Supplement for the paper entitled "A BDD-Based Approach for Designing Maximally Permissive Deadlock Avoidance Policies for Complex Resource Allocation Systems"

Zhennan Fei, Spyros Reveliotis, Sajed Miremadi and Knut Åkesson

Abstract

This electronic document provides some supportive material to the paper entitled "A BDD-Based Approach for Designing Maximally Permissive Deadlock Avoidance Policies for Complex Resource Allocation Systems" that has been submitted to IEEE Transactions on Automation Science and Engineering (T-ASE).

I. Modeling a given RAS instance Φ by the corresponding EFA $E(\Phi)$

This section provides a formal statement of the procedure for obtaining, for any RAS instance Φ coming from the RAS class that is considered in the aforementioned paper, the corresponding EFA $E(\Phi)$. This procedure is called DevEFA(RAS Φ), and it is detailed as follows:

Procedure DevEFA(RAS Φ)

Input: A RAS instance $\Phi = \langle \mathcal{R}, C, \mathcal{P}, \mathcal{A} \rangle$

Output: An EFA $E(\Phi)$ that models the (resource allocation) dynamics of Φ

1 */* Define the resource variables based on $\mathcal{R}$ and C */*

foreach R_i *in* $\mathcal{R} = \{R_1, \dots, R_m\}$ **do**

define $vR_i : \{0, \dots, C_i\}$ where $C_i = C(R_i)$

mark the value C_i as the initial and marked values of vR_i

end

foreach J_j *in* $\mathcal{P} = \{J_1, \dots, J_n\}$ **do**

2 */* Construct the EFAs modeling the sequential logic of each processing type in $\mathcal{P}$ */*

build an EFA E_j with only one location named ℓ_j

mark ℓ_j as the initial and marked location of E_j

foreach Ξ_{jk} *where* $k \le l(j)$ *in* $\mathcal{S}_j$ **do** *// recall that $J_j = \langle \mathcal{S}_j, \mathcal{G}_j \rangle$*

if Ξ_{jk} *is a non-terminal processing stage* **then**

define $v_{jk} : \{0, \dots, \theta_{jk}\}$ where:

$$\theta_{jk} = \min_i \{ \left\lfloor \frac{C_i}{\mathcal{A}_{jk}[i]} \right\rfloor : \ \mathcal{A}_{jk}[i] > 0 \}$$

mark the value 0 as the initial and the marked values of v_{jk}

end

end

foreach *stage* Ξ_{jk} *corresponding to a source node in* $\mathcal{G}_j$ **do**

build a self-loop transition tr labeled by the event $\langle \mathtt{J_j\_loading}, \Xi_{jk} \rangle$ and add it to E_j

attach the action $v_{jk} := v_{jk} + 1$ to tr

end

foreach $e = \langle \Xi_{jk}, \Xi_{jk'} \rangle$ *in* $\mathcal{G}_j$ **do**

build a self-loop transition tr labeled by the event $\langle \Xi_{jk}, \Xi_{jk'} \rangle$ and add it to E_j

attach the guard $v_{jk} \ge 1$ to tr

if $\Xi_{jk'}$ *is a non-terminal processing stage of* $\mathcal{S}_j$ **then** attach the action $v_{jk} := v_{jk} - 1; v_{jk'} := v_{jk'} + 1$ to tr

else attach the action $v_{jk} := v_{jk} - 1$ to tr *// no instance variable defined for a terminal stage*

end

3 */* Augment E_j with $\{vR_1, \dots, vR_m\}$ to represent the associated resource allocation */*

foreach *transition* tr *of* E_j **do**

let σ denote the labeling event of tr

if σ *is* $\langle \mathtt{J_j\_loading}, \Xi_{jk} \rangle$ **then** *// allocate the associated resources required at Ξ_{jk}*

append $vR_i \ge \mathcal{A}_{jk}[i]$ to the guard of tr for all i s.t. $\mathcal{A}_{jk}[i] > 0$

append $vR_i := vR_i - \mathcal{A}_{jk}[i]$ to the action of tr for all i s.t. $\mathcal{A}_{jk}[i] > 0$

end

else *// i.e., if σ is a process-advancing event $\langle \Xi_{jk}, \Xi_{jk'} \rangle$*

if $\mathcal{A}_{jk'}[i] > \mathcal{A}_{jk}[i]$ **then**

append $vR_i \ge (\mathcal{A}_{jk'}[i] - \mathcal{A}_{jk}[i])$ to the guard of tr

end

if $\Xi_{jk'}$ *is a terminal processing stage* **then** *// just deallocate the resources used at Ξ_{jk}*

append $vR_i := vR_i + \mathcal{A}_{jk}[i]$ to the action of tr for all i s.t. $\mathcal{A}_{jk}[i] > 0$

end

if $\Xi_{jk'}$ *is not a terminal processing stage* **then**

append $vR_i := vR_i - (\mathcal{A}_{jk'}[i] - \mathcal{A}_{jk}[i])$ to the action of tr for all i s.t. $\mathcal{A}_{jk'}[i] - \mathcal{A}_{jk}[i] \neq 0$

end

end

end

end

II. A COMPLETE CORRECTNESS ANALYSIS FOR ALGORITHM 2

To prove the effectiveness of Algorithm 2 w.r.t. the penultimate objective of the implementation of the maximally permissive DAP through the one-step-lookahead scheme that was outlined in the earlier parts of this manuscript, we need to show that (i) the algorithm terminates in a finite number of steps, (ii) the returned set χ_{FB} contains all the feasible boundary unsafe states, and furthermore, (iii) χ_{FB} does not contain any feasible safe state. The finiteness of Algorithm 2 depends on whether the backward search performed in Lines 6-15 can terminate in a finite number of iterations. We notice that the termination of this search is determined by the set of the new unsafe states, $\chi_{U_{new}}$, computed at each iteration; if $\chi_{U_{new}}$ is empty, the backward search terminates. We also notice that the set $\chi_{U_{new}}$ will finally be empty during the search, since the set of states in Δ_A is finite. Hence, Algorithm 2 terminates in a finite number of steps. In the following, we focus on establishing the correctness of the algorithm, by establishing items (ii) and (iii) in the above list. In the paper, these two items are addressed by Theorem IV.1. However, that manuscript provides only a sketch of the corresponding proof. Here is provide a complete treatment of the relevant results.

We begin with some lemmas that are necessary for the derivation of the final result.

Lemma II.1. *The characteristic function χ_{FD} that is obtained from the symbolic operations performed in Lines 1-4 of Algorithm 2 identifies correctly the feasible deadlock states in the transition set $\Delta_{\mathbf{E}}$ w.r.t. the process-advancing events of this set.*

Proof. First we notice that, by its construction, the set χ_D contains only states that are deadlocks w.r.t. the process-advancing events of $\Delta_{\mathbf{E}}$, and therefore, the same is true for its subset χ_{FD}. Furthermore, the states contained in χ_{FD} are feasible deadlocks since χ_{FD} is obtained from the filtration of χ_D with the characteristic function χ_F.

Next we show that χ_{FD} contains all the feasible deadlock states w.r.t. the process-advancing events of $\Delta_{\mathbf{E}}$. This part can be proved by contradiction. Let us assume that there exists a transition (s, s') in $\Delta_{\mathbf{E}}$ where s' is a feasible deadlock state that is not identified by the computation that is performed in Lines 1-4, i.e., $s' \notin FD$. Since s' is the target state of (s, s'), which is a transition in $\Delta_{\mathbf{E}} = \Delta_A \vee \Delta_L$, state s' must be in the set T. By the working assumption, state $s' \notin D$ after the computation at Line 3, since in that case it would have been retained in FD (being a feasible state). Knowing that $s' \notin D$, $s' \in T$ and $s' \neq s_0$, state s' can only be in the state set E. But this contradicts the definition of deadlock states, since states in E enable process-advancing events. □

Lemma II.2. *For every transition (s, s') of the EFA $\Delta_{\mathbf{E}}$, feasibility of the target state s' implies also the feasibility of the source state s.*

Proof. We prove the contrapositive of the above statement, i.e., every transition (s, s') of the EFA $\Delta_{\mathbf{E}}$ with an infeasible source state s has also an infeasible target state s'. Infeasibility of state s implies that there exists some resource R_i with

$$vR_i + \sum_{j=1}^{n} \sum_{k=1}^{l(j)-1} \mathcal{A}_{jk}[i] * v_{jk} = d \neq C_i,$$

for the values of the variables vR_i and $v_{jk},\ j = 1, \ldots, n,\ k = 1, \ldots, l(j) - 1$ that define state s. But it can be easily checked that every forward-advancing transition from state s preserves the invariant

$$vR_i + \sum_{j=1}^{n} \sum_{k=1}^{l(j)-1} \mathcal{A}_{jk}[i] * v_{jk} = d,$$

and therefore, it cannot restore feasibility w.r.t. to the implied allocation of resource R_i. □

Lemma II.3. *All states entering the sets U and χ_{FB} during the execution of Algorithm 2 are feasible.*

Proof. This lemma is an immediate implication of Lemmas II.1 and II.2, and of the fact that all the elements of these two sets are obtained by starting from some feasible deadlock state in χ_{FD} and backtracing upon some transitions in $\Delta_{\mathbf{E}}$. □

Lemma II.4. *The set U that is computed by Algorithm 2 contains all the feasible unsafe states in $\Delta_{\mathbf{E}}$.*

Proof. By Lemma II.1 and Line 5 of Algorithm 2, U contains all the feasible deadlocks. Next we will show that U also contains all the feasible deadlock-free unsafe states of the considered RAS.

Let us consider any such feasible deadlock-free unsafe state $\hat{u}$. The finite and acyclic nature of the paths that define the execution logic of the various process types in the considered RAS class, imply that the subspace that is reached from state $\hat{u}$ following only transitions in Δ_A has a finite, acyclic structure. This remark, when combined with the presumed unsafety of state $\hat{u}$, further implies that every path in Δ_A that emanates from state $\hat{u}$ is an acyclic path that terminates at some feasible deadlock state. Let ζ denote the longest length of these paths, where the length of a path is defined by the number of the involved transitions. Next, we will show, by induction on ζ, that state $\hat{u}$ will enter the state set U that is maintained by Algorithm 2 before the termination of the iteration in Lines 6-15.

First we consider the base case of $\zeta = 1$. Then, all the transitions emanating from $\hat{u}$ lead to a feasible deadlock state in χ_{FD}, and therefore, they will be contained in the set Δ_U computed during the first iteration of the algorithm. Hence, state $\hat{u}$ will not be in the set χ_{NU} that is computed at that iteration, and therefore, it will be correctly included in the set χ_{cur} and, eventually, in χ_U.

Next, let us suppose that all the feasible unsafe states with a maximal path of length $\zeta - 1$ from the feasible deadlock states of χ_{FD} will be correctly identified and entered in set U by Algorithm 2. Since the target state of each process-advancing transition that emanates from state $\hat{u}$ has a maximal path leading to χ_{FD} of length less than or equal to $\zeta - 1$, each of these states will be eventually identified by the algorithm. Let us consider, in particular, the iteration where the last of these states, let's say u_l, enters U. In the next iteration, u_l will be in $\chi_{U_{new}}$ and, therefore, $\hat{u}$ will be in $\chi_{S\hat{U}}$. Furthermore, $\hat{u}$ will not be in χ_{NU} since all of its emanated transitions will be either in $\Delta_{\hat{U}}$ or in $\Delta_{\hat{U}_{pre}}$. Hence, $\hat{u}$ will be included in $\chi_{U_{cur}}$ and eventually into χ_U. □

Lemma II.5. *The set U that is computed by Algorithm 2 contains no feasible safe states of $\Delta_{\mathbf{E}}$.*

Proof. We prove this result by induction on the number of iterations performed by the algorithm. The base case of zero iteration is covered by Lemma II.1. Next, suppose that the statement of Lemma II.5 is true for the set U constructed during the first n iterations. This assumption when combined with Lines 7 and 14 of Algorithm 2, further imply that the transition set $\Delta_{\hat{U}_{pre}}$ contains only transitions with target states in the already constructed state set U. But then, Lines 7-10 of the algorithm implies that the state set χ_{NU} that is constructed at iteration $n+1$ contains all the safe states that can be reached from the current set $\chi_{U_{new}}$ by backtracing on some transition of Δ_A. Hence, the set χ_{cur} constructed at the $n+1$ iteration, at Line 11, contains no safe states, and the addition of this set to state set U, at Line 13, does not introduce in U any safe states either. □

A complete proof of Theorem IV.1

Proof. Property 1 was established in Lemma II.3.

In view of Lemma II.4, to prove Property 2 we just need to show that the construction of the set FB in Lines 16-18 of Algorithm 2 retains all the boundary feasible unsafe states in U. But this can be easily checked from the facts that (a) the transition set $\Delta_{\mathcal{B}}$ contains all the transitions with target states in set U, while (b) the transition set $\Delta_{\mathcal{SB}}$ is obtained from $\Delta_{\mathcal{B}}$ by removing only transitions with source states in U (and therefore, unsafe, according to Lemma II.5).

Property 3 is also inferred from the construction of the set $\Delta_{\mathcal{SB}}$ from the set $\Delta_{\mathcal{B}}$ through the removal of all those transitions with source states in U, upon noticing that U contains all the feasible unsafe states of $\Delta_{\mathbf{E}}$ (according to Lemma II.4).

Finally, Property 4 results from Lemma II.5 and the fact that all the transitions in the set $\Delta_{\mathcal{B}}$ have target states in U. □

III. Algorithm 3 and its correctness analysis

Algorithm 3: Symbolic computation of the minimal boundary unsafe states

Input: χ_{FB}, $\{\Delta_{=}(\tilde{v}_1, v_1), \ldots, \Delta_{=}(\tilde{v}_K, v_K)\}$ and $\{\Delta_{\geq}(\tilde{v}_1, v_1), \ldots, \Delta_{\geq}(\tilde{v}_K, v_K)\}$

Output: $\chi_{\overline{FB}}$

1 $\Delta_{EQ} := \Delta_{=}(\tilde{v}_1, v_1) \wedge \ldots \wedge \Delta_{=}(\tilde{v}_K, v_K)$ *// Δ_{EQ} represents the set of pairs, which represents each state $\langle v_1, \ldots, v_K \rangle$ // by respectively using the Boolean variable sets $\tilde{X}^{\mathcal{D}}$ and $X^{\mathcal{D}}$;*

2 $\Delta_{GE} := \Delta_{\geq}(\tilde{v}_1, v_1) \wedge \ldots \wedge \Delta_{\geq}(\tilde{v}_K, v_K)$ *// Δ_{GE} represents the set of pairs, which associate each state $\langle v_1, \ldots, v_K \rangle$ // represented $\tilde{X}^{\mathcal{D}}$ with its equal and dominant states, represented by $X^{\mathcal{D}}$;*

3 $\Delta_{GT} := \Delta_{GE} \wedge \neg \Delta_{EQ}$ *// Δ_{GT} is the set of pairs where each state is associated with its dominant states;*

4 $\Delta_{BGT} := \chi_{FB}[X^{\mathcal{D}} \to \tilde{X}^{\mathcal{D}}] \wedge \Delta_{GT}$ *// Δ_{BGT} collects the pairs in Δ_{GT} with the first elements as // the feasible boundary unsafe states;*

5 $\chi_{GB} := \exists \tilde{X}^{\mathcal{D}}.\ \Delta_{BGT}$ *// χ_{GB} collects the states that are larger (component wise) than the states in χ_{FB};*

6 $\chi_{\overline{FB}} := \chi_{FB} \wedge \neg \chi_{GB}$ *// remove from χ_{FB} all the non-minimal states, which belong to the set χ_{GB};*

Algorithm 3 presents a way to compute the characteristic function of the set $\overline{FB}$ from the characteristic function χ_{FB} obtained in Eq. (8) of the main document. Before we proceed with the discussion of this algorithm, we need to introduce two auxiliary BDD sets, collectively denoted by $\{\Delta_{=}(\tilde{v}_1, v_1), \ldots, \Delta_{=}(\tilde{v}_K, v_K)\}$ and $\{\Delta_{\geq}(\tilde{v}_1, v_1), \ldots, \Delta_{\geq}(\tilde{v}_K, v_K)\}$, which will be useful for identifying state dominances, according to Eq. (9) of the main document, by the proposed algorithm. Each pair $\Delta_{=}(\tilde{v}_k, v_k)$ and $\Delta_{\geq}(\tilde{v}_k, v_k)$ pertains to the corresponding instance variable v_k, and it can be constructed as follows:

$$\Delta_{=}(\tilde{v}_k, v_k) := \bigvee_{\forall v_k \in \mathcal{D}_k} \Big(\tilde{X}^{\mathcal{D}_k}(v_k) \wedge X^{\mathcal{D}_k}(v_k) \Big) \tag{1}$$

$$\Delta_{\geq}(\tilde{v}_k, v_k) := \bigvee_{\forall v_k \in \mathcal{D}_k} \Big(\tilde{X}^{\mathcal{D}_k}(v_k) \wedge \bigvee_{\forall v'_k \geq v_k} X^{\mathcal{D}_k}(v'_k) \Big) \tag{2}$$

In (1) and (2), $\tilde{X}^{\mathcal{D}_k}(v_k)$ denotes the symbolic representation of the value of k-th variable v_k using a new set of Boolean variables denoted by $\tilde{X}^{\mathcal{D}_k}$, while $X^{\mathcal{D}_k}(v_k)$ and $X^{\mathcal{D}_k}(v'_k)$ denote the symbolic representations of the values v_k and v'_k, of the same instance variable, using the set of the Boolean variables $X^{\mathcal{D}_k}$ that represent the instance variable v_k in the original BDD $\Delta_{\mathbf{E}}$. From a conceptual standpoint, $\Delta_{\geq}(\tilde{v}_k, v_k)$ associates each value v_k with all those values $v'_k \in \mathcal{D}_k$ that are greater than or equal to v_k while $\Delta_{=}(\tilde{v}_k, v_k)$ merely associates each value v_k with itself.

Taking as input the feasible boundary unsafe state set χ_{FB} and the aforementioned auxiliary BDDs, the symbolic computation of the minimal feasible boundary unsafe states is formally expressed by Algorithm 3. Specifically, in Lines 1-2, Algorithm 3 constructs two BDDs, respectively denoted by Δ_{EQ} and Δ_{GE}, by performing the conjunction operation on $\{\Delta_{\geq}(\tilde{v}_1, v_1), \ldots, \Delta_{\geq}(\tilde{v}_K, v_K)\}$ and $\{\Delta_{=}(\tilde{v}_1, v_1), \ldots, \Delta_{=}(\tilde{v}_K, v_K)\}$. The characteristic function Δ_{EQ} associates each state $\langle v_1, \ldots, v_K \rangle$ with two different symbolic representations using the Boolean variable sets $\tilde{X}^{\mathcal{D}}$ and $X^{\mathcal{D}}$, while Δ_{GE} associates each state $\langle v_1, \ldots, v_K \rangle$, represented by $\tilde{X}^{\mathcal{D}}$, with a set of states, represented by $X^{\mathcal{D}}$, which are larger than or equal to $\langle v_1, \ldots, v_K \rangle$. Subsequently, the symbolic computation performed at Line 3 of Algorithm 3 removes all the associations of Δ_{EQ} from Δ_{GE} and the resulting set is denoted by Δ_{GT}. Line 4 of Algorithm 3 computes the characteristic function Δ_{BGT} which associates each state in χ_{FB} with the corresponding dominant states, and, subsequently, Line 5 extracts all these dominant states into the set χ_{GB}. Finally, the set of minimal feasible boundary unsafe states, $\chi_{\overline{FB}}$, is obtained in Line 6 by removing from χ_{FB} the states in χ_{GB}.

Next, we prove the correctness of Algorithm 3.

Theorem III.1. *The characteristic function $\chi_{\overline{FB}}$ returned by Algorithm 3 recognizes correctly the minimal elements of χ_{FB}.*

Proof. First we show that Algorithm 3 does not miss any minimal element of FB. For this, let us assume that there exists a minimal feasible boundary unsafe state $u \in FB$ that is not identified by Algorithm 3, i.e., $u \notin \overline{FB}$. Hence, state u is contained in the state set GB that is removed from FB. By the computation performed at Line 5, we know that the states of GB are the second elements of the relation that is encoded by set BGT, which is a subset of GT. According to Lines 1-3, GT associates each possible state $\langle v_1, \ldots, v_K \rangle$ with its dominant states (as the second elements). Then, it can be inferred from Lines 4-5 that, since $u \in GB$, there exists a state $u' \in FB$ such that the pair (u', u) is in set BGT. Therefore, we have $u' < u$, which contradicts the working assumption of the minimality of u.

Next, we show that Algorithm 3 does not include any non-minimal element of FB. For this, let us assume that there exists a non-minimal boundary unsafe state $u'' \in \overline{FB}$. Hence, by the working assumption, there exists a minimal boundary unsafe state $u \in \overline{FB}$ such that $u'' > u$. But then, state u'' must be in the set GB that is computed in Lines 3-5, and it should be one of the states removed from FB through the computation of Line 6.

Based on the above arguments, it is concluded Algorithm 3 retains in set $\overline{FB}$ all the minimal elements of FB, and it excludes from this set any non-minimal elements of FB. $\square$