Discrete Event Dynamic Systems manuscript No.
(will be inserted by the editor)

Maximally Permissive Controlled System Synthesis for Non-Determinism and Modal Logic

A. van Hulst · M. Reniers · W. Fokkink

the date of receipt and acceptance should be inserted later

Abstract We propose a new technique for controlled system synthesis on non-deterministic automata for requirements in modal logic. Synthesis, as defined in this paper, restricts a behavioral specification of the uncontrolled system such that it satisfies a given logical expression, while adhering to the rules dictated by supervisory control such as maximal permissiveness and controllability. The applied requirement formalism extends Hennessy-Milner logic with the invariant and reachability modalities from Gödel-Löb logic, and is therefore able to express a broad range of control requirements, such as marker state reachability and deadlock-freeness. This paper contributes to the field of control synthesis by achieving maximal permissiveness in a non-deterministic context for control requirements in modal logic, and treatment of controllability via partial bisimulation. We present a well-defined and complete derivation of the synthesis result, which is supported further by computer-verified proofs created using the Coq proof assistant. The synthesis method is also presented in algorithmic form, including an analysis of its computational complexity. We show that the proposed synthesis theory allows full expressibility of Ramadge-Wonham supervisory control theory and we illustrate its applicability in two small industrial case studies, including an analysis with regard to scalability.

Keywords controlled system synthesis · modal logic · non-determinism · maximal permissiveness · controllability · partial bisimulation

A.C. van Hulst · M.A. Reniers
Eindhoven University of Technology
E-mail: ahulst@tue.nl, m.a.reniers@tue.nl

W.J. Fokkink
VU University Amsterdam
E-mail: w.j.fokkink@vu.nl

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*. This resulting model represents the controlled behavior of the plant, and is therefore referred to as the *controlled system*. The automated generation, or *synthesis*, of such a restricted behavioral model incorporates a number of concepts from supervisory control theory (Ramadge and Wonham (1987)), which guarantees that the generated model is a proper controlled system with regard to the original plant specification. This includes a strict partitioning of behaviors into controllable and uncontrollable events, such that synthesis does not disable accessible uncontrollable events, thereby achieving controllability. In addition, synthesis preserves all behavior which does not invalidate the requirements, thereby inducing maximal permissiveness. The synthesis theory put forward in this paper further allows the expression of marker state reachability and deadlock-freeness, which are often employed in supervisory control (Cassandras and Lafortune (2008)).

Starting point of the synthesis construction is a non-deterministic Kripke-structure with labeled transitions, representing the uncontrolled plant model. Basic properties may be assigned to states to capture state-based information, while event-labeled transitions capture system dynamics. The required controlled behavior is expressed using modal logic with invariant and reachability operators.

A new transition relation is henceforth derived, by observing how the validity of modal expressions in consecutive states relates to event labels. From this new behavioral relation, transitions are removed if formulas assigned to target states do not satisfy a partial satisfiability test, until a stable point is reached. Most of the theoretical work in this paper involves a precise formulation of the mathematical structures involved, as well as proofs to show termination and well-definedness of the applied construction. The required controlled behavior is enforced by disallowing certain events, which coincides with the standard approach in supervisory control theory (Ramadge and Wonham (1987)), since system control should not involve a fundamental adaptation of system properties; only existing behavior may be disallowed.

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

Regarding the first contribution, it should be noted that supervisory control is often approached in a language-based setting using a deterministic model of both plant and controller. Classic Ramadge-Wonham supervisory control theory is a well-researched example of this setup (Ramadge and Wonham (1987)).

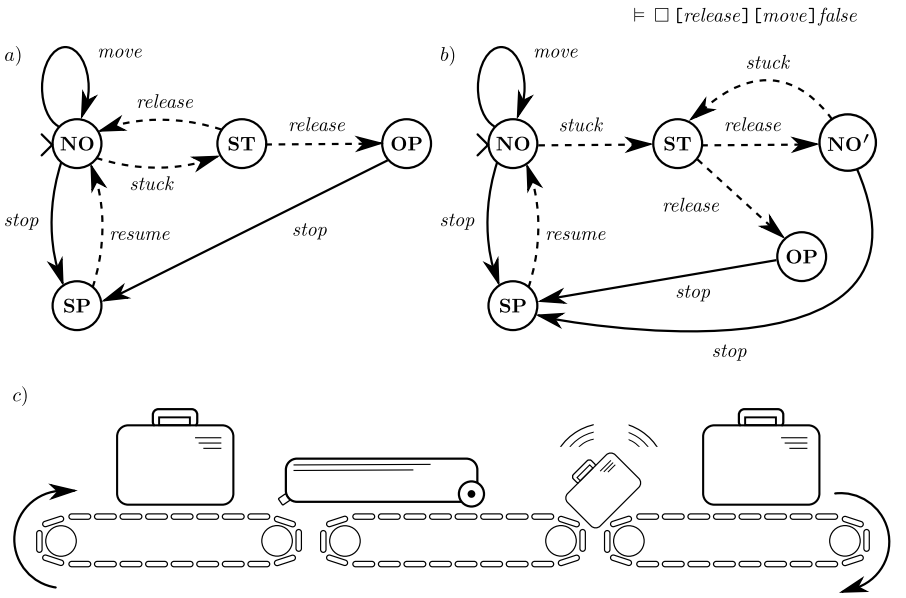

Fig. 1 Control synthesis in a non-deterministic setting. A luggage conveyor belt, depicted in Fig. 1c is modeled by the state diagram in Fig. 1a. Controlled operation such that a *release* event is not directly followed by a *move* event is shown in Fig. 1b. This controlled system model satisfies the modal expression in the upper right corner of the illustration. Note that the model in Fig. 1b incorporates a new state. Uncontrollable events are indicated using dashed lines.

The resulting controller restricts the behavior of the deterministic plant model, thereby ensuring that it operates according to the requirements. The demand for control synthesis on non-deterministic models of behavior is clearly present, as signified by a number of works on this topic (see for example Fabian and Lennartson (1997) and Kumar and Shayman (1994)). Non-determinism allows for a higher level of abstraction in modeling discrete event systems, due to identification of similar events (Cassandras and Lafortune (2008)).

In this paper, we consider non-determinism in conjunction with maximal permissiveness; a property which states that all non-invalidating behavior should be preserved, while achieving the control objective. This ensures that the controlled behavior stays as close as possible to the intended operation of the system. Since maximal permissiveness is key in achieving proper controlled behavior, we intend to improve upon previous efforts by defining controlled system synthesis in a maximal way for non-deterministic models. Since maximal permissiveness is limited or completely omitted in earlier work on control synthesis for non-deterministic models, we intend to bridge this gap towards full coverage of control synthesis in a non-deterministic setting.

We further illustrate controlled system synthesis on a non-deterministic model by the example in Fig. 1. It consists of a system of conveyor belts for luggage handling at an airport, and is loosely based on research done at

Vanderlande Industries (Kamphuis (2013); Jansen (2014)). The state diagram shown in Fig. 1a models the uncontrolled operation of this system. If the system is in normal operation (state **NO**), it repeatedly executes a *move* event. However, as depicted in Fig. 1c, a small suitcase might get stuck, halting the system (state **ST**). If the suitcase causing the obstruction is pulled loose by one of the travelers (event *release*), the conveyor belt resumes normal operation. Also, one of the operators may release the suitcase (state **OP**), *stop* the conveyor belt to make sure that everything is OK, and then *resume* its normal operation. Note that the *release* event from the state **ST** may be caused by two different situations. First, the traveler who owns the suitcase may free it from its undesirable position, and subsequently leave the airport. Second, a different traveler, who does not own the suitcase, may pull it loose and — in good faith — put it back on the conveyor belt. Since in the second situation, the suitcase still poses a threat to the desired operation of the system, we wish to control the behavior of this system in such a way that a *release* event can not be followed immediately by a *move* event, thereby forcing the system to go through the **SP** state. This required behavior is formalized by the modal expression $\Box\,[\,\mathit{release}\,]\,[\,\mathit{move}\,]\,\mathit{false}$; intuitively described as: invariantly, after every *release*, a *move* event should not be allowed.

Fig. 1b models the controlled operation of this system, and therefore satisfies $\Box\,[\,\mathit{release}\,]\,[\,\mathit{move}\,]\,\mathit{false}$, while only behavior that invalidates this property has been disallowed. Note that the adapted behavioral model incorporates a new state $\mathbf{NO}'$, modeling the new behavior of the **NO** state, after a *release* event has happened. One of the main theoretical contributions of this paper is a mathematically sound way to derive such new states.

Regarding the second contribution of this paper, a quick glance at the requirement formalism applied in this paper is provided in Fig. 1, where we used an invariant expression to disallow occurrence of *move* events after a *release*. We define the synthesis theory for a carefully chosen subset of Hennessy-Milner logic (Hennessy and Milner (1985)), and Gödel-Löb logic (Alberucci and Facchini (2009)). The choice to apply a restricted formalism for control requirements is justified by the synthesis objectives of solution uniqueness, maximal permissiveness and controllability. For instance, the μ-calculus, which is often applied in verification tasks such as symbolic model checking (McMillan (1993)), is too strong for obtaining unique and maximally permissive results for control synthesis problems. For example, a state-model consisting of a single a-loop has no single finite-state adaptation such that the expression $\mu X.\,[\,a\,]\,X$ (i.e. every a-path is finite) becomes satisfied.

Instead, we combined the invariant ($\Box\, f$) and reachable ($\Diamond\, f$) modalities from Gödel-Löb logic (Alberucci and Facchini (2009)) with the universal ($[\,e\,]\,f$) and existential ($<e>f$) lookahead from Hennessy-Milner logic (Hennessy and Milner (1985)). We restrict this requirement formalism to state-based properties for the reachability operator, and we apply the same restriction to one side of a disjunction. Fig. 5 details an example which shows that such a restriction is required to obtain unique synthesis results. This emphasis on uniqueness is a de facto standard in control theory (Ramadge and Wonham

(1987)). Earlier research shows that the restriction on disjunctive formulas and uniqueness of results are orthogonal for pure Hennessy-Milner logic (van Hulst et al (2014)). That is, all maximally permissive control synthesis results, for unrestricted disjunctions, can be obtained once the requirement for uniqueness is lifted. Two other important notions prevalent in control synthesis are marker state reachability and deadlock-freeness. To achieve flexibility in this regard, we extended the requirement formalism such that both these properties may be enforced in the synthesis result, if desired.

We provide short examples of requirements specific to the type of synthesis defined in this paper. Safety-related properties, which model the absence of faulty behavior, include deadlock-avoidance expressed as $\Box\, dlf$ (i.e., invariantly, deadlock-free). Furthermore, the logic may express safety-requirements of a more general nature, such as $\Box\, [\, send\,] \Diamond\, received$, for some type of communicating system. In addition, we may use the requirement formalism to express a limited class of fairness properties. Such as $\Box\, (busy \vee <lock> access)$, to describe the use of a particular resource in a distributed environment. The requirement formalism extends to liveness expressions such as $\Box \Diamond\, finished$, to indicate that a path to a state marked as *finished* should always exist.

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 basic formal notions up to a point where we can define the synthesis problem in a formal way. Section 4 details the synthesis method proposed in this paper, including a number of examples. Correctness theorems are subsequently developed in Section 5. Section 6 concerns the effective computation of synthesis solutions, including an algorithm and an analysis of its computational complexity. Section 7 compares the synthesis techniques in this paper to Ramadge-Wonham supervisory control, by detailing how we can express a Ramadge-Wonham controller synthesis problem using the proposed theories. A small industrial case study is provided in Section 8 and a case-based analysis of the scalability of the synthesis algorithm can be found in Section 9. Formal definitions and proofs for most of the theoretical work in this paper are presented in computer-verified form by means of Coq proofs.

2 Related Work

This paper improves upon a previous conference publication (van Hulst et al (2015)) by expanding the intuitive explanation of the synthesis setup, providing all proofs in detail, an analysis of the synthesis method in algorithmic form, a detailed comparison to Ramadge-Wonham supervisory control, and the inclusion of case studies. In addition, after a careful study we fixed an apparent too restrictive definition of the *synthesizability* predicate in van Hulst et al (2015) which relates to properly achieving controllability.

Earlier work by the same authors concerning synthesis for modal logic includes a synthesis method for Hennessy-Milner logic (van Hulst et al (2014)). The applied synthesis technique in this paper is different in an important

aspect. Due to a finite unfolding in van Hulst et al (2014) of part of the uncontrolled system into a tree-like structure, the recursive method applied in van Hulst et al (2014) is not compatible with invariant expressions. This omission lead to the derivation of the new methodology as presented in this paper.

We analyze other related work along the lines of the three intended improvements in this paper: 1) allowance of non-determinism in plant specifications, 2) expressiveness of the requirement specification formalism, and 3) adherence to maximal permissiveness.

Ramadge-Wonham supervisory control theory (Ramadge and Wonham (1987)) defines a broadly embraced methodology for controller synthesis on deterministic plant models. It identifies a number of key characteristics in the relationship between plant and controlled system, such as controllability, marker state reachability, deadlock-freeness and maximal permissiveness, which are inherited by the synthesis theory in this paper. The limitation to deterministic plant specifications in Ramadge and Wonham (1987) allows the derivation of a strictly separated unique and maximally permissive controller, but does not embrace the increased abstraction and flexibility offered by a non-deterministic plant model.

Control synthesis for non-deterministic plant models and temporal specifications is considered by Pnueli and Rosner (1989) and extended further by Arnold et al (2003), but omits maximal permissiveness as a criterion for control synthesis. Follow-up research by Arnold and Walukiewicz (2008) considers non-deterministic controllers, but limits the specification of desired behavior to alternating automata. Work by Kumar and Shayman (1994) investigates non-deterministic controllers for non-blockingness. This approach is further extended by Jiang and Kumar (2006) for CTL* control specifications, but it does not deliver maximally permissive solutions.

Similar research to the work presented in this paper is carried out in Pinchinat (2005) for control requirements in μ-calculus. The methodology presented in Pinchinat (2005) extracts a maximally permissive controller with regard to the simulation preorder. In addition, synthesis is defined to be compositional and an approach which uses iterative synthesis steps based on state-labeling is studied. However, the work in Pinchinat (2005) is limited to deterministic systems. Follow-up work in Pinchinat and Raclet (2005) does indeed incorporate non-deterministic plant specifications but a preceeding step in the synthesis process in Pinchinat and Raclet (2005) reduces the aforementioned plant model to a deterministic model. However, this step does not preserve a structural relationship between the synthesis result and the plant model, while our approach does.

Work by Kupferman and Vardi (2000) investigates the adaptation of a behavioral specification such that a μ-calculus expression becomes satisfied. This research is extended within the context of control synthesis by Kupferman et al (2000) using a tree-automata based approach which is not maximally permissive. We find the same omission of maximal permissiveness in Moor and Davoren (2001), where safety and liveness properties in μ-calculus are synthesized

for hybrid automata, in Wolff et al (2013) where LTL-requirements are synthesized for non-deterministic plant models, and in Ostroff (1989), where safety properties in real-time temporal logic are synthesized for non-deterministic plants.

A game theoretic approach to the synthesis of liveness goals stated in fluent temporal logic is the subject of several works (D'Ippolito et al (2010, 2013)). However, the pruning-based synthesis approach in D'Ippolito et al (2010, 2013) is inadequate for control of non-deterministic models, and only allows for synthesis under a weaker maximality criterion, referred to as a best effort controller.

A detailed exposure of control synthesis for deterministic automata and CTL* specifications is given in Ehlers et al (2014). However, the work by Ehlers et al (2014) asserts that in general maximally permissive controllers do not exist, while this paper defines such solutions for a reasonably expressive set of control requirements. Control synthesis via quantified atomic properties in the μ-calculus is applied by Pinchinat and Riedweg (2003), but this treatment is limited to deterministic automata. Quotient-based control synthesis in Basu and Kumar (2007) is based on the propositional μ-calculus, but applies a tableau-based synthesis method which does not result in a maximally permissive controlled system.

A comparison to work done in UPPAAL Tiga reveals interesting aspects regarding control synthesis from a mainly practical perspective. For instance, the work in Jessen et al (2007) considers synthesizing a controller for a climate control system, based on the logical specification of both existing behavior (in the form of guards) and required behavior (in the form of control requirements for safety and liveness). However, since the control-strategies derived in Jessen et al (2007) are defined as functions which deterministically choose either a specific control action or a delay, non-deterministic handling of control is not incorporated in Jessen et al (2007). Another example for control synthesis using UPPAAL Tiga is detailed in Havelund et al (1999), concerning the automated derivation of controllers for power modules in audio and video systems. One particular aspect of the work in Havelund et al (1999), which also applies to UPPAAL-based control synthesis in a general comparison with the work in this paper, is the impossibility to nest the invariant operator in logical expressions.

The interplay between the validity of modal expressions and transition removal in a control synthesis context is studied by Ziller and Schneider (2005), and the incremental effects of transition removal upon the validity of μ-calculus formulas is considered by Sokolsky and Smolka (1994) and Cleaveland and Steffen (1993). Note that the latter two works study this problem from a model-checking perspective. The limitations regarding control synthesis for disjunctive expressions, as applied in this paper, have been observed earlier (Antoniotti and Mishra (1995)).

Work by Fabian and Lennartson (1997) first identified a simulation-type refinement relation to be used in control synthesis for non-deterministic automata. Also, research done by Fainekos et al (2007) for hybrid control syn-

thesis and temporal logic applies a simulation-type relation between plant and controlled system. Constraint-based synthesis for safety properties in Pralet et al (2010) also applies a simulation-type relation for this purpose. Partial bisimulation as a means to express controllability is first described by Rutten (2000), and a subtly different variant is applied by Su (2008). Various ramifications surrounding partial bisimulation and controllability are studied by Markovski (2011) from a process-theoretic perspective.

3 Definitions

We assume a set $\mathcal{E}$ of events and a set $\mathcal{P}$ of state-assignable basic properties. In addition, we assume a 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. Events are used to capture system dynamics, and represent actions occurring when the system switches 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 or a user input. Basic properties and events are used to model plant behavior in the form of a Kripke-structure (Bull and Segerberg (2001)) with labeled transitions, to be abbreviated as *Kripke-LTS*, as formalized in Definition 1. It is essential for the well-definedness of the synthesis construction in this paper that the transition relation in this Kripke-LTS is finite. This does not exclude loops or other kinds of infinitary behavior; we solely assume finiteness of the transition relation as far as its definition as a set of triples is concerned.

Definition 1 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 $x \in X$, we define a Kripke-LTS as the four-tuple $(X, L, \longrightarrow, x)$. The set of all Kripke-LTSes 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 $\xrightarrow{s}{}^*$, for $s \in \mathcal{E}^*$, over transition relation $\longrightarrow$, is defined in the following way: For all $x \in X$ it holds that $(x, x) \in \xrightarrow{1}{}^*$, where 1 denotes the empty string; and if there exist $e \in \mathcal{E}$, $s \in \mathcal{E}^*$ and $y, x' \in X$ such that $x \xrightarrow{e} y$ and $y \xrightarrow{s}{}^* x'$, then $x \xrightarrow{es}{}^* x'$. In most cases we will use an abstraction of this reflexive-transitive closure, without reference to a particular $s \in \mathcal{E}^*$. That is, $x \longrightarrow^* x'$ if and only if there exists an $s \in \mathcal{E}^*$ such that $x \xrightarrow{s}{}^* x'$.

Partial bisimulation (Rutten (2000)) is an adaptation of bisimulation such that controllable events are simulated, while uncontrollable events are bisimulated. For plant specification $k \in \mathcal{K}$ and synthesis result $s \in \mathcal{K}$ we require that s is related to k via 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 Markovski (2011) details the nature of this partial bisimulation relation. If all events are controllable, then

partial bisimulation coincides with strong simulation. On the other hand, if all events are uncontrollable, partial bisimulation coincides with strong bisimulation (van Glabbeek (1993)).

Definition 2 For $k' = (X', L', \longrightarrow', x') \in \mathcal{K}$ and $k = (X, L, \longrightarrow, x) \in \mathcal{K}$ we say that k' and k are related via partial bisimulation (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. $L'(y') = L(y)$; and
2. if $y' \xrightarrow{e}' z'$, for $e \in \mathcal{E}$ and $z' \in X'$, then there exists a $z \in X$ such that $y \xrightarrow{e} z$ and $(z', z) \in R$; and
3. if $y \xrightarrow{e} z$, for $e \in \mathcal{U}$ and $z \in X$, then there exists a $z' \in X'$ such that $y' \xrightarrow{e}' z'$ and $(z', z) \in R$.

If the relation $R \subseteq X' \times X$ is of particular importance we will use the notation $k' \preceq_R k$ to indicate that k' and k are related via partial bisimulation as witnessed by R.

Note that both partial bisimulation as formalized in Definition 2 (Rutten (2000)) as well as a variant which omits the requirement $(z', z) \in R$ in the 3th clause in Definition 2 (Su (2008)) have been described in the literature. An explicit choice is made in this paper to apply partial bisimulation as introduced in Rutten (2000) due to the fact that it establishes a stronger control relation beyond uncontrollable events.

Furthermore, coinductive expressions for behavioral equivalence are ofted considered in conjunction with the preservation of logical properties. For instance, strong bisimulation preserves μ-calculus expressions (van Glabbeek (1993)). Partial bisimulation is *not* applied in this way in this paper. In particular, partial bisimulation does not preserve the logic for control requirements defined below, and is only to be understood as a formulation for the relationship between plant and controlled system.

Requirements are specified using a modal logic $\mathcal{F}$ given in Definition 4, which is built upon the set of state-based formulas $\mathcal{B}$, in Definition 3.

Definition 3 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 3, 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 4 The requirement specification logic $\mathcal{F}$ is defined by the grammar:

$$\mathcal{F} ::= \mathcal{B} \mid \mathcal{F} \wedge \mathcal{F} \mid \mathcal{B} \vee \mathcal{F} \mid [\mathcal{E}]\mathcal{F} \mid <\mathcal{C}>\mathcal{F} \mid \Box\mathcal{F} \mid \Diamond\mathcal{B} \mid <\mathcal{E}> \mid \mathit{dlf}$$

We briefly consider the elements of the requirement logic $\mathcal{F}$. Basic expressions $\mathcal{B}$ function as the building blocks in the modal logic $\mathcal{F}$. Conjunction is included in unrestricted form, while disjunctive formulas are restricted to those having a state-based formula from $\mathcal{B}$ in the left-hand disjunct. This restriction guarantees unique synthesis solutions, since it enables a local state-based test for retaining the appropriate transitions, as illustrated in Fig. 5. The formula $[e]f$ can be used to test whether f holds after every e-step, while the formula $<e>f$ 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 (Hennessy and Milner (1985)). The restriction for the operator $<e>$ to be limited to a controllable event $e \in \mathcal{C}$ relates to the specific synthesis for a formula $<e>f$ and is detailed in Fig. 6. 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 argument of a reachability expression is restricted to a state-based formula $b \in \mathcal{B}$. This is due to the fact that a reachability formula may be used to express a formula of type $<e>f$ with $e \in \mathcal{U}$, which leads to a problem concerning controllability, as illustrated in Fig. 6. The two operators $\Box$ and $\Diamond$ are borrowed from Gödel-Löb logic (Alberucci and Facchini (2009)), and follow the same semantics. As an addition to the formulas $<e>f$, we provide a universal existence test $<e>$, which only tests whether an e-step exists. The argument e for the operator $<e>$ may be an unrestricted event $e \in \mathcal{E}$. The deadlock-freeness expression dlf tests whether there exists an outgoing step of the current state. Combined with the invariant operator, the formula $\Box\,\mathit{dlf}$ may be used to include absence of deadlock in the enforced controlled behavior. Deadlock-freeness is not defined as a state-based expression in $\mathcal{B}$ since it requires information about the existence of outgoing transitions, which may have been removed during synthesis. These notions of validity are formalized in Definition 5.

Definition 5 Validity of formulas in $\mathcal{B}$ with respect to $\mathcal{K}$ (notation: $k \Vdash b$) is defined as follows. Assume that $k = (X, L, \longrightarrow, x) \in \mathcal{K}$, $p \in \mathcal{P}$ and $b, c \in \mathcal{B}$ in the following derivation rules:

$$\frac{}{k \Vdash \mathit{true}} \qquad \frac{p \in L(x)}{k \Vdash p} \qquad \frac{k \nVdash b}{k \Vdash \neg b} \qquad \frac{k \Vdash b \quad k \Vdash c}{k \Vdash b \wedge c} \qquad \frac{k \Vdash b}{k \Vdash b \vee c} \qquad \frac{k \Vdash c}{k \Vdash b \vee c}$$

Validity of formulas in $\mathcal{F}$ with respect to $\mathcal{K}$ (notation: $k \vDash f$) is defined in the following way. Assume that $k = (X, L, \longrightarrow, x) \in \mathcal{K}$, $b \in \mathcal{B}$, $e \in \mathcal{E}$, $x' \in X$ and $f, g \in \mathcal{F}$ in the following derivation rules:

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

$$\frac{\forall\, x \xrightarrow{e} x' \quad (X, L, \longrightarrow, x') \vDash f}{(X, L, \longrightarrow, x) \vDash [e]\, f} \qquad \frac{x \xrightarrow{e} x' \quad (X, L, \longrightarrow, x') \vDash f}{(X, L, \longrightarrow, x) \vDash <e> f}$$

$$\frac{\forall\, x \longrightarrow^* x' \quad (X, L, \longrightarrow, x') \vDash f}{(X, L, \longrightarrow, x) \vDash \Box\, f} \qquad \frac{x \longrightarrow^* x' \quad (X, L, \longrightarrow, x') \vDash b}{(X, L, \longrightarrow, x) \vDash \Diamond\, b}$$

$$\frac{x \xrightarrow{e} x'}{(X, L, \longrightarrow, x) \vDash <e>} \qquad \frac{x \xrightarrow{e} x'}{(X, L, \longrightarrow, x) \vDash dlf}$$

We may now concisely formulate the synthesis problem in terms of the previous definitions. This is the key problem which will be resolved in this paper.

Definition 6 Given $k \in \mathcal{K}$ and $f \in \mathcal{F}$, find $s \in \mathcal{K}$ such that the following properties hold: 1) $s \vDash f$, 2) $s \preceq k$, 3) For all $k' \preceq k$ and $k' \vDash f$ it holds that $k' \preceq s$; or determine that such an s does not exist.

The three properties in Definition 6 are interpreted in the context of supervisory control synthesis as follows. Property 1 (*validity*) states that the synthesis result satisfies the control requirements. Property 2 (*controllability*) ensures that no accessible uncontrollable behavior is disallowed during synthesis. Property 3 (*maximality*) states that synthesis removes the least possible behavior, and thereby induces maximal permissiveness. That is, the behavior of every alternative synthesis option (with regard to validity) is included in the behavior of the synthesis result. For clarity, please note that the term *behavior* is used here and in the rest of this paper to refer to the actual $\mathcal{K}$-model itself, and not to its language of event-sequences.

4 Synthesis

Given $k = (X, L, \longrightarrow, x) \in \mathcal{K}$ as plant specification, and control requirement $f \in \mathcal{F}$, we construct a new transition relation $\longrightarrow \subseteq (X \times \mathcal{F}) \times \mathcal{E} \times (X \times \mathcal{F})$ over the state-formula product space. This allows us to relate original states to expressions which need to be satisfied at these states, thereby employing a separate reduction process on the modal expressions in $\mathcal{F}$. The newly created transition relation is henceforth subjected to repeated transition removal, based upon a partial satisfiability test of formulas assigned to target states, as illustrated abstractly in Fig. 2. In order to further substantiate our claims, we first consider a number of examples. Particular emphasis is placed on the applied reductions of modal expressions, which are henceforth formally stated in Definition 8.

The model in Fig. 3a is adapted such that the expression $[a]p$ becomes satisfied, resulting in the model shown in Fig. 3c. The initial state space, upon

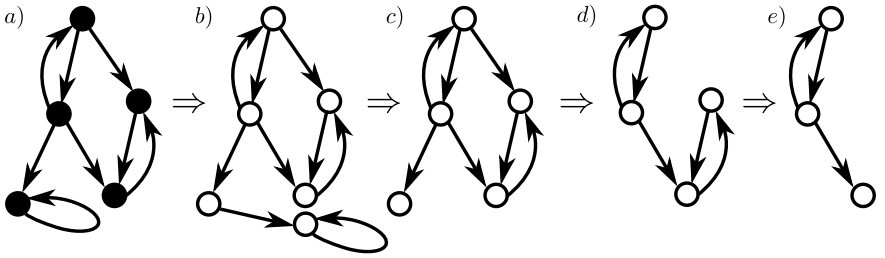

Fig. 2 Abstraction of the synthesis process. The plant specification transition relation in Fig. 2a is augmented with reductions of the control requirement in Fig. 2b, which may induce an embedded unfolding. This new transition relation is then subjected to repeated transition removal in steps Fig. 2c-2d, until a stable point has been reached in Fig. 2e.

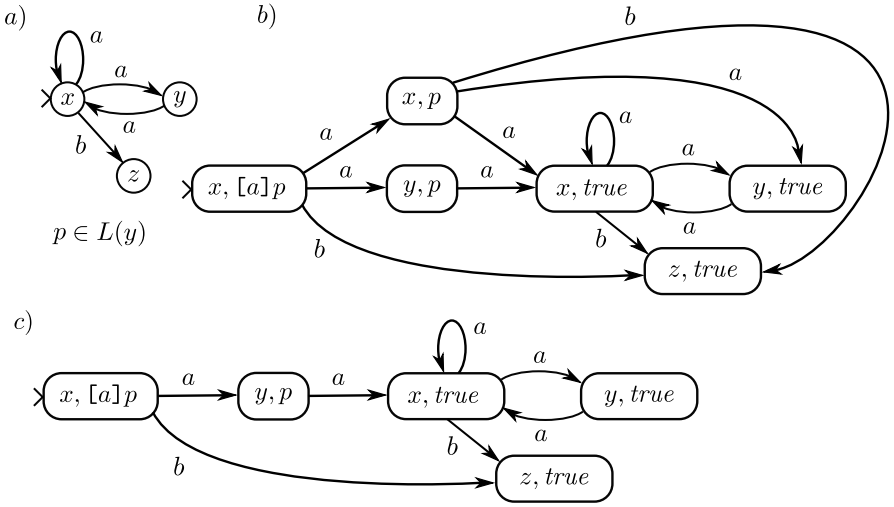

Fig. 3 Synthesis for the control requirement $[a]p$ upon the model in Fig. 3a, resulting in the model in Fig. 3c. This example illustrates the creation of a new transition relation over the state-formula product space, shown in Fig. 3b, the resulting embedded unfolding and subsequent transition removal upon Fig. 3b.

which transition removal takes place, is shown in Fig. 3b. This intermediate solution is constructed by combining the original initial state x with the formula $[a]p$, resulting in a new initial state. We then observe how the validity of the expression $[a]p$ relates to validity at succeeding states. In this case, after an a-step, p must be satisfied, while after a b-step, the formula *true* needs to hold. Transitions leading to state-formula pairs where the assigned formula cannot be satisfied are now removed, and therefore omitted in Fig. 3c. For example, the step $(x, [a]p) \xrightarrow{a} (x, p)$ is removed since property p cannot be satisfied in any state constructed from x, since $p \notin L(x)$. The thus created transition relation over the state-formula product space incorporates an embedded unfolding induced by the formula reductions. The model in Fig. 3a may also be adapted by directly removing the self-loop at state x. However, such a solution

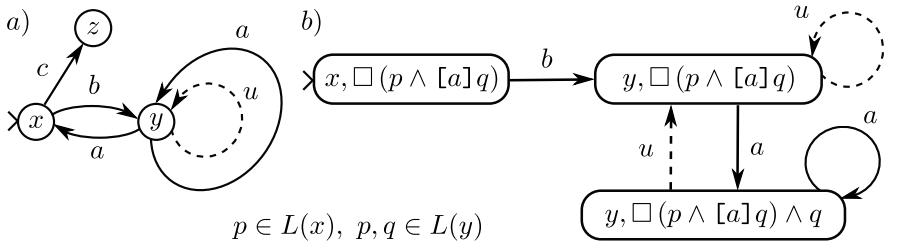

Fig. 4 Synthesis for the control requirement $\Box\,(p \wedge [a]\,q)$ upon the model in Fig. 4a, resulting in the model in Fig. 4b. Note that the invariant expression $\Box\,(p \wedge [a]\,q)$ re-occurs at succeeding states. Note that *true* conjuncts are removed in this picture for compactness, but such removal is not neccessary for ensuring finiteness.

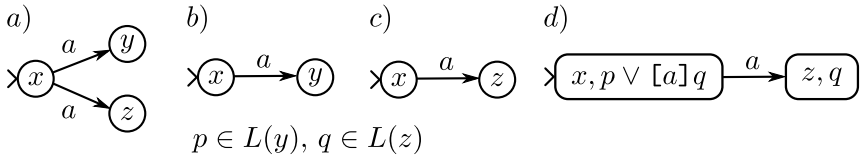

Fig. 5 Synthesis for $[a]\,p \vee [a]\,q$ upon the model in Fig. 5a would result in two different maximally permissive solutions shown in Fig. 5b and Fig. 5c. Instead, we restrict control requirements such that the left-hand disjunct may only contain elements from $\mathcal{B}$, as shown in Fig. 5d.

would clearly not be maximally permissive, while the adapted model in Fig. 3c retains this looping behavior at a later stage.

A somewhat more complicated example is shown in Fig. 4. The model in Fig. 4a is adapted such that it satisfies the control requirement $\Box\,(p \wedge [a]\,q)$, resulting in the model shown in Fig. 4b. Note that invariant formulas reduce in such a way that the entire invariant expression re-occurs at the next transition, while the formula under invariance is reduced. To counteract infinite expansion, formulas are normalized by removing all double conjuncts.

Fig. 5 is provided to shed some light on the restrictions in $\mathcal{F}$. If the formula $[a]\,p \vee [a]\,q$ were to be synthesized upon the model in Fig. 5a, this would result in two different maximally permissive solutions in Fig. 5b and Fig. 5c, which are essentially incomparable. We therefore restrict the logic $\mathcal{F}$ in such a way that the left-hand disjunct must contain an expression in $\mathcal{B}$. The formula reductions of a left-hand expression b in $b \vee f$ only reduce to *true* if b holds at the starting state of a transition, which can be readily verified.

We may now consider a formal treatment of these formula reductions as provided in Definition 8. Given the original plant transition relation $\longrightarrow \subseteq X \times \mathcal{E} \times X$, we will define a new transition relation $\longrightarrow_0 \subseteq (X \times \mathcal{F}) \times \mathcal{E} \times (X \times \mathcal{F})$ which will serve as the synthesis starting point. The specifics discussed in Fig. 5 regarding disjunctions require that the formula reductions need to be defined in terms of both formulas and states, as can be seen in Definition 8. As shown in Fig. 4, the number of formula reductions needs to be finite in order to ensure the construction of a finite synthesis starting point $\longrightarrow_0$. For this purpose

syntactical sub-formulas as given in Definition 7 are applied in Definition 8 to inhibit infinite expansion.

Definition 7 For $k = (X, L, \longrightarrow, x)$ and $f \in \mathcal{F}$ we derive the set of sub-formulas of f in state x (notation: $sub\,(x, f)$) by the rules below. Assume that $f, g, h \in \mathcal{F}$ and $b \in \mathcal{B}$ in the following definition:

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

$$\frac{f \in sub\,(x, g) \quad k \not\Vdash b}{f \in sub\,(x, b \vee g)} \qquad \frac{f \in sub\,(x, g)}{f \in sub\,(x, \Box\, g)}$$

Definition 8 We define the synthesis starting point $\longrightarrow_0 \subseteq (X \times \mathcal{F}) \times \mathcal{E} \times (X \times \mathcal{F})$ Given $k = (X, L, \longrightarrow, y) \in \mathcal{K}$, $f, g, f'g, ' \in \mathcal{F}$, $b \in \mathcal{B}$, $x, x' \in X$ and $e, e' \in \mathcal{E}$, we define the synthesis starting point $\longrightarrow_0$ by the derivation rules shown below.

$$\frac{x \xrightarrow{e} x'}{(x, b) \xrightarrow{e}_0 (x, true)} \qquad \frac{k \Vdash b \quad x \xrightarrow{e} x'}{(x, b \vee f) \xrightarrow{e}_0 (x', true)} \qquad \frac{k \not\Vdash b \quad (x, f) \xrightarrow{e}_0 (x', f')}{(x, b \vee f) \xrightarrow{e}_0 (x', f')}$$

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

$$\frac{(x, f) \xrightarrow{e}_0 (x', f') \quad (x, g) \xrightarrow{e}_0 (x', g') \quad g' \notin sub\,(x', f')}{(x, f \wedge g) \xrightarrow{e}_0 (x', f' \wedge g')}$$

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

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

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

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

We briefly consider the derivation rules in Definition 8. A basic formula $b \in \mathcal{B}$ always reduces to *true*. The details surrounding the reductions of the left-hand disjunct have been considered in Fig. 5. The right-hand reduction in a disjunctive formula is directly inherited, if the corresponding basic formula is not satisfied, as shown clearly in the third derivation rule in Definition 8.

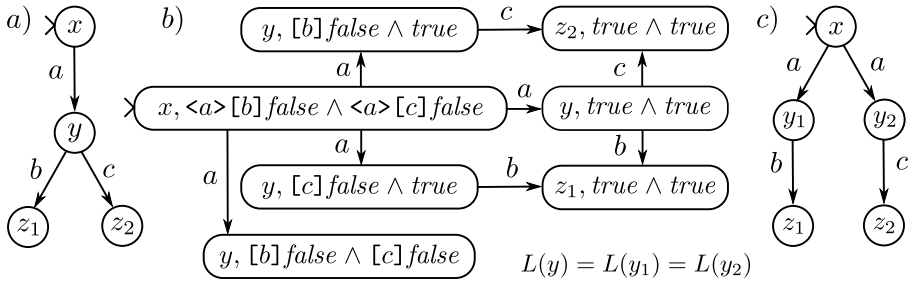

Fig. 6 Synthesis for $\langle a \rangle [b] \mathit{false} \wedge \langle a \rangle [c] \mathit{false}$ upon the model in Fig. 6a is not maximally permissive if only the b and c steps were removed from the y-state. Instead, we copy original behavior as shown in Fig. 6b, which is the correct maximal synthesis result if $a \in \mathcal{C}$. If $a \in \mathcal{U}$, then Fig. 6c is not a satisfying partial bisimulant of Fig. 6b.

Two different rules are required to define proper formula reductions under conjunction. As shown in Definition 8, the two respective rules for each conjunct are always inherited, but the right-hand reduct does not occur at the constructed target state if it is a sub-formula of the left-hand reduct. This effectively inhibits unbounded expansion of the reductions of invariant expressions. As detailed in Fig. 3a, a formula $[e]f$ reduces to f after an e-step, while it reduces to *true* after an $e' \neq e$ step. The reduction for a formula $\langle e \rangle f$ is somewhat more involved. After every e-step, an attempt is made to satisfy this formula, as signified by the reduction towards f. However, the original behavior after every e-step is also copied, which induces maximal permissiveness, as shown in Fig. 6. This is the key difference between the synthesis for a formula $[e]f$ and a formula $\langle e \rangle f$, which explains why the sixth and eighth rule are different. Synthesis for an invariant formula $\Box f$ has been considered in Fig. 4. Both the invariant formula itself, as well as its underlying reduct need to be present at the next state. The combination of reductions under conjunction then assure that appropriate modal expressions appear at later stages. Two rules are required for invariant formulas, as shown in Definition 8, to directly enforce the same type of normalization for the target reducts as shown in the rules for conjunction. The formulas $\Diamond b$, for $b \in \mathcal{B}$, $\langle e \rangle$, for $e \in \mathcal{E}$ and *dlf* each reduce to a *true* expression. Ensuring validity for these formulas relies upon the partial satisfiability test which is applied during synthesis.

Formulas of type $\langle e \rangle f$ are synthesized in such a way that original behavior is left in place. This is illustrated in Fig. 6. Synthesis for $\langle a \rangle [b] \mathit{false} \wedge \langle a \rangle [c] \mathit{false}$ upon the model in Fig. 6a would not result in a maximally permissive solution if only the b and c steps were removed from the y-state. Instead, we leave original behavior in place as shown in Fig. 6b, where new non-deterministic a-steps are introduced by applying Definition 8, followed by transition removal. Note that this is only a viable solution if $a \in \mathcal{C}$. If $a \in \mathcal{U}$, then the model in Fig. 6c is a satisfying partial bisimulant of Fig. 6a, but not of Fig. 6b. Therefore, if $a \in \mathcal{U}$ then Fig. 6b is not maximally permissive. We therefore applied the restriction of $e \in \mathcal{C}$ in Definition 4, for formulas of type $\langle e \rangle f$. The restriction that $b \in \mathcal{B}$ for formulas of type $\Diamond b$ is founded on

the same basis. The formula $<a>[b]\,\mathit{false} \wedge <a>[c]\,\mathit{false}$ may be expressed as $\Diamond(\neg x \wedge y \wedge \neg z_1 \wedge \neg z_2 \wedge [b]\,\mathit{false}) \wedge \Diamond(\neg x \wedge y \wedge \neg z_1 \wedge \neg z_2 \wedge [c]\,\mathit{false})$ and has the same synthesis solution (modulo state names) as shown in Fig. 6b. Consequently, the counterexample with regard to maximal permissiveness shown in Fig. 6c applies.

We now consider the *synthesizability* condition for removal of a constructed step $(x, f) \xrightarrow{e} (x', f')$. An initial and provably sound observation is to retain such steps if a satisfying partial bisimulant exists at the target state. That is, we should not disallow a transition $(x, f) \xrightarrow{e} (x', f')$ if a $k' \in \mathcal{K}$ exists such that $k' \preceq (X, L, \longrightarrow, x')$ and $k' \vDash f'$. However, existence of such a satisfying partial bisimulant is not a practical way from a computational perspective to express whether a constructed target state (x', f') should be retained after a step $(x, f) \xrightarrow{e} (x', f')$. Instead, we rely upon an incremental approach where we construct iterative transition relations $\longrightarrow_0$, $\longrightarrow_1$, $\longrightarrow_2$, ... until a stable point has been reached. We apply a *synthesizability* test (notation $(x', f') \uparrow_n f'$) to assess whether a constructed step $(x, f) \xrightarrow{e} (x', f')$ should be retained in step n of the iterative synthesis process. Derivation rules for this test are listed in Definition 9, and are discussed in detail thereafter.

When studying Definition 9, it might be helpful to take a glance at Definition 10, where we apply Definition 9 to create succeeding iterations $S^n_{k,f}$, $S^{n+1}_{k,f}$, $S^{n+2}_{k,f}$, ... of the synthesis result. We therefore have to define synthesizability in terms of a previously derived transition relation $\longrightarrow_n$. Therefore, the transition relation $\longrightarrow_n \subseteq (X \times \mathcal{F}) \times \mathcal{E} \times (X \times \mathcal{F})$ used in Definition 9 should be interpreted syntactically, without reference to a particular $n \in \mathbb{N}$. Definition 9 relies upon the syntactical notion of sub-formulas, as provided in Definition 7.

Definition 9 For $k = (X, L, \longrightarrow, y) \in \mathcal{K}$, $b \in \mathcal{B}$, $f, f_1, f_2, g, g' \in \mathcal{F}$, $e \in \mathcal{E}$, $x \in X$ and intermediary synthesis relation $\longrightarrow_n \subseteq (X \times \mathcal{F}) \times \mathcal{E} \times (X \times \mathcal{F})$ we derive synthesizability as follows:

$$\frac{k \Vdash b}{(x,g) \uparrow_n b} \qquad \frac{(x,g) \uparrow_n f_1 \quad (x,g) \uparrow_n f_2}{(x,g) \uparrow_n f_1 \wedge f_2} \qquad \frac{k \Vdash b}{(x,g) \uparrow_n b \vee f} \qquad \frac{(x,g) \uparrow_n f}{(x,g) \uparrow_n b \vee f}$$

$$\frac{}{(x,g) \uparrow_n [e]\,f} \qquad \frac{(x,g) \xrightarrow{e}_n (y,g') \quad f \in \mathit{sub}\,(y,g') \quad (y,g') \uparrow_n f}{(x,g) \uparrow_n <e>f}$$

$$\frac{(x,g) \uparrow_n f}{(x,g) \uparrow_n \Box f} \qquad \frac{(x,g) \longrightarrow^*_n (y,g') \quad (X, L, \longrightarrow, y) \Vdash b}{(x,g) \uparrow_n \Diamond b}$$

$$\frac{(x,g) \xrightarrow{e}_n (x',g')}{(x,g) \uparrow_n <e>} \qquad \frac{(x,g) \xrightarrow{e}_n (x',g')}{(x,g) \uparrow_n \mathit{dlf}}$$

The first derivation rule in Definition 9 expresses how synthesizability for a basic formula $b \in \mathcal{B}$ in (x, g) directly depends upon the validity of this formula at that particular state. If both conjuncts f_1 and f_2 can be synthesized,

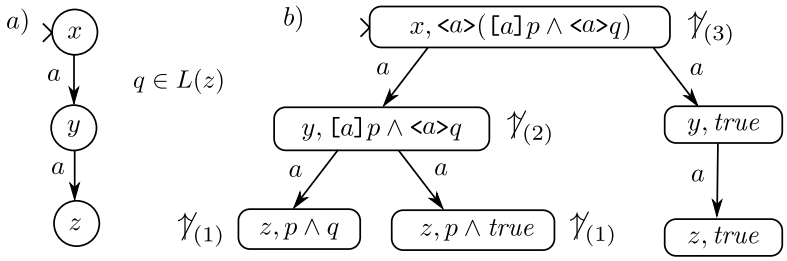

Fig. 7 Synthesis for the control requirement <a>([a]p ∧ <a>q) upon the model shown in Fig. 7a, resulting in Fig. 7b. Synthesizability at each state for various iterations in the process of transition removal is indicated using $\uparrow_i$, for $i \in \{1, 2, 3\}$. This synthesis property necessitates the use of sub-formulas in the synthesizability for formulas of type <e>f.

then the combination $f_1 \wedge f_2$ is synthesizable in (x, g). Synthesizability for disjunction is derived directly from its operands, as indicated by the third and fourth rule. A formula $[e]\, f$ is always synthesizable since every outgoing e-step may be removed. However, during the transition removal phase we have to take into account that e may be uncontrollable. For <e>f such that $e \in \mathcal{C}$, an example is considered in Fig. 7. This example shows how multiple iterations of transition removal are required to determine that the model in Fig. 7a can not be adapted to satisfy the formula <a>([a]p ∧ <a>q). Due to the introduction of a copy of the original behavior in the branch at the right-hand side in Fig. 7b, it might seem that the formula <a>([a]p ∧ <a>q) is still satisfiable, since an outgoing a-step exists. Therefore, synthesizability for a formula of type <e>f requires that f is a sub-formula of the formula assigned to the relevant step, as shown in Definition 9.

Synthesizability for a formula $\Box f$ in a combined state (x, g) requires that f is synthesizable in (x, g). The remaining expressions $\Diamond b$, <e> and *dlf* are only synthesizable if they can be directly satisfied. As justified by the intuition that no transition removal will make such an expression *more* true.

We may now define the succeeding iterations in the computational approximations $\longrightarrow_1$, $\longrightarrow_2$, ..., for which $\longrightarrow_0$ has already been given in Definition 8. The corresponding synthesis results $S^1_{k,f}$, $S^2_{k,f}$, ... are also detailed in Definition 10.

Definition 10 For $k = (X, L, \longrightarrow, x)$, $f \in \mathcal{F}$ and $n \in \mathbb{N}$, we define the n-th iteration $S^n_{k,f}$ in the computational synthesis process as follows:

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

Where $L_{XF}(y, g) = L(y)$ for all $y \in X$ and $g \in \mathcal{F}$. The transition relation $\longrightarrow_n$ is defined for $y, y' \in X$, $g, g' \in \mathcal{F}$ and $e \in \mathcal{E}$ as follows:

$$\frac{(y, g) \xrightarrow{e}_n (y', g') \quad e \in \mathcal{U}}{(y, g) \xrightarrow{e}_{n+1} (y', g')}$$

$$\frac{(y, g) \xrightarrow{e}_n (y', g') \quad \forall\, v \in \mathcal{U}^* \,:\, \forall\, (y', g') \xrightarrow{v}_n{}^{*} (y'', g'') \,:\, (y'', g'') \uparrow_n g''}{(y, g) \xrightarrow{e}_{n+1} (y', g')}$$

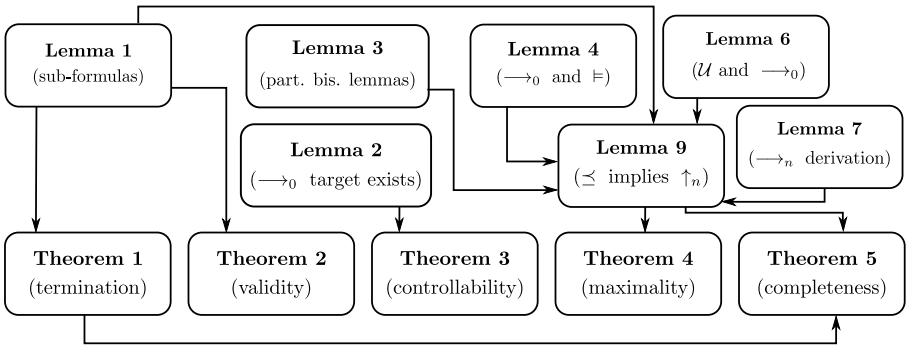

Fig. 8 A graphical illustration of the dependencies between the most important lemmas and theorems within Section 5. In-going arrows represent a dependency to proof entity which the arrow originates from.

The first rule in Definition 10 states that every uncontrollable step should be preserved. The second rule expresses the actual synthesis functionality, where a transition is only preserved if synthesizability holds at each state which is reachable by uncontrollable steps.

Transition removal in the succeeding iterations of the transition relation $\longrightarrow_0$, $\longrightarrow_1$, $\longrightarrow_2$, ... proceeds until no more target states of individual transitions are considered candidates for removal. That is, if the synthesizability predicate holds at every reachable state. If a plant model $k \in \mathcal{K}$ has finitely many transition, this process is terminating. This premise for completeness is formalized in Definition 11.

Definition 11 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 for all $(x, f) \longrightarrow^*_n (x', f')$ it holds that $(x', f') \uparrow_n f'$.

If the condition of completeness as stated in Definition 11 can not be reached a solution to the synthesis problem in Definition 6 does not exist.

5 Correctness

A number of proofs are required to establish the correctness of the synthesis theory proposed in the previous section. Theorem 1 shows that the synthesis construction in Definition 10 is terminating. We then proceed by proving that this synthesis construction satisfies the three main results from Definition 6: validity (Theorem 2), controllability (Theorem 3) and maximal permissiveness (Theorem 4). The final result in this section is shown in Theorem 5, where we prove that if a solution exists, it will be found. Fig. 8 shows the dependencies between the most important theorems and lemmas in this section. Computer verified definitions and proofs which have been created using the Coq proof assistant are available for Theorems 2, 3 and 4 at the following resource:

`https://github.com/ahulst/deds`

The Coq-formalizations are formulated in terms of inductive predicates to achieve a formalization which closely resembles the mathematical proofs stated below. This specific choice has as a drawback that the proofs for Theorem 1 and Theorem 5 cannot be directly encoded in Coq. An alternative setup of these Coq proofs where the synthesis construction would have been encoded using recursive functions, as opposed to inductive predicates, would introduce substantial overhead to such an extend that correspondence between formal definitions and the definitions in this paper would have been lost.

We first establish a number of technical results regarding sub-formulas as given in Definition 7.

Lemma 1 *For $k = (X, L, \longrightarrow, x) \in \mathcal{K}$ we have the following results regarding sub-formulas:*

(a) For $f \in sub\,(x, g)$ and $k \vDash g$ it holds that $k \vDash f$;
(b) For $f \wedge g \in sub\,(x, h)$ it holds that $f \in sub\,(x, h)$ and $g \in sub\,(x, h)$;
(c) For $[e]f \in sub\,(x, g)$ and $(x, g) \xrightarrow{e}_0 (y, g')$ it holds that $f \in sub\,(y, g')$;
(d) For $b \vee f \in sub\,(x, g)$ and $k \nVdash b$ it holds that $f \in sub\,(x, g)$;
(e) For $\Box f \in sub\,(x, g)$ it holds that $f \in sub\,(x, g)$;
(f) For $\Box f \in sub\,(x, g)$ and $(x, g) \xrightarrow{e}_0 (y, g')$ it holds that $\Box f \in sub\,(y, g')$;
(g) For $(x, h) \uparrow_n g$ and $f \in sub\,(x, g)$ it holds that $(x, h) \uparrow_n f$; and
(h) For $f \in sub\,(x, g)$ and $g \in sub\,(x, h)$ it holds that $f \in sub\,(x, h)$.

Proof These results can be obtained by induction towards the derivation depth in Definition 7. □

Theorem 1 *The derivation of the synthesis result is finite.*

Proof We prove the following result: if $k = (X, L, \longrightarrow, x) \in \mathcal{K}$, for finite $\longrightarrow$, then $S^0_{k,f}$ has finitely many transitions. We therefore have to show that the number of transitions in $\longrightarrow_0$ is finite. Every succeeding synthesis iteration removes steps until a stable point has been reached. Finiteness of $\longrightarrow_0$ is therefore sufficient to prove termination. Given that $\longrightarrow$ is finite, we will have to prove that the following set is finite:

$$\{(x', f') \in X \times \mathcal{F} \mid (x, f) \longrightarrow_0^* (x', f')\}$$

We prove this result directly by induction towards the structure of f. For the cases where $f \equiv b$ or $f \equiv \Diamond b$, for $b \in \mathcal{B}$, or $f \equiv \langle e \rangle$ or $f \equiv dlf$, the set $\{(x, f)\} \cup X \times \{true\}$ includes all newly constructed states reachable by $\longrightarrow_0^*$. Note that this is an overapproximation but still a finite set, if X is restricted to the states reachable by $\longrightarrow^*$.

If $f \equiv f_1 \wedge f_2$, then by induction we derive the following sets:

$$C_1 = \{(x', f') \in X \times \mathcal{F} \mid (x, f_1) \longrightarrow_0^* (x', f')\}$$
$$C_2 = \{(x', f') \in X \times \mathcal{F} \mid (x, f_2) \longrightarrow_0^* (x', f')\}$$

Subsequently, we will have to show that the set $C_1 \cup \{(y, g \wedge h) \mid (y, g) \in C_1, (y, h) \in C_2\}$ is finite. By induction towards the length of $(x, f_1 \wedge f_2) \longrightarrow_0^* (y, f')$, we can show that either $(y, f') \in C_1$, if the fourth rule in Definition 8 was applied, or $f' \equiv g \wedge h$ and $(y, g) \in C_1$ and $(y, h) \in C_2$. The next case to consider in the induction proof is when $f \equiv b \vee g$, for $b \in \mathcal{B}$. By induction, we derive the set:

$$C = \{(x', f') \in X \times \mathcal{F} \mid (x, g) \longrightarrow_0^* (x', g')\}$$

which is finite by induction. Henceforth, the set $\{(x, b \vee g)\} \cup C \cup X \times \{true\}$ is also finite. The inductive cases for $f \equiv [e]\, f'$ and $f \equiv \langle e \rangle f'$ are considered in parallel. By induction, for each step $(x, [e]\, f') \xrightarrow{e}_0 (y, f')$ and for each step $(x, \langle e \rangle f') \xrightarrow{e}_0 (y, f')$ a finite set C_y can be derived by induction. These sets C_y may then be combined under union and combined with $X \times \{true\}$ to finitely define the set of states reachable under $\longrightarrow_0$.

The final case for the inductive proof is where $f \equiv \Box\, f'$, which requires an additional helper function D defined below. Assume that $f \in \mathcal{F}$, $C \subseteq X \times \mathcal{F}$ and $n \in \mathbb{N}$ in the following inductive definition:

$$\begin{aligned} D(f, C, 0) &= X \times \{\Box\, f\} \\ D(f, C, n+1) &= D(f, C, n) \cup \{(x, g \wedge h) \mid (x, g) \in D(f, C, n),\, h \in C\} \end{aligned}$$

If $C = \{(x', f') \in X \times \mathcal{F} \mid (x, f) \longrightarrow_0^* (x', f')\}$ then the finite overapproximation $D(f, C, |C|)$, where $|C|$ indicates the number of elements in C, contains all states reachable from $(x, \Box\, f)$ over $\longrightarrow_0^*$.

We first show that for all $(x, \Box\, f) \longrightarrow_0^* (x', g)$ there exists an $n \in \mathbb{N}$ such that $g \in D(f, C, n)$. If we apply induction to the structure of g, there are two relevant cases: 1) if $g \equiv \Box\, f$ then $(x, g) \in D(f, C, 0)$ and, 2) if $g \equiv g_1 \wedge g_2$ then there exists an $n \in \mathbb{N}$ such that $(x', g_1) \in D(f, C, n)$ and since $(x', g_2) \in C$, it holds that $(x', g_1 \wedge g_2) \in D(f, C, n+1)$.

Subsequently, we show the following: if $(x, \Box\, f) \longrightarrow_0^* (x', g)$ then $(x', g) \in D(f, C, |C|)$. Clearly, as we just showed, there exists an $n \in \mathbb{N}$ such that $(x', g) \in D(f, C, n)$. However, if $(x', g) \in D(f, C, n)$ and $n > |C|$ then the derivation rules in Definition 8 show that $g \equiv g_1 \wedge g_2$ and $g_2 \notin sub\,(x', g_1)$, if $(x', g) \notin D(f, C, m)$ for all $m < n$. However, for all $(x, f) \longrightarrow_0^* (x', f')$ it holds that $(x', f') \in C$. If $(x', g) \notin D(f, C, m)$ for all $m < n$ then g has n different conjuncts and since $n > |C|$ it holds that $g_2 \in sub\,(x', g_1)$. $\square$

If the synthesis result satisfies the completeness premise, and thus when the synthesizability predicate holds at every reachable state, then the synthesis result satisfies the control requirement, as shown in Theorem 2.

Theorem 2 *If $k = (X, L, \longrightarrow, x) \in \mathcal{K}$, $f \in \mathcal{F}$ and $n \in \mathbb{N}$, such that $S^n_{k,f}$ is complete, then $S^n_{k,f} \vDash f$.*

Proof We will prove the following theorem: for $g \in \mathcal{F}$ such that $f \in sub\,(x, g)$ and $S^n_{k,g}$ is complete then $S^n_{k,g} \vDash f$, which is sufficient since $f \in sub\,(x, f)$. We apply induction towards the structure of $f \in \mathcal{F}$, thereby generalizing over g

and x. Note that for each inductive case we have $(x,g) \uparrow_n g \Rightarrow (x,g) \uparrow_n f$, by Lemma 1(g).

If $f \equiv b$, for $b \in \mathcal{B}$, then $(x,g) \uparrow_n b$ and thus $k \vDash b$, which implies $S^n_{k,g} \vDash b$. If $f \equiv f_1 \wedge f_2$ then $f_1 \in sub\,(x,g)$ and $f_2 \in sub\,(x,g)$, which leads to $S^n_{k,g} \vDash f_1$ and $S^n_{k,g} \vDash f_1$ by induction. If $f \equiv b \vee f'$ then we distinguish between two cases: 1) if $k \vDash b$ then $S^n_{k,g} \vDash b$, 2) if $k \nvDash b$ then by Lemma 1(d) it holds that $f' \in sub\,(x,g)$, which by induction leads to $S^n_{k,g} \vDash f'$.

If $f \equiv [e]f$, then assume there exists a step $(x,g) \xrightarrow{e}_n (x',g')$, and define $k' = (X, L, \longrightarrow, x')$. Since $(x,g) \xrightarrow{e}_0 (x',g')$, by Definitions 10 and 8, it holds that $f \in sub\,(x',g')$, by Lemma 1(c). By the induction hypothesis for f', we may now derive $S^n_{k',g'} \vDash f'$. Note that the induction premise for completeness for $S^n_{k',g'}$ follows from the assumption of $(x,g) \xrightarrow{e}_n (x',g')$ and Definition 11. If $f \equiv \langle e \rangle f'$, then by Definition 9 there exists a step $(x,g) \xrightarrow{e}_n (x',g')$ such that $f' \in sub\,(x',g')$ and $(X, L, \longrightarrow, x') \vDash g'$, where we abbreviate $k' = (X, L, \longrightarrow, x')$. Since $S^n_{k',g'}$ is complete, we may apply induction to derive $S^n_{k',g'} \vDash f'$.

If $f \equiv \Box f'$, then assume there exists a sequence of steps $(x,g) \longrightarrow^*_n (x',g')$ such that $f' \in sub\,(x',g')$ by Lemmas 1(e) and 1(f). Set $k' = (X, L, \longrightarrow, x')$, then due to the assumption of $(x,g) \longrightarrow^*_n (x',g')$, $S^n_{k',g'}$ is complete which leads to $S^n_{k',g'} \vDash f'$ by induction. For the cases $f \equiv \Diamond b$, for $b \in \mathcal{B}$, or $f \equiv \langle e \rangle$ or $f \equiv dlf$, the result $S^n_{k,g} \vDash f$ follows directly from $(x,g) \uparrow_n f$. □

Lemma 2 *If $f \in \mathcal{F}$, $e \in \mathcal{E}$ and $x, y \in X$, such that $x \xrightarrow{e} y$, there exists an $f' \in \mathcal{F}$ such that $(x,f) \xrightarrow{e}_0 (y,f')$.*

Proof By induction towards the structure of f. □

Controllability then follows directly from Lemma 2 and the construction in Definition 10, as shown in Theorem 3.

Theorem 3 *If $k = (X, L, \longrightarrow, x) \in \mathcal{K}$, $f \in \mathcal{F}$ and $n \in \mathbb{N}$ then $S^n_{k,f} \preceq k$.*

Proof We will show that $S^n_{k,f} \preceq_R k$ by defining R as follows:

$$R = \{((y,g),y) \mid (x,f) \longrightarrow^*_n (y,g)\}$$

Clearly $((x,f),x) \in R$. Assume that $((y,g),y) \in R$. If $(y,g) \xrightarrow{e}_n (z,g')$ then $y \xrightarrow{e} z$ by Definitions 10 and 8, such that $((z,g),z) \in R$.

If $y \xrightarrow{e} z$ then by Lemma 2 there exists a formula-reduct $(y,g) \xrightarrow{e}_0 (z,g')$. Since $e \in \mathcal{U}$ it follows from the construction of $\longrightarrow_n$ in Definition 10 that $(y,g) \xrightarrow{e}_n (z,g')$. □

Partial bisimulation is related to Definition 8 and Definition 7, but also implies validity for formulas in $\mathcal{B}$. These results are listed in Lemma 3.

Lemma 3 *For $k' = (X', L', \longrightarrow', x') \in \mathcal{K}$ and $k = (X, L, \longrightarrow, x) \in \mathcal{K}$, such that $k' \preceq k$, we have the following results:*

(a) For all $b \in \mathcal{B}$ it holds that $k' \Vdash b$ if and only if $k \Vdash b$;
(b) For all $f, g \in \mathcal{F}$ it holds that $f \in sub\,(x', g)$ if and only if $f \in sub\,(x, g)$; and
(c) For all $f, f' \in \mathcal{F}$ and $e \in \mathcal{E}$ and $y \in X$ it holds that $(x, f) \xrightarrow{e} (y, f')$ if and only if $(x, f) \xrightarrow{e} (y, f')$.

Proof Result (a) is obtained by induction towards the structure of $b \in \mathcal{B}$ in Definition 3, result (b) is derived by induction towards the derivation depth in Definition 7, and result (c) is shown by induction towards the derivation depth in Definition 8. □

Lemmas 4 and 5 detail how existence of a formula-reduct relates to validity. Lemma 5 can be considered a specific instance of Lemma 4, where we require the sub-formula property.

Lemma 4 *For $k = (X, L, \longrightarrow, x) \in \mathcal{K}$, $f \in \mathcal{F}$, $e \in \mathcal{E}$ and $x' \in \mathcal{X}$, such that $x \xrightarrow{e} x'$, there exists an $f' \in \mathcal{F}$ such that $(x, f) \xrightarrow{e}_0 (x', f')$ and $(X, L, \longrightarrow, x') \vDash f'$.*

Proof By induction towards the structure of $f \in \mathcal{F}$. □

Lemma 5 *For $k = (X, L, \longrightarrow, x) \in \mathcal{K}$, $e \in \mathcal{E}$ and $f, g \in \mathcal{F}$ such that $<e>f \in sub\,(x, g)$ and $k \vDash g$, there exist $x' \in X$ and $g' \in \mathcal{F}$ such that $(x, g) \xrightarrow{e}_0 (x', g')$ and $f \in sub\,(x', g')$ and $(X, L, \longrightarrow, x') \vDash g'$.*

Proof By induction towards the derivation depth of $<e>f \in sub\,(x, g)$ in Definition 7, using Lemma 4 for both cases for conjunction to cover the opposite conjunct (i.e. the case not covered by induction). □

If $(x, f) \xrightarrow{e}_0 (y, g)$ and $(x, f) \xrightarrow{e} (y, h)$ then $g \equiv h$ if $e \in \mathcal{U}$. This determinism property gives rise to a specific result between formula-reducts and validity, as shown in Lemma 6.

Lemma 6 *If $k = (X, L, \longrightarrow, x) \in \mathcal{K}$ and $(x, f) \xrightarrow{e}_0 (x', f')$, for $e \in \mathcal{U}$, $x' \in X$ and $f, f' \in \mathcal{F}$, such that $k \vDash f$, it holds that $(X, L, \longrightarrow, x') \vDash f'$.*

Proof By induction towards the derivation depth of $(x, f) \xrightarrow{e}_0 (x', f')$. □

Lemma 7 *If $(x, f) \xrightarrow{e}_0 (x', f')$, then for all $n \in \mathbb{N}$ it holds that $(x, f) \xrightarrow{e}_n (x', f')$ if for all $m < n$ and for all $v \in \mathcal{U}^*$ and $(x', f') \xrightarrow{v}{}^*_m (x'', f'')$ it holds that $(x'', f'') \uparrow_m f''$.*

Proof This result follows from the construction in Definition 10 if we apply strong induction towards n. □

Lemma 8 *If $k = (X, L, \longrightarrow, x) \in \mathcal{K}$ and $k' = (X', L', \longrightarrow', x') \in \mathcal{K}$, such that $k' \preceq k$, and if $f, f' \in \mathcal{F}$, $n \in \mathbb{N}$ and $v \in \mathcal{U}^*$, such that $(x, f) \xrightarrow{v}_n^* (y, f')$ and $(X', L', \longrightarrow', x') \vDash f$, then there exists an $y' \in X'$ such that $x'\{\longrightarrow'\}^* y'$ and $(X', L', \longrightarrow', y') \preceq (X, L, \longrightarrow, y)$ and $(X', L', \longrightarrow', y') \vDash f'$.*

Proof This result follows from Definition 2, Lemma 6 and induction towards the length of v. □

Lemma 9 details an important result between the semantic notion of synthesizability (i.e. existence of a satisfying partial bisimulant) and the syntactic notion as given in Definition 9.

Lemma 9 *For $k = (X, L, \longrightarrow, x) \in \mathcal{K}$, $k' \in \mathcal{K}$ and for $f, g \in \mathcal{F}$, such that $f \in sub\,(x, g)$ and $k' \vDash g$, it holds that $(x, g) \uparrow_n f$.*

Proof The proof is somewhat involved. We apply strong induction towards n, thereby generalizing over all other variables, and thereafter a nested induction towards the structure of f, thereby generalizing over g, x and k'. A number of cases for f can be resolved directly using the induction hypothesis for f, and do not depend upon the induction towards n. These are the cases for $f \equiv b \in \mathcal{B}$, $f \equiv f_1 \wedge f_2$, $f \equiv b \vee f'$, $f \equiv [e]\, f'$ and $f \equiv \Box f'$. Lemma 1 is required to resolve these cases.

We now consider the other inductive cases for f, where we highlight the key differences for the cases under the inductions $n \equiv 0$ and $n + 1$. Assume that $k' = (X', L', \longrightarrow', x')$ and $R \subseteq X' \times X$ such that $k' \preceq_R k$.

If $f \equiv \langle e \rangle f'$, for $e \in \mathcal{C}$, then by Lemma 1(a) it holds that $k' \vDash \langle e \rangle f'$. By Lemma 5 there exists a step $x' \xrightarrow{e} y'$ and a formula-reduction $(x', g) \xrightarrow{e}_0 (y', g')$ such that $(X', L', \longrightarrow', y') \vDash g'$ and $f' \in sub\,(y', g')$. By partial bisimulation there exists a step $x \xrightarrow{e} y$ such that $(y', y) \in R$. Definition 8 then allows the construction of a step $(x, g) \xrightarrow{e}_0 (y, g')$. By the induction hypothesis for f', we now derive $(y, g') \uparrow_n f'$. For the case $n \equiv 0$, this is sufficient to derive that $(x, g) \uparrow_n \langle e \rangle f'$. For the inductive case for n, we apply Lemma 7 to construct a step $(x, g) \xrightarrow{e}_n (y, g')$. This requires that we prove that for all $m < n$ and $v \in \mathcal{U}^*$ such that $(y, g') \xrightarrow{v}_m^* (z, g'')$ it holds that $(z, g'') \uparrow_m g''$. This can be resolved by Lemma 8 and the induction hypothesis for n.

The case for $f \equiv \Diamond b$, for $b \in \mathcal{B}$ is essentially a generalization for the case for $f \equiv \langle e \rangle f'$. If $k' \vDash \Diamond b$ then there exists a $x'\{\longrightarrow'\}^* y'$ such that $(X', L', \longrightarrow', y') \vDash b$, and by Lemma 4 there exists a $g' \in \mathcal{F}$ such that $(X', L', \longrightarrow', y') \vDash g'$. This allows us to construct $(x, g) \longrightarrow_0^* (y, g')$ by Definition 8, such that $(X, L, \longrightarrow, y) \vDash b$. For the inductive case for n we then construct $(x, g) \longrightarrow_n^* (y, g')$, which is sufficient to derive $(x, g) \uparrow_n \Diamond b$. The two remaining cases for $f \equiv \langle e \rangle$ and $f \equiv dlf$ are essentially instances for the case $f \equiv \langle e \rangle f'$. □

The main result required for the maximality Theorem 4 has been established in Lemma 9. We may now resolve this result directly.

Theorem 4 *For $k', k \in \mathcal{K}$ such that $k' \preceq k$ and $k' \vDash f$, and for all $n \in \mathbb{N}$, it holds that $k' \preceq S^n_{k,f}$.*

Proof Assume $k = (X, L, \longrightarrow, x)$ and $k' = (X', L', \longrightarrow, x')$ and further assume that $R \subseteq X' \times X$ such that $k' \preceq_R k$. We will show that $k' \preceq_{R'} S^n_{k,f}$ where R' is defined as:

$$R' = \{(y', (y, g)) \mid (y', y) \in R \wedge (X', L', \longrightarrow', y') \vDash g\}$$

Clearly $(x', (x, f)) \in R'$ and for all $(y', (y, g)) \in R'$ it holds that $L'(y') = L_{\mathrm{XF}}(y, g)$. If $y' \xrightarrow{e}' z'$ then by Lemma 4 there exists a $(y', g) \xrightarrow{e} (z', g')$ such that $(X', L', \longrightarrow', z') \vDash g'$. By partial bisimulation, there exists a step $y \xrightarrow{e} z$ such that $(z', z) \in R$. We then apply Lemma 7 and resolve that for all $m < n$ and for all $v \in \mathcal{U}^*$ such that $(z, g') \xrightarrow{v}_m (w, g'')$ it holds that $(w, g'') \uparrow_m g''$ by application of Lemma 8 and Lemma 9. We then have $(z', (z, g')) \in R'$.

For the right-to-left case, assume $(y, g) \xrightarrow{e}_n (z, g')$ for $e \in \mathcal{U}$. Since $y \xrightarrow{e} z$, there exists a step $y' \xrightarrow{e}' z'$ such that $(z', z) \in R$. By Lemma 6, it holds that $(X', L', \longrightarrow', z') \vDash g'$, and therefore $(z', (z, g')) \in R'$. □

Theorem 5 shows that if a solution exists it will eventually be found by the synthesis construction introduced before.

Theorem 5 *If $k', k \in \mathcal{K}$ and $f \in \mathcal{F}$, such that $k' \preceq k$ and $k' \vDash f$, then there exists an $n \in \mathbb{N}$ such that $S^n_{k,f}$ is complete.*

Proof Assume that $k = (X, L, \longrightarrow, x)$ and $k' = (X', L', \longrightarrow', x')$. By Theorem 1 there exists an $n \in \mathbb{N}$ such that $S^n_{k,f}$ is stable. Due to the construction in Definition 10, for all $(x, f) \longrightarrow^*_n (y, f')$ at least one of the following two observations holds:

1. There exist $v, w \in \mathcal{U}^*$ and $s \in \mathcal{C}^*$ such that $(x, f) \xrightarrow{vsw}{}^*_n (y, f')$; or
2. There exists $u \in \mathcal{U}^*$ such that $(x, f) \xrightarrow{u}{}^* (y, f')$.

In the first case, by Definition 10 it holds that $(y, f') \uparrow_n f'$. For the second case, we apply Lemma 8 to obtain an $y' \in X'$ such that $x'\{\longrightarrow'\}^* y'$ and $(X', L', \longrightarrow', y') \vDash f'$ and $(X', L', \longrightarrow', y') \preceq (X, L, \longrightarrow, y)$. By Lemma 9 it then holds that $(y, f') \uparrow_n f'$. It then follows that $S^n_{k,f}$ is complete. □

6 Computation

We propose the algorithm in Fig. 9 as a direct implementation of the theoretical synthesis construction introduced in Section 4, for which termination and correctness have already been shown. What remains to be analyzed is the computational complexity of the proposed algorithm, which is detailed in Theorem 6. We first analyze the key parts of this algorithm.

For the first part of this algorithm, shown in lines 1-11 in Fig. 9, a somewhat different approach compared to the theoretical setup in Definition 8 is applied,

```
1  procedure zero (⟶ ⊆ X × 𝓔 × X, (x, f) ∈ X × 𝓕, H ⊆ X × 𝓕)
2      returns ⟶0 ⊆ (X × 𝓕) × 𝓔 × (X × 𝓕)
3  begin
4      set ⟶0 := ∅
5      if (x, f) ∈ H
6          return ∅
7      for each (x, f) -e->0 (x', f')
8          set ⟶0 := ⟶0 ⋃ {(x, f) -e->0 (x', f')}
9          set ⟶0 := ⟶0 ⋃ zero (⟶, (x', f'), H ⋃ {(x, f)})
10     return ⟶0
11 end
12
13 procedure synthesis (k ∈ 𝓚, f ∈ 𝓕)
14     returns S^n_{k,f} ∈ 𝓚 or false
15 begin
16     let k = (X, L, ⟶, x)
17     set ⟶0 := zero (⟶, (x, f), ∅)
18     for each n > 0
19         set ⟶n := ∅
20     set n := 0
21     repeat until ⟶n−1 = ⟶n
22         for each (y, g) -e->n (y', g')
23             if e ∈ 𝓤
24                 set ⟶n+1 := ⟶n+1 ⋃ {(y, g) -e->n (y', g')}
25             else if (y'', g'') ↑n g'' for each v ∈ 𝓤* and (y', g') -v->*n (y'', g'')
26                 set ⟶n+1 := ⟶n+1 ⋃ {(y, g) -e->n (y', g')}
27         set n := n + 1
28     set S^n_{k,f} := (X × 𝓕, L_XF, ⟶n, (x, f))
29     if (x', f') ↑n f' for each (x, f) ⟶*n (x', f')
30         return S^n_{k,f}
31     else
32         return false
33 end
```

Fig. 9 Algorithm for synthesis of a formula $f \in \mathcal{F}$, applied to the Kripke-LTS $k \in \mathcal{K}$. The synthesis starting point $\longrightarrow_0$ is constructed using a recursive procedure. Following this step, the main synthesis procedure applies the iterative process of transition removal until a stable point has been reached. The completeness test then determines whether synthesis has been successful.

since these derivation rules can not be directly projected onto pseudo-code. Instead of a direct transformation of each original transition $x \xrightarrow{e} x'$ into a combined transition $(x, f) \xrightarrow{e}_0 (x', f')$, we use a recursive procedure *zero* where finiteness of formula expansion, as shown in Theorem 1, is applied in order to obtain a finite set of connected transitions which constitute $\longrightarrow_0$. This means that once every outgoing transition of a certain state (x, f) has been computed, these transitions, and successive transitions thereof, do not have to be computed again. The test in line 5 in Fig. 9, determines whether the state (x, f) has been inspected before.

The second part of the algorithm, shown in lines 13-33 in Fig. 9, applies the synthesizability test repeatedly and removes the transitions to states for which this test fails. This is done until a stable point is reached, as can be

observed in line 21 in Fig. 9. The completeness test in line 29 in Fig. 9 then determines whether synthesis has been successful.

Note that the implementation of Definition 8 and Definition 9 is not shown here, since it is assumed that these parts of the algorithm may be straightforwardly derived from their corresponding formal definitions. This includes the synthesizability test for which decidability can be derived from Definition 9 as follows. Note that it is decidable whether a formula of type $b \in \mathcal{B}$ holds at a certain state, since the Boolean algebra $\mathcal{B}$ includes only the tests for whether a label has been assigned to a particular state as well as the Boolean connectives. It then follows that it is decidable if a formula of type $\Diamond b$ holds at a certain state, by traversing all states reachable over $\longrightarrow_n$ and keeping track of which states have already been visited. The decidability of $\uparrow_n f$ for $f \in \mathcal{F}$ then follows from the inductive build-up of the derivation rules in Definition 9.

We now wish to sketch a proof for the computational complexity for the algorithm shown in Fig. 9. This requires an appropriate metric for the formula size since the number of transitions in $S^0_{k,f}$ depends upon the size of the formula. Such a metric is given in Definition 12, followed by the actual computational complexity proof.

Definition 12 We define $\mathit{size} : \mathcal{F} \mapsto \mathbb{N}$ as a metric for the size of the formulas in $\mathcal{F}$. Assume that $f, g \in \mathcal{F}$, $b \in \mathcal{B}$ and $e \in \mathcal{E}$ in the following definition:

$$
\begin{array}{ll}
\mathit{size}\,(f) = 1 \text{ for } f \in \{b, \Diamond b, \langle e \rangle, \mathit{dlf}\} & \mathit{size}\,(f \wedge g) = \mathit{size}\,(f) + \mathit{size}\,(g) \\
\mathit{size}\,(b \vee f) = 1 + \mathit{size}\,(f) & \mathit{size}\,([\,e\,]\,f) = 1 + \mathit{size}\,(f) \\
\mathit{size}\,(\langle e \rangle f) = 1 + \mathit{size}\,(f) & \mathit{size}\,(\Box\, f) = 1 + \mathit{size}\,(f)
\end{array}
$$

Theorem 6 *The algorithmic generation of $S^n_{k,f}$ terminates in $\mathcal{O}\,(m^4)$ steps, if $m = \mathit{size}\,(f) \times l$ and k has l transitions.*

Proof Let $k = (X, L, \longrightarrow, x)$ and $f \in \mathcal{F}$. Assume that l is the number of transitions in $\longrightarrow$. The starting point of synthesis $\longrightarrow_0$, as generated by the procedure `zero`, has $\mathcal{O}\,(\mathit{size}\,(f) \times l)$ transitions, since it is expanded by a factor which depends upon the size of f. The procedure `zero` is invoked only once within the procedure `synthesis`, and returns the transition relation $\longrightarrow_0$, having $\mathcal{O}\,(m)$ transitions. Since the succeeding iteration in the `synthesis` procedure operates multiple times upon this transition relation, it can be safely assumed that this part of the procedure outgrows other parts in the computational complexity. We distinguish four nested operations which may each take $\mathcal{O}\,(m)$ steps:

1. The outer loop starting in line 21, which runs until a stable transition relation has been reached, which may involve as many iterations as there are transitions in $\longrightarrow_0$.
2. The inner loop starting in line 23, which considers for each transition whether it should be removed.

3. The applied synthesizability test in line 24 can be subdivided into two nested parts:
   (a) Synthesizability is evaluated at every state reachable by uncontrollable events, which may involve a search over $\mathcal{O}(m)$ transitions.
   (b) The synthesizability test itself has complexity $\mathcal{O}(m)$ as shown in Definition 9, due to formulas of type $\Diamond b$.

This leads to the observation that the two nested loops in the `synthesis` procedure give rise to a computational complexity in the order of $\mathcal{O}(m^4)$. The succeeding invocation of the completeness test only computes the synthesizability for every remaining reachable state, and does not remove any more transitions. Its complexity is therefore superseded by the aforementioned two nested loops. □

The algorithm in Fig. 9 is presented as a direct implementation of the synthesis construction introduced in this paper and not as the most efficient or optimized implementation, since this would obscure the insight into such an algorithm. Nevertheless, the second case study in Section 9 analyzes the scalability of the algorithm presented in Fig. 9.

7 Ramadge-Wonham Supervisory Control

In this section we explain how a Ramadge-Wonham (RW) control synthesis problem (Ramadge and Wonham (1987)) may be expressed using the theory proposed in this paper. Among other adaptations, we have to establish a relation between the language-based constructs in RW-synthesis and the behavioral preorder of partial bisimulation. Note that RW-synthesis is limited to deterministic models of both plant and controller.

Recall that RW-synthesis identifies a subset $X_m \subseteq X$ of states as *marked*, modeling completed or notified tasks in the physical process the plant represents (Ramadge and Wonham (1987)). To cope with marked states, we adapt our plant model as stated in Definition 13.

Definition 13 For plant model $k = (X, L, \longrightarrow, x) \in \mathcal{K}$ and set $X_m \subseteq X$ of *marked* states we add a new label such that $\mathit{marked} \in L(y)$ for each $y \in X_m$, and $\mathit{marked} \notin L(y)$ for each $y \notin X_m$.

For the remainder of this section we assume that each $k \in \mathcal{K}$ is adapted as described in Definition 13. We now introduce two language-based notions in Definition 14 and language-based controllability in Definition 15.

Definition 14 We define the language $\mathcal{L}(k)$ and marked language $\mathcal{L}_m(k)$ of a Kripke-LTS $k = (X, L, \longrightarrow, x) \in \mathcal{K}$ as follows:

$$\begin{aligned}\mathcal{L}(k) &= \{s \in \mathcal{E}^* \mid \exists\, x' \in X : x \xrightarrow{s}{}^* x'\}\\ \mathcal{L}_m(k) &= \{s \in \mathcal{E}^* \mid \exists\, x' \in X : x \xrightarrow{s}{}^* x' \wedge \mathit{marked} \in L(x')\}\end{aligned}$$

In addition we define the language closure $\overline{L}$ of $L \subseteq \mathcal{E}^*$ as:

$$\overline{L} = \{s \in \mathcal{E}^* \mid \exists\, t \in \mathcal{E}^* \,:\, st \in L\}.$$

Definition 15 For languages $L \subseteq \mathcal{E}^*$ and $K \subseteq L$ we say that K is *controllable* with regard to L if for each $s \in K$ and $su \in L$, for $u \in \mathcal{U}$, it holds that $su \in K$.

Parallel composition with unified labels is given in Definition 16. This type of parallel composition is borrowed from process theory (Baeten et al (2010)) and may be used to define the construction between plant and controller in supervisory control.

Definition 16 For $k = (X, L, \longrightarrow, x) \in \mathcal{K}$ and $k' = (X', L', \longrightarrow', x') \in \mathcal{K}$, we define the parallel composition $k' \parallel k$, as follows:

$$k' \parallel k = (X' \times X, L'', \longrightarrow'', (x', x))$$

Where $L''(y', y) = L'(y') \cup L(y)$, for each $y' \in X'$ and $y \in X$, and $(y', y) \xrightarrow{e} {}''(z', z)$ if and only if $y' \xrightarrow{e} {}' z'$ and $y \xrightarrow{e} z$. Clearly $\mathcal{L}(k' \parallel k) \subseteq \mathcal{L}(k)$ and $\mathcal{L}(k' \parallel k) \subseteq \mathcal{L}(k')$.

To express an RW-synthesis problem, we assume a given deterministic plant specification $p \in \mathcal{K}$. We now have to construct $s \in \mathcal{K}$ such that the following properties hold: 1) $\mathcal{L}(p \parallel s)$ is controllable with regard to $\mathcal{L}(p)$; and 2) $p \parallel s$ is non-blocking; that is $\overline{\mathcal{L}(p \parallel s) \cap \mathcal{L}_m(p)} = \mathcal{L}(p \parallel s)$. In Theorem 7 we prove that if we choose $S^1_{p,\Box\,\Diamond\, marked}$ for s these two conditions are satisfied.

Lemma 10 *If $k', k \in \mathcal{K}$ such that $k' \preceq k$ then $\mathcal{L}(k')$ is controllable with regard to $\mathcal{L}(k)$.*

Proof Let $k' = (X', L', \longrightarrow', x')$ and $k = (X, L, \longrightarrow, x)$ and assume $R \subseteq X' \times X$ exists such that $k' \preceq_R k$. We follow Definition 15. Assume that $s \in \mathcal{L}(k')$ and $su \in \mathcal{L}(k)$, for $u \in \mathcal{U}$. Then clearly there exists $y' \in X'$ such that $x'\{\xrightarrow{s}'\}^* y'$ and $y \in X$ such that $x \xrightarrow{s}{}^* y$ and $(y', y) \in R$, by Definition 2. Since $y \xrightarrow{u} z$ exists, then again by Definition 2 there exists a step $y' \xrightarrow{u}' z'$ and thus $su \in \mathcal{L}(k')$. $\square$

For the remainder of this section, we only consider those plant models $p = (X, L, \longrightarrow, x) \in \mathcal{K}$ such that for all $u \in \mathcal{U}^*$ and $y \in X$ and $x \xrightarrow{u}{}^* y$, there exist $v \in \mathcal{E}^*$ and $z \in X$ such that $y \xrightarrow{v}{}^* z$ and $marked \in L(z)$. Otherwise, a solution does not exist.

Lemma 11 *For $k = (X, L, \longrightarrow, x) \in \mathcal{K}$ it holds that $S^1_{k,\Box\,\Diamond\, marked}$ is complete.*

Proof For each $(x, \Box\,\Diamond\, marked) \longrightarrow_0^* (y, \Box\,\Diamond\, marked)$ and $(y, \Box\,\Diamond\, marked) \xrightarrow{e}_0 (z, \Box\,\Diamond\, marked)$ it holds that $(y, \Box\,\Diamond\, marked) \xrightarrow{e}_1 (z, \Box\,\Diamond\, marked)$ if and only if $(z', \Box\,\Diamond\, marked) \uparrow_0 \Diamond\, marked$, for all $u \in \mathcal{U}^*$ and $(z, \Box\,\Diamond\, marked) \xrightarrow{u}_0^* (z', \Box\,\Diamond\, marked)$, as follows from Definition 10. Due to the assumption that for each $u \in \mathcal{U}^*$ and $(x, \Box\,\Diamond\, marked) \xrightarrow{u}_0^* (y, \Box\,\Diamond\, marked)$ it holds that $(y, \Box\,\Diamond\, marked) \uparrow_0 \Diamond\, marked$, we conclude that $S^1_{k,\Box\,\Diamond\, marked}$ is complete.

Theorem 7 *Given $p \in \mathcal{K}$ then $\mathcal{L}(p \parallel S^1_{p,\Box\Diamond marked})$ is controllable with regard to $\mathcal{L}(p)$, and in addition $p \parallel S^1_{p,\Box\Diamond marked}$ is non-blocking.*

Proof By Lemma 11, $S^1_{p,\Box\Diamond marked}$ is complete. Then by Lemma 10 and Theorem 3 it holds that $\mathcal{L}(p \parallel S^1_{p,\Box\Diamond marked})$ is controllable with regard to $\mathcal{L}(p)$. We now consider non-blockingness. Observe that:

$$\overline{\mathcal{L}(p \parallel S^1_{p,\Box\Diamond marked}) \cap \mathcal{L}_m(p)} \subseteq \mathcal{L}(p \parallel S^1_{p,\Box\Diamond marked})$$

already holds, so therefore:

$$\mathcal{L}(p \parallel S^1_{p,\Box\Diamond marked}) \subseteq \overline{\mathcal{L}(p \parallel S^1_{p,\Box\Diamond marked}) \cap \mathcal{L}_m(p)}$$

is what remains to be shown. Assume that $p = (X, L, \longrightarrow, x)$ and $t \in \mathcal{L}(p \parallel S^1_{p,\Box\Diamond marked})$, then $(x, \Box\Diamond marked) \xrightarrow{t}{}^*_1 (y, \Box\Diamond marked)$. Since $(y, \Box\Diamond marked) \uparrow_0 \Diamond marked$ there exists $v \in \mathcal{E}^*$ such that $(y, \Box\Diamond marked) \xrightarrow{v}{}^*_1 (z, \Box\Diamond marked)$ and $marked \in \mathcal{L}(z)$. Then by Definition 8 it holds that $x \xrightarrow{tv}{}^* z$ and therefore $tv \in \mathcal{L}_m(p)$. □

8 Case Study

Controlled system synthesis as achieved by the proposed theories in this paper may be employed in an actual application setting as detailed in this section, where we will model the coordination of maintenance procedures in the printing process of a high-end industrial printer (Markovski et al (2010)), and also in the next section where we will use a case study for automated guided vehicles to analyze scalability. An implementation of the synthesis algorithm was written in the C programming language. Source files for this program and input files for the various analyzed models are available at the following location:

`https://github.com/ahulst/deds`

Several distributed independent components make up the printing process of an industrial printer, as shown in Fig. 10. The main task of the printing process is to apply the toner image onto the toner transfuse belt, followed by the actual printing task where it is fused onto the paper sheet. Preserving printing quality over numerous print jobs involves several maintenance operations. For instance, the toner transfuse belt is jittered periodically, which ensures an even spread of the wear induced by sharp paper edges, occasional printing of completely black pages takes place, thereby removing paper dust, and various techniques are applied to remove coarse toner particles. Maintenance scheduling is mainly based upon the number of prints which have taken place, and maintenance scheduling is therefore related to various strict, as well as, postponable thresholds. If a maintenance operation has to be performed, the printing process has to switch its power mode from **Run** to **Standby**. However, such a power mode switch may actually trigger the activation of other

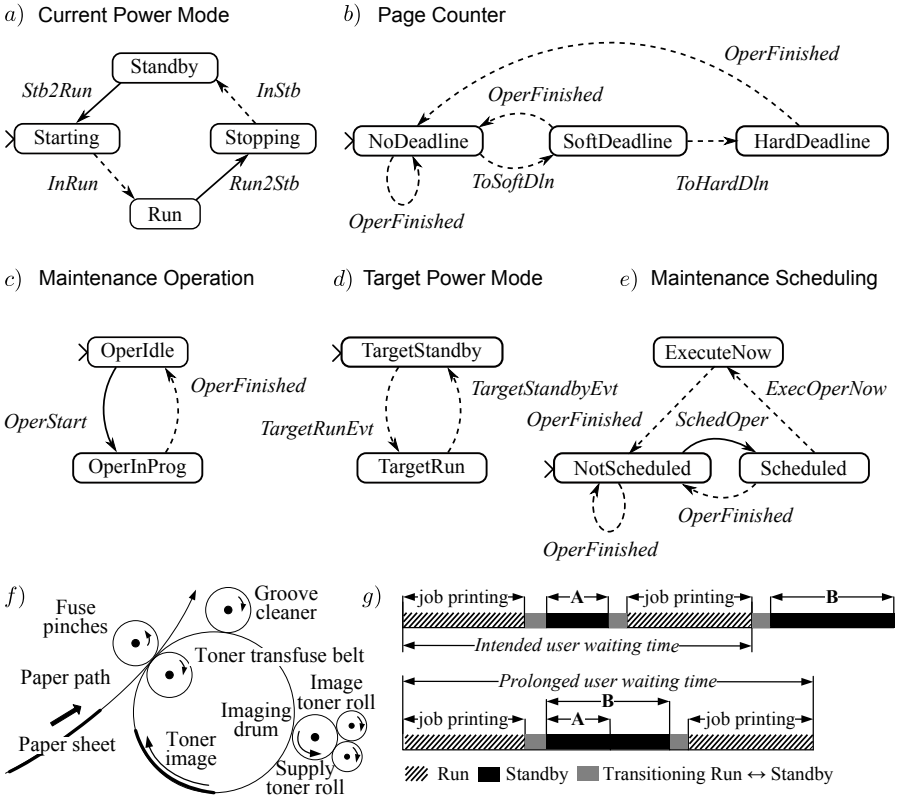

Fig. 10 Five automata shown in Fig. 10a-10e constitute the main components in the maintenance scheduling of an industrial printer. Events *Stb2Run*, *Run2Stb*, *OperStart*, and, *SchedOper* are controllable, while the other events are uncontrollable. Abstracted functionality is shown in Fig. 10f, while scheduling of maintenance tasks is depicted in Fig. 10g.

queued maintenance operations, which may lead to users of the printer having to wait unnecessarily long before the printer becomes usable again.

In Fig. 10g, the occurrence of a non-delayable maintenance operation **A** suspends the current print task, followed by a power mode transition to **Standby**. However, the power mode change triggers the execution of another maintenance operation **B**, having a longer duration than operation **A**. A realistic example of a practical situation where this occurs is when a black image is printed (**A**), taking the exact time required to print a single page, while the significantly longer transfuse belt jittering (**B**) is initiated due to the power mode switch. This combined behavior gives rise to a prolonged user wait time between print jobs, as shown in Fig. 10g.

In this case, we would like to enforce this uncontrolled system to behave in such a way that undesired emergent behaviors does not occur. In addition, all other behavior of the otherwise correctly functioning distributed components which make up the printing process should be left in place. That is, the controlled system affected by the imposed restrictions due to more strin-

gent maintenance coordination should be maximally permissive. The various distributed components which model the printing process are shown in Fig. 10a-10e.

We briefly consider the functionality of the various components depicted in Fig. 10 at an informal level. Note that these system models comprise a natural abstraction in the sense that the scheduling of only one maintenance operation is considered and timing issues are not taken into account. The components Target Power Mode and Maintenance Scheduling execute scheduling tasks. The Current Power Mode, Maintenance Operation and Page Counter components are responsible for handling maintenance tasks and actuating underlying hardware control. The Current Power Mode sets the power mode to **Run** or **Standby**, in reaction to the enabling signals *Stb2Run* and *Run2Stb* respectively. A confirmation is replied by means of *InStb* and *InRun*. The Maintenance Operation component switches between carrying out maintenance, triggered by *OperStart*, or being idle, as confirmed by the *OperFinished* signal. The component Page Counter is responsible for counting the number of printed pages since maintenance has taken place. It sends the signals *ToSoftDln* and *ToHardDln* when soft and hard deadlines are reached. Once the maintenance has finished, the page counter module is reset by receiving the *OperFinished* from the Maintenance Operation component. The controller Target Power Mode defines which mode is requested by the manager by sending the control signals *TargetStandbyEvt* and *TargetRunEvt* respectively. The Maintenance Scheduling receives a request for maintenance via the signal *SchedOper*, which is forwarded to the manager. Confirmation is send by the manager using the *ExecOperNow* event. In addition, it receives feedback from Maintenance Operation to confirm that maintenance has finished in order to reset the scheduling.

As the starting point for synthesis, we construct the synchronized product of these five components. Upon this intermediate result, we apply synthesis for six separate control requirements which are partly based on earlier research done by Markovski et al (2010). Informal and formal interpretations for these control requirements are shown below. We will use $p \Rightarrow q$ as an abbreviation for $\neg p \vee q$ and we use bold face in the formal requirements to indicate state name propositions. For requirements 2-4, we invert $\neg\langle e \rangle$ as $[e]\mathit{false}$.

1. Maintenance operations can be performed only when the printing process is in standby, formalized as:

$$\Box\,(\mathbf{OperInProg} \Rightarrow \mathbf{Standby})$$

2. Maintenance operations can be scheduled only when a soft deadline has been reached, and there are no print jobs in progress, or when a hard deadline has passed, formalized as:

$$\Box\,(\langle \mathit{SchedOper} \rangle \Rightarrow ((\mathbf{SoftDeadline} \wedge \neg \mathbf{TargetRun}) \vee \mathbf{HardDeadline}))$$

3. Maintenance operations can be started only after being scheduled, formalized as:

$$\Box\,(\langle \mathit{OperStart} \rangle \Rightarrow \mathbf{ExecuteNow})$$

4. The power mode of the printing process must follow the power mode dictated by the managers, unless overridden by any pending maintenance operation, formalized using two requirements as:

$$\Box\,(<\mathit{Stb2Run}> \Rightarrow (\mathbf{TargetRun} \wedge \neg\mathbf{ExecuteNow}))$$
$$\Box\,(<\mathit{Run2Stb}> \Rightarrow (\mathbf{TargetStandby} \vee \mathbf{ExecuteNow}))$$

5. After a maintenance operation has finished, the system should be in the TargetStandby state, formalized as:

$$\Box\,[\,\mathit{OperFinished}\,]\,\mathbf{TargetStandby}$$

6. Once a maintenance operation has started, its completion should end immediately such that no new maintenance operation is scheduled and the system is not in the Standby state, formalized as:

$$\Box\,[\,\mathit{OperStart}\,]\,[\,\mathit{OperFinished}\,]\,(\mathbf{NotScheduled} \wedge \neg\mathbf{Standby})$$

A controlled system conforming to control requirements 1-4 may be synthesized using both the traditional event-based supervisory control framework (Ramadge and Wonham (1987)), but also using the theories proposed in this paper. In the first case, the initial state of each model in Fig. 10a-10e needs to be interpreted as a marked state. Both synthesis techniques result in the exact same model consisting of 60 states and 172 transitions. Requirements 5-6 are used to illustrate the extended expressibility for control requirements the theories in this paper provides. Synthesis for requirements 1-6 results in a further restricted controlled system consisting of 56 states and 158 transitions. This case study, albeit small, reflects that the synthesis theory put forward in this paper is able to express earlier research and supersedes earlier work by allowing for more expressive control requirements.

9 Scalability Analysis

The purpose of this section is to analyze the scalability of the algorithm introduced in Section 6 by means of an extendable case study, which is loosely based on the work in Mushtaq (2000). We will compute a control coordination strategy for a variable number of automated guided vehicles (AGVs) and a mixing station within a chemical production plant. Since this model can be instantiated at variable plant sizes and due to the fact that each new AGV introduces separate control objectives, this case is well-suited to analyze the scalability of the type of synthesis presented in this paper.

We will now introduce the case study at hand, as illustrated in Fig. 11. A single mixing station (Fig. 11a) in a chemical production plant is combined with several AGVs. A single instance of the AGV model is shown in Fig. 11b. As shown in Fig. 11b, for each $1 \leq n \leq N$, the label agv_n is added to the connected state in AGV n, if there are N AGVs in the entire model. Synchronization takes place under parallel composition in such a way that the mixing station synchronizes with precisely one AGV over the *connect* event.

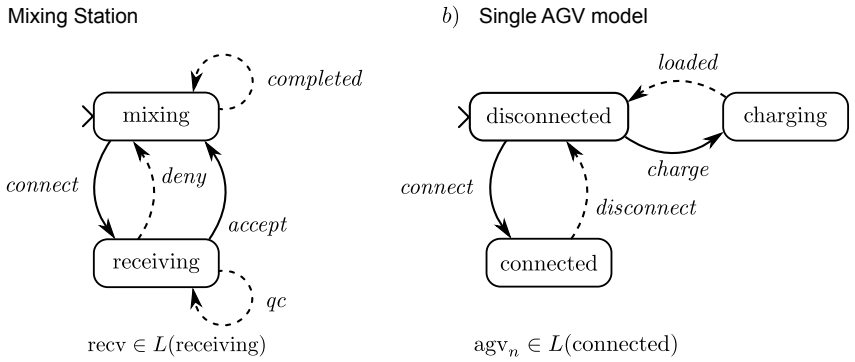

Fig. 11 Components in a chemical production plant for which a guiding control strategy is derived. A single mixing station (Fig. 11a) is combined with multiple instances of the AGV model (Fig. 11b). Models are combined under parallel composition, such that a *connect* event between the mixing station and precisely one AGV is the only synchronizing event.

This is the only point of synchronization in the constructed parallel model. The *connect* event represents the AGV connecting to the station where it delivers a chemical component. The AGV may then disconnect after which it can be ordered to charge its batteries. If the mixing station has received a chemical component from the AGV, it performs quality checks after which the chemical may be either accepted or denied for mixing. The *completed* event represents the situation where the station has completed its task of mixing, after which its mixing tank is supposed to be immediately cleaned. For this purpose, a cleaning agent is again delivered by an AGV.

We intend to synthesize a controlled system for the behavior of both the AGVs and the mixing station, which needs to adhere to several control requirements, as listed below:

1. If there are N AGVs, then after N direct subsequential quality checks at least one AGV should have been disconnected. This requirement ensures that the system does not solely perform quality checks while not sending the AGVs back. We may formalize this control objective as follows:

$$\Box \left(([\, qc \,])^N \left(\bigvee_{1 \leq n \leq N} \neg agv_n \right) \right)$$

2. If the mixing process is finished, as signalled by the *completed* event, cleaning should happen immediately. In this case, AGV_1 should either be already connected or immediately available for connecting, since this is the only AGV which is allowed to deliver the cleaning agent:

$$\Box \, [\, completed \,] \, (agv_1 \vee <connect> agv_1)$$

3. If no AGV is connected and if the mixing staion is not receiving, then at least one AGV should be immediately available for connecting. This may be expressed in terms of the agv_n labels:

$$\Box \left(\left(recv \vee \bigvee_{1 \leq n \leq N} agv_n \right) \vee <connect> \right)$$

The table below shows the computation results for increasing numbers of AGVs in terms of original plant size (as a single $\mathcal{K}$-model, integrated under the aforementioned form of parallel composition), the size of $S^0_{k,f}$ and the size of the final resulting models. All data for model sizes is expressed as S/T, where S refers to the number of states and T refers to the number of transitions. In addition, maximal memory consumption and the number of iterations required to achieve the final synthesis result are mentioned. Note that memory consumption here refers to the maximal number of bytes allocated at some point during the running of the program.

#AGVs	plant (k) size	$S^0_{k,f}$ size	$S^n_{k,f}$ size	memory	iterations
2	18/78	51/236	23/93	2Mb	3
3	54/297	182/1060	105/558	17Mb	3
4	162/1080	633/4406	487/2956	87Mb	3

These results show that the synthesis method in its current form expands fast in terms of memory consumption. A possible improvement may be implemented in terms of on-the-fly computations of the respective transition relations. That is, instead of computing the entire initial model $S^0_{k,f}$ at once, before any transition removal takes place, an improved algorithm may entirely avoid adding certain transitions to $S^0_{k,f}$ if these transitions are due to be removed anyway. For instance, a transition towards a state (y, false) is in the current synthesis setup added to $S^0_{k,f}$ but removed in $S^1_{k,f}$. This requires a more complicated, but possibly also more efficient approach of integrating the construction of the initial model $S^0_{k,f}$ with a partial implementation of the synthesizability test.

We briefly characterize the obtained results in terms of related research. In Wolff et al (2013), the synthesis for a comparable formula consisting of a conjunction of invariants of reachability expressions is considered. An interesting similarity to our research occurs when synthesis for variable-size plant models in the orders of tens of states results in large (up to several gigabytest) state models which remain computable in minutes. Work in Ehlers et al (2014) also shows exponential growth of the controller in terms of the size of the synthesized formula. Many comparable works indicate exponential growth of both the resulting model and running time once either the plant model or the synthesized formula expands linearly (Arnold et al (2003); Kupferman et al (2000); Fabian and Lennartson (1997)). Different results were obtained as well. D'Ippolito et al (2010) and Pralet et al (2010) do not observe such strong increases in memory consumption, although in the first case this might be partly due to the specific approach applied in D'Ippolito et al (2010). An offset in results for synthesis for formulas in temporal logic may be noticed based on work in Su (2008), where abstractions due to non-determinism result in effective plant models having significantly smaller state spaces. Based on these observations, the hypothesis arises that if a state space reduction via

abstraction through the introduction of non-determinism preceeds a memory demanding computation, the net result may be more efficient compared to observations of the synthesis problem in itself.

10 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 controlled behavior, expressed as a formula in modal logic. The relationship between the synthesis result and the original plant specification adheres to important notions in control synthesis: controllability and maximal permissiveness. The controlled behavior 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 which may be straightforwardly implemented. The synthesis theory presented in this paper allows the full expressibility of Ramadge-Wonham control synthesis on deterministic plant models.

An implementation of the synthesis technique has been developed and analyzed in this paper. We will assess various parameters regarding tractability and efficiency of the proposed algorithm, or improvements thereof, in further case studies. In particular, the synthesizability of reachability formulas may be modified such that recomputations can be avoided, using dynamic programming or memoization techniques. To enable a clear focus on the theoretical results as presented here, we did not include an embedding of such optimizations in the synthesis theory. Partial bisimulation as a means to express controllability, as well as other behavioral preorders for this purpose, will also be studied further. We also intend to compare the efficiency of the constructed implementation to other toolsets, in particular UPPAAL Tiga.

Acknowledgments

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

References

Alberucci L, Facchini A (2009) On modal μ-calculus and Gödel-Löb logic. Studia Logica 91(2):145–169

Antoniotti M, Mishra B (1995) Discrete event models+temporal logic=supervisory controller: Automatic synthesis of locomotion controllers. In: Proceedings of Robotics and Automation, IEEE, pp 1441–1446

Arnold A, Walukiewicz I (2008) Nondeterministic controllers of nondeterministic processes. In: Proceedings of Logic and Automata, Amsterdam University Press, pp 29–52

Arnold A, Vincent A, Walukiewicz I (2003) Games for synthesis of controllers with partial observation. Theoretical Computer Science 303(1):7–34

Baeten J, Basten T, Reniers M (2010) Process algebra: equational theories of communicating processes. Cambridge University Press

Basu S, Kumar R (2007) Quotient-based control synthesis for partially observed non-deterministic plants with mu-calculus specifications. In: Proceedings of CDC, IEEE, pp 5294–5299

Bull R, Segerberg K (2001) Basic modal logic. In: Handbook of philosophical logic, Springer, pp 1–81

Cassandras C, Lafortune S (2008) Introduction to discrete event systems. Springer

Cleaveland R, Steffen B (1993) A linear-time model-checking algorithm for the alternation-free modal mu-calculus. Formal Methods in System Design 2(2):121–147

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

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

Ehlers R, Lafortune S, Tripakis S, Vardi M (2014) Bridging the gap between supervisory control and reactive synthesis: case of full observation and centralized control. In: Proceedings of WODES, IFAC, pp 222–227

Fabian M, Lennartson B (1997) A class of non-deterministic specifications for supervisory control. European Journal of Control 3(1):81–90

Fainekos G, Girard A, Pappas G (2007) Hierarchical synthesis of hybrid controllers from temporal logic specifications. In: Proceedings of HSCC, Springer, pp 203–216

van Glabbeek R (1993) The linear time branching time spectrum II. In: Proceedings of CONCUR, Springer, pp 66–81

Havelund K, Larsen K, Skou A (1999) Formal verification of a power controller using the real-time model checker UPPAAL. In: Proceedings of AMAST, Springer, pp 277–298

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

van Hulst A, Reniers M, Fokkink W (2014) Maximal synthesis for Hennessy-Milner logic. ACM Transactions on Embedded Computing 14(1):10:1–10:21

van Hulst A, Reniers M, Fokkink W (2015) Maximally permissive controlled system synthesis for modal logic. In: Proceedings of SOFSEM, Springer, vol 8939, pp 230–241

Jansen S (2014) Design and implementation of model-based controllers for baggage handling systems. TU/e Master Thesis Archive at http://www.tue.nl

Jessen J, Rasmussen J, Larsen K, David A (2007) Guided controller synthesis for climate controller using UPPAAL Tiga. In: Proceedings of FORMATS, Springer, vol 4763, pp 227–240

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

Kamphuis R (2013) Design and real-time implementation of a supervisory controller for a part of Veghel airport. TU/e Master Thesis Archive at http://www.tue.nl

Kumar R, Shayman M (1994) Non-blocking supervisory control of nondeterministic discrete event systems. In: Proceedings of ACC, IEEE, pp 1089–1093

Kupferman O, Vardi M (2000) μ-calculus synthesis. In: Proceedings of MFCS, Springer, pp 497–507

Kupferman O, Madhusudan P, Thiagarajan P, Vardi M (2000) Open systems in reactive environments: control and synthesis. In: Proceedings of CONCUR, Springer, pp 92–107

Markovski J (2011) A process-theoretic approach to supervisory control theory. In: Proceedings of ACC, IEEE, pp 4496–4501

Markovski J, Jacobs K, van Beek D, Somers L, Rooda J (2010) Coordination of resources using generalized state-based requirements. In: Proceedings of WODES, pp 287–292

McMillan K (1993) Symbolic model checking. Springer

Moor T, Davoren J (2001) Robust controller synthesis for hybrid systems using modal logic. In: Proceedings of HSCC, Springer, pp 433–446

Mushtaq F (2000) The safe design of computer controlled pipeless batch plants. PhD thesis, Loughborough University

Ostroff J (1989) Synthesis of controllers for real-time discrete event systems. In: Proceedings of CDC, IEEE, pp 138–144

Pinchinat S (2005) You can always compute maximally permissive controllers under partial observation when they exist. In: Proceedings of ACC, IEEE, pp 2287–2292

Pinchinat S, Raclet J (2005) Supervisory control problems for nondeterministic discrete-event systems: a logical approach. In: Proceedings of the IFAC world congress, Elsevier, vol 16, pp 295–301

Pinchinat S, Riedweg S (2003) Quantified mu-calculus for control synthesis. In: Proceedings of MFCS, Springer, pp 642–651

Pnueli A, Rosner R (1989) On the synthesis of a reactive module. In: Proceedings of POPL, ACM, pp 179–190

Pralet C, Verfaillie G, Lemaître M, Infantes G (2010) Constraint-based controller synthesis in non-deterministic and partially observable domains. In: Proceedings of ECAI, IOS Press, vol 215, pp 681–686

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

Rutten J (2000) Coalgebra, concurrency, and control. In: Discrete Event Systems: Analysis and Control, Springer, pp 31–38

Sokolsky O, Smolka S (1994) Incremental model checking in the modal mu-calculus. In: Proceedings of CAV, Springer, pp 351–363

Su R (2008) Supervisor synthesis based on abstractions of nondeterministic automata. In: Proceedings of WODES, IEEE, pp 412–418

Wolff E, Topcu U, Murray R (2013) Efficient reactive controller synthesis for a fragment of linear temporal logic. In: Proceedings of ICRA, IEEE, pp 5033–5040

Ziller R, Schneider K (2005) Combining supervisor synthesis and model checking. ACM Transactions on Embedded Computing Systems 4(2):331–362