Maximal Synthesis for Hennessy-Milner Logic with the Box Modality

A.C. van Hulst [*] **M.A. Reniers** [*] **W.J. Fokkink** [**,*]

[*] *Eindhoven University of Technology*
(email: {ahulst,m.a.reniers}@tue.nl)
[**] *VU University Amsterdam* (email: w.j.fokkink@vu.nl)

Abstract: This paper presents a novel approach to adapt a behavioral model in order to satisfy a requirement in Hennessy-Milner Logic, including an additional box modality operator, expressing an invariant formula. Control system synthesis, as defined in this way, retains all non-invalidating behavior, and thereby guarantees maximal permissiveness for supervisory control. This research extends earlier work by embracing a broader synthesized logic, enabling synthesis with respect to invariant formulas for non-deterministic behavioral models. All definitions and proofs in this paper have been computer verified using the Coq proof assistant.

Keywords: Control system synthesis, Supervisory control, Temporal logic, Invariants

1. INTRODUCTION

Software for the control of hardware or embedded systems is usually created via the translation of informal requirements into concrete implementations. This workflow is error-prone, as correspondence between desired functionality and actual realization is unverified. This issue gave rise to the field of supervisory control theory (Ramadge and Wonham (1987)), where a controller is synthesized, based upon an automata-like behavioral specification of the uncontrolled system (plant). A logical specification is then enforced by disabling certain events in particular states. Within this context, the term *controller* should be understood as a separate entity which regulates system behavior by imposing restrictions upon actions that can be performed at certain states. Controller synthesis then becomes the automated generation or derivation of this entity, based upon a description of the original model and its desired behavior.

The approach to supervisory control theory via event-disabling, as described first in Ramadge and Wonham (1987), is inherently limited in its expressiveness. Since required behavior is specified in Ramadge and Wonham (1987) as an automaton, one can only limit the behavior of the plant, but it is not possible to express and guarantee desired properties. For instance, using an automaton, it is not possible to express that in each state, an e-step should exist. However, such a requirement can clearly be expressed in modal logic. Therefore, in this research, we aim to extend the logic for which supervisory control can be applied. It is important to stress the difference between this problem and the generation of an arbitrary formula-satisfying model. Also, the research described in this paper differs from the standard supervisory control approach as described in Ramadge and Wonham (1987), where a controller is considered a separate entity. Instead, control system synthesis strives to construct an adaptation of an existing behavioral model, where all non-invalidating steps are retained. The latter property is referred to as maximal permissiveness and is of key importance if additional analysis needs to be applied to the synthesized model. For instance, if the resulting system needs to be subjected to further design, optimization or synthesis for liveness properties.

This paper addresses the maximal synthesis problem for Hennessy-Milner Logic (HML), including an additional box modality operator, used to express invariant properties. We introduce a novel approach for adapting a behavioral model in order to satisfy a logical requirement. In order to achieve this, synthesis is defined by constructing a restricted mapping of the original behavioral relation onto the state-formula product space. As a formula might not be satisfiable with respect to a particular model, a separate test is introduced for *synthesizability*. If this test deems the formula to be realizable, synthesis results in a satisfying model. We show that the obtained synthesized system is maximal with respect to all formula satisfying models which are related to the original system via simulation. Various restrictions lead to a unique solution and a tractable and efficient derivation of synthesizability. All definitions and proofs in this paper have been computer verified using the Coq proof assistant (see Van Hulst (2013)).

In Van Hulst et al. (2013) we proposed a recursive construction for synthesis for Hennessy-Milner Logic, which lacks the ability to express that a property holds for all reachable states. This method incorporates a separate unfolding step up to the applicable reach of the synthesized modal formula. This approach is therefore inherently not applicable to formulas reaching over an arbitrary number of transitions and states, such as the box operator. Also, maximality was only obtained up to all deterministic models. The current paper presents a novel synthesis technique which addresses each of these issues.

Due to setup of the proposed synthesis construction here, unique outcomes are enforced via applying restrictions

upon the synthesized logic. Synthesis as presented in Van Hulst et al. (2013) possibly results in a set of multiple maximal solutions, in particular for disjunctive formulas. As these solutions are essentially incomparable, a choice was made in Van Hulst et al. (2013) to include all of these in the result set. This is a key difference between the research presented in this paper and the unrestricted setting for disjunctive formulas in Van Hulst et al. (2013).

Previous and ongoing similar research revolves around varieties of the logical framework considered in this paper. In Antoniotti (1995), the supervisor synthesis problem is addressed for a fragment of CTL, limiting disjunctive formulas, among other restrictions. This approach uses a labeling algorithm from model checking introduced in Kupferman et al. (2001). The problem introduced in Antoniotti (1995) is shown to be NP-complete in Antoniotti and Mishra (1995). The work in Clarke and Emerson (2008) concerns the generation of abstract program descriptions, called synchronization skeletons, from specifications in branching-time temporal logic. It is shown how a satisfiable program requirement ensures a consistent specification via a bounded finite model property. Semantic tableaux are used as part of the state of a controller constructed using a propositional LTL specification in Deshpande and Varaiya (1996). In this work action inhibition is ensured via blocking of controllable signals. Propositional temporal formulas are also used in Manna and Wolper (1984) in order to synthesize synchronizations in communicating processes using a tableau-like method. This method employs a separate unfolding step, similar to Van Hulst et al. (2013). Using ready simulation as a behavioral relation, the work in Lüttgen and Vogler (2011) embeds a temporal specification in a Logic LTS. This introduces a way of system specification via a mixed operational and specification formalism. In Vardi (1995), a behavioral specification is mapped to a Rabin automaton. A program is then extracted, based on a proof that the given LTL specification is realizable. The latter construction is comparable to the notion of synthesizability as will be introduced in this paper. The approach in Vardi (1995) is somewhat comparable to the technique introduced in Jiang and Kumar (2006), where a restricted CTL* formula is mapped to a Rabin tree automaton, which is then transformed algorithmically into a deterministic supervisor. Important work by D'Ippolito et al. (2010) (see also D'Ippolito et al. (2013)) cannot be left unmentioned. Within the framework of the world machine model, it distinguishes between controlled and monitored actions, and between system goals and environment assumptions. It achieves synthesis of behavior models for a relatively expressive set of liveness properties stated in fluent temporal logic. The employed technique is to derive a controller from a winning strategy in a two-player game between original and required behavior, expressed using the notion of generalized reactivity, as introduced in D'Ippolito et al. (2013).

The approach in this paper differs in several aspects and improves upon earlier work. Previous research essentially equates synthesis to requirement satisfiability in Clarke and Emerson (2008), Kupferman and Vardi (2000) and Vardi (1995), instead of adapting existing behavior. Other approaches do so, but disjunctive formulas are considered to a lesser extent in Antoniotti (1995) and Jiang and Kumar (2006), among other logical restrictions. In this paper, the type of behavioral model is invariant under synthesis, while a different post-synthesis formalism is used in Lüttgen and Vogler (2011). Additionally, other approaches do not consider maximal permissiveness, such as in Manna and Wolper (1984), Lüttgen and Vogler (2011) and Deshpande and Varaiya (1996). As an important addition to previous research, the approach in this paper incorporates non-determinism in behavioral models, while other research is limited to deterministic transition systems, as described in Kupferman and Vardi (2000) and Jiang and Kumar (2006). Supervisory control as described in Ramadge and Wonham (1987) synthesizes for absence of deadlock and marker state reachability. Furthermore, it differentiates between controllable and uncontrollable events. These properties are not taken into account here for the sake of simplicity.

The remainder of this paper is set up as follows. In Section 2, a number of formal definitions for the behavioral model, synthesized logic and formula validity are introduced, as these are required to properly interpret the synthesis problem as well as the formal definition of the proposed solution. An intuitive explanation of the applied synthesis technique is the subject of Section 3, as well as a discussion of the employed logical restrictions. A formal definition of synthesis is then provided per coinductive construction in Section 4. Synthesis is shown to result in a formula-satisfying system in Section 5. Maximal permissiveness is shown in Section 6, while Section 7 presents concluding remarks as well as an outline of future work.

2. DEFINITIONS

A number of initial definitions are stated here to aid in understanding the chosen approach to synthesis, as well as the formal definition thereof. As our computational model we merge the standard formalisms of Kripke-structure and Labeled Transition System (LTS). The computational model of *Kripke-LTS* in Definition 1 (introduced as *KTS* in Müller-Olm et al. (1999)), consists of system states and transitions between states labeled by events, which are here assumed to be chosen from a global event-set $\mathcal{E}$. Basic properties from the global set $\mathcal{P}$ can be assigned to states by a labeling function $L : X \mapsto 2^{\mathcal{P}}$, for state-space X.

Definition 1. A Kripke-LTS is a four-tuple $(X, L, \rightarrow, x)$ for state space X, labeling function $L : X \mapsto 2^{\mathcal{P}}$, transition relation $\rightarrow \subseteq X \times \mathcal{E} \times X$ and initial state $x \in X$. The universe of all Kripke-LTSes is denoted as $\mathcal{K}$. As usual, the notation $x \xrightarrow{e} x'$ will be used to denote that $(x, e, x') \in \rightarrow$.

The definitions for two fundamental behavioral relations are provided below. These are analogous to definitions provided in Van Glabbeek (1990).

Definition 2. Let $k' = (X', L', \rightarrow', x') \in \mathcal{K}$ and $k = (X, L, \rightarrow, x) \in \mathcal{K}$. The models k' and k are related via *simulation* (notation: $k' \preceq k$) if there exists a relation $R \subseteq X' \times X$ such that $(x', x) \in R$ and for all $(y', y) \in R$, the following holds:

(1) $L'(y') = L(y)$
(2) For each $y' \xrightarrow{e}{}' z'$ there exists a $z \in X$ such that $y \xrightarrow{e} z$ and $(z', z) \in R$.

If R is not obvious from the context, or of particular importance, the notation $k' \preceq_R k$ will be used.

Definition 3. If there exists an R such that $k' \preceq_R k$ and $k \preceq_{R^{-1}} k'$ for $k', k \in \mathcal{K}$, according to Definition 2, then k' and k are related via *bisimulation* (notation $k' \underline{\leftrightarrow} k$).

The main logic $\mathcal{F}$ that will be considered in this paper is provided in Definition 4.

Definition 4. The set of formulas in Hennessy-Milner Logic, as introduced in Hennessy and Milner (1985), with an additional box modality operator, is inductively defined as the set $\mathcal{F}$, where p and e range over $\mathcal{P}$ and $\mathcal{E}$ respectively.

$$\mathcal{F} ::= \mathit{true} \mid \mathit{false} \mid p \mid \neg p \mid \mathcal{F} \wedge \mathcal{F} \mid \mathcal{F} \vee \mathcal{F} \mid [e]\mathcal{F} \mid \langle e \rangle \mathcal{F} \mid \Box \mathcal{F}$$

The elements of the logic $\mathcal{F}$ are introduced briefly. The operators $[e]$, $\langle e \rangle$, and $\Box$ will be referred to as *modal* operators or *modalities*, as is common in the literature. The operators $[e]$ and $\Box$ are characterized here as being *positive*, while $\langle e \rangle$ will be referred to as a *negative* operator in the remainder of this paper. This characterization is due to the fact that positive operators remain satisfied under additional transition removal.

The formulas *true* and *false* are self-explanatory. The formulas $p \in \mathcal{F}$ and $\neg p \in \mathcal{F}$ for $p \in \mathcal{P}$ can be used to test whether basic property p respectively holds in a certain state, or not. The operators for conjunction and disjunction are interpreted via their usual semantics. The positive modality $[e]\, f$ tests whether formula f holds after every e-step, while the negative formula $\langle e \rangle f$ expresses that there exists an e-step after which f holds. Finally, the second positive modality $\Box\, f$ expresses that f holds in every reachable state. Note that we only have negation at the level of basic formulas $\neg p$, for simplicity. This can be lifted to negation on formulas without modal operators. Validity is made formal in Definition 5.

Definition 5. Formula validity $\vDash \subset \mathcal{K} \times \mathcal{F}$ is defined by the following set of deduction rules. The notation $\rightarrow^*$ refers to the reflexive, transitive closure of $\rightarrow$; that is, $(x, y) \in \rightarrow^*$ if $x = y$ or there exist x' and e such that $x \xrightarrow{e} x'$ and $(x', y) \in \rightarrow^*$.

$$\frac{}{k \vDash \mathit{true}} \qquad \frac{p \in L(x)}{(X, L, \rightarrow, x) \vDash p} \qquad \frac{p \notin L(x)}{(X, L, \rightarrow, x) \vDash \neg p}$$

$$\frac{k \vDash f \quad k \vDash g}{k \vDash f \wedge g} \qquad \frac{k \vDash f}{k \vDash f \vee g} \qquad \frac{k \vDash g}{k \vDash f \vee g}$$

$$\frac{\forall\, x \xrightarrow{e} x' \ (X, L, \rightarrow, x') \vDash f}{(X, L, \rightarrow, x) \vDash [e]\, f} \qquad \frac{x \xrightarrow{e} x' \ (X, L, \rightarrow, x') \vDash f}{(X, L, \rightarrow, x) \vDash \langle e \rangle f}$$

$$\frac{\forall\, x \rightarrow^* x' \ (X, L, \rightarrow, x') \vDash f}{(X, L, \rightarrow, x) \vDash \Box\, f}$$

3. SYNTHESIS

Supervisory control as defined in Ramadge and Wonham (1987) disables events, based on requirements specified as an automaton. The resulting controlled system as the product of synthesis satisfies the stated requirements. Synthesis as defined in this paper abstracts from a separate controlling entity, and is defined as an adaptation of existing behavior in order to satisfy a given logical requirement. This adaptation is realized by removing transitions in an on-the-fly created, bisimulation-equivalent model of the original transition system. One of the general observations introduced in this section is that the decision to allow a transition in the synthesized system depends upon a reduced formula being realizable in the target state. This leads to a two-level approach for synthesis and *synthesizability*. The purpose of this section is to illustrate this two-folded setup at an intuitive level.

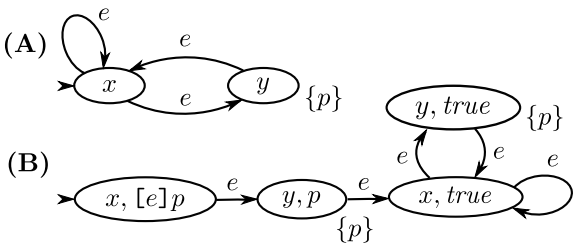

Fig. 1. Illustrating formula reductions and loop unwinding. Synthesis for formula $[e]p$ on original model **(A)** results in synthesized system **(B)**.

Viewed from an algorithmical perspective, synthesis starts in the initial state, for the given formula. It then progresses through the transition system, thereby applying formula reduction induced by transition labels. The purpose of Algorithm 1 is purely illustrative, but it does highlight the key principles of the synthesis approach introduced here. If the original transition relation is finite and normalization of invariant reducts is applied, this method is linear in the product of the number of transitions and the formula depth. Also, under these conditions, the proposed method is terminating. Nevertheless, it is important to understand Algorithm 1 as an intuitive interpretation of the formal construction detailed in section 4, and not as the actual realization thereof.

Algorithm 1: Simplified synthesis algorithm

```
1  x ← initial state
2  f ← synthesized formula
3  for each x -e-> x'
4     for each formula e-reduct f' of f
5        if f' is synthesizable in x'
6           add (x, f) -e-» (x', f') to solution
7     end for
8  end for
9  for each constructed (x', f')
10    repeat at 1
```

We consider a number of examples concerning the formula reductions. Fig. 1 shows the synthesis for original system **(A)**, resulting in solution **(B)**. It details how states may play different roles at various stages of synthesis, by initially disabling the self-loop in state x, while it re-occurs at a later stage, since it does not invalidate $[e]p$ at this point. Note that the state-formula pairs thereby induce an embedded unfolding in order to preserve as much behavior as possible, due to the $[e]p \twoheadrightarrow p \twoheadrightarrow \mathit{true}$ reduction sequence.

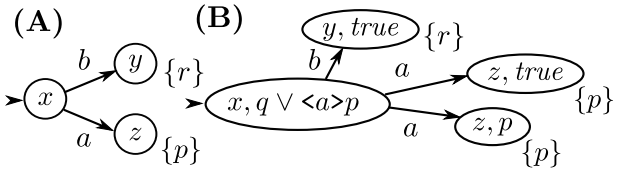

Fig. 2. Non-synthesizable disjunct q does not induce any transitions, while $\langle a\rangle p$ creates a witness step to (z, p) as well as a copy of original behavior in order to achieve maximal permissiveness.

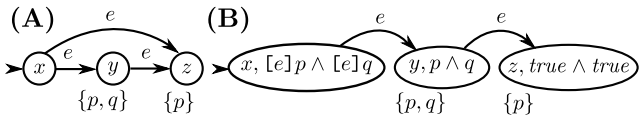

Fig. 3. Illustrating preservation of conjunctive formulas over modal operator reductions for synthesis of the formula $[e]p \wedge [e]q$.

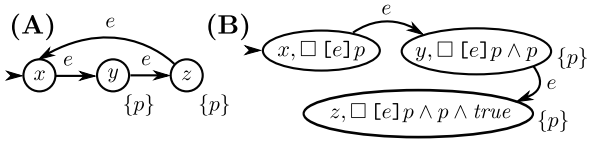

Fig. 4. Synthesis for $\Box\,[e]p$ results in transitions to states which have to satisfy the invariant formula itself, as well as a corresponding reduct of $[e]p$.

Fig. 2 depicts how transitions can be induced by both disjuncts, provided that they are synthesizable. Since the formula q is not synthesizable in x, it does not create any outgoing transition. The other disjunct $\langle a\rangle p$ creates transitions to (y, true) and (z, true), as original behavior should be copied for the existential modality, in order to achieve maximal permissiveness. The actual synthesized transition for the formula $\langle a\rangle p$ is obtained by creating a transition towards (z, p).

As shown in Fig. 3, conjunctive formulas are preserved under synthesis. This ensures formula validity for the entire reach of both conjuncts. The modal operator reductions are as shown previously. A similar approach is used for the box modality operator as shown in Fig. 4. In general, for $f \in \mathcal{F}$ the formula $\Box\, f$ reduces to $\Box\, f \wedge f'$ if f itself reduces to f'. This coaligns with the very nature of an invariant expression, where a formula is required to be true in every reachable state.

Restrictions apply to the synthesized logic $\mathcal{F}$ in order to obtain a unique solution and to avoid a codependent definition of synthesis and synthesizability. Unique solutions are enforced by forbidding initial positive modalities in both operands of a disjunction. For instance, the formula $[a]p \vee \langle b\rangle [a]p$ is allowed, while $[a]p \vee [a]q$ is not allowed. This is illustrated by the following example. Suppose the formula $[a]r \vee [b]p$ is synthesized on the original model **(A)**, as shown in Fig. 2. Observe that removing each of the two different transitions leads to two maximal solutions, which are essentially incomparable. One might argue that both solutions should be included in the set of synthesis results, as we proposed in Van Hulst et al. (2013). However, the synthesis technique outlined in this paper constructs a single solution, and therefore a unique outcome is enforced by the aforementioned restriction.

A further restriction is applied by requiring a strict partition between events under the modality $[e]$ and $\langle e\rangle$. This allows formulas such as $[a]\langle b\rangle p$, but forbids the formula $[a]\langle a\rangle p$. Again, the necessity for this restriction is explained by example. Suppose synthesis for $[e]r \wedge \langle e\rangle p$ is applied to the model **(A)** in Fig. 3. It is clear that both conjuncts can be realized, and are thus synthesizable. However, the conjunctive formula in its entirety cannot be synthesized, since the operand $\langle e\rangle p$ requires the existence of a transition which the operand $[e]r$ disables. A distinct possibility is to employ a recursive or fixed point definition for synthesizability in terms of synthesis itself. However, at this point it is not clear whether all required results can be obtained in such a way. Furthermore, synthesizability in its current form is guaranteed to be tractable. We argue that these arguments justify the proposed restriction for event partitioning. The same underlying problem emerges for the box modality operator. For instance, synthesizability for the formula $\Box\langle e\rangle q$ on model **(A)** in Fig. 4 depends upon the transitions left in place by synthesis of the formula $\langle e\rangle q$. Again, this would require a codependent definition of synthesis and synthesizability.

4. CONSTRUCTION

As explained in the previous section, a number of restrictions is imposed upon the synthesized logic $\mathcal{F}$, in order to obtain a unique solution and to avoid codependent definitions for synthesis and synthesizability. This overcomes the difference between synthesizability and solution existence via the interpretation of existence of a formula satisfying model, which is related to the original model via simulation. The restrictions are categorized as follows:

(1) A restriction upon disjunctive formulas such that a unique synthesis solution is guaranteed. This restriction forbids initial positive modalities in *both* disjuncts. For instance, the formula $[a]p \vee \langle b\rangle [a]p$ is allowed, while the formula $[a]p \vee [b]q$ is not allowed, since the later one contains the operator $[e]$, for some $e \in \mathcal{E}$, in both disjuncts. Likewise, the formula $\Box\, p \vee \langle a\rangle p$ is allowed while the, formula $\Box\, p \vee [a]p$ is not allowed since, again, both disjuncts start with a positive modality.

(2) A number of other restrictions which are required in order to show that the obtained synthesis result indeed satisfies the synthesized formula. These restrictions include a strict partition between events under the $[e]$ and $\langle e\rangle$ operators. For instance, the formula $[a]\langle b\rangle p$ is allowed, while the formula $[a]\langle a\rangle p$ is not allowed, since the latter contains the event a under both the operators $[a]$ and $\langle a\rangle$. Another restriction is to exclude negative formulas from appearing under the box modality operator. This allows formulas such as $\langle e\rangle \Box\, p$, while $\Box\langle e\rangle p$ is not allowed. As a last restriction we introduce a strict choice between the types of formulas which may constitute a conjunction. A conjunctive formula, in its entirety, is either a strict HML-formula (excluding $\Box$) or a strictly positive formula. This allows formulas such as $[a]p \wedge \langle b\rangle q$ and $[a]p \wedge \Box\, q$, while the formula $\Box\, p \wedge \langle a\rangle q$ is not allowed.

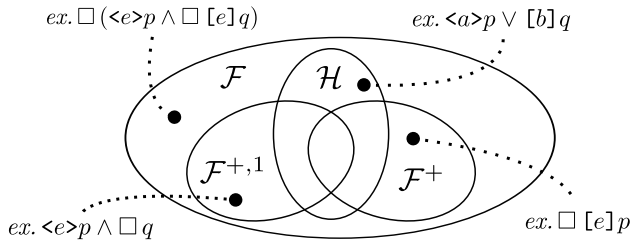

Fig. 5. Various subsets of the synthesized logic $\mathcal{F}$, including typical examples of formulas in these sets. The set of positive formulas $\mathcal{F}^+$, the set of formulas having an initial positive modality $\mathcal{F}^{+,1}$ and the set of formulas in Hennessy-Milner Logic $\mathcal{H}$.

Restrictions on formulas are given by means of three separate grammars in Definition 6, which should be interpreted via their natural embedding in $\mathcal{F}$, as shown in Fig. 5. Based upon these restrictions, a definition of well-formedness for formulas is provided in Definition 7. This notion of well-formedness will serve as a premise for the theorem expressing validity of the synthesis results, later on. For clarity, note that well-formedness is defined with regard to $\mathcal{F}^{+,1}$ via an *exclusive* definition.

Definition 6. The set of *initially positive* formulas $\mathcal{F}^{+,1}$, characterized as having a positive modality at the first level, is defined here. Additionally, sets of *strictly positive* formulas $\mathcal{F}^+$ and HML formulas $\mathcal{H}$ are defined here. Let $e \in \mathcal{E}$ and $p \in \mathcal{P}$ in the following definition:

$$
\begin{array}{lll}
\mathcal{F}^{+,1} & ::= & \mathcal{F}^{+,1} \wedge \mathcal{F} \mid \mathcal{F} \wedge \mathcal{F}^{+,1} \mid \mathcal{F}^{+,1} \wedge \mathcal{F}^{+,1} \mid \mathcal{F}^{+,1} \vee \mathcal{F} \mid \\
 & & \mathcal{F} \vee \mathcal{F}^{+,1} \mid \mathcal{F}^{+,1} \vee \mathcal{F}^{+,1} \mid [e]\mathcal{F} \mid \Box\mathcal{F} \\
\mathcal{F}^{+} & ::= & \mathit{true} \mid \mathit{false} \mid p \mid \neg p \mid \mathcal{F}^{+} \wedge \mathcal{F}^{+} \mid \mathcal{F}^{+} \vee \mathcal{F}^{+} \mid \\
 & & [e]\mathcal{F}^{+} \mid \Box\mathcal{F}^{+} \\
\mathcal{H} & ::= & \mathit{true} \mid \mathit{false} \mid p \mid \neg p \mid \mathcal{H} \wedge \mathcal{H} \mid \mathcal{H} \vee \mathcal{H} \mid [e]\mathcal{H} \mid \langle e\rangle\mathcal{H}
\end{array}
$$

Definition 7. A formula $f \in \mathcal{F}$ is *well-formed* if there exists a set $U \subseteq \mathcal{E}$ such that the following definition holds. If f is well-formed with respect to event-set U, then the notation $wf_U(f)$ is employed. Note that well-formedness for $\mathit{true}, \mathit{false}, p$ and $\neg p$ holds vacuously with respect to any event set U. Let $e \in \mathcal{E}$ and $f, g \in \mathcal{F}$ in the following definition:

$$
\begin{array}{lcl}
wf_U(f \wedge g) & \iff & wf_U(f) \text{ and } wf_U(g) \text{ and} \\
 & & ((f \wedge g) \in \mathcal{F}^{+} \text{ or } (f \wedge g) \in \mathcal{H}) \\
wf_U(f \vee g) & \iff & wf_U(f) \text{ and } wf_U(g) \text{ and} \\
 & & (f \notin \mathcal{F}^{+,1} \text{ or } g \notin \mathcal{F}^{+,1}) \\
wf_U([e]f) & \iff & e \notin U \text{ and } wf_U(f) \\
wf_U(\langle e\rangle f) & \iff & e \in U \text{ and } wf_U(f) \\
wf_U(\Box f) & \iff & wf_U(f) \text{ and } f \in \mathcal{F}^{+}
\end{array}
$$

Given a transition system $k = (X, L, \rightarrow, x)$ and formula $f \in \mathcal{F}$, a new transition relation over $X \times \mathcal{F}$ is constructed in Definition 9. This synthesized transition relation is in turn defined in terms of a predicate for *synthesizability* in Definition 8.

Definition 8. For state space X, labeling function L, and transition relation $\rightarrow$, the predicate $\uparrow \subset X \times \mathcal{F}$ is defined to capture the notion of synthesizability. The formula $f \in \mathcal{F}$ is *synthesizable* in state $x \in X$ (notation: $(x, f) \uparrow$) as derived by the following set of deduction rules:

$$
\frac{}{(x, \mathit{true}) \uparrow} \qquad \frac{p \in L(x)}{(x, p) \uparrow} \qquad \frac{p \notin L(x)}{(x, \neg p) \uparrow}
$$

$$
\frac{(x, f) \uparrow \quad (x, g) \uparrow}{(x, f \wedge g) \uparrow} \qquad \frac{(x, f) \uparrow}{(x, f \vee g) \uparrow} \qquad \frac{(x, g) \uparrow}{(x, f \vee g) \uparrow}
$$

$$
\frac{}{(x, [e]f) \uparrow} \qquad \frac{x \xrightarrow{e} x' \quad (x', f) \uparrow}{(x, \langle e\rangle f) \uparrow} \qquad \frac{(x, f) \uparrow}{(x, \Box f) \uparrow}
$$

Definition 9. Given a transition system $(X, L, \rightarrow, x)$ a new transition relation $\twoheadrightarrow \subset (X \times \mathcal{F}) \times \mathcal{E} \times (X \times \mathcal{F})$ is defined by the following deduction rules:

$$
\frac{x \xrightarrow{e} x'}{(x, \mathit{true}) \xrightarrow{e} (x', \mathit{true})} \qquad \frac{x \xrightarrow{e} x'}{(x, p) \xrightarrow{e} (x', \mathit{true})}
$$

$$
\frac{x \xrightarrow{e} x'}{(x, \neg p) \xrightarrow{e} (x', \mathit{true})} \qquad \frac{(x, f) \xrightarrow{e} (x', f') \quad (x, g) \xrightarrow{e} (x', g')}{(x, f \wedge g) \xrightarrow{e} (x', f' \wedge g')}
$$

$$
\frac{(x, f) \uparrow \quad (x, f) \xrightarrow{e} (x', f')}{(x, f \vee g) \xrightarrow{e} (x', f')} \qquad \frac{(x, g) \uparrow \quad (x, g) \xrightarrow{e} (x', g')}{(x, f \vee g) \xrightarrow{e} (x', g')}
$$

$$
\frac{x \xrightarrow{e} x' \quad (x', f) \uparrow}{(x, [e]f) \xrightarrow{e} (x', f)} \qquad \frac{x \xrightarrow{e} x' \quad e \neq e'}{(x, [e']f) \xrightarrow{e} (x', \mathit{true})}
$$

$$
\frac{x \xrightarrow{e} x' \quad (x', f) \uparrow}{(x, \langle e\rangle f) \xrightarrow{e} (x', f)} \qquad \frac{x \xrightarrow{e} x'}{(x, \langle e'\rangle f) \xrightarrow{e} (x', \mathit{true})}
$$

$$
\frac{(x, f) \xrightarrow{e} (x', f') \quad (x', f) \uparrow}{(x, \Box f) \xrightarrow{e} (x', \Box f \wedge f')}
$$

In order to incorporate the synthesized transition relation into a newly created transition system, the construction as given in Definition 10 is used.

Definition 10. For state space X, labeling function $L : X \mapsto 2^{\mathcal{P}}$, transition relation $\rightarrow \subseteq X \times \mathcal{E} \times X$ and initial state $x \in X$, the projection for L is defined on state-formula pairs as $L_{\text{proj}}(y, f) = L(y)$ for each $y \in X$ and $f \in \mathcal{F}$. The *synthesized system* is then defined as: $(X \times \mathcal{F}, L_{\text{proj}}, \twoheadrightarrow, (x, f)) \in \mathcal{K}$. Note that $\twoheadrightarrow$ in this construction refers to the transition relation in Definition 9. In the remainder of this paper, the notation (x, f) will be used for the synthesized system. Also, we will incidentally use the initial state x to refer to an unmodified LTS as given in Definition 1.

5. VALIDITY

Towards a theorem expressing that the synthesis result indeed satisfies the required formula, a number of additional lemmas is required. A variant of the covariant-contravariant simulation, as described in Fábregas et al. (2010), is given in Definition 11, and a helpful notation for conjunctive formulas in Definition 12.

Definition 11. Let $U \subseteq \mathcal{E}$ and $k' = (X', L', \rightarrow', x'), k = (X, L, \rightarrow, x) \in \mathcal{K}$. The models k' and k are related via covariant-contravariant simulation (notation: $k' \precapprox_U k$) with respect to event set U, if there exists a relation $R \subseteq X' \times X$ such that $(x', x) \in R$ and for all $(y', y) \in R$ the following holds:

(1) $L'(y') = L(y)$.
(2) For all $e \in \mathcal{E}$ and $z' \in X'$ such that $y' \xrightarrow{e}{}' z'$ there exists a $z \in X$ such that $y \xrightarrow{e} z$ and $(z', z) \in R$.
(3) For all $e \in U$ and $z \in X$ such that $y \xrightarrow{e} z$ there exists a $z' \in X'$ such that $y' \xrightarrow{e}{}' z'$ and $(z', z) \in R$.

The sole purpose of introducing Definition 11 is for proof-technical reasons. Simulation as given in Definition 2 does preserve validity of formulas containing only positive operators (see Lemma 1(A)). However, in order to transfer validity of a formula containing $\langle e \rangle f$, a different coinductive relation such as $\precsim\!\!\approx_U$ needs to be established. If $k' \precsim\!\!\approx_U k$ and $k \vDash \langle e \rangle f$ then, indeed, $k' \vDash \langle e \rangle f$, provided that $e \in U$ (see Lemma 1(C)).

Definition 12. For $f, f_1, f_2, \ldots, f_n \in \mathcal{F}$ the notation $f \in_\wedge f_1 \wedge f_2 \wedge \ldots \wedge f_n$ is used to denote that there exists at least one $1 \leq i \leq n$ such that $f_i \equiv f$.

For the three behavioral (pre)congruences as defined before, simulation, bisimulation, and covariant-contravariant simulation, formula validity is concerned in Lemma 1.

Lemma 1. Let k', k in $\mathcal{K}$ and $U \subseteq \mathcal{E}$. The following correspondences exist between the respective behavioral relations and preservation of formula validity:

(A) If $k' \preceq k$ and $f \in \mathcal{F}^+$ then $k \vDash f$ implies $k' \vDash f$.
(B) If $k' \leftrightarrows k$ and $f \in \mathcal{F}$ then $k' \vDash f$ if and only if $k \vDash f$.
(C) If $k' \precsim\!\!\approx_U k$ and $f \in \mathcal{H}$ such that $wf_U(f)$ then $k \vDash f$ implies $k' \vDash f$.

Proof. The first two results can be straightforwardly obtained using structural induction on f. The proof for (C) can be derived via results in Fábregas et al. (2010). ■

A number of results can be obtained if two state-formula pairs are related via the synthesized transition relation as given in Definition 9. These properties are characterized in Lemma 2.

Lemma 2. If $(x, f) \overset{e}{\rightarrowtriangle} (x', f')$ for $x, x' \in X$, $e \in \mathcal{E}$ and $f, f' \in \mathcal{F}$, then the following properties can be derived:

(A) $x \xrightarrow{e} x'$
(B) If $f \in \mathcal{H}$ then $f' \in \mathcal{H}$
(C) If $f \in \mathcal{F}^+$ then $f' \in \mathcal{F}^+$
(D) If $wf_U(f)$ and $f \in \mathcal{H}$ then $wf_U(f')$
(E) $(x', f') \uparrow$

Proof. These results can be obtained by induction towards the structure of f. ■

Lemma 3. If $wf_U(f)$, $e \in U$, $f \in \mathcal{H}$, $(x, f) \uparrow$ and $x \xrightarrow{e} x'$ then there exists an $f' \in \mathcal{F}$ such that $(x, f) \overset{e}{\rightarrowtriangle} (x', f')$.

Proof. By structural induction on f. For the cases where $f \equiv \mathit{true}$, $f \equiv p$ or $f \equiv \neg p$, we choose $f' = \mathit{true}$ and apply the corresponding rule from Definition 9. Also, it is clear that $(x, \mathit{false}) \not\uparrow$. The cases $f \equiv f_1 \wedge f_2$ and $f \equiv f_1 \vee f_2$ are solved by induction. Note that it follows from Definition 6 and 8 that the induction premises are satisfied. Let $f \equiv [e']f'$ for some $e' \in \mathcal{E}$ and $f' \in \mathcal{F}$. If $e = e'$ then $e \notin U$ according to $wf_U([e']f')$, which clearly leads to a contradiction. If $e \neq e'$ then there exists a step $(x, [e']) \overset{e}{\rightarrowtriangle} (x', \mathit{true})$ by Definition 9. For $f \equiv \langle e' \rangle f'$ there always exists a *true*-step, as shown in Definition 9, while the case for $f \equiv \Box f'$ is prevented by $f \in \mathcal{H}$. ■

The next two lemmas are required to prove validity for conjunction.

Lemma 4. For $f, g \in \mathcal{H}$ such that $wf_U(f \wedge g)$ and $(x, f \wedge g) \uparrow$ it holds that $(x, f \wedge g) \precsim\!\!\approx_U (x, f)$ and $(x, f \wedge g) \precsim\!\!\approx_U (x, g)$.

Proof. Choose $R = \{((y, f \wedge g), (y, f)) \mid f, g \in \mathcal{F}, (y, f \wedge g) \uparrow, wf_U(f \wedge g), (f \wedge g) \in \mathcal{H}\}$ as a witness relation in order to obtain the first result. If $(y, f \wedge g) \overset{e}{\rightarrowtriangle} (y', f' \wedge g')$ then there also exists a step $(y, f) \overset{e}{\rightarrowtriangle} (y', f')$. By Lemma 2(B),(E),(D) preservation of inclusion in HML, synthesizability and well-formedness with respect to $f' \wedge g'$ is obtained. If $(y, f) \overset{e}{\rightarrowtriangle} (y', f')$ for some $e \in U$ then there exists a step $(y, g) \overset{e}{\rightarrowtriangle} (y', g')$ by Lemma 3, since $g \in \mathcal{H}$, $(x, g) \uparrow$ and $x \xrightarrow{e} x'$. This leads to $(y, f \wedge g) \overset{e}{\rightarrowtriangle} (y', f' \wedge g')$, by Definition 9. A step $x \xrightarrow{e} x'$ is obtained by Lemma 2(A), to satisfy all premises for the application of Lemma 3. ■

Lemma 5. For each $f \in \mathcal{F}^+$ and $g, h \in \mathcal{F}$ with $g \in_\wedge h$ it holds that $(x, g) \vDash f$ implies $(x, h) \vDash f$.

Proof. Choose $R = \{((y, h'), (y, g')) \mid g' \in_\wedge h', y \in X\}$ as a witness for $(x, h) \preceq_R (x, g)$. By Lemma 1(A) it is clear that $(x, h) \vDash f$. ■

The following two lemmas are required for validity of disjunction.

Lemma 6. If $f, g, h \in \mathcal{F}$ such that $(x, g) \vDash f$, $(x, g) \uparrow$ and $f \notin \mathcal{F}^{+,1}$ then $(x, g \vee h) \vDash f$ and $(x, h \vee g) \vDash f$.

Proof. The first conclusion is shown by structural induction on f. The cases $f \equiv \mathit{true}$, $f \equiv \mathit{false}$, $f \equiv p$ and $f \equiv \neg p$ are solved directly, as the validity of these formulas does not depend on the transition relation. The cases $f \equiv f_1 \wedge f_2$ and $f \equiv f_1 \vee f_2$ are solved by application of the induction hypotheses, and Definition 6, to satisfy the induction premises for containment in $\mathcal{F}^{+,1}$. Let $f \equiv \langle e \rangle f'$ for some $f' \in \mathcal{F}$. As $(x, g) \vDash f$, there exists a step $(x, g) \overset{e}{\rightarrowtriangle} (x', g')$ such that $(x', g') \vDash f'$. This leads to $(x, g \vee h) \overset{e}{\rightarrowtriangle} (x', g')$ since $(x, g) \uparrow$. Note that the cases $f \equiv [e]f'$ and $f \equiv \Box f'$ can be omitted, as these formulas are contained in $\mathcal{F}^{+,1}$. The result $(x, h \vee g) \vDash f$ is obtained using exactly the same reasoning. ■

Lemma 7. If $f, g \in \mathcal{F}$ such that $(x, f) \vDash f$, $(x, g) \not\uparrow$ and $(x, f) \uparrow$, then $(x, f \vee g) \vDash f$ and $(x, g \vee f) \vDash f$.

Proof. Since $(x, g) \not\uparrow$, it is clear from Definition 9 that $(x, f \vee g)$ cannot do a step originating from g. This allows the construction of a bisimulation between (x, f) and $(x, f \vee g)$, which directly leads to the first result by Lemma 1(B). The result $(x, g \vee f) \vDash f$ is obtained in precisely the same way. ■

Lemma 8. For each $g \in \mathcal{F}$ and $f \in \mathcal{F}^+$ such that $(x, g) \vDash f$ it holds that $(x, \Box g) \vDash f$.

Proof. Pick $R = \{((x, \Box g), (x, g))\} \cup \{((y, h'), (y, g')) \mid g' \in_\wedge h', g', h' \in \mathcal{F}\}$ as a witness for $(x, \Box g) \preceq_R (x, g)$. Then the result follows from Lemma 1(A). ■

Theorem 9. For each $f \in \mathcal{F}$ and $U \subseteq \mathcal{E}$ such that $wf_U(f)$ it holds that $(x, f) \uparrow$ implies $(x, f) \vDash f$.

Proof. The proof for this theorem is by induction towards the structure of f. Note that for the cases $f \equiv \mathit{true}$, $f \equiv \mathit{false}$, $f \equiv p$ and $f \equiv \neg p$ for $p \in \mathcal{P}$ the result follows directly from the $(x, f) \uparrow$ premise, which leaves the next case $f \equiv f_1 \wedge f_2$ to consider.

Assume $(x, f_1) \uparrow$ and $(x, f_2) \uparrow$. Then, by induction: $(x, f_1) \vDash f_1$ and $(x, f_2) \vDash f_2$. Since $wf_U(f_1 \wedge f_2)$, it is clear from Definition 6 that either $(f_1 \wedge f_2) \in \mathcal{F}^+$ or $(f_1 \wedge f_2) \in \mathcal{H}$. For the first case, it is directly clear that $(x, f_1 \wedge f_2) \vDash f_1$ and $(x, f_1 \wedge f_2) \vDash f_2$ by Lemma 5. For the second case, it holds that $(x, f_1 \wedge f_2) \precapprox_U (x, f_1)$ and $(x, f_1 \wedge f_2) \precapprox_U (x, f_2)$ by Lemma 4. Preservance of formula validity for HML-formulas, as shown in Lemma 1(C), gives $(x, f_1 \wedge f_2) \vDash f_1$ and $(x, f_1 \wedge f_2) \vDash f_2$.

Consider the case where $f \equiv f_1 \vee f_2$. By Definition 6 and $wf_U(f_1 \vee f_2)$ it holds that that either $f_1 \notin \mathcal{F}^{+,1}$ or $f_2 \notin \mathcal{F}^{+,1}$. The proof is shown here for the first of these two cases only. Since $(x, f_1 \vee f_2) \uparrow$, then by Definition 8 this leads to the two cases $(x, f_1) \uparrow$ or $(x, f_2) \uparrow$. If $(x, f_1) \uparrow$, then $(x, f_1) \vDash f_1$ by induction. Via Lemma 6 the result $(x, f_1 \vee f_2) \vDash f_1$ is obtained. Suppose that $(x, f_2) \uparrow$. If it also holds that $(x, f_1) \uparrow$, the result $(x, f_1) \vDash f_1$ is obtained by induction and, again, $(x, f_1 \vee f_2) \vDash f_1$ by Lemma 6. Otherwise, if $(x, f_1) \nvDash$ and $(x, f_2) \uparrow$, then $(x, f_2) \vDash f_2$ by induction, and $(x, f_1 \vee f_2) \vDash f_2$ by Lemma 7.

The remaining cases consider the three modalities. Assume $f \equiv [e]\, f'$, for some $f' \in \mathcal{F}$. Then it needs to be shown that $(x, [e]\, f') \vDash [e]\, f'$. Let $(x, [e]\, f') \xrightarrow{e} (x', f')$, which is the only relevant e-step according to Definition 9. By induction it holds that $(x', f') \vDash f'$. If $f \equiv \langle e \rangle f'$ for some $f' \in \mathcal{F}$, then it is clear that $(x', \langle e \rangle f') \uparrow$, and by Definition 8 there exists a step $x \xrightarrow{e} x'$ such that $(x', f') \uparrow$. Application of the relevant rule in Definition 9 gives $(x, \langle e \rangle f') \xrightarrow{e} (x', f')$. Note that $(x', f') \vDash f'$ is resolved by application of the induction hypothesis.

The remaining case to consider is when $f \equiv \Box f'$ for some $f' \in \mathcal{F}$. Assume $(x, \Box f') \twoheadrightarrow^* (x', \Box f' \wedge f_1 \wedge f_2 \wedge \ldots \wedge f_n)$, for some $n \geq 0$. Use either $(x, \Box f) \uparrow$ or Lemma 1(E) to obtain that $(x', f') \uparrow$. Then, by induction hypothesis, it is clear that $(x', f') \vDash f'$. Application of Lemma 8 gives $(x', \Box f') \vDash f'$ and Lemma 5 gives $(x', \Box f' \wedge f_1 \wedge f_2 \ldots f_n) \vDash f$. Note that $f' \in \mathcal{F}^+$ is obtained via $wf_U(\Box f')$. ∎

Two additional theorems are required to establish the validity of this technique for synthesis. In Theorem 10 it is shown that synthesis results are, indeed, simulations of the original system. Theorem 11 shows that the intuitive understanding of solution existence implies a positive result for the formal synthesizability predicate as given in Definition 8. Note that the reverse implication of Theorem 11, is shown by Theorems 9 and 10.

Theorem 10. Given state space X. For each $x \in X$ and $f \in \mathcal{F}$ it holds that $(x, f) \preceq x$.

Proof. Choose $R = \{((y, g), y) \mid y \in X, g \in \mathcal{F}\}$ as a witness, and apply Lemma 2(A). ∎

Theorem 11. If $f \in \mathcal{F}$ and $k' \preceq x$ such that $k' \vDash f$ then $(x, f) \uparrow$.

Proof. Apply structural induction on f. Observe that the cases for both $(x, p) \uparrow$ and $(x, \neg p) \uparrow$ can be solved because strict equivalence of labels is required in Definition 2. The cases $f \equiv f_1 \wedge f_2$ and $f \equiv f_1 \vee f_2$ are quickly solved by application of the respective induction hypotheses. Formulas $[e]\, f$ are immediately synthesizable as shown in Definition 8. If $k' \vDash \langle e \rangle f$ and $k' = (X', L', \rightarrow', x')$, then there exists a step $x' \xrightarrow{e} {}' y'$ such that $(X', L', \rightarrow', y') \vDash f$. By Definition 2 there exists a step $x \xrightarrow{e} y$ such that $(X', L', \rightarrow', y') \preceq (X, L, \rightarrow, y)$. Application of the induction hypothesis leads to $(y, f) \uparrow$, from which it can be obtained that $(x, \langle e \rangle f) \uparrow$ by Definition 8. The only requirement for $(x, \Box f) \uparrow$ is $(x, f) \uparrow$, which is readily obtained by induction and because $k' \vDash \Box f$, which immediately leads to $k' \vDash f$. ∎

6. MAXIMALITY

This section works towards a coinductive proof for the maximality requirement of synthesis as defined in this paper. If there exists a formula satisfying model, which is related to the original model via simulation, then it is also related to the synthesis result via simulation. A sub-result about formula reductions is obtained via Lemma 12, while the actual maximality property is shown in Theorem 13

Lemma 12. If $x' \preceq x$, $y' \preceq y$, $x' \xrightarrow{e} y'$, $x \xrightarrow{e} y$ and $x' \vDash f$, then there exists an $f' \in \mathcal{F}$ such that $(x, f) \xrightarrow{e} (y, f')$ and $y' \vDash f'$.

Proof. The proof is by induction towards the structure of f. Note that for the cases $f \equiv \mathit{true}$, $f \equiv p$ and $f \equiv \neg p$, choose $f' \equiv \mathit{true}$, and the result follows directly by application of the corresponding rule in Definition 9. For $f \equiv \mathit{false}$, it is clear that $x' \nvDash f$. The case for $f \equiv f_1 \wedge f_2$ is solved straightforwardly by application of the induction hypothesis.

For $f \equiv f_1 \vee f_2$, only the case for $x' \vDash f_1$ is considered here. Application of the induction hypothesis results in an $f' \in \mathcal{F}$ such that $(x, f_1) \xrightarrow{e} (y, f')$. Note that the rule for construction of the step $(x, f_1 \vee f_2) \xrightarrow{e} (y, f')$ may now be applied, as $(x, f_1) \uparrow$ can be obtained via Theorem 11.

If $f \equiv [e']\, f'$, then a distinction is made between the following cases. If $e = e'$, then f' satisfies the existential result. Again $(y, f') \uparrow$ is obtained via Theorem 11. If $e \neq e'$, then *true* is the appropriate choice. The latter argument is repeated for the case where $f \equiv \langle e' \rangle f'$, as Definition 9 clearly shows that *true* can always be chosen in this case. If $f \equiv \Box f'$, then induction results in an $f'' \in \mathcal{F}$ such that $(x, f') \xrightarrow{e} (y, f'')$. Note that the induction hypothesis can be applied here since $x' \vDash \Box f'$ and therefore $x' \vDash f'$. Choosing $\Box f' \wedge f''$ resolves the existential question. Again, $(y, f) \uparrow$ is obtained via Theorem 11. ∎

Theorem 13. Let $f \in \mathcal{F}$ and $k' \in \mathcal{K}$. If $k' \preceq_R x$ and $k' \vDash f$, then $k' \preceq (x, f)$.

Proof. Assume $k' = (X', L', \rightarrow', x')$ and choose $R' = \{(y', (y, g)) \mid (y', y) \in R, y' \vDash g\}$ as a witness relation for the required simulation result. If $y' \xrightarrow{e} {}' z'$ then, since $(y', y) \in R$, there exists a corresponding step $y \xrightarrow{e} z$ such that $(z', z) \in R$. As $y' \vDash g$, a $g' \in \mathcal{F}$ is obtained via Lemma 12 such that $(y, g) \xrightarrow{e} (z, g')$ and $z' \vDash g'$. The inclusion $(z', (z, g)) \in R'$ completes the proof. ∎

7. CONCLUSIONS

In this paper, a synthesis technique for Hennessy-Milner Logic with the box modality is presented. The constructed synthesis result satisfies the synthesized formula, and is related via simulation to the original system. The intuitive understanding of solution availability, that is, existence of a formula satisfying model which is related to the original model via simulation. This intuitive interpretation of solution existence coincides with the formally defined predicate for synthesizability. The synthesis result is maximally permissive with respect to all simulating models which satisfy the required formula. These results show that the outlined approach of a restricted map of the behavioral relation onto the state-formula product space is viable. The presented results extend existing work by embracing a broader synthesized logic. It also introduces new insights on synthesis of a coinductive nature and preserves uniformity of the type of non-deterministic behavior models before and after synthesis.

The proposed limitations on the synthesized logic are induced for several reasons. Multiple solutions are avoided by limiting disjunctive formulas to contain a first-level positive operator in at most one disjunct. In Van Hulst et al. (2013) we proposed a synthesis method for unrestricted disjunctions of HML formulas. It remains an open question whether this recursive construction is compatible with the techniques introduced in this paper. The two other limitations are a strict partition of events under opposing modalities, and not allowing the existential modality under box modality.

Besides attempts to subdue the logical limitations considered in this paper, we also intend to focus our attempts on extensions of a more applicative nature. For instance, one may add variables and guarded transitions to the behavioral model subject to synthesis. Yet another research step might be the integration of the presented synthesis technique with a strict partition between controllable and uncontrollable events, as is a customary point of view in the field of supervisory control theory Ramadge and Wonham (1987). This would limit the synthesis method to only disable transitions labeled by controllable events. A different problem exists with regard to algorithmic realization of the proposed synthesis method for the box modality. For instance, the formula $\Box p$ has the infinite reduction sequence $\Box p \twoheadrightarrow \Box p \wedge \mathit{true} \twoheadrightarrow \Box p \wedge \mathit{true} \wedge \mathit{true} \ldots$. This leads to an infinite-state result for a finite-state original model having a loop. Such an unbounded number of states is clearly undesired, and we strive to investigate solutions by considering formulas under logical equivalence or normalization.

ACKNOWLEDGEMENTS

This work is supported by the EU FP7 Programme under grant agreement no. 295261 (MEALS)

REFERENCES

Antoniotti, M. (1995). *Synthesis and Verification of Discrete Controllers for Robotics and Manufacturing Devices with Temporal Logic and the Control-D System.* Ph.D. thesis, New York University.

Antoniotti, M. and Mishra, B. (1995). The supervisor synthesis problem for unrestricted CTL is NP-complete. Technical report, New York University.

Clarke, E. and Emerson, A. (2008). Design and synthesis of synchronization skeletons using branching time temporal logic. In *25 Years of Model Checking*, volume 5000, 196–215. Springer.

Deshpande, A. and Varaiya, P. (1996). Semantic tableau for control of PLTL formulae. In *Proceedings of CDC*, volume 2, 2243–2248. IEEE.

D'Ippolito, N., Braberman, V., Piterman, N., and Uchitel, S. (2010). Synthesis of live behaviour models. In *Proceedings of FSE*, 77–86. ACM.

D'Ippolito, N., Braberman, V., Piterman, N., and Uchitel, S. (2013). Synthesizing nonanomalous event-based controllers for liveness goals. *Transactions on Software Engineering and Methodology*, 22(1), 9.

Fábregas, I., de Frutos Escrig, D., and Palomino, M. (2010). Logics for contravariant simulations. In *Proceedings of FORTE*, volume 6117, 224–231. Springer.

Hennessy, M. and Milner, R. (1985). Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1), 137–161.

Jiang, S. and Kumar, R. (2006). Supervisory control of discrete event systems with CTL* temporal logic specifications. *Journal on Control and Optimization*, 44(6), 2079–2103.

Kupferman, O. and Vardi, M. (2000). μ-calculus synthesis. In *Proceedings of MFCS*, volume 1893, 497–507. Springer.

Kupferman, O., Vardi, M., and Wolper, P. (2001). Module checking. *Information and Computation*, 164(2), 322–344.

Lüttgen, G. and Vogler, W. (2011). Safe reasoning with Logic LTS. *Theoretical Computer Science*, 412(28), 3337–3357.

Manna, Z. and Wolper, P. (1984). Synthesis of communicating processes from temporal logic specifications. *Transactions on Programming Languages and Systems*, 6(1), 68–93.

Müller-Olm, M., Schmidt, D., and Steffen, B. (1999). Model-checking: A tutorial introduction. In *Proceedings of SAS*, volume 1694, 330–354. Springer.

Ramadge, P. and Wonham, W. (1987). Supervisory control of a class of discrete event processes. *Journal on Control and Optimization*, 25(1), 206–230.

Van Glabbeek, R. (1990). The linear time-branching time spectrum. In *Proceedings of CONCUR*, volume 458, 278–297. Springer.

Van Hulst, A. (2013). Computer verified proofs for this paper (using coq version 8.3). `http://seweb.se.wtb.tue.nl/~ahulst/wodes/`.

Van Hulst, A., Reniers, M., and Fokkink, W. (2013). Maximal Synthesis for Hennessy-Milner Logic. In *Proceedings of ACSD*, 1–10. IEEE.

Vardi, M. (1995). An automata-theoretic approach to linear temporal logic. In *Proceedings of Logics for Concurrency*, volume 1043, 238–266. Springer.