Maximally Permissive Controlled System Synthesis for Modal Logic [⋆]

A.C. van Hulst, M.A. Reniers, and W.J. Fokkink

Eindhoven University of Technology, The Netherlands

Abstract. We propose a new method for controlled system synthesis on non-deterministic automata, which includes the synthesis for deadlock-freeness, as well as invariant and reachability expressions. Our technique restricts the behavior of a Kripke-structure with labeled transitions, representing the uncontrolled system, such that it adheres to a given requirement specification in an expressive modal logic, while all non-invalidating behavior is retained. This induces maximal permissiveness in the context of supervisory control. Research presented in this paper allows a system model to be constrained according to a broad set of liveness, safety and fairness specifications of desired behavior, and embraces most concepts from Ramadge-Wonham supervisory control, including controllability and marker-state reachability. The synthesis construction is formally verified using the Coq proof assistant.

1 Introduction

This paper concerns the controlled system synthesis on non-deterministic automata for requirements in modal logic. The controlled systems perspective treats the system under control — the *plant* — and a system component which restricts the plant behavior — the *controller* — as a single integrated entity. This means that we take a model of all possible plant behavior, and construct a new model which is constrained according to a logical specification of desired behavior — the *requirements*. The *synthesis*, of such a restricted behavioral model incorporates a number of concepts from supervisory control theory [6], which affirm the generated model as being a proper controlled system, in relation to the original plant specification. Events are strictly partitioned into being either controllable or uncontrollable, such that synthesis only disallows events of the first type. In addition, synthesis preserves all behavior which does not invalidate the requirements, thereby inducing maximal permissiveness [6]. The requirement specification formalism extends Hennessy-Milner Logic [10] with invariant, reachability, and deadlock-freeness expressions, and is also able to express the supervisory control concept of marker-state reachability [13].

The intended contribution of this paper is two-fold. First, it presents a technique for controlled system synthesis in a non-deterministic context. Second, it defines synthesis for a modal logic which is able to capture a broad set of

[⋆] Supported by the EU FP7 Programme under grant agreement no. 295261 (MEALS).

requirements. Regarding the first contribution, it should be noted that supervisory control synthesis is often approached using a deterministic model of both plant and controller. Notably, the classic Ramadge-Wonham supervisory control theory [13] is a well-researched example of this setup. The resulting controller restricts the behavior of the deterministic plant model, thereby ensuring that it operates according to the requirements via event-based synchronization. A controlled system can not be constructed in this way for a non-deterministic model, as illustrated by example in Fig. 1. Assume that we wish to restrict all technically possible behavior of an indicator light of a printer (Fig. 1a) such that after a single *refill* event, the indicator light turns *green* immediately. In the solution shown in Fig. 1b, the self-loop at the right-most state is disallowed, as indicated using dashed lines, while all other behavior is preserved. Note that it is not possible to construct this maximally-permissive solution using event-based synchronization, as shown in [4]. However, an outcome as shown in Fig. 1b can be obtained by applying synthesis for the property $\Box\,[\mathit{refill}]\,\mathit{green}$, using the method described in this paper. As this example shows, the strict separation between plant and controller is not possible for non-deterministic models, and therefore we interpret the controlled system as a singular entity.

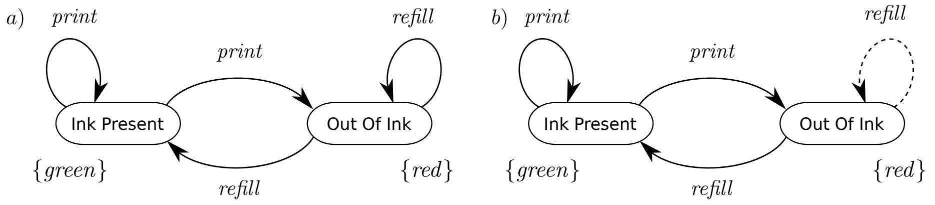


Fig. 1. Example of control synthesis in a non-deterministic context. A model for all possible behavior of an ink presence indicator light of a printer is restricted in such a way that after every *refill*, the state labeled with *green* is reached directly. Synthesis, as defined in this paper, of the property $\Box\,[\mathit{refill}]\,\mathit{green}$ upon the model in Fig. 1a, results in a synthesis outcome as in Fig. 1b., where disallowed behavior is indicated using dashed lines.

This requirement formalism applied in this paper, which extends Hennessy-Milner Logic with invariant and reachability operators, and also includes a test for deadlock-freeness, is able to express a broad set of liveness, safety, and fairness properties. For instance, an important liveness concept in supervisory control theory involves marker-state reachability, which is informally expressed as the requirement that it is always possible to reach a state which is said to be *marked*. This requirement is modeled as $\Box\,\Diamond\,\mathit{marked}$, using the requirement specification logic, in conjunction with assigning *marked* as a separate property to the designated states in the Kripke-model.

Safety-related requirements, which model the absence of faulty behavior, include deadlock-avoidance, expressed as $\Box\,\mathit{dlf}$ (i.e., invariantly, deadlock-free) and safety requirements of a more general nature. For instance, one might require that some type of communicating system is always able to perform a *receive* step, directly after every *send* step. Such a property is expressed as $\Box\,[send]\texttt{<}receive\texttt{>}true$, using the requirement specification logic applied in this paper. In addition, we argue that this logic is able to model a limited class of fairness properties. One might require from a system which uses a shared resource that in every state, the system has access to the resource (the state has the *access* property), or it can do a *lock* step to claim the resource, after which access is achieved immediately. We synthesize the property $\Box\,(access \vee \texttt{<}lock\texttt{>}access)$ in order to constrain the plant behavior in this way.

The remainder of this paper is set up as follows. We consider a number of related works on control synthesis in Section 2. Preliminary definitions in Section 3 introduce formal notions up to a clear statement of the synthesis problem. Section 4 concerns the formal definition of the synthesis construction while Section 5 lists a number of important theorems indicating correctness of the synthesis approach, with detailed proofs being available in textual form [19], as well as in computer-verified form [16].

2 Related Work

Earlier work by the same authors concerning synthesis for modal logic includes a recursive synthesis method for Hennessy-Milner Logic [17], and a synthesis method for a subset of the logic considered in this paper, with additional restrictions on combinations of modal operators [18]. This paper vastly improves previous efforts by allowing unrestricted synthesis for invariant formulas and including the new operator $\Diamond$ for reachability. It also takes into account deadlock-freeness, and uncontrollable events, thereby achieving controllability.

We analyze related work alongside the three intended improvements in this paper: 1) Allowance of non-determinism in plant specifications, 2) Expressiveness of the requirement specification formalism, and, 3) Adhering to some form of maximal permissiveness.

Ramadge-Wonham supervisory control [13] defines a broadly-embraced methodology for controller synthesis on deterministic plant models for requirements specified using automata. It defines a number of key elements in the relationship between plant and controlled system, such as controllability, marker-state reachability, deadlock-freeness and maximal permissiveness. Despite the fact that a strictly separated controller offers advantages from a developmental or implementational point of view, we argue that increased abstraction and flexibility justifies research into control synthesis for non-deterministic models. In addition, we emphasize that the automata-based description of desired behavior in the Ramadge-Wonham framework [13] does not allow the specification of requirements of existential nature. For instance, in this framework it is not possible to specify that a step labeled with a particular event *must* exist, hence the choice of modal logic as our requirement formalism.

Work by Pnueli and Rosner [12] concerns a treatment of synthesis for reactive systems, based upon a finite transducer model of the plant, and a temporal specification of desired behavior. This synthesis construction is developed further for deterministic automata in [12], but the treatment remains non-maximal. This research is extended in [2], which connects reactive synthesis to Ramadge-Wonham supervisory control using a parity-game based approach. The methodology described in [2] transforms the synthesis control problem for μ-calculus formulas in such a way that the set of satisfying models of a μ-calculus formula coincides with the set of controllers which enforce the controlled behavior. Although non-determinism is allowed in plant-specifications in [2], the treatment via loop-automata does not allow straightforward modeling of all (infinite) behaviors. Also, maximal permissiveness is not specified as a criterion for control synthesis in [2]. Interesting follow-up research is found in [3], for non-deterministic controllers over non-deterministic processes. However, the specification of desired behavior is limited to alternating automata [3], which do not allow complete coverage of invariant expressions over all modalities, or an equivalent thereof. Reactive synthesis is further applied to hierarchical [1] and recursive [11] component-based specifications. These works, which both are based upon a deterministic setting, provide a quite interesting setup from a developmental perspective, due to their focus on the re-usability of components.

Research in [20] relates Ramadge-Wonham supervisory control to an equivalent model-checking problem, resulting in important observations regarding the mutual exchangeability and complexity analysis of both problems. Despite the fact that research in [20] is limited to a deterministic setting, and synthesis results are not guaranteed to be maximally permissive, it does incorporate a quite expressible set of μ-calculus requirements. Other research based upon a dual approach between control synthesis and model checking studies the incremental effects of transition removal on the validity of μ-calculus formulas [14], [7].

Research by D'Ippolito and others [8], [9] is based upon the framework of the world machine model for the synthesis of liveness properties, stated in fluent temporal logic. A distinction is made between controlled and monitored behavior, and between system goals and environment assumptions [8]. A controller is then derived from a winning strategy in a two-player game between original and required behavior, as expressed in terms of the notion of generalized reactivity, as introduced in [8]. Research in [8] also emphasizes the fact that pruning-based synthesis is not adequate for control of non-deterministic models, and it defines synthesis of liveness goals under a maximality criterion, referred to as best-effort controller. However, this maximality requirement is trace-based and is therefore not able to signify inclusion of all possible infinite behaviors. In addition, some results in [8] are based upon the assumption of a deterministic plant specification.

3 Definitions

We assume a set $\mathcal{E}$ of events and a set $\mathcal{P}$ of state-based properties. In addition, we assume a strict partition of $\mathcal{E}$ into controllable events $\mathcal{C}$ and uncontrollable events

$\mathcal{U}$, such that $\mathcal{C} \cup \mathcal{U} = \mathcal{E}$ and $\mathcal{C} \cap \mathcal{U} = \emptyset$. State-based properties are used to capture state-based information, and are assigned to states using a labeling function. Example properties are shown in Fig. 1, as *red* and *green*. Fig. 1 also shows examples of the events *print* and *refill*, which are assumed to be controllable in this example. Events are used to capture system dynamics, and represent actions occurring when the system transitions between states. Controllable events may be used to model actuator actions in the plant, while an uncontrollable event may represent, for instance, a sensor reading. Basic properties and events are used to model plant behavior in the form of a Kripke-structure [5] with labeled transitions, to be abbreviated as Kripke-LTS, as formalized in Definition 1. Note that we assume finiteness of the given transition relation.

Definition 1. *We define a Kripke-LTS as a four-tuple $(X, L, \longrightarrow, x)$ for state-space X, labeling function $L : X \mapsto 2^{\mathcal{P}}$, finite transition relation $\longrightarrow \subseteq X \times \mathcal{E} \times X$, and initial state $x \in X$. The universe of all Kripke-LTSs is denoted by $\mathcal{K}$.*

As usual, we will use the notation $x \xrightarrow{e} x'$ to denote that $(x, e, x') \in \longrightarrow$. The reflexive-transitive closure $\longrightarrow^*$ of a transition relation $\longrightarrow$ is defined in the following way: For all $x \in X$ it holds that $(x, x) \in \longrightarrow^*$ and if there exist $e \in \mathcal{E}$ and $y, x' \in X$ such that $x \xrightarrow{e} y$ and $y \longrightarrow^* x'$ then $(x, x') \in \longrightarrow^*$.

Two different behavioral preorders are applied in this paper. The first is the simulation preorder, which is reiterated in Definition 2. Simulation is used to signify inclusion of behavior, while synthesis may alter the transition structure due to, for instance, unfolding. Simulation as applied in this paper is a straightforward adaptation of the definition of simulation in [15].

Definition 2. *For $k' = (X', L', \longrightarrow', x')$ and $k = (X, L, \longrightarrow, x)$ we say that 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. *We have $L'(y') = L(y)$; and*
2. *If $y' \xrightarrow{e}{}' z'$ then there exists a step $y \xrightarrow{e} z$ such that $(z', z) \in R$.*

Partial bisimulation [4] is an extension of simulation such that the subset of uncontrollable events is bisimulated. For plant specification $k \in \mathcal{K}$ and synthesis result $s \in \mathcal{K}$ we require that s is related to k via a partial bisimulation. This signifies the fact that synthesis did not disallow any uncontrollable event, which implies controllability in the context of supervisory control. Research in [4] details the nature of this partial bisimulation preorder.

Definition 3. *If $k' = (X', L', \longrightarrow', x')$ and $k = (X, L, \longrightarrow, x)$, then k' and k are related via a partial bisimulation (notation: $k' \precapprox k$) 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. *We have $L'(y') = L(y)$;*
2. *If $y' \xrightarrow{e}{}' z'$ then there exists a step $y \xrightarrow{e} z$ such that $(z', z) \in R$; and*
3. *If $y \xrightarrow{e} z$ for $e \in \mathcal{U}$ then there exists a step $y' \xrightarrow{e}{}' z'$ such that $(z', z) \in R$.*

Requirements are specified using a modal logic $\mathcal{F}$ given in Definition 5, which is built upon the set of state-based formulas $\mathcal{B}$ in Definition 4.

Definition 4. *The set of state-based formulas $\mathcal{B}$ is defined by the grammar:*

$$\mathcal{B} ::= \mathit{true} \mid \mathit{false} \mid \mathcal{P} \mid \neg\mathcal{B} \mid \mathcal{B} \wedge \mathcal{B} \mid \mathcal{B} \vee \mathcal{B}$$

As indicated in Definition 4, state-based formulas are constructed from a straightforward Boolean algebra which includes the basic expressions *true* and *false*, as well as a state-based property test for $p \in \mathcal{P}$. Formulas in $\mathcal{B}$ are then combined using the standard Boolean operators $\neg$, $\wedge$ and $\vee$.

Definition 5. *The requirement specification logic $\mathcal{F}$ is defined by the grammar:*

$$\mathcal{F} ::= \mathcal{B} \mid \mathcal{F} \wedge \mathcal{F} \mid \mathcal{B} \vee \mathcal{F} \mid [\mathcal{E}]\,\mathcal{F} \mid \langle\mathcal{E}\rangle\mathcal{F} \mid \Box\,\mathcal{F} \mid \Diamond\,\mathcal{B} \mid \mathit{dlf}$$

We briefly consider the elements of the requirement logic $\mathcal{F}$. Basic expressions in Definition 4 function as the basic building blocks in the modal logic $\mathcal{F}$. Conjunction is included, having its usual semantics, while disjunctive formulas are restricted to those having a state-based formula in the left-hand disjunct. This restriction guarantees correct synthesis solutions, since it enables a local state-based test for retaining the appropriate transitions. The formula $[e]\,f$ can be used to test whether f holds after every e-step, while the formula $\langle e\rangle f$ is used to assess whether there exists an e-step after which f holds. These two operators thereby follow their standard semantics from Hennessy-Milner Logic [10]. An invariant formula $\Box\,f$ tests whether f holds in every reachable state, while a reachability expression $\Diamond\,b$ may be used to check whether there exists a path such that the state-based formula b holds at some state on this path. Note that the sub-formula b of a reachability expression $\Diamond\,b$ is restricted to a state-based formula $b \in \mathcal{B}$. This is used to acquire unique synthesis solutions, due to the fact that for unrestricted reachability expressions, only an indefinite unfolding coincides with a maximal solution. The deadlock-free test *dlf* tests whether there exists an outgoing step of a particular state. Combined with the invariant operator, the formula $\Box\,\mathit{dlf}$ can be used to specify that the entire synthesized system should be deadlock-free. Deadlock-freeness is not defined as a state-based expression here since it requires information about (the existence of) outgoing transitions, which may have been removed during synthesis. Validity of formulas in $\mathcal{B}$ and $\mathcal{F}$, with respect to a Kripke-LTS $k \in \mathcal{K}$, is as shown in Definition 6.

Definition 6. *For $k = (X, L, \longrightarrow, x) \in \mathcal{K}$ and $f \in \mathcal{F}$ we define if k* satisfies *f (notation: $k \models f$) as follows:*

$$\frac{}{k \models \mathit{true}} \qquad \frac{p \in L(x)}{(X, L, \longrightarrow, x) \models p} \qquad \frac{k \not\models b}{k \models \neg b} \qquad \frac{k \models f \quad k \models g}{k \models f \wedge g} \qquad \frac{k \models f}{k \models f \vee g} \qquad \frac{k \models g}{k \models f \vee g}$$

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

$$\frac{\forall\, x \longrightarrow^{*} x' \quad (X, L, \longrightarrow, x') \models f}{(X, L, \longrightarrow, x) \models \Box\,f} \qquad \frac{x \longrightarrow^{*} x' \quad (X, L, \longrightarrow, x') \models b}{(X, L, \longrightarrow, x) \models \Diamond\,b} \qquad \frac{x \xrightarrow{e} x'}{(X, L, \longrightarrow, x) \models \mathit{dlf}}$$

We may now formulate the synthesis problem in terms of the previous definitions in Definition 7. Research in this paper focuses on resolving this problem.

Definition 7. *Given $k \in \mathcal{K}$ and $f \in \mathcal{F}$, find $s \in \mathcal{K}$ in a finite method such that the following holds: 1) $s \vDash f$, 2) $s \preceq k$, 3) $s \precapprox k$, 4) For all $k' \preceq k$ and $k' \vDash f$ holds $k' \preceq s$; or determine that such an s does not exist.*

These four properties are interpreted in the context of supervisory control synthesis as follows. Property 1 (*validity*) states that the synthesis result satisfies the synthesized formula. Property 2 (*simulation*) asserts that the synthesis result is a restriction of the original behavior, while property 3 (*controllability*) ensures that no accessible uncontrollable behavior is disallowed during synthesis. Controllability is achieved if the synthesis result is related to the original plant-model via a partial bisimulation, which adds bisimulation of all uncontrollable events to the second property. Note that the third property implies the second property, as can be observed in Definitions 2 and 3. However, both these properties are of importance since the former is related to the synthesis result being a restriction of original behavior, while the latter signifies achievement of controllability. Property 4 (*maximality*) states that synthesis removes the least possible behavior, and thereby induces maximal permissiveness. That is, the behavior of every alternative synthesis option is included in the behavior of the synthesis result.

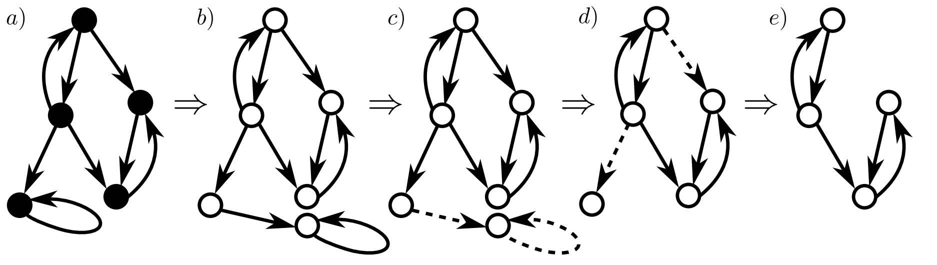


Fig. 2. Overview of the synthesis process. Steps in the original transition relation (Fig. 2a) of type $x \xrightarrow{e} x'$ are combined with reductions of the synthesized requirement (Fig. 2b), resulting in transitions of type $(x, f) \xrightarrow{e}_0 (x', f')$, and possibly inducing unfoldings. Transition are then removed (Fig. 2c-2d) based upon a local synthesizability test for formulas assigned to target states, until synthesizability holds in every reachable state (Fig. 2e).

4 Synthesis

The purpose of this section is to illustrate the formal definition of the synthesis construction. Synthesis as defined in this paper involves three major steps, after which a modified Kripke-LTS is obtained. If synthesis is successful, the resulting structure satisfies all synthesis requirements, as stated in Definition 7. The first stage of synthesis transforms the original transition relation $\longrightarrow \subseteq X \times \mathcal{E} \times X$, for state-space X, into a new transition relation $\longrightarrow_0 \subseteq (X \times \mathcal{F}) \times \mathcal{E} \times (X \times \mathcal{F})$ over the state-formula product space. This allows us to indicate precisely which modal (sub-)formula needs to hold at each point in the new transition relation. The second step removes transitions based upon an assertion of *synthesizability* of formulas assigned to the target states of transitions. This second step is repeated until no more transitions are removed. The third and final synthesis step tests whether synthesis has been successful by evaluating whether the *synthesizability* predicate holds for every remaining state. An overview of the synthesis process is shown in Fig. 2.

A formal derivation of the starting point in the synthesis process $\longrightarrow_0$ is shown in Definition 9. This definition relies upon sub-formulas under conjunction and invariant operators, as formalized in Definition 8.

Definition 8. *We say that $f \in \mathcal{F}$ is a sub-formula of $g \in \mathcal{F}$ (notation $f \in sub\,(g)$) if this can be derived by the following rules:*

$$\frac{}{f \in sub\,(f)} \qquad \frac{f \in sub\,(g)}{f \in sub\,(g \wedge h)} \qquad \frac{f \in sub\,(h)}{f \in sub\,(g \wedge h)} \qquad \frac{f \in sub\,(g)}{f \in sub\,(\Box\, g)}$$

Definition 9. *For state-space X and original transition relation $\longrightarrow \subseteq X \times \mathcal{E} \times X$, we define the starting point of synthesis $\longrightarrow_0 \subseteq (X \times \mathcal{F}) \times \mathcal{E} \times (X \times \mathcal{F})$ as follows:*

$$\frac{x \xrightarrow{e} x'}{(x, b) \xrightarrow{e}_0 (x, true)} \qquad \frac{(x, f) \xrightarrow{e}_0 (x', f') \quad (x, g) \xrightarrow{e}_0 (x', g') \quad g' \in sub\,(f')}{(x, f \wedge g) \xrightarrow{e}_0 (x', f')}$$

$$\frac{(x, f) \xrightarrow{e}_0 (x', f') \quad (x, g) \xrightarrow{e}_0 (x', g') \quad g' \notin sub\,(f')}{(x, f \wedge g) \xrightarrow{e}_0 (x', f' \wedge g')} \qquad \frac{x \xrightarrow{e} x' \quad x \vDash b}{(x, b \vee f) \xrightarrow{e}_0 (x', true)}$$

$$\frac{(x, f) \xrightarrow{e}_0 (x', f')}{(x, b \vee f) \xrightarrow{e}_0 (x', f')} \qquad \frac{x \xrightarrow{e} x'}{(x, \texttt{[}e\texttt{]}\, f) \xrightarrow{e}_0 (x', f)} \qquad \frac{x \xrightarrow{e} x' \quad e \neq e'}{(x, \texttt{[}e'\texttt{]}\, f) \xrightarrow{e}_0 (x', true)}$$

$$\frac{x \xrightarrow{e} x'}{(x, \texttt{<}e\texttt{>} f) \xrightarrow{e}_0 (x', f)} \qquad \frac{x \xrightarrow{e} x'}{(x, \texttt{<}e'\texttt{>} f) \xrightarrow{e}_0 (x', true)} \qquad \frac{(x, f) \xrightarrow{e}_0 (x', f') \quad f' \in sub\,(\Box\, f)}{(x, \Box\, f) \xrightarrow{e}_0 (x', \Box\, f)}$$

$$\frac{(x, f) \xrightarrow{e}_0 (x', f') \quad f' \notin sub\,(\Box\, f)}{(x, \Box\, f) \xrightarrow{e}_0 (x', \Box\, f \wedge f')} \qquad \frac{x \xrightarrow{e} x'}{(x, \Diamond\, b) \xrightarrow{e}_0 (x', true)}$$

$$\frac{x \xrightarrow{e} x'}{(x, \Diamond\, b) \xrightarrow{e}_0 (x', \Diamond\, b)} \qquad \frac{x \xrightarrow{e} x'}{(x, dlf) \xrightarrow{e}_0 (x', true)}$$

The intuitive interpretation of a derivation rule for $(x, f) \xrightarrow{e}_0 (x', f')$ in Definition 9 is an assignment of the formula f' to the state x', if f' is required for the validity of f in x, after event e. This is particularly recognizable in the derivation rules for $\texttt{[}e\texttt{]}\, f$. The derivation rules for conjunction ensure the validity of reductions for both operands. However, in order to achieve a terminating synthesis procedure, reductions of conjunctive formulas are prevented from expanding infinitely often using the sub-formula relation. This applies also to reductions of invariant formulas. The prevention of indefinite formula expansion under conjunction is essential, and only required for, finiteness of the formula-reductions for invariant expressions. Other reduction strategies typical for the synthesis approach in Definition 9 include a limitation of outgoing transitions based upon the state-based validity of left-hand disjuncts. In addition, reductions towards *true* are included for $\texttt{<}e\texttt{>}f$ and $\Diamond\, p$, in order to achieve maximal permissiveness, since the synthesis for a single witness does not affect other outgoing transitions, which should be left in place.

The starting point of synthesis $\longrightarrow_0$ is subjected to transition removal via a synthesizability test for formulas assigned to the target states of transitions. In generalized form, we define a formula $f \in \mathcal{F}$ to be synthesizable in the state-formula pair (x, g) if this can be derived by the rules in Definition 11. For an appropriate definition of synthesizability, it is necessary to extend the notion of sub-formulas in such a way that a state-based evaluation can be incorporated, in order to handle disjunctive formulas correctly. This leads to the sub-formula notion called *part*, which is shown in Definition 10.

Definition 10. *We say that a formula $f \in \mathcal{F}$ is a part of a formula $g \in \mathcal{F}$ in the context of a state based evaluation for $(X, L, \longrightarrow, x)$ if $f \equiv g$, or a derivation can be obtained by the following rules:*

$$\frac{f \in \mathit{part}\,(x, g)}{f \in \mathit{part}\,(x, g \wedge h)} \qquad \frac{f \in \mathit{part}\,(x, h)}{f \in \mathit{part}\,(x, g \wedge h)} \qquad \frac{x \not\models b \quad f \in \mathit{part}\,(x, g)}{f \in \mathit{part}\,(x, b \vee g)} \qquad \frac{f \in \mathit{part}\,(x, g)}{f \in \mathit{part}\,(x, \Box\, g)}$$

Partial formulas as shown in Definition 10 are used in the definition of synthesizability as shown in Definition 11. In particular, this is used in the definition of synthesizability for formulas of type $\texttt{<}e\texttt{>}f$. In addition, partial formulas play a major role in the correctness proofs of the synthesis method.

Definition 11. *With regard to an intermediate relation $\longrightarrow_n \subseteq (X \times \mathcal{F}) \times \mathcal{E} \times (X \times \mathcal{F})$ in the synthesis procedure, we say that a formula $f \in \mathcal{F}$ is* synthesizable *in the state-formula pair (x, g) (notation: $(x, g) \uparrow f$) if this can be derived as follows:*

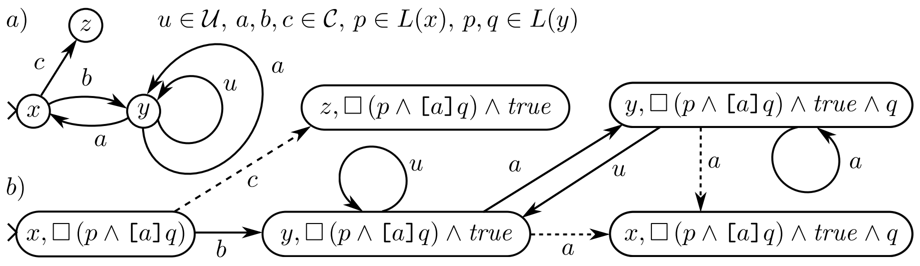
$$\frac{x \models b}{(x, g) \uparrow b} \qquad \frac{(x, g) \uparrow f_1 \quad (x, g) \uparrow f_2}{(x, g) \uparrow f_1 \wedge f_2} \qquad \frac{x \models b}{(x, g) \uparrow b \vee f} \qquad \frac{(x, g) \uparrow f}{(x, g) \uparrow b \vee f}$$

$$\frac{}{(x, g) \uparrow \texttt{[}e\texttt{]}\, f} \qquad \frac{(x', g') \uparrow f \quad (x, g) \xrightarrow{e}_n (x', g') \quad f \in \mathit{part}\,(x', g')}{(x, g) \uparrow \texttt{<}e\texttt{>}f}$$

$$\frac{(x, g) \uparrow f}{(x, g) \uparrow \Box\, f} \qquad \frac{(x, g) \longrightarrow_n^* (x', g') \quad x' \models b}{(x, g) \uparrow \Diamond\, b} \qquad \frac{(x, g) \xrightarrow{e}_n (x', g')}{(x, g) \uparrow \mathit{dlf}}$$

It is important to note here that the *synthesizability* test serves as a partial assessment. The synthesizability predicate for f holds in the state-formula pair

(x, g) if it is possible to modify outgoing transitions of (x, g) in such a way that f becomes satisfied in (x, g). However, synthesizability is not straightforwardly definable for a number of formulas. For instance, it can not be directly assessed whether it is possible to satisfy an invariant formula. Therefore, the synthesizability test in Definition 11 is designed to operate in conjunction with the process of repeated transition removal, as shown in Fig. 2. This is reflected, for instance, in the definition of synthesizability for an invariant formula $\Box f$, which only relies upon f being synthesizable. However, since synthesizability needs to hold at every reachable state for synthesis to be successful, such a definition of synthesizability for invariant formulas is appropriate due to its role in the entire synthesis process. A synthesis example for the invariant formula $\Box p \wedge [a] q$ is shown in Fig. 3.



a)
b)
$u \in \mathcal{U}$, $a, b, c \in \mathcal{C}$, $p \in L(x)$, $p, q \in L(y)$
x
y
z
a
b
c
u
$z, \Box(p \wedge [a] q) \wedge true$
$y, \Box(p \wedge [a] q) \wedge true \wedge q$
$x, \Box(p \wedge [a] q)$
$y, \Box(p \wedge [a] q) \wedge true$
$x, \Box(p \wedge [a] q) \wedge true \wedge q$

Fig. 3. Synthesis for the formula $\Box p \wedge [a] q$ upon the model in Fig. 3a, resulting in the restricted behavioral model shown in Fig. 3b. Note the unfolding for $[a] q$, the restricted formula-expansion for invariant formulas, and transition disabling, indicated by dashed lines, due to the state-based formula q not being synthesizable in x, and p not being synthesizable in z.

Using the definitions stated before, we are now ready to define the main synthesis construction. That is, how transitions are removed from the synthesis starting point $\longrightarrow_0$, and how are the subsequent intermediate transition relations $\longrightarrow_1, \longrightarrow_2, \ldots$ constructed. In addition, more clarity is required with regard to reaching a stable point during synthesis, and verifying whether the synthesis construction has been completed successfully.

Definition 12. *For $k = (X, L, \longrightarrow, x) \in \mathcal{K}$ and $f \in \mathcal{F}$, we define the n-th iteration in the synthesis construction as follows:*

$$\frac{(x, f) \xrightarrow{e}_n (x', f') \quad e \in \mathcal{U}}{(x, f) \xrightarrow{e}_{n+1} (x', f')} \qquad \frac{(x, f) \xrightarrow{e}_n (x', f') \quad (x, f) \uparrow f}{(x, f) \xrightarrow{e}_{n+1} (x', f')}$$

The corresponding system model $S^n_{k,f}$ is defined as stated below, using the labeling function L_{proj}, such that $L_{proj}(y, g) = L(y)$, for all $y \in X$ and $g \in \mathcal{F}$.

$$S^n_{k,f} = (X \times \mathcal{F}, L_{proj}, \longrightarrow_n, (x, f))$$

One last definition remains, namely *completeness* of the synthesis construction. The formula reductions induced by Definition 9 are finite, which implies a terminating construction of the transition relation $\longrightarrow_0$. Since $\longrightarrow_0$ consists of finitely many transitions, only finitely many steps may be removed. This means that at some point, no more transitions are removed, and a stable point will be reached. If at this point, synthesizability holds at every reachable state, synthesis is successful. Otherwise, it is not. It is natural that a formal notion representing the first situation serves as a premise for a number of correctness results. This notion is formalized as *completeness* in Definition 13.

Definition 13. *For $k = (X, L, \longrightarrow, x) \in \mathcal{K}$, $f \in \mathcal{F}$ and $n \in \mathbb{N}$, we say that $S^n_{k,f}$ is* complete *if the following holds:*

$$\text{For all } (x, f) \longrightarrow^*_n (x', f') \text{ it holds that } (x', f') \uparrow f'.$$

5 Correctness

We show that synthesis as defined in this paper results in a controlled system adhering to the conditions in Definition 7. Detailed proofs are available in [19], while computer-verified proofs are available as well [16]. The synthesis method is finite (Theorem 1), and the result satisfies the synthesized requirement (Theorem 2). In addition, we show that the synthesis result is related to the original model via partial bisimulation (Theorem 3), which implies simulation. As a final result, we prove maximal permissiveness (Theorem 4). Assessing whether synthesis has been successful is done by checking whether synthesizability holds at every reachable state in the fixed point obtained as a result of Theorem 1.

Theorem 1. *For $k = (X, L, \longrightarrow, x) \in \mathcal{K}$, having finite $\longrightarrow$, and $f \in \mathcal{F}$, there exists an $n \in \mathbb{N}$ such that $S^n_{k,f} = S^m_{k,f}$ for all $m > n$.*

Theorem 2. *If $S^n_{k,f}$ is complete then $S^n_{k,f} \models f$.*

Theorem 3. *If $S^n_{k,f}$ is complete then $S^n_{k,f} \precapprox k$.*

Theorem 4. *If $k' \preceq k$ and $k' \models f$ then $k' \preceq S^n_{k,f}$.*

6 Conclusions

This paper presents a novel approach to controlled system synthesis for modal logic on non-deterministic plant models. The behavior of a Kripke-structure with labeled transitions is adapted such that it satisfies the synthesized requirement. The relationship between the synthesis result and the original plant specification adheres to important notions from Ramadge-Wonham supervisory control: controllability and maximal permissiveness. The requirement specification logic also allows expressibility of deadlock-freeness and marker-state reachability. The synthesis approach, via a reduction on modal expressions combined with an iteratively applied synthesizability test for formulas assigned to target states of transitions results in an effective synthesis procedure. Our next research efforts will focus on determining the effectiveness of this procedure as well as its applicability in case studies.

References

1. B. Aminof, F. Mogavero, and A. Murano. Synthesis of hierarchical systems. In *Formal Aspects of Component Software*, pages 42–60. Springer, 2011.
2. A. Arnold, I. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 1(303):7–34, 2003.
3. A. Arnold and I. Walukiewicz. Nondeterministic controllers of nondeterministic processes. In *Logic and Automata*, pages 29–52. Amsterdam University Press, 2008.
4. J. Baeten, B. van Beek, A. van Hulst, and J. Markovski. A process algebra for supervisory coordination. In *Process Algebra and Coordination*, pages 36–55. EPTCS, 2011.
5. R. Bull and K. Segerberg. Basic modal logic. In *Handbook of Philosophical Logic*, pages 1–88. Springer, 1994.
6. C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 1999.
7. R. Cleaveland and B. Steffen. A linear-time model checking algorithm for the alternation-free modal mu-calculus. *Formal Methods in System Design*, 2:121–147, 1993.
8. N. D'Ippolito, V. Braberman, N. Piterman, and S. Uchitel. Synthesis of live behaviour models. In *Foundations of Software Engineering*, pages 77–86. ACM, 2010.
9. N. D'Ippolito, V. Braberman, N. Piterman, and S. Uchitel. Synthesizing nonanomalous event-based controllers for liveness goals. *ACM Transactions on Software Engineering Methodology*, 22(1):1–36, 2013.
10. M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.
11. Y. Lustig and M. Vardi. Synthesis from recursive-components libraries. In *Games, Automata, Logics and Formal Verification*, pages 1–16. EPTCS, 2011.
12. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Principles of Programming Languages*, pages 179–190. ACM, 1989.
13. P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230, 1987.
14. O. Sokolsky and S. Smolka. Incremental model checking in the modal mu-calculus. In *Computer Aided Verification*, pages 351–363. Springer, 1994.
15. R. van Glabbeek. The linear time-branching time spectrum II. In *Conference on Concurrency Theory*, pages 66–81. Springer, 1993.
16. A. van Hulst. Coq v8.3 proofs:. `http://seweb.se.wtb.tue.nl/~ahulst/sofsem/`, August 2014.
17. A. van Hulst, M. Reniers, and W. Fokkink. Maximal synthesis for Hennessy-Milner logic. In *Application of Concurrency to System Design*, pages 1–10. IEEE, 2013.
18. A. van Hulst, M. Reniers, and W. Fokkink. Maximal synthesis for Hennessy-Milner logic with the box-modality. In *Workshop on Discrete Event Systems*, pages 278–285. IEEE, 2014.
19. A. van Hulst, M. Reniers, and W. Fokkink. Maximally permissive controlled system synthesis for modal logic. Preprint at `http://arxiv.org/abs/1408.3317/`, August 2014.
20. R. Ziller and K. Schneider. Combining supervisory synthesis and model checking. *ACM Transactions on Embedded Computing Systems*, 4(2):331–362, 2005.