Compositional Control Synthesis for Partially Observable Systems

Wouter Kuijper and Jaco van de Pol

University of Twente Formal Methods and Tools
Dept. of EEMCS
{W.Kuijper**, J.C.vandePol}@ewi.utwente.nl

Abstract. We present a compositional method for deriving control constraints on a network of interconnected, partially observable and partially controllable plant components. The constraint derivation method works in conjunction with an antichain–based, symbolic algorithm for computing weakest strategies in safety games of imperfect information. We demonstrate how the technique allows a reactive controller to be synthesized in an incremental manner, exploiting locality and independence in the problem specification.

1 Introduction

Control Synthesis [23] is the idea of automatically synthesizing a controller for enforcing some desired behaviour in a plant. This problem can be phrased logically as follows. Given a plant description P and a desired property ϕ, construct a controller C such that $P \| C \vDash \phi$. Control synthesis is a close cousin of the model checking problem. Where model checking is about establishing whether or not a model supports a given property, control synthesis is about generating a model on which the property will hold.

The main difficulty that any effective procedure for controller synthesis must face is that the uncontrolled state space generated by the plant description is typically large. This is mainly due to concurrency in the model, which is a central issue also in model checking. However, for synthesis the problem is amplified by two additional, complicating factors. First, we typically see a higher degree of non–determinism because a priori no control constraints are given. Second, it is often the case that the state of the plant P is only partially observable for the controller C. Resolving this may incur another exponential blowup. On instances, this blowup may be avoided by using smart, symbolic methods [26].

Contribution In this paper we focus on the compositional synthesis of a reactive controller under the assumption of partial observability. Our main contributions are a compositional framework for describing control synthesis problems as a network of interconnected, partially controllable, partially observable plant components, and a compositional method for synthesizing safety controllers over such a plant model.

We believe there are at least two novel aspects to our approach. First, there is the combination of imperfect information with compositionality. In particular, we make

** This author is supported by NWO project 600.065.120.24N20

sure that context assumptions take into account partial observability of the components. Second, our framework ensures context assumptions gradually shift in the direction of control constraints as the scope widens. In this way we avoid, to some extent, unrealistic assumptions, and generally obtain a less permissive context. Note that the size of the context assumptions is an important factor [19] in the efficiency of assume–guarantee based methods.

This work is complementary to our work in [13]. The game solving algorithm we present there is designed to be applied in a compositional setting. It is an efficient counterexample driven algorithm for computing sparse context assumptions. As such, it is especially suitable for the higher levels in the composition hierarchy where abstraction is often possible (and useful). However, its succesful application hinges on the fact that control constraints local to any of the considered subcomponents should already be incorporated into the game board. Here we show how this can be achieved by proceeding compositionally.

Related Work Synthesis of reactive systems was first considered by Church [10] who suggested the problem of finding a set of restricted recursion equivalences mapping an input signal to an output signal satisfying a given requirement [25]. The classical solutions to Church's Problem [4, 22] in principle solve the synthesis problem for omega–regular specifications. Since then, much of the subsequent work has focussed on extending these results to richer classes of properties and systems, and on making synthesis more scalable.

Pioneering work on synthesis of *closed* reactive systems [18, 11] uses a reduction to the satisfiability of a temporal logic formula. That it is also possible to synthesize *open* reactive systems is shown [20, 21] using a reduction to the satisfiability of a CTL* formula, where path operators force alternation between the system and the environment. Around the same time, another branch of work [23, 16] considers the synthesis problem specifically in the context of control of discrete event systems, this introduces many important control theoretical concepts, such as *observability*, *controllability*, and the notion of a *control hierarchy*.

More recently several contributions have widened the scope of the field and at the same time addressed several scalabilty issues. Symbolic methods, already proven succesful in a verification setting, can be applied also for synthesis [2]. Symbolic techniques also enable synthesis for hybrid systems which incorporate continuous as well as discrete behaviour [1]. Controller synthesis under partial information can be done by a reduction to the emptyness of an alternating tree automaton [15]. This method is very general and works in branching and linear settings. However, scalability issues remain as the authors note that most of the combinatorial complexity is shifted to the emptyness check for the alternating tree automaton. In [14] a compositional synthesis method is presented that reduces the synthesis problem to the emptyness of a non–deterministic Büchi tree automaton. For the specific case of hard real time systems the full expressive power of omega regular languages may not be necessary [17], since a bounded response requirement can be expressed as a safety property.

Even the earliest solutions to Church's problem, essentially involve solving a game between the environment and the control [24]. As such there is a need to study the (sym-

bolic) representation and manipulation of games and strategies as first class citizens. In [7] a symbolic algorithm for games with imperfect information is developed based on fixed point iteration over antichains. In [5] there are efficient on–the–fly algorithms for solving games of imperfect information.

Compositionality adds another dimension to the synthesis problem: in order to obtain more scalable solutions it is desirable to solve the synthesis problem in an incremental manner, treating first subproblems in isolation before combining their results. In general this requires a form of assume–guarantee reasoning. There exists an important line of related work that addresses such issues.

One such recent development that aims to deal with component based designs are interface automata [12]. This work introduces *interfaces* as the weakest behavioural assumptions/guarantees between components. A synchronous, bidirectional interface model is presented in [6]. Our component model is similar, but differs on the in–/output partition of the variables to be able to handle partially observable systems. Besides providing a clean theory of important software engineering concepts like *incremental design* and *independent implementability*, interfaces also have nice algorithmic properties allowing for instance automated refinement and compatibility checking. Several algorithms for interface synthesis are discussed in [3].

Authors in [9] describe a co–synthesis method based on assume–guarantee reasoning. Their solution is interesting in that it addresses non–zero–sum games where processes compete, but not at all cost. In [8] the same authors explore ways in which to compute evironment assumptions for specifications involving liveness properties, removal of unsafe transitions constitutes a pre–processing step. Note that, for a liveness property, in general, there does not exist a unique weakest assumption on the context, so this is a non–trivial problem.

Structure The rest of the paper is structured as follows. In Section 2 we build up a game–theoretic, semantical framework for control synthesis problems. We also present our running example used to illustrate all the definitions. In Section 3 we propose a sound and complete compositional method for control synthesis. We demonstrate the method on the running example. In Section 4 we conclude with a summary of the contribution, and perspectives on future work.

2 Compositional Framework

In this section we give a formal semantics for our central object of interest, which is the *plant under control*, or PuC. First we give an example.

A Motivating Example We consider the *parcel plant* illustrated in Figure 1. This fictive plant consists of a *feeder* and two *stamps* connected together with a conveyor belt. A parcel is fed onto the belt by the feeder. The belt transports the parcel over to stamp 1 which prints a tracking code. The belt then transports the parcel over to stamp 2 which stamps the shipping company's logo. Next to the illustration in Figure 1 we list all the *propositions* we used to model this particular example. For each proposition we

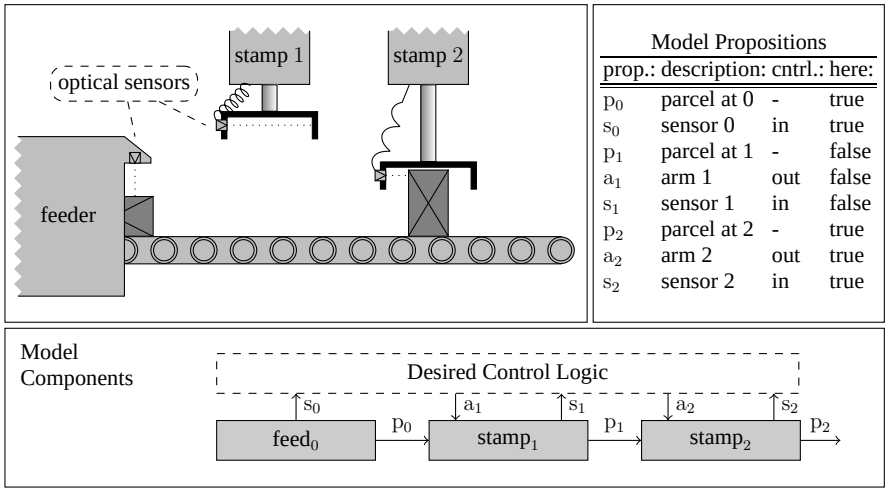

Fig. 1. A modular parcel stamping plant. (top left) The legend (top right) shows all the propositions we used to model this example. The decomposition structure (bottom) shows the components we identified in the model and their interconnections, in terms of input and output propositions.

indicate whether it is a control output/input, and whether or not it holds for the current state of the plant as it is shown in the picture. We specify the behaviour of the three components in the parcel stamp using Kripke structures over these atomic propositions in Figure 2.

Definition 1 (Kripke Structures) We let $\mathcal{X}$ be a background set of *propositions*. For a given (finite) subset $X \subseteq \mathcal{X}$ of propositions we define $\mathcal{S}[X] = 2^X$ as the set of *states*, or *valuations* over X. We define shorthand $\mathcal{S} = \mathcal{S}[\mathcal{X}]$. A *Kripke structure* is a tuple $A = (L, X, \gamma, \delta, L^{\text{init}})$, consisting of a set of *locations* L, a finite set of *relevant propositions* $X \subseteq \mathcal{X}$, a *propositional labeling* $\gamma : L \to \mathcal{S}[X]$, a *transition relation* $\delta \subseteq L \times L$, and a set of *initial locations* $L^{\text{init}} \subseteq L$. For any two Kripke structures A_1, A_2 the *composition* $A_{12} = A_1 \| A_2$ is defined with $L_{12} = \{(\ell_1, \ell_2) \in L_1 \times L_2 \mid \gamma_1(\ell_1) \cap X_2 = \gamma_2(\ell_2) \cap X_1\}$, $X_{12} = X_1 \cup X_2$, for all $(\ell_1, \ell_2) \in L_{12}$ it holds $\gamma_{12}(\ell_1, \ell_2) = \gamma_1(\ell_1) \cup \gamma_2(\ell_2)$, for all $(\ell_1, \ell_2), (\ell_1', \ell_2') \in L_{12}$ it holds $(\ell_1, \ell_2)\delta_{12}(\ell_1', \ell_2')$ iff $\ell_1 \delta_1 \ell_1'$ and $\ell_2 \delta_2 \ell_2'$, and, for the initial locations, it holds $L_{12}^{\text{init}} = (L_1^{\text{init}} \times L_2^{\text{init}}) \cap L_{12}$. Note that L_{12} contains all the pairs of locations in the Kripke structures that are *consistent*, meaning that they agree on the truth of all *shared propositions* $X_1 \cap X_2$. $\lhd$

We note that, in our notation, we use a horizontal bar to denote the negation of a proposition letter, i.e.: $\overline{\mathrm{a}_1}$ should be read as *not* a_1 which, in turn, is interpreted as *stamp number 1 has not activated its stamping arm.* Next, we note that the models for the stamps contain deadlock locations. We use this to encode a *safety property* into the model. The safety property here simply says that the stamp *must* activate whenever there is a parcel present on the belt, otherwise the system will deadlock.

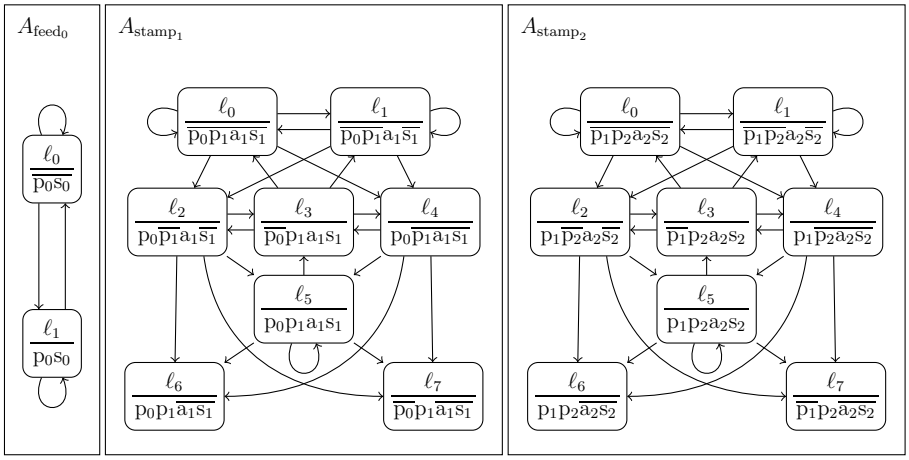

Fig. 2. Kripke structures modeling the feeder (left), stamp 1 (center), and stamp 2 (right).

Game Semantics We now turn to assigning a game semantics to the type of plant models we have just introduced. We start with some prerequisites.

Definition 2 (Strategy) A *strategy* $f : \mathcal{S}^* \to 2^{\mathcal{S}}$ is a function mapping *histories* of states to the sets of allowed successor states. With $f_\top$ we denote the strategy that maps all histories to $\mathcal{S}$. A *trace* $\sigma = s_0 \dots s_n \in \mathcal{S}^+$ is *consistent* with f iff for all $0 \leq i \leq n$ it holds $f(s_0 \dots s_{i-1}) \ni s_i$. With $\mathsf{Reach}(f)$ we denote the set of traces consistent with f. We require strategies to be *prefix–closed* meaning that $f(\sigma) \neq \varnothing$ and $\sigma \neq \epsilon$ implies $\sigma \in \mathsf{Reach}(f)$. A strategy f is *safe* iff for all $\sigma \in \mathsf{Reach}(f)$ it holds $f(\sigma) \neq \varnothing$. With Safe we denote the set of all safe strategies. $\lhd$

Any given Kripke structure can be viewed as a strategy by considering the restriction that the Kripke structure places on the next state given the history of states that came before.

Definition 3 (Regular Strategies) To a given Krikpke structure A, we assign the strategy $[\![A]\!] : \mathcal{S}^* \to 2^{\mathcal{S}}$ such that for the empty trace ϵ it holds $s \in [\![A]\!](\epsilon)$ iff there exists an initial location $\ell_0 \in L_A^{\text{init}}$ such that ℓ_0 is consistent with s, and for a given history $\sigma = s_0 \dots s_n \in \mathcal{S}^+$ it holds $s' \in [\![A]\!](\sigma)$ iff there exists a computation $\ell_0 \dots \ell_{n+1}$ in the Kripke structure such that $\ell_0 \in L_A^{\text{init}}$, each location ℓ_i for $i \leq n$ is consistent with s_i, and s' is consistent with ℓ_{n+1}. A strategy $f \in \mathcal{F}$ is regular iff there exists a finite Kripke structure A such that $f = [\![A]\!]$. $\lhd$

It is often important that Kripke structures are input enabled for a subset of the propositions $X^{\text{i}} \subseteq X$ as is the case for the Kripke structures we defined for the parcel stamp. In practice this is easy to enforce using some syntactic criteria on the specification. On a semantic level we prefer to abstract from the particular way the strategy is represented. For this reason we introduce the notion of *permissibility*.

Definition 4 (Permissibility) For a given set of propositions $X \subseteq \mathcal{X}$ we define the *indistinguishability relation* $\sim_X \subseteq \mathcal{S} \times \mathcal{S}$ such that $s \sim_X s'$ iff $s \cap X = s' \cap X$, we lift $\sim_X$ to *traces* of states $\sigma = s_1 \dots s_n \in \mathcal{S}^*$ and $\sigma' = s'_1 \dots s'_m \in \mathcal{S}^*$ such that $\sigma \sim_X \sigma'$ iff $n = m$ and for all $0 < i \leq n$ it holds $s_i \sim_X s'_i$. A *signature* is a pair $[X^{\mathrm{i}} \to X^{\mathrm{o}}]$ such that $X^{\mathrm{i}} \subseteq^{\mathrm{finite}} \mathcal{X}$ and $X^{\mathrm{o}} \subseteq^{\mathrm{finite}} \mathcal{X}$, we let $X^{\mathrm{io}} = X^{\mathrm{i}} \cup X^{\mathrm{o}}$. A strategy $f \in \mathcal{F}$ is *permissible* for a signature $[X^{\mathrm{i}} \to X^{\mathrm{o}}]$ iff for all $\sigma, \sigma' \in \mathcal{S}^*$ such that $\sigma \sim_{X^{\mathrm{io}}} \sigma'$ and all $s, s' \in \mathcal{S}$ such that $s \sim_{X^{\mathrm{o}}} s'$ we have $s \in f(\sigma)$ iff $s' \in f(\sigma')$. With $\mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$ we denote the set of strategies that are permissible for $[X^{\mathrm{i}} \to X^{\mathrm{o}}]$. $\lhd$

Note that, for any Kripke structure, it holds $[\![A]\!] \in \mathcal{F}[X_A \to X_A]$. To illustrate, we can now formalize a suitable notion of *input enabledness* for Kripke structures in terms of permissibility. We say that A is *input enabled* for a subset of the relevant propositions $X^{\mathrm{i}} \subset X_A$ iff $[\![A]\!] \in \mathcal{F}[X^{\mathrm{i}} \to X_A \setminus X^{\mathrm{i}}]$.

Definition 5 (Lattice of Strategies) We fix a partial order $\sqsubseteq$ on the set of strategies such that $f \sqsubseteq f'$ iff for all $\sigma \in \mathcal{S}^*$ it holds $f(\sigma) \subseteq f'(\sigma)$ we say f' is *more permissive* or *weaker* than f. The set of strategies ordered by permissiveness forms a complete lattice. The *join* of a set of strategies $F \subseteq \mathcal{F}$ is denoted $\sqcup F$ and is defined as $f \in \mathcal{F}$ such that for all $\sigma \in \mathcal{S}^*$ it holds $f(\sigma) = \cup_{f' \in F} f'(\sigma)$. The *meet* is denoted $\sqcap F$ and is defined dually. $\lhd$

We have now all the prerequisites to introduce the concept of a *plant under control* or PuC. A PuC is a semantic object that encodes the behaviour of a system of interacting plant components. In addition it also specifies a set of control localities which are selected subsets of plant components that are of special interest because of their control dependencies. For each such control locality the PuC encodes context assumptions and control constraints. Later we will show how these assumptions and constraints can be automatically derived by solving games.

Definition 6 (Plant under Control) A *plant under control (PuC)* M is a tuple

$$M = (P, \{f_p, [X^{\mathrm{i}}_p \to X^{\mathrm{o}}_p]\}_{p \in P}, C, \{g_K, h_K\}_{K \in C})$$

Consisting of a finite set of *plant components* P, for each plant component $p \in P$ the PuC represents the *component behaviour* as the strategy $f_p \in \mathcal{F}[X^{\mathrm{i}}_p \to X^{\mathrm{o}}_p]$. We require for all $p_1, p_2 \in P$ such that $p_1 \neq p_2$ it holds $X^{\mathrm{i}}_{p_1} \cap X^{\mathrm{i}}_{p_2} = X^{\mathrm{o}}_{p_1} \cap X^{\mathrm{o}}_{p_2} = \varnothing$. The PuC has a selected set of *control localities* $C \subseteq 2^P$ such that $P \in C$. For a given control locality $K \in C$ we define $X^{\mathrm{i}}_K = \cup_{p \in K} X^{\mathrm{i}}_p$, and $X^{\mathrm{o}}_K = \cup_{p \in K} X^{\mathrm{o}}_p$, and $f_K = \sqcap_{p \in K} f_p$. We define the *control signature* $[X^{\mathrm{ci}} \to X^{\mathrm{co}}]$ such that $X^{\mathrm{ci}} = X^{\mathrm{o}}_P \setminus X^{\mathrm{i}}_P$ and $X^{\mathrm{co}} = X^{\mathrm{i}}_P \setminus X^{\mathrm{o}}_P$. For each control locality $K \in C$, the PuC represents the current *context assumptions* as the strategy $g_K \in \mathcal{F}[X^{\mathrm{o}}_K \setminus X^{\mathrm{i}}_K \to X^{\mathrm{i}}_K \setminus X^{\mathrm{o}}_K]$, and the current *control constraints* as the strategy $h_K \in \mathcal{F}[X^{\mathrm{o}}_K \cap X^{\mathrm{ci}} \to X^{\mathrm{i}}_K \cap X^{\mathrm{co}}]$. $\lhd$

In this definition we are assuming a set of interacting plant components that communicate to each other and to the controller by means of their signature of input/output propositions. If a proposition is both input and output to the same component we say it is *internal* to the plant component. The definition ensures that no other plant component may synchronize on such an internal proposition. We assume that all non–internal

propositions that are not used for synchronization among plant components are control propositions (open plant output propositions are control input propositions, and open plant input propositions are control output propositions).

Note that we are assuming a *given* set of control localities. This information should be added to an existing componentized model of the plant. Either manually or by looking for interesting clusters of plant components that have some mutual dependencies. Such an automatic clustering approach has already been investigated for compostinal verification in [19].

Example 1 (Parcel Stamp) We define a PuC M_{parcel} for the parcel stamp example. We first fix the plant components $P_{\mathrm{parcel}} = \{\mathrm{feed}_0, \mathrm{stamp}_1, \mathrm{stamp}_2\}$. Their signatures are $X^{\mathrm{o}}_{\mathrm{feed}_0} = \{\mathrm{p}_0, \mathrm{s}_0\}$, $X^{\mathrm{i}}_{\mathrm{feed}_0} = \varnothing$, $X^{\mathrm{o}}_{\mathrm{stamp}_1} = \{\mathrm{p}_1, \mathrm{s}_1\}$, $X^{\mathrm{i}}_{\mathrm{stamp}_1} = \{\mathrm{p}_0, \mathrm{a}_1\}$, $X^{\mathrm{o}}_{\mathrm{stamp}_2} = \{\mathrm{p}_2, \mathrm{s}_2\}$, $X^{\mathrm{i}}_{\mathrm{stamp}_2} = \{\mathrm{p}_1, \mathrm{a}_2, \mathrm{p}_2\}$, Note that we make p_2 an internal variable of stamp_2 since it is not input to any other component, in this way the control signature becomes $X^{\mathrm{ci}} = \{\mathrm{s}_0, \mathrm{s}_1, \mathrm{s}_0\}$ and $X^{\mathrm{co}} = \{\mathrm{a}_1, \mathrm{a}_2\}$. The component behaviour is given by the Kripke structures in Figure 2, $f_{\mathrm{feed}_0} = [\![A_{\mathrm{feed}_0}]\!]$, and $f_{\mathrm{stamp}_1} = [\![A_{\mathrm{stamp}_1}]\!]$, and $f_{\mathrm{stamp}_2} = [\![A_{\mathrm{stamp}_2}]\!]$. We define the control localities

$$\begin{aligned} C_{\mathrm{parcel}} = \{&\{\mathrm{feed}_0\}, \{\mathrm{stamp}_1\}, \{\mathrm{stamp}_2\}, \\ &\{\mathrm{feed}_0, \mathrm{stamp}_1\}, \{\mathrm{stamp}_1, \mathrm{stamp}_2\}, \\ &\{\mathrm{feed}_0, \mathrm{stamp}_1, \mathrm{stamp}_2\}\} \end{aligned}$$

The context assumptions g_K and control guarantees h_K for each locality $K \in C$ are initially set to the vacuous strategy $g_K = h_K = f_\top$. ◁

Global Control Constraints For a given PuC M we are interested in computing the weakest global control constraints $\hat{h}_P$ such that $f_P \sqcap \hat{h}_P \in \mathsf{Safe}$. In principle this can be done by viewing the PuC as a safety game of imperfect information where the safety player may, at each turn, observe the value of the control input propositions and determine the value of the control output propositions. In this way we obtain a game graph that can be solved using conventional game solving algorithms. The result will be the weakest strategy $\hat{h}_P$ for the safety player.

Definition 7 (Weakest Safe Global Strategy) For a given PuC M we define the *weakest safe global control constraints* $\hat{h}_P$ as follows

$$\hat{h}_P = \sqcup\{h \in \mathcal{F}[X^{\mathrm{ci}} \to X^{\mathrm{co}}] \mid f_P \sqcap h \in \mathsf{Safe}\}$$

i.e. the weakest global control strategy that is sufficient to keep the system safe. ◁

Computing $\hat{h}_P$ directly by solving the global safety game does not scale very well to larger systems. For this reason we want to proceed compositionally and start with smaller control localities $K \in C$ such that $K \subset P$ before treating the plant $P \in C$ as a whole. As it turns out solving the local safety game over the control signature $[X^{\mathrm{o}}_K \cap X^{\mathrm{ci}} \to X^{\mathrm{i}}_K \cap X^{\mathrm{co}}]$ will yield control constraints that are too strong in the sense that not every possible safe control solution $h_P \in \mathcal{F}[X^{\mathrm{ci}} \to X^{\mathrm{co}}]$ on the global level will be preserved. In Example 2 we illustrate this phenomenon.

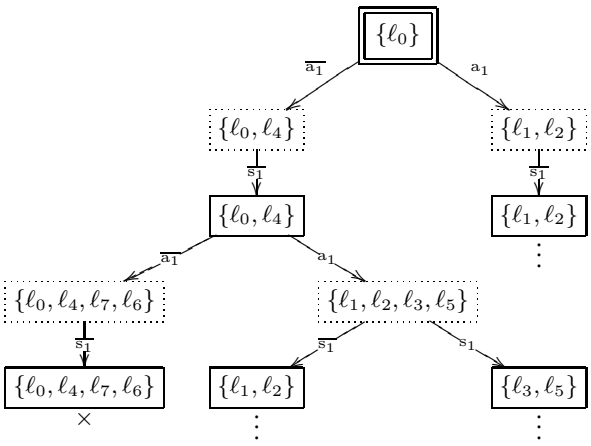

Fig. 3. Partial game tree for A_{stamp_1}, where the safety player is restricted to use the control signature $[X^{\mathrm{o}}_{\mathrm{stamp}_1} \cap X^{\mathrm{ci}} \to X^{\mathrm{i}}_{\mathrm{stamp}_1} \cap X^{\mathrm{co}}] = [\{\mathrm{s}_1\} \to \{\mathrm{a}_1\}]$.

Example 2 (Overrestrictive Control) In Figure 3 we show a partial unravelling of the game board A_{stamp_1} into a *game tree*. The nodes in the game tree are partitioned into nodes for the *safety player* shown as solid boxes and nodes for the *reachability player* shown as dotted boxes. The nodes for the safety player are annotated with the *knowledge* or *information set* that the safety player has given the observation history. The nodes for the reachability player are labeled with the *forcing sets* which are all locations to which there exists a trace that is consistent with the observation history *and* the control output as chosen by the safety player. From a forcing set, the reachability player fixes the control input by choosing one of the locations in the forcing set. Note that the subset construction does not show the concrete successor locations. Rather it shows the resulting information set for the safety player, which is the smallest set of locations that are consistent with the input/output that has been exchanged.

As can be seen the safety player is forced, after 1 iteration, to always play a_1. Meaning that, she is *always* activating the stamp. She cannot, based on her observations, determine for sure whether or not there is a parcel present. Note however, if we would have taken also the feeder component A_{feed_0} into account, it would have been possible for the safety player to deduce this information based on the sensor in the feeder. So the strategy for A_{stamp_1} that we got from solving this game does not respect the strategy for $A_{\mathrm{feed}_1} \| A_{\mathrm{stamp}_1}$ which activates only if the optical sensor in the feeder is triggered. $\lhd$

3 Compositional Synthesis Method

Our solution approach to the problems sketched in the previous section is based on an over approximation of the allowable behaviour followed by an under approximation of

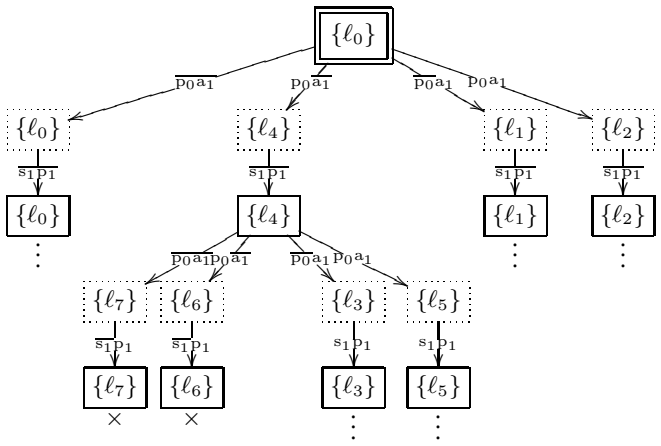

Fig. 4. Partial game tree for A_{stamp_1}, where the safety player is allowed to use the control signature $[X^{\mathrm{o}}_{\mathrm{stamp}_1} \setminus X^{\mathrm{i}}_{\mathrm{stamp}_1} \to X^{\mathrm{i}}_{\mathrm{stamp}_1} \cap X^{\mathrm{o}}_{\mathrm{stamp}_1}] = [\{\mathrm{s}_1, \mathrm{p}_1\} \to \{\mathrm{p}_0, \mathrm{a}_1\}]$.

the denyable behaviour. The soundness of our approach rests on the notion of *conservativity*.

Definition 8 (Conservative Systems) A PuC is *conservatively constrained* iff for all $K \in C$ both the local assumptions g_K and the local constraints h_K are *conservative*, meaning that $f_P \sqcap \hat{h}_P \sqsubseteq g_K$ and $f_P \sqcap \hat{h}_P \sqsubseteq h_K$, i.e.: both the local assumptions and the local constraints allow all the behaviour that would be allowed by the weakest safe global control constraints. A system that is not conservatively constrained is *over constrained*. ◁

For a conservatively constrained PuC we may always take into account the existing control constraints and context assumptions while computing new control constraint or context assumptions. This unlocks possibilities that allows more efficient symbolic game solving. For larger systems there may exist control localities that need highly non–trivial context assumptions which require a lot of computation time and storage space. This problem is sometimes referred to as the problem of *assumption explosion*.

To prevent this we rely on two mechanisms. The first is the fact that the signature for the context assumptions g_K tends to the signature for the control constraints h_K as K approaches P. Note that, at the highest level of composition, P, the signatures for the control constraints and the context assumptions coincide. This means that context assumptions become more and more like control constraints as we progress upward in the decomposition hierarchy C. The second mechanism we rely on is a synergistic relation between context assumptions and control constraints. In particular, for conservative systems, it is possible while computing weakest context assumptions for a control locality $K \in C$ to take into account the conservative control constraints of all lower control

localities $K' \subset K$ that have been previously computed. We refer to this as *subordinate* control.

Definition 9 (Conservative Local Context Assumptions) For a given PuC M and control locality $K \in C$ we define the *subordinate localities* $K{\downarrow} = \{K' \in C \mid K' \subset K\}$, and the *subordinate control* $h_{K\downarrow} = \sqcap_{K' \in K\downarrow} h_{K'}$. Now $\mathsf{WeakestContext}_K(M) = M'$ such that

$$g'_K = \sqcup\{g' \in \mathcal{F}[X^{\mathrm{o}}_K \setminus X^{\mathrm{i}}_K \to X^{\mathrm{i}}_K \setminus X^{\mathrm{o}}_K] \mid (f_K \sqcap h_{K\downarrow}) \sqcap g' \in \mathsf{Safe}\}$$

and M' is equal to M otherwise. $\lhd$

Lemma 1 (WeakestContext) The operation $\mathsf{WeakestContext}_K(\cdot)$ on PuCs preserves conservativity. $\lhd$

Proof Sketch We define $\hat{g}_K = \sqcap\{g \in \mathcal{F}[X^{\mathrm{o}}_K \setminus X^{\mathrm{i}}_K \to X^{\mathrm{i}}_K \setminus X^{\mathrm{o}}_K] \mid (f_P \sqcap \hat{h}_P) \sqsubseteq g\}$. For this context assumption we can prove that it is conservative and safe in the sense that $(f_K \sqcap h_{K\downarrow}) \sqcap \hat{g}_K \in \mathsf{Safe}$, it follows, by Definition 9, that $\hat{g}_K \sqsubseteq g'_K$, hence g'_K is also conservative. $\square$

Example 3 (Computing Conservative Local Context Assumptions) In Figure 4 we show a partial unravelling of the game board A_{stamp_1} into a game tree, this time for the control signature $[X^{\mathrm{o}}_{\mathrm{stamp}_1} \setminus X^{\mathrm{i}}_{\mathrm{stamp}_1} \to X^{\mathrm{i}}_{\mathrm{stamp}_1} \setminus X^{\mathrm{o}}_{\mathrm{stamp}_1}] = [\{\mathrm{s}_1, \mathrm{p}_1\} \to \{\mathrm{a}_1, \mathrm{p}_0\}]$.

When we solve this game and determine the weakest safe strategy we obtain the strategy which, in modal logic notation, can be defined as follows: $\mathrm{p}_0 \to \bigcirc \mathrm{a}_1$, i.e.: when there is a parcel in the feeder the stamp *must* activate in the next state. This regular strategy is shown in Figure 5 (left) encoded as a Kripke structure.

Note however, that this strategy relies on observation of p_0 which is not in the control signature. This means that this strategy encodes an *assumption* on the context, rather than a *guarantee* on the control. Assumptions may or may not be realizable depending on the rest of the plant. In this example, for this particular constraint, a control is realizable because the feeder component indeed provides us with an observation s_0 that allows the control to derive the status of p_0 by causality. $\lhd$

To fully exploit the synergistic relationship between context assumptions and control constraints we need to obtain also control constraints on a local level. So far we have shown (in Example 3) how local context assumptions can be computed, at the same time we have shown (in Example 2) that the direct approach to computing local control guarantees breaks down because it may yield constraints that are not conservative. However, as it turns out, it is possible to approximate conservative control constraints based on conservative context assumptions. Intuitively, this is done by *under approximating* the denyable behaviour.

Definition 10 (Conservative Local Control) For a given PuC M and a control locality $K \in C$ we define $\mathsf{StrongestControl}_K(M) = M'$ such that

$$h'_K = \sqcap\{h' \in \mathcal{F}[X^{\mathrm{o}}_K \cap X^{\mathrm{ci}} \to X^{\mathrm{i}}_K \cap X^{\mathrm{co}}] \mid (f_K \sqcap h_{K\downarrow}) \sqcap g_K \sqsubseteq h'\}$$

and M' is equal to M otherwise. $\lhd$

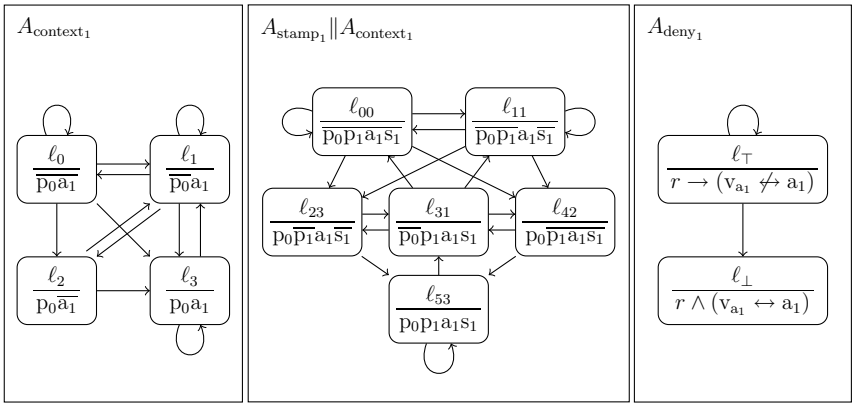

Fig. 5. Context assumptions for stamp_1 component (left), the behaviour of the stamp_1 component in its idealized context (center), the special Kripke structure encoding the rules of the dual deny game for the stamp_1 component (right).

Lemma 2 (StrongestControl) The operation $\mathsf{StrongestControl}_K(\cdot)$ on PuCs preserves conservativity. $\lhd$

Proof By conservativity of M it holds $f_P \sqcap \hat{h}_P \sqsubseteq (f_K \sqcap h_{K\downarrow}) \sqcap g_K$. By Definition 10 it holds $(f_K \sqcap h_{K\downarrow}) \sqcap g_K \sqsubseteq h'_K$. It follows $f_P \sqcap \hat{h}_P \sqsubseteq h'_K$. $\square$

Example 4 (Computing Strongest Local Control) We can effectively compute an approximate local control strategy by exploiting a natural duality that exists between *allow strategies* and *deny strategies*. Where allow strategies determine what is the set of *allowed* control outputs based on the observation history, deny strategies work by mapping an observation history to the set of *denied* control outputs, which is just the complement of the allowed set. We can exploit this duality because the *weakest conservative deny strategy* is the *strongest conservative allow strategy*.

The construction that turns an *allow game* into a *deny game* is then done as follows. First we turn all the control outputs $a_1, a_2 \in \mathrm{X}^{\mathrm{co}}$ into control inputs $a_1, a_2 \in \mathrm{X}^{\mathrm{ci}'}$ and replace them with a fresh set of *deny outputs* $\mathrm{v}_{a_1}, \mathrm{v}_{a_2} \cdots \in X^{\mathrm{co}'}$. We add one special control output $\mathrm{r} \in X^{\mathrm{co}'}$ which is called *restrict*. The rules of the game are as follows: if the safety player plays $\overline{r}$ the next state is *not restricted*. And the plant in its idealized context progresses normally. If the safety player plays r, *restrict*, we require that, in the next state, *at least one of the deny outputs* $\mathrm{v}_{a_1}, \mathrm{v}_{a_2}, \ldots$ *differs from the original control outputs* $a_1, a_2 \ldots$ *as chosen by the plant in its idealized context.* In this way the safety player is forced to be conservative in the restriction that she puts since she can only deny some sequence of control outputs whenever she is sure that the idealized context will never allow this sequence.

We may construct the game board for this deny game by taking the composition of the Kripke structures for the plant, the context strategy, and the rule that forces at least one of the deny outputs to be distinct from the control output as chosen by the plant in its

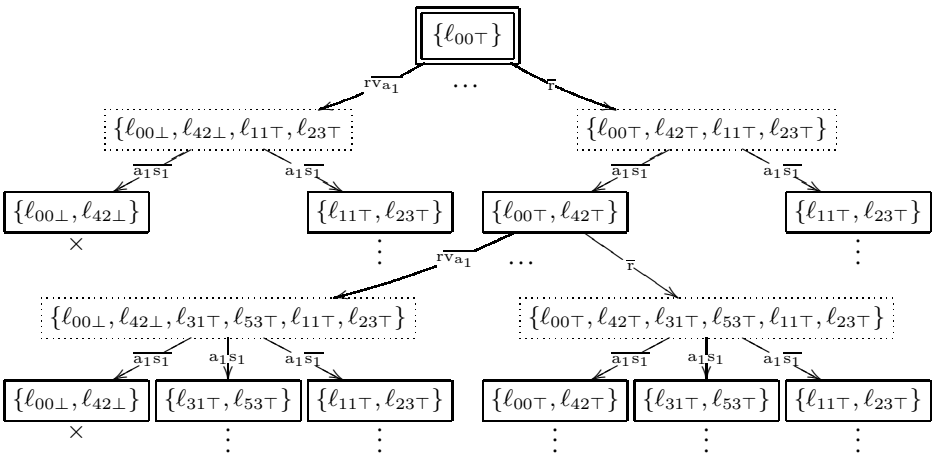

Fig. 6. Game tree for the deny game $A_{\mathrm{stamp}_1} \| A_{\mathrm{context}_1} \| A_{\mathrm{deny}_1}$, over the control signature $[\{\mathrm{a}_1, \mathrm{s}_1\} \to \{\mathrm{r}, \mathrm{v}_{\mathrm{a}_1}\}]$, here $\mathrm{r}\overline{\mathrm{v}_{\mathrm{a}_1}}$ means *restrict by denying* $\overline{\mathrm{a}_1}$, and $\overline{\mathrm{r}}$ means *unrestricted*.

idealized context. For stamp_1 this is $A_{\mathrm{stamp}_1} \| A_{\mathrm{context}_1} \| A_{\mathrm{deny}_1}$. A partial unraveling of the resulting game tree over this product game is shown in Figure 6. When we work this out further we quickly see that, in this game, the safety player is always forced to play $\overline{\mathrm{r}}$. This means she cannot put any restriction on the control outputs. Which, in turn, means that the resulting control strategy (after projecting out r, and projecting the temporary $\mathrm{v}_{\mathrm{a}_1}$ back to a_1) will be $f_\top$, i.e.: we cannot put any control constraints using the control signature for this locality. ◁

Compositional Controller Synthesis Algorithm We have now established all prerequisites to present Compositional Controller Synthesis Algorithm 1 (COCOS). The

Algorithm 1: COCOS (Compositional Control Synthesis)

Data: A PuC $M = (P, \{f_p, [X_p^{\mathrm{i}} \to X_p^{\mathrm{o}}]\}_{p \in P}, C, \{g_K, h_K\}_{K \in C})$ such that for all $K \in C$ it holds $g_k = h_K = f_\top$

Result: A maximally permissive, safe control strategy for the given system of plant components.

```
Visited ← ∅
while P ∉ Visited do
    select some K ∈ C such that K ∉ Visited and K↓ ⊆ Visited
    M ← StrongestControl_K(WeakestContext_K(M))
    Visited ← Visited ∪ {K}
return h_P
```

algorithm starts with a PuC M that is initially unconstrained, that is: the context assumptions and control constraints for each control locality are vacuous. The algorithm then works by making a single bottom–up pass over the control localities. It will start at the lowest localities (for which $K{\downarrow} = \varnothing$) progressing up to the highest control locality P. For each locality the weakest local context assumptions are computed (simplified using the subordinate control constraints) and subsequently the strongest local control constraints are computed (based on the weakest local control assumptions and the subordinate control constraints). The while loop terminates when the highest control locality $P \in C$ has been visited. At this point it holds $h_P = \hat{h}_P$.

Theorem 3 (Correctness) Algorithm 1 always terminates, and after termination it will hold $(f_P \sqcap h_P) = (f_P \sqcap \hat{h}_P)$. $\lhd$

Proof Completeness follows from the fact that there are only a finite number of control localities. Soundness follows from Lemmas 1 and 2, and the fact that for the highest control locality P the signatures of the weakest context assumptions, the strongest control constraints and the global control constraints $\hat{h}_P$ coincide. $\square$

Example 5 (Compositional Controller Synthesis) In Figure 7 we show how COCOS treats the PuC M_{parcel} as defined in Example 1. The plant components are shown as gray boxes connected by horizontal arrows denoting the unchecked plant propositions. The control localities are shown as white boxes connected to the plant components by vertical arrows denoting the control propositions. Each control locality is labeled with the weakest local context assumptions and the strongest local control constraints in modal logic notation. Since the algorithm performs a single, bottom–up pass over the control localities this picture represents the entire run.

For locality $\{\mathrm{feed}_0\}$ we get two vacuous strategies. The reason is that the feeder plant component does not contain any deadlocks. As such it needs no assumptions or control constraints to function safely. Control locality $\{\mathrm{stamp}_1\}$ has been treated more extensively as the running example in the previous sections. It requires a single context

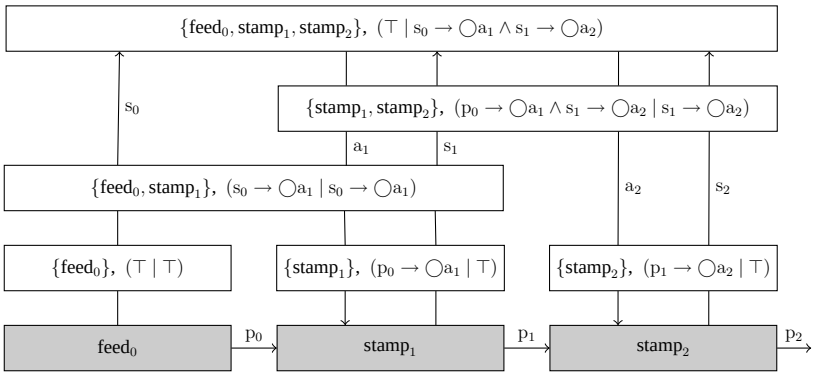

Fig. 7. PuC, Legend: (weakest local context assumptions | strongest local control guarantees).

assumption saying that the arm will activate when there is a parcel queuing in the feeder. As we have seen in the previous example we cannot enforce this assumption on a local level, yet. The situation for $\{\mathrm{stamp}_2\}$ is completely symmetrical. After treating the lower localities COCOS proceeds with the intermediate two localities.

For locality $\{\mathrm{feed}_0, \mathrm{stamp}_1\}$ new context assumptions are computed. This computation cannot yet benefit from subordinate control, since subordinate control is still vacuous. However, we do note that, since the signature of the context assumptions changes non–monotonically, the results will be different this time. In particular the p_0 proposition has become an internal plant proposition which is no longer visible to the context. At the same time the feeder component has added the control input proposition s_0. This means that the weakest context assumption has changed from $\mathrm{p}_0 \rightarrow \bigcirc \mathrm{a}_1$ into $\mathrm{s}_0 \rightarrow \bigcirc \mathrm{a}_1$. Intuitively, by restricting the context signature as much as we can (without sacrificing conservativity) the context assumptions have shifted into the direction of something that can be turned into a control constraint.

For locality $\{\mathrm{stamp}_1, \mathrm{stamp}_2\}$ the situation is almost symmetrical except for the fact that the assumption $\mathrm{p}_0 \rightarrow \bigcirc a_1$ on stamp_1 still has to be made by the context. Note that even though the weakest local context assumptions are over approximating the allowable behaviour by assuming plant proposition p_0 to be observable, COCOS is still able to recover the constraint $\mathrm{s}_1 \rightarrow \bigcirc \mathrm{a}_2$ which has a clear causal dependency on this proposition p_0.

Finally, we treat the topmost locality $\{\mathrm{feed}_0, \mathrm{stamp}_1, \mathrm{stamp}_2\} = P$. The weakest context assumptions for this locality are vacuous, since subordinate control already ensures safety. In this case, the strongest control constraints are simply the conjunction of the subordinate control constraints. Note that this is where compositionality really helps us: computing the assumptions for higher control localities becomes much easier in the presence of subordinate control, especially using a counterexample driven algorithm. In this case subordinate control ensures that there are no unsafe transitions anymore at the highest level of composition. $\lhd$

4 Conclusion

We have presented a semantical framework for compositional control synthesis problems. Based on the framework we have developed a compositional control synthesis method that is sound and complete for computing most permissive safety controllers on regular models under the assumption of partial observation. The novel aspects of the method are:

1. The signature of the local context assumptions changes non–monotonically with increasing scope, tending to the signature of local control constraints. In this way we obtain more realistic assumptions as the scope widens.
2. The local control assumptions are simplified with respect to subordinate control. In this way, we make it possible to efficiently apply a counterexample driven refinement algorithm for finding the weakest context assumptions.
3. Subordinate control is approximated based on local context assumptions. In this way, we enable a synergistic relationship between local context assumptions and local control constraints.

Output Confusion and Subordinate Control To simplify exposition in this paper we explicitly forbid in– and output confusion among plant components. For some applications, such as circuit synthesis, it may be desirable to allow in– and output confusion. For instance, because a single output signal is shared by two or more components, or because a single component treats a signal as input or output depending on its internal state. Note that, to accomodate this, we need to consider moves of the safety player as *sets of control outputs* (*allow sets*) as opposed to concrete control outputs.

Consider for instance a plant component over a single proposition x that at even clockcycles treats x as *output* by writing a random bit $s_{2j}(x) \in \{0,1\}$ to the controller, and at odd clockcycles treats x as *input* by reading $s_{2j+1}(x) \in \{0,1\}$ back from the controller. Next consider a safety invariant that requires: $s_{2j+1}(x) = s_{2j}(x)$. The resulting game cannot be won by the safety player if she needs to choose concrete outputs for x at each cycle t, since at even clockcycles $t = 2j$ she cannot predict what the component is going to write as output. However if we consider moves as proper allow sets this problem disappears. At even clockcycles the safety player may simply allow x to vary freely by allowing x to be either high or low: $h(s_0 \dots s_{2j+1}) = \{x, \overline{x}\}$, and at odd clockcycles the safety player may restrict x depending on what she observed in the previous state: $h(s_0 \dots s_{2j}) = \{x\}$ in case $s_{2j}(x) = 1$ or $h(s_0 \dots s_{2j}) = \{\overline{x}\}$ in case $s_{2j}(x) = 0$.

The algorithm in [13] already treats moves for the safety player as allow sets. This facilitates abstraction for the higher levels in the control hierarchy. Note that, if we require the safety player to choose *concrete* control outputs she has no choice but to emulate all subordinate control that we incorporated in the game board. If she would choose a concrete control ouput that is forbidden by subordinate control the system will block. Such an interpretation would defeat the purpose of an incremental algorithm like COCOS. Instead in our interpretation the safety player can safely allow a control output that is forbidden by subordinate control. Intuitively we can see this as a form of output confusion among a control locality and its subordinate control localities.

Future Work We are currently working on an implementation of the framework in a prototype tool. For the operations that we defined here on a semantic level to be implemented efficiently, we are using a symbolic representation of context assumptions as antichains over info/allow pairs, and the dual representation for control constraints as antichains over info/deny pairs.

For solving games we are using a combination of forward and backward methods. The lower control localities can typically be handled by forward methods, which has an advantage that only reachable states are considered. For the higher control localities, where there is more room for abstraction, we switch to a backward counter example driven algorithm like in [13]. In this way we fully exploit the subordinate control constraints.

References

1. Eugene Asarin, Olivier Bournez, Thao Dang, Oded Maler, and Amir Pnueli. Effective synthesis of switching controllers for linear systems. *Proceedings of the IEEE*, 88(7):1011–1025, 2000.

2. Eugene Asarin, Oded Maler, and Amir Pnueli. Symbolic controller synthesis for discrete and timed systems. volume 999 of *LNCS*, pages 1–20. Springer-Verlag, 1995.
3. Dirk Beyer, Thomas A. Henzinger, and Vasu Singh. Algorithms for interface synthesis. volume 4590 of *LNCS*, pages 4–19. Springer, 2007.
4. J. Richard Buchi and Lawrence H. Landweber. Solving sequential conditions by Finite-State strategies. *Transactions of the American Mathematical Society*, 138:295–311, April 1969. ArticleType: primary_article / Full publication date: Apr., 1969 / Copyright 1969 American Mathematical Society.
5. Franck Cassez. Efficient On-the-Fly algorithms for partially observable timed games. volume 4763 of *LNCS*, pages 5–24. Springer, 2007.
6. Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Freddy Y. C. Mang. Synchronous and bidirectional component interfaces. volume 2404 of *LNCS*, pages 711–745. Springer, 2002.
7. Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-Franois Raskin. Algorithms for Omega-Regular games with imperfect information,. volume 4207 of *LNCS*, pages 287–302. Springer-Verlag, September 2006.
8. Krishnendu Chatterjee, Thomas Henzinger, and Barbara Jobstmann. *Environment Assumptions for Synthesis*, pages 147–161. 2008.
9. Krishnendu Chatterjee and Thomas A. Henzinger. Assume-Guarantee synthesis. volume 4424 of *LNCS*, pages 261–275. Springer, 2007.
10. A. Church. Application of recursive arithmetic to the problem of circuit synthesis. *Summaries of the Summer Institute of Symbolic Logic, Cornell Univ., Ithaca, NY*, 1:3–50, 1957.
11. Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using Branching-Time temporal logic. In *Logic of Programs, Workshop*, pages 52–71. Springer-Verlag, 1982.
12. Luca de Alfaro and Thomas A. Henzinger. Interface automata. *SIGSOFT Softw. Eng. Notes*, 26(5):109–120, 2001.
13. Wouter Kuijper and Jaco van de Pol. *Computing Weakest Strategies for Safety Games of Imperfect Information*, pages 92–106. 2009.
14. Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. Safraless compositional synthesis. volume 4144 of *LNCS*, pages 31–44. Springer, 2006.
15. Orna Kupferman and Moshe Y. Vardi. Synthesis with incomplete information. volume 16 of *Applied Logic Series*, pages 109–127. Kluwer, 2000.
16. F. Lin and W. M. Wonham. Decentralized control and coordination of discrete-event systemswith partial observation. *IEEE Transactions on automatic control*, 35(12):1330–1337, 1990.
17. Oded Maler, Dejan Nickovic, and Amir Pnueli. On synthesizing controllers from Bounded-Response properties. volume 4590 of *LNCS*, pages 95–107. Springer, 2007.
18. Z. Manna and P. Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 6(1):68–93, 1984.
19. Wonhong Nam, P. Madhusudan, and Rajeev Alur. Automatic symbolic compositional verification by learning assumptions. *Form. Methods Syst. Des.*, 32(3):207–234, 2008.
20. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 179–190, Austin, Texas, United States, 1989. ACM.
21. Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. volume 372 of *LNCS*, pages 652–671. Springer, 1989.
22. M. O. Rabin. *Automata on infinite objects and Church's problem*. American Mathematical Society, 1972.

23. Peter J.G. Ramadge and W. Murray Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.
24. W. Thomas. On the synthesis of strategies in infinite games. *Lecture Notes in Computer Science*, 900(1-13):12, 1995.
25. W. Thomas. Facets of synthesis: Revisiting church's problem. In *Foundations of Software Science and Computational Structures: 12th International Conference, Fossacs 2009, Held As Part of the Joint European Conferences on Theory and Practice of Software, Etaps 2009, York, Uk*, page 1. Springer-Verlag New York Inc, 2009.
26. Martin De Wulf, Laurent Doyen, and Jean-Franois Raskin. A lattice theory for solving games of imperfect information. volume 3927 of *LNCS*, pages 153–168. Springer, 2006.

A Proofs

A.1 Sanity Lemmas

Lemma 4 (Joins and Meets Preserve Signatures) For a given $F \subseteq \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$, it holds $\sqcup F \in \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$ and $\sqcap F \in \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$. $\lhd$

Proof (Lemma 4) Let $F \subseteq \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$, $f_{\sqcup} = \sqcup F$, and $f_{\sqcap} = \sqcap F$. We need to show that $f_{\sqcup} \in \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$ and $f_{\sqcap} \in \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$.

First, we prove $f_{\sqcup} \in \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$. Assume, for contradiction, $f_{\sqcup} \notin \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$. By Definition 4 (Permissibility) it follows there exists $\sigma, \sigma' \in \mathcal{S}^*$ and $s, s' \in \mathcal{S}$ such that $\sigma \sim_{X^{\mathrm{io}}} \sigma'$, and, wlog, $s \in f_{\sqcup}(\sigma)$, and $s' \notin f_{\sqcup}(\sigma')$. It follows by definition of $f_{\sqcup}$ that there exists $f \in F$ such that $s \in f(\sigma)$ and for all $f' \in F$ it holds $s' \notin f'(\sigma')$ in particular $s' \notin f(\sigma')$. But this would mean that f is not permissible for the signature $[X^{\mathrm{i}} \to X^{\mathrm{o}}]$ which establishes the required contradiction.

Next, we prove $f_{\sqcap} \in \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$. Assume, for contradiction, $f_{\sqcap} \notin \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$. By Definition 4 (Permissibility) it follows there exists $\sigma, \sigma' \in \mathcal{S}^*$ and $s, s' \in \mathcal{S}$ such that $\sigma \sim_{X^{\mathrm{io}}} \sigma'$, and, wlog, $s \notin f_{\sqcap}(\sigma)$, and $s' \in f_{\sqcap}(\sigma')$. It follows by definition of $f_{\sqcap}$ that there exists $f \in F$ such that $s \notin f(\sigma)$ and for all $f' \in F$ it holds $s' \in f(\sigma')$ in particular $s' \in f(\sigma')$. But this would mean that f is not permissible for the signature $[X^{\mathrm{i}} \to X^{\mathrm{o}}]$, which establishes the required contradiction. $\square$

Lemma 5 (Joins Preserve Safety) For a given $f \in \mathcal{F}$, a subset $F \subseteq \mathcal{F}$, and $\hat{g} = \sqcup\{g \in F \mid f \sqcap g \in \mathsf{Safe}\}$ it holds $f \sqcap \hat{g} \in \mathsf{Safe}$. $\lhd$

Proof (Lemma 5) Let $f \in \mathcal{F}$ and $\hat{g} = \sqcup\{g \in \mathcal{F} \mid f \sqcap \hat{g} \in \mathsf{Safe}\}$. Assume, for contradiction, that $f \sqcap \hat{g} \notin \mathsf{Safe}$. It follows by Definition 2 (Safety) there must exist $\sigma^\frown s' \in \mathsf{Reach}(f \sqcap \hat{g})$ such that $(f \sqcap \hat{g})(\sigma^\frown s') = \varnothing$. Then by Definition 2 (Consistency) it must hold that $(f \sqcap \hat{g})(\sigma) \ni s'$. Hence, $\hat{g}(\sigma) \ni s'$. It follows by definition of $\hat{g}$ that there exists $g \in F$ such that $f \sqcap g \in \mathsf{Safe}$ and $g(\sigma) \ni s'$. Then, again by Definition 2 (Prefix–closedness) it follows $\sigma^\frown s' \in \mathsf{Reach}(f \sqcap g)$. But, because $g \sqsubseteq \hat{g}$, it follows $(f \sqcap g)(\sigma^\frown s') = \varnothing$, this would mean $f \sqcap g \notin \mathsf{Safe}$ which establishes the required contradiction. $\square$

Lemma 6 The weakest safe global strategy $\hat{h}_P$, from Definition 7, is well–defined. $\lhd$

Proof (Lemma 6) First, we need that $\hat{h}_P$ is of the right signature: $\hat{h}_P \in \mathcal{F}[X^{\mathrm{ci}} \to X^{\mathrm{co}}]$, this is ensured by Lemma 4. Second, we need that $\hat{h}_P$ is safe: $f_P \sqcap \hat{h}_P \in \mathsf{Safe}$, this is ensured by Lemma 5. $\square$

Lemma 7 The operation $\mathsf{WeakestContext}_K(\cdot)$, from Definition 9, is well–defined. $\lhd$

Proof (Lemma 7) This proof is completely analogous to the proof of Lemma 6. First, we need that g'_K is of the right signature: $g'_K \in \mathcal{F}[X_K^{\mathrm{po}} \setminus X_K^{\mathrm{pi}} \to X_K^{\mathrm{pi}} \setminus X_K^{\mathrm{po}}]$, this is ensured by Lemma 4. Second, we need that g'_K is safe: $f_K \sqcap g'_K \in \mathsf{Safe}$, this is ensured by Lemma 5. $\square$

Lemma 8 The operation $\mathsf{StrongestControl}_K(\cdot)$, from Definition 10, is well–defined.$\lhd$

Proof (Lemma 8) First, we need that h'_K is of the right signature: $h'_K \in \mathcal{F}[X^{\mathrm{po}}_K \cap X^{\mathrm{ci}} \to X^{\mathrm{pi}}_K \cap X^{\mathrm{co}}]$, this is ensured by Lemma 4. Second, we need that h'_K is conservative: $(f_K \sqcap \hat{h}_{K\downarrow}) \sqcap g_K \sqsubseteq h'_K$, this follows directly from the definition of the infimum strategy. □

A.2 Preservation Lemma

For the preservation results we rely on the existence of infimum and supremum strategies over given signatures. But this is a rather technical explanation. What intuitively happens, is that we are conservatively *following* a strategy $g \in \mathcal{F}$ with a given signature $[X^{\mathrm{i}} \to X^{\mathrm{o}}]$. Conservatively following a strategy with a given signature sometimes entails making an over approximation of the allowed successors in order not to exclude possible behaviour of the underlying strategy.

Definition 11 (Follower Strategy) For a given strategy $g \in \mathcal{F}$ and a given signature $[X^{\mathrm{i}} \to X^{\mathrm{o}}]$ we define the *follower strategy* $g_{[X^{\mathrm{i}} \to X^{\mathrm{o}}]}$ such that for all $\sigma \in \mathcal{S}^*$ we have

$$g_{[X^{\mathrm{i}} \to X^{\mathrm{o}}]}(\sigma) = \{s \in \mathcal{S} \mid \exists \sigma' \in \mathcal{S}^*.\sigma \sim_{X^{\mathrm{io}}} \sigma' \text{ and } \exists s' \in g(\sigma').s \sim_{X^o} s'\}$$

◁

Lemma 9 For a given strategy $g \in \mathcal{F}$ and a given signature $[X^{\mathrm{i}} \to X^{\mathrm{o}}]$ it holds $g_{[X^{\mathrm{i}} \to X^{\mathrm{o}}]} = \sqcap\{g' \in \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}] \mid g \sqsubseteq g'\}$, i.e.: the follower strategy is the infimum strategy above g over the given signature. ◁

Proof (Lemma 9) It is clear that $g \sqsubseteq g_{[X^{\mathrm{i}} \to X^{\mathrm{o}}]}$ and $g_{[X^{\mathrm{i}} \to X^{\mathrm{o}}]} \in \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$. To see that it is really the *least* upper bound let us assume some $h \in \mathcal{F}[X^{\mathrm{i}} \to X^{\mathrm{o}}]$ such that $g \sqsubseteq h$. Now we need to show that $g_{[X^{\mathrm{i}} \to X^{\mathrm{o}}]} \sqsubseteq h$. So assume, for contradiction, $g_{[X^{\mathrm{i}} \to X^{\mathrm{o}}]} \not\sqsubseteq h$. It follows there exists some $\sigma \in \mathcal{S}^*$ such that $g_{[X^{\mathrm{i}} \to X^{\mathrm{o}}]}(\sigma) \not\subseteq h(\sigma)$. So, wlog, assume $s \notin h(\sigma)$ and $s \in g_{[X^{\mathrm{i}} \to X^{\mathrm{o}}]}(\sigma)$. From the latter we obtain existence of an alternative trace $\sigma' \in \mathcal{S}^*$ and an alternative successor $s' \in g(\sigma')$ such that $\sigma' \sim_{X^{\mathrm{io}}} \sigma$ and $s' \sim_{X^o} s$. Now by permissibility of h and our assumption that $s \notin h(\sigma)$ we obtain that $s' \notin h(\sigma')$, since by assumption $s' \in g(\sigma')$, it would follow that $g \not\sqsubseteq h$, which establishes the required contradiction. □

We may now go back and work out the proof–sketch we have already given for Lemma 1. We recall that the lemma says that the $\mathsf{WeakestContext}_K(\cdot)$ operation on PuCs preserves conservativity.

Proof (Lemma 1) Let M be a conservatively constrained PuC. Let $K \in C$ be some control locality. Now for $M' = \mathsf{WeakestContext}_K(M)$ we must show that M' is conservative. We recall that by definition the only difference between M' and M lies in the strengthened context assumptions $g'_K \sqsubseteq g_K$ (cf. Definition 9). So for conservativity it would suffice to prove $f_P \sqcap \hat{h}_P \sqsubseteq g'_K$ (cf. Definition 8).

The main proof idea is to turn the behaviour of the global system — consisting of the global plant behaviour f_P restricted by the weakest safe global control strategy $\hat{g}_P$ — into a context assumption $\hat{g}_K$ for control locality K, by following the global

system's behaviour $f_P \sqcap \hat{g}_P$ with the context signature for locality K. We then show that the assumptions that we constructed in this way are *conservative*: $f_P \sqcap \hat{h}_P \sqsubseteq \hat{g}_K$, and we show that they are *safe*: $(f_K \sqcap h_{K\downarrow}) \sqcap \hat{g}_K \in \mathsf{Safe}$. This suffices since g'_K, by Definition 9, is the *weakest safe context assumption* so, in particular, it will be weaker than the constructed assumptions: $\hat{g}_K \sqsubseteq g'_K$, since these are conservative by construction it will follow that g'_K is conservative as well: $f_P \sqcap \hat{h}_P \sqsubseteq g'_K$.

First, we introduce some shorthand for the context signature for control locality K, so let $[X_K^{\mathrm{gi}} \to X_K^{\mathrm{go}}] = [X_K^{\mathrm{po}} \setminus X_K^{\mathrm{pi}} \to X_K^{\mathrm{pi}} \setminus X_K^{\mathrm{po}}]$. Next, we construct the required context assumptions for control locality K, as follows: $\hat{g}_K = (f_P \sqcap \hat{h}_P)_{[X^{\mathrm{gi}} \to X^{\mathrm{go}}]}$.

By Lemma 9 we obtain $f_P \sqcap \hat{h}_P \sqsubseteq \hat{g}_K$. It remains to show $(f_K \sqcap h_{K\downarrow}) \sqcap \hat{g}_K \in \mathsf{Safe}$. For this, we will first prove that (1) $(f_K \sqcap h_{K\downarrow}) \sqcap \hat{g}_K = (f_K \sqcap \hat{g}_K)$, and then we will prove that (2) $f_K \sqcap \hat{g}_K \in \mathsf{Safe}$.

For (1), it would suffice to show $\hat{g}_K \sqsubseteq h_{K\downarrow}$. We will use the assumption that M is conservative, and the fact that $h_{K\downarrow} \in \mathcal{F}[X_K^{\mathrm{gi}} \to X_K^{\mathrm{go}}]$, i.e.: the signature for the context assumptions is wider than the signature for the subordinate control constraints. Now assume, for contradiction, $\hat{g}_K \not\sqsubseteq h_{K\downarrow}$ from this it would follow that there exists some trace $\sigma \in \mathcal{S}^*$ and some state $s \notin h_{K\downarrow}(\sigma)$ while $s \in \hat{g}_K(\sigma)$. From the latter it follows that there exists an alternative trace $\sigma' \in \mathcal{S}^*$ and an alternative successor state $s' \in (f_P \sqcap \hat{h}_P)(\sigma')$ such that $\sigma' \sim_{X_K^{\mathrm{gio}}} \sigma$ and $s' \sim_{X_K^{\mathrm{go}}} s$. But then, by the fact that $h_{K\downarrow}$ is permissible for $[X^{\mathrm{gi}} \to X^{\mathrm{go}}]$ and our assumption that $s \notin h_{K\downarrow}(\sigma)$ it would follow that $s' \notin h_{K\downarrow}(\sigma')$, and this would imply that $f_P \sqcap \hat{h}_P \not\sqsubseteq h_{K\downarrow}$, which in turn would imply that $\hat{h}_P \not\sqsubseteq h_{K\downarrow}$, which contradicts our assumption that $h_{K\downarrow}$ is conservative.

For (2), assume, $\sigma^\frown s \in \mathsf{Reach}(f_K \sqcap \hat{g}_K)$. It follows there exists $\sigma' \in \mathcal{S}^*$ and $s' \in (f_P \sqcap \hat{h}_P)(\sigma')$ such that $\sigma' \sim_{X^{\mathrm{gio}}} \sigma$ and $s' \sim_{X^{\mathrm{go}}} s$. By Prefix–closedness this implies $\sigma'^\frown s' \in \mathsf{Reach}(f_P \sqcap \hat{h}_P)$. We now construct a third trace $\sigma''^\frown s'' \in \mathcal{S}^*$ such that for all $i \leq |\sigma''|$ it holds $\sigma''_i \cap X_K^{\mathrm{pio}} = \sigma_i \cap X_K^{\mathrm{pio}}$ and $\sigma''_i \cap \overline{X}_K^{\mathrm{pio}} = \sigma'_i \cap \overline{X}_K^{\mathrm{pio}}$ and $s'' \cap X_K^{\mathrm{pio}} = s \cap X_K^{\mathrm{pio}}$ and $s'' \cap \overline{X}_K^{\mathrm{pio}} = s' \cap \overline{X}_K^{\mathrm{pio}}$, where $\overline{X}_K^{\mathrm{pio}} = \mathcal{X} \setminus X_K^{\mathrm{pio}}$ i.e.: this third trace $\sigma''^\frown s''$ is consistent with σ on all plant in– and outputs of locality K and it is consistent with σ' on all other propositions. Moreover, since $\sigma \sim_{X^{\mathrm{gio}}} \sigma'$ and $X_{\overline{K}}^{\mathrm{pio}} \subseteq \overline{X}_K^{\mathrm{pio}} \cup X_K^{\mathrm{gio}}$ it holds, in addition, for all $i \leq |\sigma''|$ that $\sigma''_i \cap X_{\overline{K}}^{\mathrm{pio}} = \sigma'_i \cap X_{\overline{K}}^{\mathrm{pio}}$, i.e.: σ'' is consistent with σ' on all in– and outputs of locality $\overline{K} = P \setminus K$. And finally, since $\sigma \sim_{X^{\mathrm{gio}}} \sigma'$ and $X_P^{\mathrm{cio}} \subseteq \overline{X}_K^{\mathrm{pio}} \cup X_K^{\mathrm{gio}}$ it holds, in addition, $\sigma''_i \cap X_P^{\mathrm{cio}} = \sigma'_i \cap X_P^{\mathrm{cio}}$, i.e.: σ'' is consistent with σ' on all control in– and outputs. The latter proves that $\hat{h}_P(\sigma'') = \hat{h}_P(\sigma')$. Now note that $f_P = f_K \sqcap f_{\overline{K}}$, moreover $f_K \in \mathcal{F}[X_K^{\mathrm{pi}} \to X_K^{\mathrm{po}}]$ and $f_{\overline{K}} \in \mathcal{F}[X_{\overline{K}}^{\mathrm{pi}} \to X_{\overline{K}}]$. Hence, by permissibility, $f_K(\sigma'') = f_K(\sigma')$ and $f_{\overline{K}}(\sigma'') = f_{\overline{K}}(\sigma')$, these two facts together prove $f_P(\sigma'') = f_P(\sigma')$. Now note that for the successor state s'' it holds $s'' \sim_{X_K^{\mathrm{po}}} s'$ and $s'' \sim_{X_{\overline{K}}^{\mathrm{po}}} s'$ and $s'' \sim_{X_P^{\mathrm{co}}} s'$. And, since $s' \in (f_P \sqcap \hat{h}_P)(\sigma')$, it follows $s'' \in (f_P \sqcap \hat{h}_P)(\sigma'')$. It follows, $\sigma''^\frown s'' \in \mathsf{Reach}(f_P \sqcap \hat{h}_P)$. By safety of $f_P \sqcap \hat{h}_P$ this implies $f_K(\sigma''^\frown s'') \cap \hat{h}_P(\sigma''^\frown s'') \neq \varnothing$. Now note that $f_K(\sigma''^\frown s'') = f_K(\sigma^\frown s)$ and $\hat{g}_K(\sigma''^\frown s'') = \hat{g}_K(\sigma^\frown s)$ and, by definition, $\hat{g}_K(\sigma''^\frown s'') \supseteq \hat{h}_K(\sigma''^\frown s'')$. It follows $f_K(\sigma''^\frown s'') \cap \hat{g}_K(\sigma''^\frown s'') \neq \varnothing$. Hence, $(f_K \sqcap \hat{g}_K)(\sigma^\frown s) \neq \varnothing$. □