12th IFAC/IEEE Workshop on Discrete Event Systems
Cachan, France. May 14-16, 2014

Extending Supervisory Controller Synthesis to Deterministic Pushdown Automata— Enforcing Controllability Least Restrictively

A.-K. Schmuck [**] **S. Schneider** [*] **J. Raisch** [**,***] **U. Nestmann** [*]

[*] *Modelle und Theorie Verteilter Systeme, Technische Universität Berlin*
[**] *Regelungssysteme, Technische Universität Berlin, Germany*
[***] *Max-Planck-Institut für Dynamik komplexer technischer Systeme, Magdeburg, Germany*

Abstract: In this paper a step towards the generalization of supervisory control theory to situations where the specification is modeled by a deterministic pushdown automaton (DPDA) is provided. In particular, this paper presents an algorithm to calculate the largest controllable marked sublanguage of a given deterministic context free language (DCFL) by least restrictively removing controllability problems in a DPDA realization of this DCFL. It also provides a counterexample which shows that the algorithm by Griffin (2008) intended to solve the considered problem is not minimally restrictive.

Keywords: Supervisory Control Theory, Supremal Controllable Sublanguage, Minimally Restrictive Supervisor, Deterministic Context Free Languages, Pushdown Automata

1. INTRODUCTION

Ramadge and Wonham (1984) established *supervisory control theory* (SCT) for controller synthesis on formal languages. Given a plant and a specification, SCT defines a *proper minimally restrictive supervisor* as a controller which generates a closed loop system (i.e., a plant restricted by a controller) that contains as many words allowed by the plant as possible while respecting the specification, not preventing uncontrollable events and always guiding the system to a satisfactory (marking) state. Wonham and Ramadge (1987) presented an implementable fixed point algorithm to calculate the desired marked closed loop language L_{clm} using finite automaton representations of the involved languages and therefore restricting its applicability to regular plant and specification languages. This fixed point algorithm iteratively executes the following on the product automaton of plant and specification: (i) it removes controllability problems, i.e., situations where the controller attempts to prevent uncontrollable events. (ii) It resolves blocking issues, i.e., situations where the closed loop cannot reach a marking state. Obviously, step (i) may generate new blocking issues, while step (ii) may lead to new controllability problems. The algorithm terminates iff no more controllability problems or blocking situations are present.

The class of deterministic context free languages (DCFL) contains the class of regular languages, and DCFL can be represented by deterministic pushdown automata (DPDA). It was shown by Sreenivas (1993) and Masopust (2012) that step (i) of the fixed point algorithm to calculate L_{clm} cannot be realized for the case where both the plant and the specification language are DCFL. However, this is possible for the case where only the specification language is generalized to DCFL. The resulting generalized supervisory control problem is practically relevant, as it allows to consider a broader class of specifications.

The main contribution of this paper is an algorithm which realizes step (i) for DPDA. Step (ii) is discussed by Schneider and Nestmann (2014). Soundness issues involving the overall iteration, including the existence of a unique fixed point, are discussed in a companion paper by Schneider, Schmuck, Raisch, and Nestmann (2014). Griffin (2008) suggested an algorithm for the more restrictive setting of a prefix closed regular plant and a prefix closed deterministic context free specification language. However, as shown by the counterexample in Appendix A, Griffin's algorithm does not construct the minimally restrictive supervisor.

The paper is structured as follows. After introducing all required notation in Section 2, we summarize the necessary parts of SCT in Section 3. In Section 4, we present an algorithm working on DPDA, realizing step (i) of the fixed point algorithm to calculate L_{clm}.

2. PRELIMINARIES

Let Σ be the external alphabet. Then Σ^* denotes the set of all finite-length strings over symbols from Σ. Furthermore, we use the abbreviations $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$ and $\Sigma^{\leq 1} = \Sigma \cup \{\lambda\}$, where λ is the empty string. Throughout this paper, we denote elements of the set $\Sigma^{\leq 1}$ by σ, i.e., σ can also be the empty string. We denote the *projection* of a tuple or string a on its ith element by $\pi_i(a)$ and the concatenation of two strings $w, w' \in \Sigma^*$ by $w{\cdot}w'$, meaning that w' is appended to w. The *prefix relations* on strings are defined by $w \sqsubseteq w'$ if [1] $\exists w'' \in \Sigma^* \,.\, w{\cdot}w'' = w'$ and by $w \sqsubset w'$ if $\exists w'' \in \Sigma^+ \,.\, w{\cdot}w'' = w'$. Any subset of

⋆ This work was partially supported by the HYCON2 Network of excellence (FP7 ICT 257462).

[1] Throughout this paper we use the notation "$\forall \cdot \,.\, \cdot$", meaning that all statements after the dot hold for all variables in front of the dot. "$\exists \cdot \,.\, \cdot$" is interpreted analogously.

Σ^* is called a *language*. The *prefix closure* of a language L is defined by $\overline{L} = \{w \in \Sigma^* \mid \exists w' \in L \,.\, w \sqsubseteq w'\}$. A language L_1 is *nonblocking* w.r.t. a language L_2 iff $L_1 \subseteq \overline{L_2}$. On an alphabet Σ partitioned into controllable (in the sense of preventable) and uncontrollable events, i.e., $\Sigma = \Sigma_{\mathsf{c}} \cup \Sigma_{\mathsf{uc}}$, $\Sigma_{\mathsf{c}} \cap \Sigma_{\mathsf{uc}} = \emptyset$, a prefix-closed language $L_1 = \overline{L_1}$ is *controllable* w.r.t. a prefix-closed language $L_2 = \overline{L_2}$ iff $(L_1 \cdot \Sigma_{\mathsf{uc}}) \cap L_2 \subseteq L_1$. In particular, a word $w \in L_1$ is controllable w.r.t. L_2 (written $\mathrm{ContW}(L_1, L_2, \Sigma_{\mathsf{uc}}, w)$) iff

$$\forall \mu \in \Sigma_{\mathsf{uc}} \,.\, w\mu \in L_2 \rightarrow w\mu \in L_1. \tag{1}$$

A *discrete event system* (DES) is a tuple $D = (\Sigma_{\mathsf{c}}, \Sigma_{\mathsf{uc}}, L_{\mathrm{um}}, L_{\mathrm{m}})$, where $L_{\mathrm{um}} = \overline{L_{\mathrm{um}}} \subseteq \Sigma^*$ is the prefix closed unmarked language modeling the step by step evolution of the system while the marked language $L_{\mathrm{m}} \subseteq L_{\mathrm{um}}$ models only the satisfactory words. Since all DES in this paper evolve on the same external alphabet $\Sigma = \Sigma_{\mathsf{c}} \cup \Sigma_{\mathsf{uc}}$, we can characterize D by its marked and unmarked language only and write $D = (\!| L_{\mathrm{um}}, L_{\mathrm{m}} |\!)$. We say that D is nonblocking, iff L_{um} is nonblocking w.r.t. L_{m}.

A *pushdown automaton* (PDA) is a tuple

$$M := (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$$

s.t. (i) the state set Q, the external alphabet Σ, the stack alphabet Γ, and the set of transitions δ are finite, (ii) $\delta \subseteq Q \times \Sigma^{\leq 1} \times \Gamma \times \Gamma^* \times Q$, (iii) the end-of-stack marker $\square$ is contained in Γ, (iv) the set of marking states F and the initial state q_0 are contained in Q and (v) $\square$ is never removed from the stack (i.e., $(q, \sigma, \square, s, q')$ implies [2] $\exists s' \in \Gamma^* \,.\, s = s' \cdot \square$).

Example 1. Consider the external alphabet $\Sigma = \{a, b, c, d, u\}$, the stack alphabet $\Gamma = \{\square, \bullet\}$, the state set $Q = \{q_0, \ldots, q_5\}$ and the set of marking states $F = \{q_2, q_5\}$. Then M_O in Figure 1 is a PDA, and the transition $(q, \sigma, \gamma, s, q') \in \delta$ is depicted by an edge from q to q' labeled by σ, γ, s, denoting that by taking this transition, $\sigma \in \Sigma^{\leq 1}$ is generated, $\gamma \in \Gamma$ (called "stack-top") is popped from the top of the stack, and $s \in \Gamma^*$ is pushed onto the stack (with the right-most symbol first). Two transitions with the same pre and post state are depicted by one edge with two labels. ◁

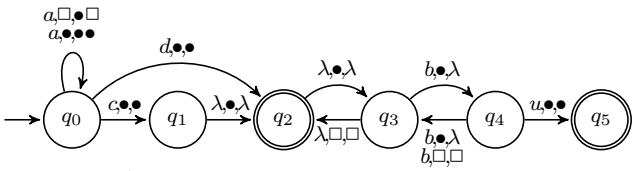

Figure 1. PDA M_O in Example 1.

Note that M can do silent moves (called λ-transitions), possibly modifying the stack but not generating an external symbol (e.g., see Figure 1, from q_1 to q_2). We collect the starting states of all λ-transitions in the set

$$Q_\lambda(M) := \{q \in Q \mid (q, \lambda, \gamma, s, q') \in \delta\}.$$

A PDA M is *livelock free* (written $M \in \mathrm{LLF}$) iff there exists no reachable infinite sequence of λ-transitions. Since livelock free PDA and PDA are equally expressive, we only consider $M \in \mathrm{LLF}$.

From a system theoretic point of view, a system state would be a pair (q, s), where $q \in Q$ and $s \in \Gamma^*$ represents the current stack content. Hence, from this point of view, a PDA with $|Q| < \infty$ has infinite state space since the stack is not restricted. The pair (q, s) is sometimes referred to as a configuration. For our purposes, it turns out to be convenient to add the string of generated external symbols to this pair. We therefore define the set of *configurations* of $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F) \in \mathrm{PDA}$ by

$$\mathcal{C}(M) := Q \times \Sigma^* \times \Gamma^*$$

where $(q, w, s) \in \mathcal{C}(M)$, consists of a state q, a history variable w (storing the external symbols generated), and a stack variable s (storing the current stack content). The initial configuration is $(q_0, \lambda, \square)$. Now observe that a transition from one configuration to another occurs when M takes a transition $e \in \delta$. To make this step visible we choose to include the taken transition e into the definition of a transition from one configuration to another. The *single-step transition relation* $\vdash_M \subseteq ((\delta \cup \{\bot\}) \times \mathcal{C}(M))^2$ is therefore defined by

$$(e, (q, w, \gamma \cdot s')) \vdash_M (e', (q', w \cdot \sigma, s \cdot s')) \text{ for } e' = (q, \sigma, \gamma, s, q'),$$

where $\bot$ denotes an undefined [3] pre-transition.

The set $\mathcal{D}(M)$ contains all finite-length [4] *derivations* $f : \mathbf{N} \rightharpoonup ((\delta \cup \{\bot\}) \times \mathcal{C}(M))$ s.t.

$$\forall n < \max(\mathrm{dom}(f)) \,.\, f(n) \vdash_M f(n+1)$$

with $\mathrm{dom}(f)$ being the domain of f. The set $\mathcal{D}_{\mathrm{I}}(M)$ contains all elements of $\mathcal{D}(M)$ starting with the initial element $f(0) = (\bot, (q_0, \lambda, \square))$. Furthermore, the set $\mathcal{D}_{max}(M, w)$ of maximal finite-length derivations of a word $w \in \Sigma^*$ is defined by

$$\mathcal{D}_{max}(M, w) := \left\{ f \in \mathcal{D}_{\mathrm{I}}(M) \,\middle|\, \begin{pmatrix} \exists e, q, s \,.\, f(\max(\mathrm{dom}(f))) = (e, (q, w, s)) \\ \wedge \nexists e', q', s' \,.\, (e, (q, w, s)) \vdash_M (e', (q', w, s')) \end{pmatrix} \right\}.$$

The set of reachable configurations is defined by

$$\mathcal{C}_{\mathrm{reach}}(M) := \{c \in \mathcal{C}(M) | \exists d \in \mathcal{D}_{\mathrm{I}}(M), e, n \,.\, d(n) = (e, c)\}.$$

Using this definition, we define the *marked and the unmarked languages* generated by $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F) \in \mathrm{PDA}$ by

$$\mathrm{L_m}(M) := \{w \mid (q, w, s) \in \mathcal{C}_{\mathrm{reach}}(M) \wedge q \in F\} \text{ and} \tag{2}$$

$$\mathrm{L_{um}}(M) := \{w \mid (q, w, s) \in \mathcal{C}_{\mathrm{reach}}(M)\}, \tag{3}$$

respectively. The class of languages generated by PDA is the class of context free languages (CFL). We say that a PDA M realizes a DES $D = (\!| L_{\mathrm{um}}, L_{\mathrm{m}} |\!)$, iff $\mathrm{L_m}(M) = L_{\mathrm{m}}$ and $\mathrm{L_{um}}(M) = L_{\mathrm{um}}$ and, with some abuse of terminology, we say $D \in \mathrm{CFL}$ iff $L_{\mathrm{um}}, L_{\mathrm{m}} \in \mathrm{CFL}$.

A PDA M is a *deterministic pushdown automaton* (DPDA) (written $M \in \mathrm{DPDA}$) iff distinct steps from a reachable configuration append distinct elements of Σ to the history variable. Note that this implies that the existence of an outgoing λ-transition in state q requiring stack-top γ prevents other outgoing transition in q requiring stack-top γ. The class of languages generated by DPDA is the class of deterministic context free languages (DCFL).

Example 2. The PDA M_O depicted in Figure 1 is also a DPDA since all required properties hold. Furthermore, the marked and unmarked language of M_O are given by

$$\begin{aligned} \mathrm{L_m}(M_O) = \big\{ & ac, aac, a^{2k+1}c(bb)^l, a^{2k+2}c(bb)^l, a^{2k+3}c(bb)^l bu, \\ & a^{2k+2}c(bb)^l bu, ad, a^{2k}d(bb)^l, a^{2k+1}d(bb)^l, \\ & a^{2k+2}d(bb)^l bu, a^{2k+1}d(bb)^l bu \;\big|\; k, l \in \mathbf{N}, k > 0, l \leq k \big\} \end{aligned}$$

[2] s is always pushed onto the stack with the right-most symbol first.

[3] We use the dummy symbol $\bot$ to define the initial transition, and, occasionally, when the pre-transition is irrelevant for the context.

[4] since $M \in \mathrm{LLF}$

and $\mathrm{L_{um}}(M_O) = \overline{\mathrm{L_m}(M_O)}$, since no blocking situations occur in M_O. This can be verified in Figure 1 (i) by checking that in every non-marking state $\tilde{q}$ either (a) an outgoing transition exists for both stack-tops $\square$ and $\bullet$, or (b) the configuration $(\tilde{q}, \cdot, \gamma)$ is not reachable if no outgoing transition with stack-top γ exists (e.g., $(q_1, \cdot, \square)$ is not reachable) and (ii) observing that no dead locks or infinite loops visiting only non-marking states exist. ◁

A *nondeterministic finite automaton* (NFA) can be viewed as a special PDA which does neither have λ-transitions, nor a stack (see Figure 2). Therefore, we can formally define a PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ to be an NFA (written $M \in$ NFA) iff whenever $(q, \sigma, \gamma, s, q') \in \delta$, then $\gamma = s = \square$ and $\sigma \in \Sigma$. Additionally, *deterministic finite automata* (DFA) are NFA which are deterministic. The class of languages generated by DFA is the class of regular languages (REG) [5].

Example 3. Consider the input alphabet $\Sigma = \{a, b, c, d, u\}$, the state set $Q = \{p_0, p_1, p_2, p_3\}$, the set of marking states $F = \{p_1, p_3\}$ and the initial state p_0. Then the automaton M_P in Figure 2 is a DFA. The transition $(p, \sigma, \square, \square, p') \in \delta$ is depicted by an edge from p to p' labeled by σ.

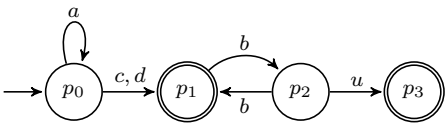

Figure 2. DFA M_P in Example 3.

The marked and unmarked languages of M_P are given by [6]

$\mathrm{L_m}(M_P) = \{a^n(c+d), a^n(c+d)(bb)^m(bu+\lambda) \mid n, m \in \mathbf{N}\}$

and $\mathrm{L_{um}}(M_P) = \overline{\mathrm{L_m}(M_P)}$, since no blocking situations occur in M_P. ◁

3. SUPERVISORY CONTROL REVISITED

In the context of SCT, a controller $D_C = \langle\!| L_{\mathrm{Cum}}, L_{\mathrm{Cm}} |\!\rangle$ is a proper minimally restrictive supervisor for a plant $D_P = \langle\!| L_{\mathrm{Pum}}, L_{\mathrm{Pm}} |\!\rangle$ and a specification $D_S = \langle\!| L_{\mathrm{Sum}}, L_{\mathrm{Sm}} |\!\rangle$, if D_C (i) is not preventing uncontrollable events (i.e., $((L_{\mathrm{Pum}} \cap L_{\mathrm{Cum}}) \cdot \Sigma_{\mathsf{uc}}) \cap L_{\mathrm{Pum}} \subseteq (L_{\mathrm{Pum}} \cap L_{\mathrm{Cum}})$), (ii) generates a closed loop $D_{cl} = \langle\!| L_{\mathrm{clum}}, L_{\mathrm{clm}} |\!\rangle$, with $L_{\mathrm{clum}} = L_{\mathrm{Pum}} \cap L_{\mathrm{Cum}}$ and $L_{\mathrm{clm}} = L_{\mathrm{Pm}} \cap L_{\mathrm{Cm}}$, which contains as many words generated by the plant as possible while respecting the specification, and (iii) always guides the system to a marking state (i.e., $L_{\mathrm{clum}} \subseteq \overline{L_{\mathrm{clm}}}$). In this case L_{clm} is the so called marked supremal controllable (nonblocking) sublanguage of $L_{\mathrm{Pm}} \cap L_{\mathrm{Sm}}$ (see Wonham and Ramadge, 1987). Since different controllers can generate the same closed loop, there exists no unique minimally restrictive supervisor for D_P and D_S. Obviously, the closed loop itself is a proper supervisor, giving $D_C = \langle\!| L_{\mathrm{clum}}, L_{\mathrm{clm}} |\!\rangle$. Wonham and Ramadge (1987, Lemma 2.1) introduced the monotonic operator $\Omega : \Sigma^* \to \Sigma^*$ defined by

$$\Omega(L_{\mathrm{m}}) = \{w \in L_{\mathrm{m}} \mid \forall w' \sqsubseteq w \,.\, \mathrm{ContW}(\overline{L_{\mathrm{m}}}, L_{\mathrm{Pum}}, \Sigma_{\mathsf{uc}}, w')\} \subseteq L_{\mathrm{m}}. \quad (4)$$

[5] Note that for every language L_{um} generated by a NFA, there also exists a DFA generating L_{um} (see Hopcroft and Ullman, 1979, p.22). However, this is not true for PDA and DPDA, implying DCFL $\subset$ CFL.

[6] The term $(x + y)$ denotes "x or y".

By iteratively applying Ω, starting with $L_{\mathrm{Sm}} \cap L_{\mathrm{Pm}}$, one obtains L_{clm} as the (unique) greatest fixed point. Wonham and Ramadge (1987, Lemma 2.1) showed that for $L_{\mathrm{Sm}}, L_{\mathrm{Pm}} \in$ REG an implementable algorithm working on DFA to calculate $L_{\mathrm{clm}} \in$ REG exists.

Now consider a plant $D_P = \langle\!| L_{\mathrm{Pum}}, L_{\mathrm{Pm}} |\!\rangle \in$ REG and a specification $D_S = \langle\!| L_{\mathrm{Sum}}, L_{\mathrm{Sm}} |\!\rangle \in$ DCFL. In this case we have $\langle\!| L_{\mathrm{Sum}} \cap L_{\mathrm{Pum}}, L_{\mathrm{Sm}} \cap L_{\mathrm{Pm}} |\!\rangle \in$ DCFL (see Hopcroft and Ullman, 1979, p.135), realizable by a DPDA. Now observe the following: in one iteration of the fixed-point algorithm the language $\overline{L_{\mathrm{m}}}$ is needed to calculate $\Omega(L_{\mathrm{m}})$. When implementing this algorithm using a DPDA-realization of the DES $D = \langle\!| L_{\mathrm{um}}, L_{\mathrm{m}} |\!\rangle$, the language $\overline{L_{\mathrm{m}}}$ is (in general) not easily obtained. However, if D is nonblocking, we can use $L_{\mathrm{um}} = \overline{L_{\mathrm{m}}}$. Therefore, to implement the iterative calculation of Ω, starting with $L_{\mathrm{Sm}} \cap L_{\mathrm{Pm}} \in$ DCFL, one has to iteratively (i) construct a DPDA M'_O from a nonblocking DPDA M_O s.t. $\Omega(\mathrm{L_m}(M_O)) = \mathrm{L_m}(M'_O)$ and (ii) make M'_O nonblocking. The connection between the language characterization of the supervisory control problem and its automaton based construction using these two basic subfunctions is investigated in detail in the companion paper by Schneider, Schmuck, Raisch, and Nestmann (2014). There, explicit fixed-point constructions and soundness proofs are given and it is shown that the resulting controller can be realized by a DPDA.

We show in the remainder of this paper that there exists an algorithm working on DPDA which realizes step (i) and returns a DPDA. An algorithm realizing step (ii) is presented by Schneider and Nestmann (2014).

Remark 1. The implementable algorithm to calculate $L_{\mathrm{clm}} \in$ REG presented by Wonham and Ramadge (1987, Lemma 5.1) iteratively calculates the *unmarked* controllable sublanguage of $L_{\mathrm{um}} = \overline{L_{\mathrm{m}}}$, i.e.,

$$\Omega(L_{\mathrm{um}}) = \{w \in L_{\mathrm{um}} \mid \mathrm{ContW}(L_{\mathrm{um}}, L_{\mathrm{Pum}}, \Sigma_{\mathsf{uc}}, w)\},$$

and its nonblocking sublanguage $\overline{\Omega(\overline{L_{\mathrm{m}}}) \cap L_{\mathrm{Sm}} \cap L_{\mathrm{Pm}}}$. In contrast, the algorithm introduced in the next section iteratively calculates the *marked* controllable sublanguage $\Omega(L_{\mathrm{m}})$ and its prefix closure. ◁

4. COMPUTABILITY OF Ω FOR DCFL

In this section, using the DFA M_P realizing $D_P = \langle\!| L_{\mathrm{Pum}}, L_{\mathrm{Pm}} |\!\rangle$, we derive a sequence of automaton manipulations starting with the DPDA M_O and generating a DPDA M'_O s.t. $\Omega(\mathrm{L_m}(M_O)) = \mathrm{L_m}(M'_O)$. We assume that $\mathrm{L_m}(M_O) \subseteq L_{\mathrm{Pm}}$. This is not a restriction of generality, as Ω is monotonic and its iterative application is initialized with $L_{\mathrm{Pm}} \cap L_{\mathrm{Sm}}$. We furthermore assume that M_O is nonblocking, i.e., $\mathrm{L_{um}}(M_O) \subseteq \overline{\mathrm{L_m}(M_O)}$. If this assumption does not hold, the algorithm to remove blocking situations is applied first.

The first task is to find all controllability problems in M_O. Intuitively, a controllability problem occurs when a word $w \in \Sigma^*$ is generated by M_P and M_O reaching states p and q (by using maximal derivations), respectively, and M_P can generate an uncontrollable symbol $\mu \in \Sigma_{\mathsf{uc}}$ in p while M_O cannot generate this symbol in q. To obtain the states p and q reached when generating the same word $w \in \Sigma^*$, a product automaton is constructed, following, e.g., Hopcroft and Ullman (1979, p.135).

Definition 1. Let $M_P = (Q_P, \Sigma, \{\Box\}, \delta_P, q_{P0}, \Box, F_P) \in$ DFA and $M_O = (Q_O, \Sigma, \Gamma, \delta_O, q_{O0}, \Box, F_O) \in$ DPDA. Then the product automaton $M_\times = M_P \times M_O$ is defined by $M_\times = (Q_\times, \Sigma, \Gamma, \delta_\times, q_{\times 0}, \Box, F_\times)$ with $Q_\times = Q_P \times Q_O$, $q_{\times 0} = (q_{P0}, q_{O0})$, $F_\times = F_P \times F_O$ and

$$\delta_\times = \left\{ ((p,q),\sigma,\gamma,s,(p',q')) \;\middle|\; \begin{pmatrix} \sigma \in \Sigma \\ \wedge (q,\sigma,\gamma,s,q') \in \delta_O \\ \wedge (p,\sigma,\Box,\Box,p') \in \delta_P \end{pmatrix} \vee \begin{pmatrix} \sigma = \lambda \\ \wedge p = p' \\ \wedge (q,\lambda,\gamma,s,q') \in \delta_O \end{pmatrix} \right\}.$$

◁

Lemma 1. Let $M_P \in$ DFA and $M_O \in$ DPDA s.t. $\mathrm{L_m}(M_O) \subseteq \mathrm{L_m}(M_P)$ and $\mathrm{L_{um}}(M_O) \subseteq \overline{\mathrm{L_m}(M_O)}$. Then (i) $\mathrm{L_m}(M_P \times M_O) = \mathrm{L_m}(M_O)$, (ii) $\mathrm{L_{um}}(M_P \times M_O) = \mathrm{L_{um}}(M_O)$, (iii) $M_\times = M_P \times M_O \in$ DPDA, and (iv) $\times$ is implementable [7]. ◁

Proof 1. Since $M_\times = M_P \times M_O$ is a product automaton in the usual sense, we have $\mathrm{L_m}(M_P \times M_O) = \mathrm{L_m}(M_P) \cap \mathrm{L_m}(M_O)$ and $\mathrm{L_{um}}(M_P \times M_O) = \mathrm{L_{um}}(M_P) \cap \overline{\mathrm{L_{um}}(M_O)}$. Having $\mathrm{L_m}(M_O) \subseteq \mathrm{L_m}(M_P)$, $\mathrm{L_{um}}(M_O) \subseteq \overline{\mathrm{L_m}(M_O)}$ and $\overline{\mathrm{L_{um}}(M_P)} = \mathrm{L_{um}}(M_P)$ implies $\mathrm{L_{um}}(M_O) \subseteq \mathrm{L_{um}}(M_P)$, which immediately proves (i) and (ii). (iii) and (iv) follow from Hopcroft and Ullman (1979, p.135). □

Example 4. Consider the DPDA M_O in Figure 1 and the DFA M_P in Figure 2. It can be easily verified that $\mathrm{L_m}(M_O) \subseteq \mathrm{L_m}(M_P)$, since the state and transition structures of M_O and M_P are identical and the usage of a stack only prevents certain transitions. Furthermore, $\mathrm{L_{um}}(M_O) \subseteq \overline{\mathrm{L_m}(M_O)}$ from Example 2. The (accessible part of the) product automaton $M_\times = M_P \times M_O$ is depicted in Figure 3 and does obviously generate the same marked and unmarked language as M_O. ◁

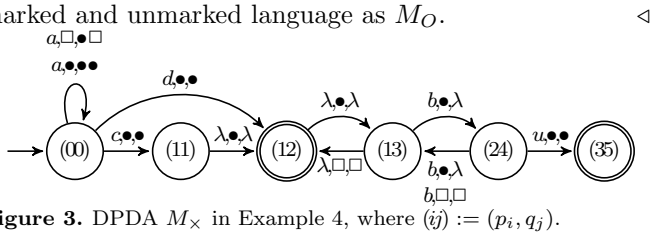

Figure 3. DPDA $M_\times$ in Example 4, where $(ij) := (p_i, q_j)$.

Unfortunately, in contrast to the DFA algorithm by Wonham and Ramadge (1987, Lemma 5.1), it is not possible to remove controllability problems in M_O in a minimally restrictive fashion by deleting states or edges in $M_\times$. This is due to the following observations: (i) it is possible that controllability problems occur in one state for a subset of possible stack-tops only (e.g., if $u \in \Sigma_{\mathsf{uc}}$, $M_\times$ in Figure 3 has a controllability problem in (p_2, q_4) for stack-top $\Box$, only). Therefore, removing this state may falsely delete controllable words. (ii) It is generally not possible to uniquely prevent a certain stack-top symbol in a given state by removing certain pre-transitions as one incoming transition can generate more than one stack-top [8] (e.g., the transition $((p_1,q_3),b,\bullet,\lambda,(p_2,q_4))$ in Figure 3 generates as stack-top the symbol which is currently in the stack underneath $\bullet$). (iii) Controllability problems are not easily observable in states $(\tilde{p},\tilde{q}) \in Q_\lambda$ (e.g., (p_1,q_1), (p_1,q_2) in Figure 3) as it is not possible to determine in $(\tilde{p},\tilde{q})$ whether an uncontrollable event $\mu \in \Sigma_{\mathsf{uc}}$ generated by M_P in $\tilde{p}$ is also generated by $M_\times$ after a (finite [9]) sequence of λ-transitions starting in $(\tilde{p},\tilde{q})$. If such a controllability problem occurs, it will be resolved at the final state $(\tilde{p},\tilde{q}')$ of the λ-transition sequence. However, observe that $(\tilde{p},\tilde{q}) \in F_\times$ and $(\tilde{p},\tilde{q}') \notin F_\times$ implies that the word $\tilde{w} \in \mathrm{L_m}(M_\times)$ with $((\tilde{p},\tilde{q}),\tilde{w},s),((\tilde{p},\tilde{q}'),\tilde{w},s') \in \mathcal{C}_{\mathrm{reach}}(M_\times)$ (having a controllability problem) is not removed from the marked language by removing $(\tilde{p},\tilde{q}')$ in $M_\times$. To remove words $w \in \mathrm{L_m}(M_\times)$ with a controllability problem from the marked language, we will ensure that *only* its maximal derivation $f \in \mathcal{D}_{max}(M_\times, w)$ ends in a marking state.
We therefore split states and redirect transitions in a particular way, such that deleting states with a controllability problem deletes *all* words $w \in \mathrm{L_m}(M_\times)$ *(and only those)* having a controllability problem, as required by Ω in (4). For this purpose we introduce four new state types: *regular* $(\cdot_r)$ and *special* $(\cdot_s)$ *main* states

$$\mathcal{M}_r(Q) := \{\langle q\rangle_r \mid q \in Q\}$$
$$\mathcal{M}_s(Q) := \{\langle q\rangle_s \mid q \in Q\}$$

and *regular* $(\cdot_r)$ and *special* $(\cdot_s)$ *auxiliary* states

$$\mathcal{A}_r(Q,\Gamma) := \{\langle q,\gamma\rangle_r \mid q \in Q \wedge \gamma \in \Gamma\}$$
$$\mathcal{A}_s(Q,\Gamma) := \{\langle q,\gamma\rangle_s \mid q \in Q \wedge \gamma \in \Gamma\}$$

where $\mathcal{M}(Q) = \mathcal{M}_r(Q) \cup \mathcal{M}_s(Q)$ and $\mathcal{A}(Q,\Gamma) = \mathcal{A}_r(Q,\Gamma) \cup \mathcal{A}_s(Q,\Gamma)$ are the sets of all main and all auxiliary states, respectively. Hence, every state is split into two entities consisting of $|\Gamma|+1$ states each. Using these new states, we define a function splitting states and redirecting transitions.

Definition 2. Let $M = (Q,\Sigma,\Gamma,\delta,q_0,\Box,F)$. Then the split automaton $M_{Sp} = \mathrm{SPLIT}(M)$ is defined by $M_{Sp} = (Q_{Sp},\Sigma,\Gamma,\delta_{Sp},q_{Sp\,0},\Box,F_{Sp})$ with $Q_{Sp} = \mathcal{M}(Q) \cup \mathcal{A}(Q,\Gamma)$, $q_{Sp\,0} = \langle q_0\rangle_r$, $F_{Sp} = (\mathcal{A}_r(F,\Gamma) \cup \mathcal{A}_s(Q,\Gamma)) \setminus Q_\lambda(M_{Sp})$ and

$$\delta_{Sp} = \left(\begin{array}{l} \{(\langle p\rangle_r,\lambda,\gamma,\gamma,\langle p,\gamma\rangle_r) | p \in Q, \gamma \in \Gamma\} \\ \cup \{(\langle p\rangle_s,\lambda,\gamma,\gamma,\langle p,\gamma\rangle_s) | p \in Q, \gamma \in \Gamma\} \\ \cup \left\{ (\langle p,\gamma\rangle_r,\sigma,\gamma,s,\langle p'\rangle_r) \;\middle|\; \begin{pmatrix} (p,\sigma,\gamma,s,p') \in \delta \\ \wedge (p \notin F \vee \sigma \neq \lambda) \end{pmatrix} \right\} \\ \cup \{(\langle p,\gamma\rangle_r,\lambda,\gamma,s,\langle p'\rangle_s) | (p,\lambda,\gamma,s,p') \in \delta \wedge p \in F\} \\ \cup \{(\langle p,\gamma\rangle_s,\lambda,\gamma,s,\langle p'\rangle_s) | (p,\lambda,\gamma,s,p') \in \delta\} \\ \cup \{(\langle p,\gamma\rangle_s,\sigma,\gamma,s,\langle p'\rangle_r) | (p,\sigma,\gamma,s,p') \in \delta \wedge \sigma \neq \lambda\} \end{array} \right)$$

◁

Example 5. Consider the product automaton $M_\times$ in Example 4 depicted in Figure 3. By using the abbreviation in Figure 4, its split version $M_{Sp} = \mathrm{SPLIT}(M_\times)$ is depicted in Figure 5, where all "obviously useless" entities (i.e., main with corresponding auxiliary states that are not connected by a path (ignoring the labels) to the initial and a marking state) were removed. ◁

Intuitively, the newly introduced auxiliary states work as a stack-top observer, separating outgoing transitions by their required stack-top, as shown in Example 5. Therefore, if a controllability problem occurs for one stack-top, we can delete the respective auxiliary state without falsely deleting controllable words. Furthermore, observe that all main states belong to $Q_\lambda(M_{Sp})$ while, due to determinism, auxiliary states can either belong to

[7] We say an algorithm is implementable if it can be realized by a computer program.
[8] That this is the reason why the algorithm presented by Griffin (2008) does not give a minimally restrictive controller.
[9] As $M_O \in$ LLF.

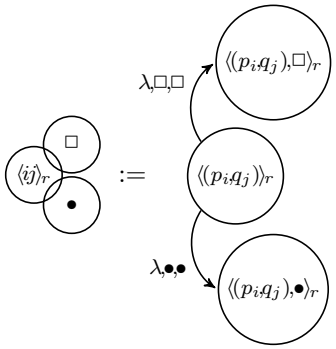

Figure 4. Graphical abbreviation of one entity.

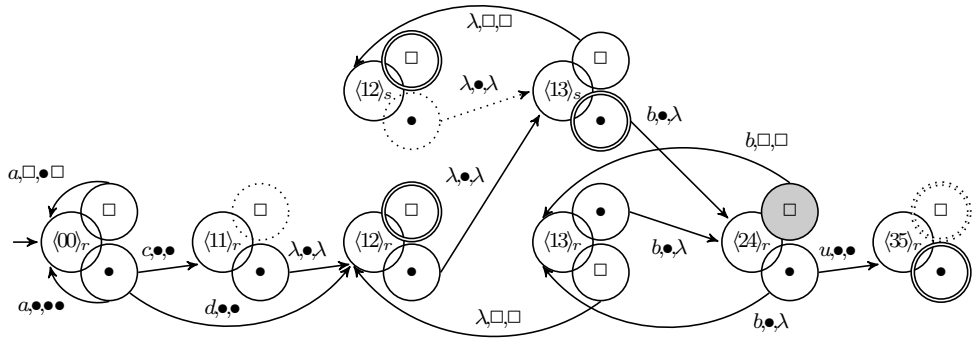

Figure 5. DPDA M_{Sp} in Example 5. Non-accessible states and transitions are dotted. The uncontrollable state NCS = $\{\langle(p_2,q_4),\square\rangle_r\}$ is indicated in gray.

the set $Q_\lambda(M_{Sp})$ or do not have outgoing λ-transitions at all. This uniquely defines the subset of states (i.e., $\mathcal{A}(Q,\Gamma)\setminus Q_\lambda(M_{Sp})$) for which controllability can and must efficiently be tested. Finally, the new special states are used to ensure that only the states reached by maximal derivations of words $w \in \mathrm{L_m}(M_O)$ are marking. Observe that the construction in SPLIT ensures, that final states of maximal derivations always end in the set $\mathcal{A}(Q,\Gamma) \setminus Q_\lambda(M_{Sp})$, formally,

$$\forall\, w \in \mathrm{L_{um}}(M_{Sp}), f \in \mathcal{D}_{max,M_{Sp}}(w)\,.\, \pi_1(\pi_2(f(\max(\mathrm{dom}(f))))) \in (\mathcal{A}(Q,\Gamma)\setminus Q_\lambda(M_{Sp}))\,. \tag{5}$$

Therefore, shifting the marking into those states ensures that uncontrollable words can be removed from the marked language by removing states in the set $\mathcal{A}(Q,\Gamma)\setminus Q_\lambda(M_{Sp})$. Before removing states with controllability problems we show that SPLIT does not change the marked and unmarked language of its input.

Lemma 2. Let $M = (Q,\Sigma,\Gamma,\delta,q_0,\square,F)$ and $M_{Sp} = \mathrm{SPLIT}(M)$. Then (i) $\mathrm{L_{um}}(M_{Sp}) = \mathrm{L_{um}}(M)$, (ii) $\mathrm{L_m}(M_{Sp}) = \mathrm{L_m}(M)$, (iii) $\mathrm{SPLIT}(M) \in$ DPDA, and (iv) SPLIT is implementable. ◁

Proof 2. see Appendix B. □

Technically, we are now ready to delete all states that have a controllability problem. However, both $\times$ and SPLIT introduce non-accessible states and transitions. Trimming the automaton M_{Sp} prior to the removal of controllability problems has the advantage that termination of the fixed point algorithm over Ω can be easily verified, since no states are removed, if the automaton is trim, nonblocking and no further controllability problems are present. Therefore, following Griffin (2006, Thm.4.1), we introduce an algorithm to remove non-accessible states and transitions in DPDA. Here, in contrast to DFA, transitions can be non-accessible even if they connect accessible states, since the required stack-top might not be available at its pre-state.

Definition 3. Let $M = (Q,\Sigma,\Gamma,\delta,q_0,\square,F)$. Then the accessible part $\mathrm{AC}(M)$ of M is defined by $M_{Ac} = (Q_{Ac},\Sigma,\Gamma,\delta_{Ac},q_0,\square,F_{Ac})$ s.t. $Q_{Ac} = Q \setminus \mathrm{Q_{na}}(M)$, $F_{Ac} = F \setminus \mathrm{Q_{na}}(M)$ and $\delta_{Ac} = \delta \setminus \delta_{\mathrm{na}}(M)$, where

$$\mathrm{Q_{na}}(M) := \{q \in Q\setminus\{q_0\} \mid \mathrm{L_m}((Q,\Sigma,\Gamma,\delta,q_0,\square,\{q\})) = \emptyset\}$$

are the non accessible states and

$$\delta_{\mathrm{na}}(M) := \left\{ e=(q,x,\gamma,s,q') \in \delta \mid \forall\, r \notin Q, \delta' = (\delta \cup \{(q,\lambda,\gamma,\gamma,r)\}) \setminus \{e\}\,.\, \mathrm{L_m}((Q\cup\{r\},\Sigma,\Gamma,\delta',q_0,\square,\{r\})) = \emptyset \right\}$$

are the non accessible transitions. ◁

Example 6. The accessible part of the automaton M_{Sp} in Example 5, depicted in Figure 5, is obtained by removing the dotted states, giving $M_{Ac} = \mathrm{AC}(M_{Sp})$. ◁

Trimming a DPDA does not change its marked and unmarked languages.

Lemma 3. Let $M = (Q,\Sigma,\Gamma,\delta,q_0,\square,F)$. Then (i) $\mathrm{L_m}(\mathrm{AC}(M)) = \mathrm{L_m}(M)$, (ii) $\mathrm{L_{um}}(\mathrm{AC}(M)) = \mathrm{L_{um}}(M)$, (iii) $\mathrm{AC}(M) \in$ DPDA, and (iv) AC is implementable. ◁

Proof 3. Observe that $\mathrm{Q_{na}}(M)$ is the set of states not reachable by derivations of words $w \in \mathrm{L_{um}}(M)$ and $\delta_{\mathrm{na}}(M)$ is the set of transitions that are not part of derivations of words $w \in \mathrm{L_{um}}(M)$. Therefore removing these states and transitions does not change the unmarked language, implying (ii). Since accessible marking states are preserved, also the marked language (as a subset of the unmarked language) is left unchanged, implying (i). Since $M \in$ DPDA, (iii) immediately follows as we only remove states and transitions. (iv) follows, as the emptiness of a CFL is decidable(see Hopcroft and Ullman, 1979, p.137). □

As the final step of our algorithm, we will now identify auxiliary states that have a controllability problem with respect to the plant, and remove them. This is done in analogy to the DFA algorithm by Wonham and Ramadge (1987), while ignoring all states in Q_λ.

Definition 4. Let $\Sigma_{\mathsf{uc}} \subseteq \Sigma$, $M_P \in$ DFA, $M_O \in$ DPDA, $M_\times = M_P \times M_O = (Q_\times,\Sigma,\Gamma,\delta_\times,q_{\times 0},\square,F_\times)$, and $M_{Ac} = \mathrm{AC}(\mathrm{SPLIT}(M_\times)) = (Q_{Ac},\Sigma,\Gamma,\delta_{Ac},q_0,\square,F_{Ac})$.
Then the automaton with removed non-controllable auxiliary states is defined by $\mathrm{RNCS}(M_{Ac}) = (Q_R,\Sigma,\Gamma,\delta_R,q_0,\square,F_R)$, where $Q_R = Q_{Ac} \setminus \mathrm{NCS}$, $F_R = F_{Ac} \setminus \mathrm{NCS}$ and $\delta_R = \{(q,x,\gamma,w,q') \in \delta' | q,q' \in Q_R\}$, where

$$\mathrm{NCS} := \left\{ \langle(p,q),\gamma\rangle \in ((Q_{Ac}\cap\mathcal{A}(Q_\times,\Gamma))\setminus Q_\lambda(M_{Ac})) \mid \exists \mu \in \Sigma_{\mathsf{uc}}\,.\, \begin{pmatrix} \exists p'\,.\,(p,\mu,\square,\square,p') \in \delta_P \\ \wedge \forall s,r\,.\,(\langle(p,q),\gamma\rangle,\mu,\gamma,s,r) \notin \delta_{Ac} \end{pmatrix} \right\}$$

◁

Example 7. Consider the accessible split automaton M_{Ac} in Example 6 depicted in Figure 5 and let $\Sigma_{\mathsf{uc}} = \{u\}$. Then the set of uncontrollable auxiliary states of M_{Ac} is given by NCS $= \{\langle(p_2, q_4), \Box\rangle_r\}$, indicated in gray in Figure 5. Removing this state and all ingoing and outgoing transitions generates the automaton $M'_O = \mathrm{RNCS}(M_{Ac})$. Observe that M'_O has a blocking situation in $\langle(p_2, q_4)\rangle_r$ for all words w whose derivations generate stack-top $\Box$ in $\langle(p_2, q_4)\rangle_r$, which are $w \in \{a^{2k+1}c(bb)^k, a^{2k}d(bb)^k \mid k > 0\}$. Note that removing these words (and their extensions) from $\mathrm{L_{um}}(M'_O)$, i.e., making M'_O nonblocking, requires, depending on the number of occurrences of a and b in the past, the restriction of possible future steps. This implies that the nonblocking version of M'_O has to be structurally different from M_O in a non-obvious manner. This makes the problem of *automatically* removing blocking situations in DPDA challenging. ◁

As our main result, we now prove that the introduced sequence of automaton manipulations removes those (and only those) marked words which have a prefix which is uncontrollable w.r.t. the plant.

Theorem 1. Let $\Sigma = \Sigma_{\mathsf{c}} \cup \Sigma_{\mathsf{uc}}$ s.t. $\Sigma_{\mathsf{c}} \cap \Sigma_{\mathsf{uc}} = \emptyset$; $M_P \in$ DFA and $M_O \in$ DPDA s.t. $\mathrm{L_m}(M_O) \subseteq \mathrm{L_m}(M_P)$ and $\mathrm{L_{um}}(M_O) \subseteq \overline{\mathrm{L_m}(M_O)}$. Then

$$\mathrm{L_m}(\mathrm{RNCS}(\mathrm{AC}(\mathrm{SPLIT}(M_P \times M_O)))) = \Omega(\mathrm{L_m}(M_O)).$$

◁

Proof 4. Using $M_{Ac} = \mathrm{AC}(\mathrm{SPLIT}(M_P \times M_O))$ and $M'_O = \mathrm{RNCS}(M_{Ac})$, we have the following observations:
(A) From Lemma 1, 2 and 3, follows that $\mathrm{L_{um}}(M_{Ac}) = \mathrm{L_{um}}(M_O)$ and $\mathrm{L_m}(M_{Ac}) = \mathrm{L_m}(M_O)$. **(B)** Pick $w \in \mathrm{L_{um}}(M_O)$ with $\neg\,\mathrm{ContW}(\mathrm{L_{um}}(M_O), L_{\mathrm{P\,um}}, \Sigma_{\mathsf{uc}}, w)$. Using (5) , we can fix $f_w \in \mathcal{D}_{max,M_{Ac}}(w)$ and its final state $\langle(p, q), \gamma\rangle \in ((Q_{Ac} \cap \mathcal{A}(Q_\times, \Gamma)) \setminus Q_\lambda(M_{Ac}))$. Using Definition 1 this implies that M_P uniquely accepts w in p. Now (1), (3) and (A) imply that there exists $\mu \in \Sigma_{\mathsf{uc}}$ s.t. $(p, \mu, \Box, \Box, p') \in \delta_P$ and $\forall s, r\ .\ (\langle(p, q), \gamma\rangle, \mu, \gamma, s, r) \notin \delta_{Ac}$, giving that $\langle(p, q), \gamma\rangle \in$ NCS. **(C)** It follows from the construction in Definition 2 that there exist derivations generating w and ending in a main state $\langle(p, q)\rangle$ and possibly also in other preceding states in $Q_\lambda(M_{Ac})$. Observe that all states reached by generating w, except for $\langle(p, q), \gamma\rangle$, are in $Q_\lambda(M_{Ac})$ and therefore, by construction, not in F_{Ac}. **(D)** Definition 4 implies that $w \in \mathrm{L_{um}}(M'_O)$ iff there exists a (possibly non-maximal) derivation $f' \in \mathcal{D}(M_{Ac})$ with final configuration (r, w, s) s.t. $\forall n \in \mathrm{dom}(f'), n \leq \max(\mathrm{dom}(f'))\ .\ \pi_2(f'(n)) \notin$ NCS. $M_{Ac} \in$ DPDA implies that for all prefixes $w' \sqsubset w$ (but not for w itself) holds that their maximal derivations are a prefix of f'. Therefore, using (B) implies

$$w \in \mathrm{L_{um}}(M'_O) \ \rightarrow\ \big(\forall w' \sqsubset w\ .\ \mathrm{ContW}(\mathrm{L_{um}}(M_O), L_{\mathrm{P\,um}}, \Sigma_{\mathsf{uc}}, w')\big) \quad \text{and}$$

$$\big(\forall w' \sqsubseteq w\ .\ \mathrm{ContW}(\mathrm{L_{um}}(M_O), L_{\mathrm{P\,um}}, \Sigma_{\mathsf{uc}}, w')\big) \ \rightarrow\ w \in \mathrm{L_{um}}(M'_O).$$

(E) Now assume $\neg\,\mathrm{ContW}(\mathrm{L_{um}}(M_O), L_{\mathrm{P\,um}}, \Sigma_{\mathsf{uc}}, w)$ and all $w' \sqsubset w$ are controllable. From (B) follows that $\langle(p, q), \gamma\rangle \in$ NCS is removed. This has two consequences: **(a)** we still have $w \in \mathrm{L_{um}}(M'_O)$ from (C),(D), and **(b)** if $w \in \mathrm{L_m}(M_O)$ we have $w \notin \mathrm{L_m}(M'_O)$ from (C), (D).

Combining (A), (D) and (E)(b) implies

$$w \in \mathrm{L_m}(\mathrm{RNCS}(\mathrm{AC}(\mathrm{SPLIT}(M_P \times M_O)))) \ \leftrightarrow\ \big(\forall w' \sqsubseteq w\ .\ \mathrm{ContW}(\mathrm{L_{um}}(M_O), L_{\mathrm{P\,um}}, \Sigma_{\mathsf{uc}}, w')\big),$$

which proves the statement. □

Remark 2. Observation (E)(a) in the proof of Theorem 1 is the reason why we do not implement the removal of all uncontrollable *unmarked* words, i.e., $\Omega(\mathrm{L_{um}}(M_O))$, used by Wonham and Ramadge (1987) as discussed in Remark 1. ◁

5. CONCLUSION

In this paper, we have presented an extension of SCT to situations, where the plant is realized by a DFA, but the specification is modeled as a DPDA, i.e., the specification language is a DCFL. In particular, we have presented an algorithm consisting of a sequence of automaton manipulations which, starting with a nonblocking DPDA, removes all (and only those) marked words having a prefix which is uncontrollable w.r.t. the plant. This algorithm was implemented as a plug-in in libFAUDES (2006-2013). The remaining part of a procedure to obtain a proper minimally restrictive supervisor, namely an algorithm removing blocking situations in DPDA, is presented by Schneider and Nestmann (2014). The connection between a language-theoretic characterization and the automata-based implementation is investigated in detail in the companion paper by Schneider, Schmuck, Raisch, and Nestmann (2014).

Appendix A. COUNTEREXAMPLE

In this section, the algorithm presented in (Griffin, 2008, p.827) and (Griffin, 2007, p.64) is applied to an example. It will be shown that this represents a counterexample to (Griffin, 2008, Theorem 3.5) and (Griffin, 2007, Theorem 5.2.5) since the final DPDA does not realize the supremal controllable sublanguage of the given prefix closed deterministic context free specification language (which is required to be a subset of the given prefix closed regular plant language). Therefore we claim that the problem of automatically calculating a supremal controllable sublanguage of a DCFL was not solved by Griffin (2007, 2008). The algorithm is initialized with a DFA G and a DPDA M realizing the plant and the specification, respectively, s.t. $\mathrm{L_m}(G) = \mathrm{L_{um}}(G)$, $\mathrm{L_m}(M) = \mathrm{L_{um}}(M)$ and $\mathrm{L_m}(M) \subseteq \mathrm{L_m}(G)$. Observe that the DFA G and the DPDA M depicted in Figure A.1 satisfy these requirements since their languages are given by

$$\mathrm{L_m}(G) = \mathrm{L_{um}}(G) = \{a^n, a^n b, a^n bu | n \in \mathbf{N}\} \quad \text{and}$$

$$\mathrm{L_m}(M) = \mathrm{L_{um}}(M) = \left\{ a^n, a^m b, a^k bu \mid n, m, k \in \mathbf{N}, m{>}0, k{>}1 \right\}.$$

Using these automata, the construction follows seven steps.

1. Construct M', depicted in Figure A.2, by making M scan its entire input, using the algorithm by Hopcroft and Ullman (1979, Lem.10.3.).

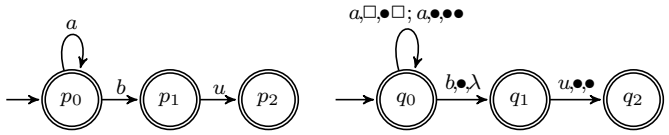

Figure A.1. DFA G (left) and DPDA M (right) realizing plant and specification, respectively.

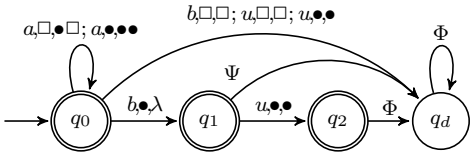

Figure A.2. DPDA M', with $\Psi = \Phi \setminus u,\bullet,\bullet$ and $\Phi = \{a,\square,\square; a,\bullet,\bullet; b,\square,\square; b,\bullet,\bullet; u,\square,\square; u,\bullet,\bullet\}$.

2. Construct a DPDA M'' that accepts the complement of $\mathrm{L_m}(M)$ using the algorithm by Hopcroft and Ullman (1979), Thm.10.1. Here M'' is identical to M' in Figure A.2 but with exchanged non-marking and marking states, i.e., $F'' = \{q_d\}$.
3. Construct a DPDA M''' that accepts $\mathrm{L_m^c}(M) \cap \mathrm{L_m}(G)$, i.e., calculate the cross product of G and M'' using the algorithm by Hopcroft and Ullman (1979), Thm.6.5.
4. Construct M_1, depicted in Figure A.3, as the accessible part of M''', using the algorithm by Griffin (2006), Thm.4.1.

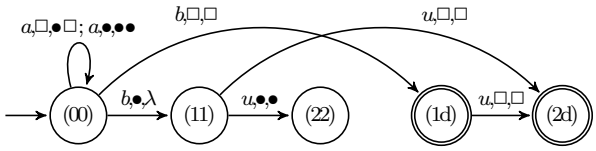

Figure A.3. DPDA M_1, with $(ij) := (p_i, q_j)$

5. Construct a predicting machine to observe so called μ-reverse paths using the algorithm by Hopcroft and Ullman (1979), p.240. Here, the construction simply defines an additional stack symbol μ_γ, $\forall \gamma \in \Gamma$ s.t.

$$\mu_\gamma := \left\{ q \in Q_1 \setminus F_1 \;\middle|\; \begin{array}{l} \exists\, q' \in F_1, v \in \Sigma_{\mathsf{uc}}^* \,. \\ \quad (\bot, (q, w, \gamma{\cdot}s)) \vdash_{M_1}^* (\bot, (q', w{\cdot}v, s')) \end{array} \right\}$$

which denotes the set of unmarked states in M_1 from which a derivation starting with stack-top γ, generating a sequence of uncontrollable symbols $v \in \Sigma_{\mathsf{uc}}^*$ and reaching a marking state q' (i.e., a so called μ-reverse path), exists. For M_1, depicted in Figure A.3, this gives $\mu_\square = \{(p_1, q_1)\}$ and $\mu_\bullet = \emptyset$. The predicting machine M_1^μ, depicted in Figure A.4, is then identical to M_1 but uses pairs $[\gamma, \mu_\gamma]$ as stack.

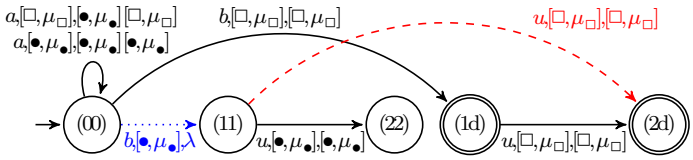

Figure A.4. DPDA M_1^μ with $\mu_\square = \{(p_1, q_1)\}$ and $\mu_\bullet = \emptyset$. The set of μ-reverse paths is depicted in red (dashed) while the set of edges in δ_{cp} is depicted in blue (dotted).

6. Construct M_2, depicted in Figure A.5, by deleting all transitions

$$\delta_{cp} := \left\{ (q, \sigma, \gamma, \gamma'{\cdot}s, q') \in \delta_M \;\middle|\; \begin{array}{l} \\ \left(\begin{array}{l} (q, \sigma, \gamma, [\gamma', \mu_{\gamma'}]{\cdot}s, q') \in \delta_{M_1^\mu} \\ \wedge \sigma \in \Sigma_{\mathsf{c}} \wedge q' \in \mu_{\gamma'} \end{array} \right) \end{array} \right\}$$

in M which produce a stack-top in q' which enables a μ-reverse path starting in q'. For M and M_1^μ, depicted in Figure A.1 and A.4, respectively, observe that $e = ((p_0, q_0), b, [\bullet, \mu_\bullet], \lambda, (p_1, q_1)) \in \delta_{M_1^\mu}$ is the only ingoing transition to (p_1, q_1) (where the only μ-reverse path starts for stack-top $\square$, since $\mu_\square = \{(p_1, q_1)\}$) and, since M_1 is trim, eventually leads to the stack-top $\square$ in (p_1, q_1). Using the corresponding transition to e in M, this gives $\delta_{cp} = \{(q_0, b, \bullet, \lambda, q_1)\}$. By deleting δ_{cp} in M, we obtain M_2, depicted in Figure A.5.
7. Construct M_3, depicted in Figure A.6, as the accessible part of M_2, using the algorithm by Griffin (2006), Thm.4.1. If δ_{cp} in step 6 is empty, the algorithm terminates. Otherwise, the algorithm is restarted with $M = M_3$.

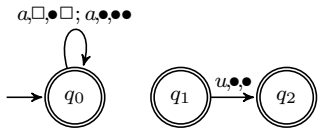

Figure A.5. DPDA M_2

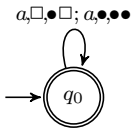

Figure A.6. DPDA M_3

Obviously, M_3 does not have further controllability problems. Therefore, the algorithm would redo steps 1-6 and then return M_3.

Now observe that the specification language $\mathrm{L_m}(M)$ restricts the plant language $\mathrm{L_m}(G)$ such that u cannot occur after exactly one a. This generates a controllability for the word ab only. Using (4) in Section 3, the supremal controllable sublanguage of $\mathrm{L_m}(M)$ for this example is given by

$$\begin{aligned} L_{\mathrm{clm}} &= \{w \in \mathrm{L_m}(M) \mid \forall w' \sqsubseteq w \,.\, w' \neq ab\} \\ &= \mathrm{L_m}(M) \setminus \{ab\} \\ &= \left\{a^n, a^m b, a^k bu \middle| n, m, k \in \mathbf{N}, m, k > 1\right\} \end{aligned} \tag{A.1}$$

implying that $\mathrm{L_m}(M_3) = \{a^n \mid n \in \mathbf{N}\}$ is a strict subset of L_{clm} which is an obvious contradiction to (Griffin, 2008, Thm.3.5). Furthermore, L_{clm} in (A.1) cannot be realized using the state and transition structure of M and only deleting existing transitions.

The automatic synthesis of a DPDA realizing L_{clm} for this example is provided as an example within our pushdown-plug-in for libFAUDES (2006-2013).

Appendix B. PROOF OF LEMMA 2

Here we give the proof of Lemma 2.

Proof 2. To simplify notation, we collect all states q, reachable by a finite sequence of λ-transitions from a configuration $(\tilde{q}, w, \tilde{s})$ with $\tilde{q} \in F$ in the set

$$Q_{rlt}(M, w) := \left\{ q \in Q \;\middle|\; \begin{array}{l} \exists\, f \in \mathcal{D}_1(M), n \in \mathbf{N}, \tilde{q} \in F \,. \\ \quad \left(\begin{array}{l} f(n) = (\tilde{e}, (\tilde{q}, w, \tilde{s})) \\ \wedge \exists n' > n \,.\, f(n') = (e, (q, w, s)) \end{array} \right) \end{array} \right\}.$$

(i) $\mathrm{L_{um}}(M_{Sp}) = \mathrm{L_{um}}(M)$: Let $q \in Q$, $w \in \Sigma^*$, $\gamma \in \Gamma$, $s \in \Gamma^*$ and $(q, w, \gamma{\cdot}s)$ be an arbitrary configuration reachable

by the initial derivation $f \in \mathcal{D}_{\mathrm{I}}(M)$, implying $w \in \mathrm{L}_{\mathrm{um}}(M)$. Observe that the state q is mapped to the states $\{\langle q\rangle_r, \langle q,\gamma\rangle_r\}$ if $q \notin Q_{rlt}(M,w)$ and to $\{\langle q\rangle_s, \langle q,\gamma\rangle_s\}$ if $q \in Q_{rlt}(M,w)$ [10]. Using this, we can show that the single-step relation

$$(e,(q,w,\gamma{\cdot}s)) \vdash_M ((q,\sigma,\gamma,s',q'),(q',w{\cdot}\sigma,s'{\cdot}s)) \quad \text{(B.1)}$$

is mimicked by a sequence of two single-step relations in M_{Sp} in four different ways

(a) $(q \notin Q_{rlt}(M,w) \wedge (\sigma \neq \lambda \vee q \notin F))$:

$$\begin{aligned}&(\check{e},(\langle q\rangle_r,w,\gamma{\cdot}s)) \vdash_{M_{Sp}} ((\langle q\rangle_r,\lambda,\gamma,\gamma,\langle q,\gamma\rangle_r),(\langle q,\gamma\rangle_r,w,\gamma{\cdot}s))\\ &\vdash_{M_{Sp}} ((\langle q,\gamma\rangle_r,\sigma,\gamma,s',\langle q'\rangle_r),(\langle q'\rangle_r,w{\cdot}\sigma,s'{\cdot}s))\end{aligned}$$

(b) $(q \in F \wedge \sigma = \lambda)$:

$$\begin{aligned}&(\check{e},(\langle q\rangle_r,w,\gamma{\cdot}s)) \vdash_{M_{Sp}} ((\langle q\rangle_r,\lambda,\gamma,\gamma,\langle q,\gamma\rangle_r),(\langle q,\gamma\rangle_r,w,\gamma{\cdot}s))\\ &\vdash_{M_{Sp}} ((\langle q,\gamma\rangle_r,\sigma,\gamma,s',\langle q'\rangle_s),(\langle q'\rangle_s,w{\cdot}\sigma,s'{\cdot}s))\end{aligned}$$

(c) $(q \in Q_{rlt}(M,w) \wedge \sigma = \lambda)$:

$$\begin{aligned}&(\check{e},(\langle q\rangle_s,w,\gamma{\cdot}s)) \vdash_{M_{Sp}} ((\langle q\rangle_s,\lambda,\gamma,\gamma,\langle q,\gamma\rangle_s),(\langle q,\gamma\rangle_s,w,\gamma{\cdot}s))\\ &\vdash_{M_{Sp}} ((\langle q,\gamma\rangle_s,\sigma,\gamma,s',\langle q'\rangle_s),(\langle q'\rangle_s,w{\cdot}\sigma,s'{\cdot}s))\end{aligned}$$

(d) $(q \in Q_{rlt}(M,w) \wedge \sigma \neq \lambda)$:

$$\begin{aligned}&(\check{e},(\langle q\rangle_s,w,\gamma{\cdot}s)) \vdash_{M_{Sp}} ((\langle q\rangle_s,\lambda,\gamma,\gamma,\langle q,\gamma\rangle_s),(\langle q,\gamma\rangle_s,w,\gamma{\cdot}s))\\ &\vdash_{M_{Sp}} ((\langle q,\gamma\rangle_s,\sigma,\gamma,s',\langle q'\rangle_r),(\langle q'\rangle_r,w{\cdot}\sigma,s'{\cdot}s))\end{aligned}$$

Therefore, using induction, we can always construct a derivation $f' \in \mathcal{D}_{\mathrm{I}}(M_{Sp})$, $n' \in \mathbf{N}$ s.t. $f'(n') = (e',(q',w,\gamma{\cdot}s))$ giving $w \in \mathrm{L}_{\mathrm{um}}(M_{Sp})$ and therefore, $\mathrm{L}_{\mathrm{um}}(M_{Sp}) \subseteq \mathrm{L}_{\mathrm{um}}(M)$. Using the fact that the relations in case (a)-(d) only occur, iff there exists a matching relation (B.1) in M and derivations in M_{Sp} can only be concatenated iff this is possible in M, the construction of single-step relations in M matching single-step relations in M_{Sp} gives the same cases as before, implying $\mathrm{L}_{\mathrm{um}}(M) \subseteq \mathrm{L}_{\mathrm{um}}(M_{Sp})$.

(ii) Show $\mathrm{L}_{\mathrm{m}}(M_{Sp}) = \mathrm{L}_{\mathrm{m}}(M)$: Let $w \in \mathrm{L}_{\mathrm{m}}(M)$ and $f \in \mathcal{D}_{\mathrm{I}}(M)$, $n \in \mathbf{N}$, $q \in F$ s.t. $f(n) = (e,(q,w,\gamma{\cdot}s))$ is the starting configuration in (B.1). If $\sigma \neq \lambda$ in (B.1), we know from part (i) and cases (a) and (d) that there exists a derivation $f' \in \mathcal{D}_{\mathrm{I}}(M_{Sp})$, $n' \in \mathbf{N}$ s.t. $f'(n') = (\check{e},(\langle q,\gamma\rangle_r,w,\gamma{\cdot}s))$ with $\langle q,\gamma\rangle_r \in F_{Sp}$ (since $q \in F$ and $\langle q,\gamma\rangle_r \notin Q_\lambda(M_{Sp})$), giving $w \in \mathrm{L}_{\mathrm{m}}(M_{Sp})$. Now let $\sigma = \lambda$. Since $M \in \mathrm{LLF}$, there exists $\sigma' \neq \lambda$ and a finite chain of singe-step relations, s.t.

$$\begin{aligned}&(\check{e},(q,w,\gamma{\cdot}s)) \vdash^*_M (\tilde{e},(\tilde{q},w,\tilde{\gamma}{\cdot}\tilde{s}))\\ &\vdash_M ((\tilde{q},\sigma',\tilde{\gamma},\tilde{s}',\tilde{q}'),(\tilde{q}',w{\cdot}\sigma',\tilde{s}'{\cdot}\tilde{s}))\end{aligned} \quad \text{or} \quad \text{(B.2)}$$

$$(\check{e},(q,w,\gamma{\cdot}s)) \vdash^*_M (\hat{e},(\hat{q},w,\hat{\gamma}{\cdot}\hat{s})) \not\vdash_M . \quad \text{(B.3)}$$

Then we can combine case (b),(c) and (d) from the proof of part (i) to mimic (B.2) by the finite chain

$$\begin{aligned}&(\check{e},(\langle q\rangle,w,\gamma{\cdot}s)) \vdash^*_{M_{Sp}} (\tilde{h},(\langle\tilde{q},\tilde{\gamma}\rangle_s,w,\tilde{\gamma}{\cdot}\tilde{s}))\\ &\vdash_{M_{Sp}} ((\langle\tilde{q},\tilde{\gamma}\rangle_s,\sigma',\tilde{\gamma},\tilde{s}',\langle\tilde{q}'\rangle_r),(\langle\tilde{q}'\rangle_r,w\sigma',\tilde{s}'{\cdot}\tilde{s}))\end{aligned}$$

and (B.3) by the finite chain

$$(\bot,(\langle q\rangle,w,\gamma{\cdot}s)) \vdash^*_{M_{Sp}} (\hat{h},(\langle\hat{q},\hat{\gamma}\rangle_s,w,\hat{\gamma}{\cdot}\hat{s})) \not\vdash_{M_{Sp}} .$$

[10] Observe that reaching q with a different initial derivation f' might result in a different mapping. E.g., the state (p_1,q_2) in Figure 3 is mapped to $\{\langle(p_1,q_2)\rangle_r, \langle(p_1,q_2),\bullet\rangle_r\}$ and $\{\langle(p_1,q_2)\rangle_s, \langle(p_1,q_2),\square\rangle_s\}$ when reached by an initial derivation generating the words $w = ad$ and $w = ac$, respectively.

Now observe that all states in $\mathcal{A}_s(Q,\Gamma) \cap Q_\lambda(Q_{Sp})$ can *only* have outgoing λ-transitions, due to the determinism of Q that is preserved in Q_{Sp}. Therefore, we have $\langle\tilde{q},\tilde{\gamma}\rangle_s, \langle\hat{q},\hat{\gamma}\rangle_s \notin Q_\lambda(M_{Sp})$ and therefore $\langle\tilde{q},\tilde{\gamma}\rangle_s, \langle\hat{q},\hat{\gamma}\rangle_s \in F_{Sp}$ by definition. This implies that there exists a derivation $f' \in \mathcal{D}_{\mathrm{I}}(M_{Sp})$, $n'' \in \mathbf{N}$ s.t. $f'(n') = (h,(p,w,r))$ s.t. $p \in F_{Sp}$, implying $w \in \mathrm{L}_{\mathrm{m}}(M_{Sp})$ and therefore $\mathrm{L}_{\mathrm{m}}(M) \subseteq \mathrm{L}_{\mathrm{m}}(\mathrm{SPLIT}(M))$. For the proof of $\mathrm{L}_{\mathrm{m}}(\mathrm{SPLIT}(M)) \subseteq \mathrm{L}_{\mathrm{m}}(M)$ observe that if w is accepted by a state $\langle q,\gamma\rangle_r \in F_{Sp}$, it follows from (i) and the construction of F_{Sp} that w is accepted by $q \in F$ in M. Now let w be accepted by a state $\langle q,\gamma\rangle_s \in F_{Sp}$. Then it follows from case (b) in the proof of (i) that there exists some state $p \in F$ that accepts w, since otherwise, $\langle q,\gamma\rangle_s \in F_{Sp}$ is not reachable by w. This again gives $w \in \mathrm{L}_{\mathrm{m}}(M)$.

Since we only split states, redirect existing transitions and add unique λ-transitions, **(iii)** and **(iv)** follow immediately from the construction. □

REFERENCES

Griffin, C. (2006). A note on deciding controllability in pushdown systems. *IEEE Transactions on Automatic Control*, 51(2), 334 – 337.

Griffin, C. (2008). A note on the properties of the supremal controllable sublanguage in pushdown systems. *IEEE Transactions on Automatic Control*, 53(3), 826 –829.

Griffin, C. (2007). *Decidability and optimality in pushdown control systems: A new approach to discrete event control*. Ph.D. thesis, The Pensylvania State University.

Hopcroft, J.E. and Ullman, J.D. (1979). *Introduction to Automata Theory, languages and computation*. Addison-Wesley Publishing company.

libFAUDES (2006-2013). Software library for discrete event systems. URL `http://www.rt.eei.uni-erlangen.de/FGdes/faudes`.

Masopust, T. (2012). A note on controllability of deterministic context-free systems. *Automatica*, 48(8), 1934–1937.

Ramadge, P. and Wonham, W. (1984). Supervisory control of a class of discrete event processes. In A. Bensoussan and J. Lions (eds.), *Analysis and Optimization of Systems*, volume 63 of *Lecture Notes in Control and Information Sciences*, 475–498. Springer Berlin Heidelberg.

Schneider, S. and Nestmann, U. (2014). Enforcing operational properties including blockfreeness for deterministic pushdown automata. `http://arxiv.org/abs/1403.5081`.

Schneider, S., Schmuck, A.-K., Raisch, J., and Nestmann, U. (2014). Reducing an operational supervisory control problem by decomposition for deterministic pushdown automata. *Proceedings of the 12th IFAC - IEEE International Workshop on Discrete Event Systems*.

Sreenivas, R.S. (1993). On a weaker notion of controllability of a language k with respect to a language l. *Automatic Control, IEEE Transactions on*, 38(9), 1446–1447.

Wonham, W.M. and Ramadge, P.J. (1987). On the supremal controllable sublanguage of a given language. In *SIAM Journal on Control and Optimization*, volume 25, 637–659.