML formulae, while retaining the fact that $\Delta(x)$ is a finite HML formula. The result of Theorem 3 could then be extended with the statement that these possibly infinite $\nu$HML formulae are equivalent to general DMTS/NAA.

For a DMTS $S = (S, S^0, \dashrightarrow, \longrightarrow)$, let $\mathrm{Tran}(s) = \{M \subseteq \Sigma \times S \mid \exists N : s \longrightarrow N, N \subseteq M; \forall (a,t) \in M : s \overset{a}{\dashrightarrow} t\}$ and define the NAA $dn(S) = (S, S^0, \mathrm{Tran})$.

Conversely, for an NAA $(S, S^0, \mathrm{Tran})$, define the DMTS $nd(S) = (T, T^0, \dashrightarrow, \longrightarrow)$ as follows:

- $T = \{M \in \mathrm{Tran}(s) \mid s \in S\}$, $T^0 = \{M \in \mathrm{Tran}(s^0) \mid s^0 \in S^0\}$,
- $\longrightarrow = \{(M, \{(a, M') \mid M' \in \mathrm{Tran}(s')\} \mid (a, s') \in M\}$,
- $\dashrightarrow = \{(t, a, t') \mid t \in T, \exists (t, N) \in \longrightarrow : (a, t') \in N\}$.

Note that both *nd* and *dn* preserve finiteness. Both translation are exponential in their respective arguments.

**Lemma 4.** *For every DMTS $S$, $S \equiv_{\mathrm{t}} dn(S)$. For every NAA $S$, $S \equiv_{\mathrm{t}} nd(S)$.*

For a set of pairs of actions and states $M$ we use $M_a$ to denote the set $\{s \mid (a,s) \in M\}$. Let $(S, S^0, \mathrm{Tran})$ be a finite NAA and let $s \in S$, we then define

$$\Delta_{\mathrm{Tran}}(s) = \bigvee_{M \in \mathrm{Tran}(s)} \Big( \bigwedge_{(a,t) \in M} \langle a \rangle t \wedge \bigwedge_{a \in \Sigma} [a] \Big( \bigvee_{u \in M_a} u \Big) \Big)$$

We then define the $\nu$HML formula $nh(S) = (S, S^0, \Delta_{\mathrm{Tran}})$. Notice that variables in $nh(S)$ are states of $S$.

**Lemma 5.** *For all NAA $S$, $S \equiv_{\mathrm{t}} nh(S)$.*

Our translation from $\nu$HML to DMTS is based on the constructions in [9]. First, we need a variant of a disjunctive normal form for HML formulae:

**Lemma 6.** *For any $\nu$HML formula $(X_1, X_1^0, \Delta_1)$, there exists another formula $(X_2, X_2^0, \Delta_2)$ with $[\![\Delta_1]\!] = [\![\Delta_2]\!]$ and such that any formula $\Delta_2(x)$, for $x \in X_2$, is $\mathbf{tt}$ or of the form $\Delta_2(x) = \bigvee_{i\in I} \left( \bigwedge_{j\in J_i} \langle a_{ij}\rangle x_{ij} \wedge \bigwedge_{a\in\Sigma} [a] y_{i,a} \right)$ for finite (possibly empty) index sets $I$ and $J_i$, $i \in I$, and all $x_{ij}, y_{i,a} \in X_2$. Additionally we can assume that for all $i \in I$, $j \in J_i$, $a \in \Sigma$, $a_{ij} = a$ implies $[\![x_{ij}]\!] \subseteq [\![y_{i,a}]\!]$.*

Let now $(X, X^0, \Delta)$ be a $\nu$HML formula in the form introduced above, then we define a DMTS $hd(\Delta) = (S, S^0, \dashrightarrow, \longrightarrow)$ as follows:

- $S = \{(x,k) \mid x \in X, \Delta(x) = \bigvee_{i\in I} \phi_i, k \in I \neq \emptyset\} \cup \{\bot, \top\}$,
- $S^0 = \{(x^0, k) \mid x^0 \in X^0\}$.
- For each $(x,k) \in S$ with $\Delta(x) = \bigvee_{i\in I}(\bigwedge_{j\in J_i} \langle a_{ij}\rangle x_{ij} \wedge \bigwedge_{a\in\Sigma} [a] y_{i,a})$ and $I \neq \emptyset$,
  - for each $j \in J_i$, let $\mathrm{Must}_j(x,k) = \{(a_{ij}, (x_{ij}, i')) \in \Sigma \times S\}$,
  - for each $a \in \Sigma$, let $\mathrm{May}_a(x,k) = \{(x', i') \in S \mid [\![x']\!] \subseteq [\![y_{i,a}]\!]\}$.
- Let $\dashrightarrow = \{(s, a, s') \mid s \in S, a \in \Sigma, s' \in \mathrm{May}_a(s)\} \cup \{(\top, a, \top) \mid a \in \Sigma\}$ and $\longrightarrow = \{(s, \mathrm{Must}_j(s)) \mid s = (x,i) \in S, j \in J_i\} \cup \{(\bot, \emptyset)\}$.

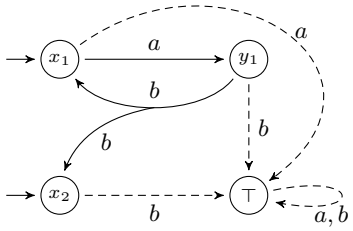
**Lemma 7.** *For all $\nu$HML formulae $\Delta$, $\Delta \equiv_{\mathrm{t}} hd(\Delta)$.*

Further, we remark that the overall translation from DMTS to $\nu$HML is quadratic and in the other direction inevitably exponential.

*Example 8.* Consider the $\nu$HML formula $X = (\langle a\rangle(\langle b\rangle X \wedge [a]\mathbf{ff}) \wedge [b]\mathbf{ff}) \vee [a]\mathbf{ff}$. Changing the formula into the normal form of Lemma 6 introduces a new variable $Y$ as illustrated below; $X$ remains the sole initial variable. The translation $hd$ then gives a DMTS with two initial states (the inconsistent state $\bot$ and redundant may transitions such as $x_1 \overset{a}{\dashrightarrow} x_2$, $x_2 \overset{b}{\dashrightarrow} x_1$, etc. have been omitted):

$$X = (\overbrace{\langle a\rangle Y \wedge [a]\mathbf{tt} \wedge [b]\mathbf{ff}}^{x_1}) \vee (\underbrace{[a]\mathbf{ff} \wedge [b]\mathbf{tt}}_{x_2})$$

$$Y = \underbrace{\langle b\rangle X \wedge [a]\mathbf{ff} \wedge [b]\mathbf{tt}}_{y_1}$$



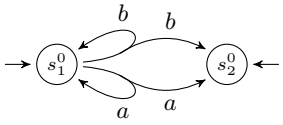

## 3 Specification Theory

In this section, we introduce operations of conjunction, disjunction, structural composition and quotient for NAA, DMTS and $\nu$HML. Together, these operations yield a *complete specification theory* in the sense of [1], which allows for compositional design and verification using both logical and structural operations. We remark that conjunction and disjunction are straightforward for logical formalisms such as $\nu$HML, whereas structural composition is more readily defined on behavioural formalisms such as (D)MTS. For the mixed formalism of NAA, disjunction is trivial as we permit multiple initial states, but conjunction requires some work. Note that our construction of conjunction works for nondeterministic systems in contrast to all the work in this area except for [5, 25].

### 3.1 Disjunction

The disjunction of NAA $S_1 = (S_1, S_1^0, \mathrm{Tran}_1)$ and $S_2 = (S_2, S_2^0, \mathrm{Tran}_2)$ is $S_1 \vee S_2 = (S_1 \cup S_2, S_1^0 \cup S_2^0, \mathrm{Tran}_1 \cup \mathrm{Tran}_2)$. Similarly, the disjunction of two DMTS $S_1 = (S_1, S_1^0, \dashrightarrow_1, \longrightarrow_1)$ and $S_2 = (S_2, S_2^0, \dashrightarrow_2, \longrightarrow_2)$ is $S_1 \vee S_2 = (S_1 \cup S_2, S_1^0 \cup S_2^0, \dashrightarrow_1 \cup \dashrightarrow_2, \longrightarrow_1 \cup \longrightarrow_2)$. It follows that disjunction respects the translation mappings *dn* and *nd* from the previous section.

**Theorem 9.** *Let $S_1$, $S_2$, $S_3$ be NAA or DMTS. Then $[\![S_1 \vee S_2]\!] = [\![S_1]\!] \cup [\![S_2]\!]$. Further, $S_1 \vee S_2 \leq_\mathrm{m} S_3$ iff $S_1 \leq_\mathrm{m} S_3$ and $S_2 \leq_\mathrm{m} S_3$.*




We point out one important distinction between NAA and DMTS: NAA with a *single* initial state are equally expressive as general NAA, while for DMTS, this is not the case. The example on the right shows a DMTS $(S, S^0, \dashrightarrow, \longrightarrow)$, with $S = S^0 = \{s_1^0, s_2^0\}$, $s_1^0 \longrightarrow \{(a, s_1^0), (a, s_2^0)\}$ and $s_1^0 \longrightarrow \{(b, s_1^0), (b, s_2^0)\}$ (and the corresponding may-transitions). Two initial states are necessary for capturing $[\![S]\!]$.

**Lemma 10.** *For any NAA $S$ there is a NAA $T = (T, T^0, \Psi)$ with $T^0 = \{t^0\}$ a singleton and $S \equiv_\mathrm{m} T$.*

### 3.2 Conjunction

Conjunction for DMTS is an extension of the construction from [5] for multiple initial states. Given two DMTS $(S_1, S_1^0, \dashrightarrow_1, \longrightarrow_1)$, $(S_2, S_2^0, \dashrightarrow_2, \longrightarrow_2)$, we define $S_1 \wedge S_2 = (S, S^0, \dashrightarrow, \longrightarrow)$ with $S = S_1 \times S_2$, $S^0 = S_1^0 \times S_2^0$, and

- $(s_1, s_2) \overset{a}{\dashrightarrow} (t_1, t_2)$ iff $s_1 \overset{a}{\dashrightarrow}_1 t_1$ and $s_2 \overset{a}{\dashrightarrow}_2 t_2$,
- for all $s_1 \longrightarrow N_1$, $(s_1, s_2) \longrightarrow \{(a, (t_1, t_2)) \mid (a, t_1) \in N_1, (s_1, s_2) \overset{a}{\dashrightarrow} (t_1, t_2)\}$,
- for all $s_2 \longrightarrow N_2$, $(s_1, s_2) \longrightarrow \{(a, (t_1, t_2)) \mid (a, t_2) \in N_2, (s_1, s_2) \overset{a}{\dashrightarrow} (t_1, t_2)\}$.

To define conjunction for NAA, we need auxiliary projection functions $\pi_i : \Sigma \times S_1 \times S_2 \to \Sigma \times S_i$. These are defined by

$$\pi_1(M) = \{(a, s_1) \mid \exists s_2 \in S_2 : (a, s_1, s_2) \in M\}$$
$$\pi_2(M) = \{(a, s_2) \mid \exists s_1 \in S_1 : (a, s_1, s_2) \in M\}$$

Given NAA $(S_1, S_1^0, \mathrm{Tran}_1)$, $(S_2, S_2^0, \mathrm{Tran}_2)$, define $S_1 \wedge S_2 = (S, S^0, \mathrm{Tran})$, with $S = S_1 \times S_2$, $S^0 = S_1^0 \times S_2^0$ and $\mathrm{Tran}((s_1, s_2)) = \{M \subseteq \Sigma \times S_1 \times S_2 \mid \pi_1(M) \in \mathrm{Tran}_1(s_1), \pi_2(M) \in \mathrm{Tran}_2(s_2)\}$.

**Lemma 11.** *For DMTS $S_1$, $S_2$, $dn(S_1 \wedge S_2) = dn(S_1) \wedge dn(S_2)$.*

For the translation from NAA to DMTS, $nd(S_1 \wedge S_2) = nd(S_1) \wedge nd(S_2)$ does not necessarily hold, as the translation changes the state space. However, Theorem 12 below will ensure that $nd(S_1 \wedge S_2) \equiv_\mathrm{t} nd(S_1) \wedge nd(S_2)$.

**Theorem 12.** *Let $S_1$, $S_2$, $S_3$ be NAA or DMTS. Then $[\![S_1 \wedge S_2]\!] = [\![S_1]\!] \cap [\![S_2]\!]$. Further, $S_1 \leq_\mathrm{m} S_2 \wedge S_3$ iff $S_1 \leq_\mathrm{m} S_2$ and $S_1 \leq_\mathrm{m} S_3$.*

**Theorem 13.** *With operations $\wedge$ and $\vee$, the sets of DMTS and NAA form bounded distributive lattices up to $\equiv_\mathrm{m}$.*

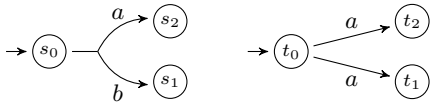
### 3.3 Structural Composition

We define structural composition for NAA. For NAA $S_1 = (S_1, S_1^0, \mathrm{Tran}_1)$, $S_2 = (S_2, S_2^0, \mathrm{Tran}_2)$, we define $S_1 \| S_2 = (S, S^0, \mathrm{Tran})$ with $S = S_1 \times S_2$, $S^0 = S_1^0 \times S_2^0$, and for all $(s_1, s_2) \in S$, $\mathrm{Tran}((s_1, s_2)) = \{M_1 \| M_2 \mid M_1 \in \mathrm{Tran}_1(s_1), M_2 \in \mathrm{Tran}_2(s_2)\}$, where $M_1 \| M_2 = \{(a, (t_1, t_2)) \mid (a, t_1) \in M_1, (a, t_2) \in M_2\}$.

**Lemma 14.** *Up to $\equiv_{\mathrm{m}}$, the operator $\|$ on NAA is associative and commutative, distributes over $\vee$, and has unit $\mathsf{U}$, where $\mathsf{U}$ is the LTS $(\{s\}, s, \longrightarrow)$ with $s \xrightarrow{a} s$ for all $a \in \Sigma$.*

**Theorem 15.** *For all NAA $S_1$, $S_2$, $S_3$, $S_4$, $S_1 \leq_{\mathrm{m}} S_3$ and $S_2 \leq_{\mathrm{m}} S_4$ imply $S_1 \| S_2 \leq_{\mathrm{m}} S_3 \| S_4$.*

We remark that structural composition on MTS [26] coincides with our NAA composition, so that for MTS $S_1$, $S_2$, $dn(S_1) \| dn(S_2) = dn(S_1 \| S_2)$. On the other hand, structural composition for DMTS (with single initial states) as defined in [5] is *weaker* than NAA composition, *i.e.* for DMTS $S_1$, $S_2$, and denoting by $\|'$ the composition from [5], only $dn(S_1) \| dn(S_2) \leq_{\mathrm{t}} dn(S_1 \|' S_2)$ holds. Consider for example the DMTS $S$ and $S'$ in the figure below. When considering their NAA composition, the initial state is the pair $(s_0, t_0)$ with $\mathrm{Tran}((s_0, t_0)) = \{\emptyset, \{(a, (s_2, t_1)), (a, (s_2, t_2))\}$. Since this constraint cannot be represented as a disjunctive must, there is no DMTS with a single initial state which can represent the NAA composition precisely.




Hence the DMTS composition of [5] is a DMTS over-approximation of the NAA composition, and translating from DMTS to NAA before composing (and back again) will generally give a tighter specification. However, as noted already in [23], MTS composition itself is an over-approximation, in the sense that there will generally be implementations $I \in [\![S_1 \| S_2]\!]$ which cannot be written $I = I_1 \| I_2$ for $I_1 \in [\![S_1]\!]$ and $I_2 \in [\![S_2]\!]$; the same is the case for NAA and DMTS.

### 3.4 Quotient

We now present one of the central contributions of this paper, the construction of quotient. The quotient $S/T$ is to be the most general specification that, when composed with $T$, refines $S$. In other words, it must satisfy the property that for all specifications $X$, $X \leq_{\mathrm{m}} S/T$ iff $X \parallel T \leq_{\mathrm{m}} S$. Quotient has been defined for deterministic MTS and for deterministic acceptance automata in [34]; here we extend it to the nondeterministic case (*i.e.* NAA). The construction incurs an exponential blow-up, which however is local and depends on the degree of nondeterminism. We also provide a quotient construction for nondeterministic MTS; this is useful because MTS encodings for NAA can be very compact.

Let $(S, S^0, \mathrm{Tran}_S)$, $(T, T^0, \mathrm{Tran}_T)$ be two NAA. We define the quotient $S/T = (Q, \{q^0\}, \mathrm{Tran}_Q)$. Let $Q = 2^{S\times T}_{\mathrm{Fin}}$ and $q^0 = \{(s^0, t^0) \mid s^0 \in S^0, t^0 \in T^0\}$. States in $Q$ will be written $\{s_1/t_1, \dots, s_n/t_n\}$ instead of $\{(s_1, t_1), \dots, (s_n, t_n)\}$.

In the following, we use the notation $x \in\in z$ as a shortcut for the fact that there exists $y$ with $x \in y \in z$. We first define $\mathrm{Tran}_Q(\emptyset) = 2^{\Sigma\times\{\emptyset\}}$. This means that the empty set of pairs is the universal state $\top$. Now let $q = \{s_1/t_1, \dots, s_n/t_n\} \in Q$. We first define the auxiliary set of possible transitions $pt(q)$ as follows. For $x \in S \cup T$, let $\alpha(x) = \{a \in \Sigma \mid \exists y : (a, y) \in\in \mathrm{Tran}(x)\}$ and $\gamma(q) = \bigcap_i \big(\alpha(s_i) \cup (\Sigma \setminus \alpha(t_i))\big)$. Let further $\pi_a(X) = \{x \mid (a, x) \in X\}$.

Let now $a \in \gamma(q)$. For all $i \in \{1, \dots, n\}$, let $\{t_{i,1}, \dots, t_{i,m_i}\} = \pi_a(\bigcup \mathrm{Tran}_T(t_i))$ be the possible next states from $t_i$ after an $a$-transition, and define

$$pt_a(q) = \big\{\{s_{i,j}/t_{i,j} \mid i \in \{1, \dots, n\}, j \in \{1, \dots, m_i\}\} \mid \\ \forall i \in \{1, \dots, n\} : \forall j \in \{1, \dots, m_i\} : (a, s_{i,j}) \in\in \mathrm{Tran}_S(s_i)\big\}$$

and $pt(q) = \bigcup_{a\in\Sigma}(\{a\} \times pt_a(q))$. Hence $pt_a(q)$ contains sets of possible next quotient states after an $a$-transition, each obtained by combining the $t_{i,j}$ with some permutation of possible next $a$-states in $S$. We then define

$$\mathrm{Tran}_Q(q) = \{X \subseteq pt(q) \mid \forall i : \forall Y \in \mathrm{Tran}_T(t_i) : X \triangleright Y \in \mathrm{Tran}_S(s_i)\},$$

where the operator $\triangleright$ is defined by $\{s_1/t_1, \dots, s_k/t_k\} \triangleright t_\ell = s_\ell$ and $X \triangleright Y = \{(a, x \triangleright y) \mid (a, x) \in X, (a, y) \in Y\}$. Hence $\mathrm{Tran}_Q(q)$ contains all sets of (possible) transitions which are compatible with all $t_i$ in the sense that (the projection of) their parallel composition with any set $Y \in \mathrm{Tran}_T(t_i)$ is in $\mathrm{Tran}_S(s_i)$.

**Theorem 16.** *For all NAA $S$, $T$ and $X$, $X \| T \leq_{\mathrm{m}} S$ iff $X \leq_{\mathrm{m}} S/T$.*

**Theorem 17.** *With operations $\wedge$, $\vee$, $\|$ and $/$, the set of NAA forms a commutative residuated lattice up to $\equiv_{\mathrm{m}}$.*

This theorem makes clear the relation of NAA to linear logic [19]: except for completeness of the lattice induced by $\wedge$ and $\vee$ (*cf.* Theorem 13), NAA form a *commutative unital Girard quantale* [39], the standard algebraic setting for linear logic. Completeness of the lattice can be obtained by allowing infinite conjunctions and disjunctions (and infinite NAA).

### 3.5 Quotient for MTS

We now give a quotient algorithm for the important special case of MTS, which results in a much more compact quotient than the NAA construction in the previous section. However, MTS are not closed under quotient; *cf.* [27, Thm. 5.5]. We show that the quotient of two MTS will generally be a DMTS.

Let $(S, s^0, \dashrightarrow_S, \longrightarrow_S)$ and $(T, t^0, \dashrightarrow_T, \longrightarrow_T)$ be nondeterministic MTS. We define the quotient $S/T = (Q, \{q^0\}, \dashrightarrow_Q, \longrightarrow_Q)$. We let $Q = 2^{S\times T}_{\mathrm{Fin}}$ as before, and $q^0 = \{(s^0, t^0)\}$. The state $\emptyset \in Q$ is again universal, so we define $\emptyset \overset{a}{\dashrightarrow} \emptyset$ for all $a \in \Sigma$. There are no must transitions from $\emptyset$.

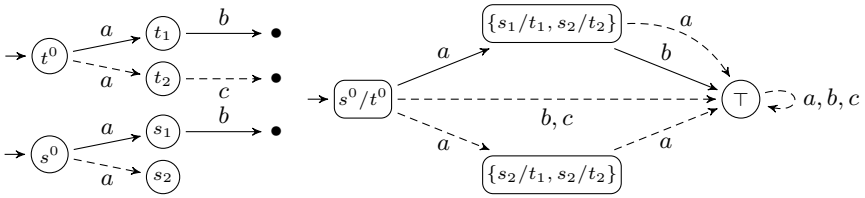



**Fig. 4.** Two nondeterministic MTS and their quotient

Let $\alpha(s)$, $\gamma(q)$ be as in the previous section. For convenience, we work with sets $\mathrm{May}_a(s)$, for $a \in \Sigma$ and states $s$, instead of may transitions, *i.e.* we have $\mathrm{May}_a(s) = \{t \mid s \xrightarrow{a} t\}$.

Let $q = \{s_1/t_1, \ldots, s_n/t_n\} \in Q$ and $a \in \Sigma$. First we define the may transitions. If $a \in \gamma(q)$ then for each $i \in \{1, \ldots, n\}$, write $\mathrm{May}_a(t_i) = \{t_{i,1}, \ldots, t_{i,m_i}\}$, and define

$$\mathrm{May}_a(q) = \big\{\{s_{i,j}/t_{i,j} \mid i \in \{1,\ldots,n\}, j \in \{1,\ldots,m_i\}\} \mid \\ \forall i \in \{1,\ldots,n\} : \forall j \in \{1,\ldots,m_i\} : s_{i,j} \in \mathrm{May}_a(s_i)\big\}.$$

For the (disjunctive) must-transitions, we let, for every $s_i \xrightarrow{a} s'$,

$$q \longrightarrow \{(a, M) \in \{a\} \times \mathrm{May}_a(q) \mid \exists t' : s'/t' \in M,\ t_i \xrightarrow{a} t'\}.$$

*Example 18.* We illustrate the construction on an example. Let $S$ and $T$ be the MTS in the left part of Fig. 4. We construct $S/T$; the end result is displayed in the right part of the figure.

First we construct the may-successors of $s^0/t^0$. Under $b$ and $c$ there are no constraints, hence we go to $\top$. For $a$, we have all permutations of assignments of successors of $s$ to successors of $t$, namely $\{s_1/t_1, s_1/t_2\}$, $\{s_1/t_1, s_2/t_2\}$, $\{s_2/t_1, s_1/t_2\}$ and $\{s_2/t_1, s_2/t_2\}$. Since there is a must-transition from $s$ (to $s_1$), we create a disjunctive must-transition to all successors that can be used to yield a must-transition when composed with the must-transition from $t$ to $t_1$. These are all successors where $t_1$ is mapped to $s_1$, hence the first two. However, $\{s_1/t_1, s_1/t_2\}$ will turn out inconsistent, as it requires to refine $s_1$ by a composition with $t_2$. As $t_2$ has no must under $b$, the composition has none either, hence the must of $s_1$ can never be matched. As a result, after pruning, the disjunctive must from $\{s^0/t^0\}$ leads only to $\{s_1/t_1, s_2/t_2\}$. Further, $\{s_2/t_1, s_1/t_2\}$ is inconsistent for the same reason, so that we only have one other may-transition under $a$ from $\{s^0/t^0\}$.

Now $\{s_1/t_1, s_2/t_2\}$ is obliged to have a must under $b$ so that it refines $s_1$ when composed with $t_1$, but cannot have any $c$ in order to match $s_2$ when composed with $t_2$. Similarly, $\{s_2/t_1, s_2/t_2\}$ has neither $c$ nor $b$. One can easily verify that $T \| (S/T) \equiv_{\mathrm{m}} S$ in this case.

Note that the constructions may create inconsistent states, which have no implementation. In order to get a consistent system, it needs to be pruned. This is standard and the details can be found in Appendix **??**. The pruning can be done in polynomial time.

**Theorem 19.** *For all MTS $S$, $T$ and $X$, $X \leq_{\mathrm{m}} S/T$ iff $T \| X \leq_{\mathrm{m}} S$.*

## 4 Conclusion and Future Work

In this paper we have introduced a general specification framework whose basis consists of three different but equally expressive formalisms: one of a graphical behavioural kind (DMTS), one logic-based ($\nu$HML) and one an intermediate language between the former two (NAA). We have shown that the framework possesses a rich algebraic structure that includes logical (conjunction, disjunction) and structural operations (parallel composition and quotient). Moreover, the construction of the quotient solves an open problem in the area of MTS. As for future work, we hope to establish the exact complexity of the quotient constructions. We conjecture that the exponential blow-up of the construction is in general unavoidable.

## References

1. S.S. Bauer, A. David, R. Hennicker, K.G. Larsen, A. Legay, U. Nyman, and A. Wasowski. Moving from specifications to contracts in component-based design. In *FASE*, pages 43–58, 2012.
2. S.S. Bauer, P. Mayer, and A. Legay. MIO workbench: A tool for compositional design with modal input/output interfaces. In *ATVA*, pages 418–421, 2011.
3. N. Beneš and J. Křetínský. Process algebra for modal transition systemses. In *MEMICS*, pages 9–18, 2010.
4. N. Beneš, J. Křetínský, K. G. Larsen, M. H. Møller, and J. Srba. Parametric modal transition systems. In *ATVA*, pages 275–289, 2011.
5. N. Beneš, I. Černá, and J. Křetínský. Modal transition systems: Composition and LTL model checking. In *ATVA*, pages 228–242, 2011.
6. P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic J. IGPL*, 8(3):339–365, 2000.
7. BMoTras. http://delahaye.benoit.free.fr/BMoTraS.tar.
8. A. Børjesson, K.G. Larsen, and A. Skou. Generality in design and compositional verification using TAV. *Formal Meth. Syst. Design*, 6(3):239–258, 1995.
9. G. Boudol and K.G. Larsen. Graphical versus logical specifications. *Theor. Comput. Sci.*, 106(1):3–20, 1992.
10. G. Bruns. An industrial application of modal process logic. *Sci. Comput. Program.*, 29(1-2):3–22, 1997.
11. G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *CAV*, pages 274–287, 1999.
12. L. Caires and L. Cardelli. A spatial logic for concurrency (part I). *Inf. Comput.*, 186(2):194–235, 2003.
13. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, pages 52–71, 1981.
14. P. Darondeau, J. Dubreil, and H. Marchand. Supervisory control for modal specifications of services. In *WODES*, pages 428–435, 2010.
15. N. D'Ippolito, D. Fischbein, H. Foster, and S. Uchitel. MTSA: Eclipse support for modal transition systems construction, analysis and elaboration. In *ETX*, pages 6–10, 2007.

16. H. Fecher and H. Schmidt. Comparing disjunctive modal transition systems with an one-selecting variant. *J. Logic Algebr. Program.*, 77(1-2):20–39, 2008.
17. H. Fecher and M. Steffen. Characteristic mu-calculus formulas for underspecified transition systems. *Electr. Notes Theor. Comput. Sci.*, 128(2):103–116, 2005.
18. W. Fokkink, R.J. van Glabbeek, and P. de Wind. Compositionality of Hennessy-Milner logic by structural operational semantics. *Theor. Comput. Sci.*, 354(3):421–440, 2006.
19. Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.
20. J.B. Hart, L. Rafter, and C. Tsinakis. The structure of commutative residuated lattices. *Internat. J. Algebra Comput.*, 12(4):509–524, 2002.
21. M. Hennessy. Acceptance trees. *J. ACM*, 32(4):896–928, 1985.
22. M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.
23. H. Hüttel and K.G. Larsen. The use of static constructs in a modal process logic. In *Logic at Botik*, pages 163–180, 1989.
24. D. Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.
25. K. G. Larsen and Liu X. Equation solving using modal transition systems. In *LICS*, pages 108–117, 1990.
26. K.G. Larsen. Modal specifications. In *Automatic Verification Methods for Finite State Systems*, pages 232–246, 1989.
27. K.G. Larsen. Ideal specification formalism = expressivity + compositionality + decidability + testability + ... In *CONCUR*, pages 33–56, 1990.
28. K.G. Larsen. Proof systems for satisfiability in Hennessy-Milner logic with recursion. *Theor. Comput. Sci.*, 72:265–288, 1990.
29. K.G. Larsen and Liu X. Compositionality through an operational semantics of contexts. In *ICALP*, pages 526–539, 1990.
30. U. Nyman. *Modal Transition Systems as the Basis for Interface Theories and Product Lines*. PhD thesis, Institut for Datalogi, Aalborg Universitet, 2008.
31. P.W. O'Hearn, J.C. Reynolds, and H. Yang. Local reasoning about programs that alter data structures. In *CSL*, pages 1–19, 2001.
32. A.N. Prior. *Papers on Time and Tense*. Oxford: Clarendon Press, 1968.
33. J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Symp. Program.*, pages 337–351, 1982.
34. J.-B. Raclet. Residual for component specifications. *Electr. Notes Theor. Comput. Sci.*, 215:93–110, 2008.
35. J.-B. Raclet, E. Badouel, A. Benveniste, B. Caillaud, and R. Passerone. Why are modalities good for interface theories? In *ACSD*, pages 119–127, 2009.
36. J.C. Reynolds. Separation logic: A logic for shared mutable data structures. In *LICS*, pages 55–74, 2002.
37. S. Uchitel and M. Chechik. Merging partial behavioural models. In *SIGSOFT FSE*, pages 43–52, 2004.
38. M. Ward and R. P. Dilworth. Residuated lattices. *Trans. AMS*, 45(3):335–354, 1939.
39. David N. Yetter. Quantales and (noncommutative) linear logic. *J. Symb. Log.*, 55(1):41–64, 1990.