Logical Control of Complex Resource Allocation Systems

Spyros Reveliotis
School of Industrial & Systems Engineering
Georgia Institute of Technology

WODES 2018
Sorrento, Italy

Some major underlying “themes”

- **Real-time management of complex resource allocation systems:**
  - An important and interesting application domain for broader DES theory.
  - A significant specialization and extension of the general DES theory focused on
    - the particular challenges and
    - the special structure

    that are encountered in the considered class of problems.
  - Particular emphasis placed on the computational complexity of the defined problem(s) and the derived solutions.
  - Considerable potential for transferring the derived results to other application domains of DES theory.

Acknowledging the contributors of the presented research

Past and Current Students:

- Jonghun Park
- Jin Young Choi
- Theologos Bountourelis
- Ahmed Nazeem
- Hongwei Liao
  (co-adv. with S. Lafortune)
- Zhennan Fei
  (co-adv. with K. Akesson)
- Ran Li
- Stephen Daugherty
- Michael Ibrahim

Research Sponsors:

- Various NSF grants from
  - CMMI / MES
  - ECS & ECCS
- The ISyE Virtual Factory Lab and the Keck Foundation
- The Georgia Tech Research Institute

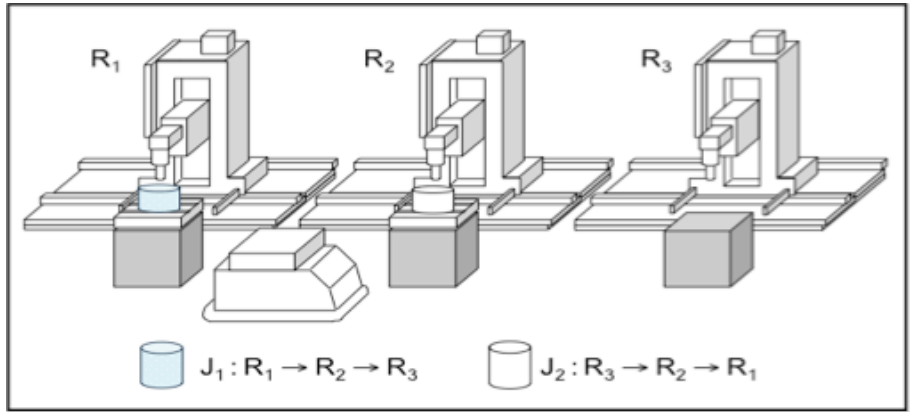
Other research groups and collaborators:

- E. Dijkstra, A. Habermann, J. Havender, R. Holt, E. Coffman, M. Elphick, A. Shoshanni,
- M. Gold, T. Araki, Y. Sugiyama and T. Kasami

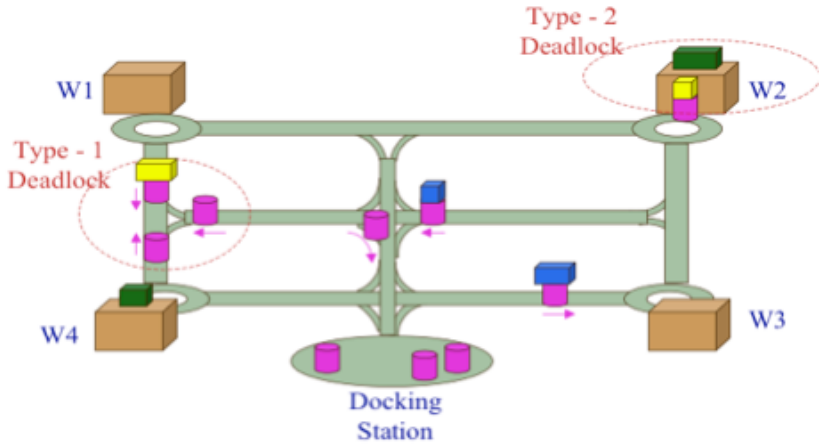
- P. Ramadge and W. M. Wonham
- J. Ezpeleta, J.-M. Colom, M. Silva and their colleagues
- N. Viswanadham,Y. Narahari and T. Johnson
- Z. Banaszak and B. Krogh
- P. Ferreira, M. Lawley and his colleagues
- E. Roszkowska
- M. P. Fanti and her colleagues
- M. Zhou, Z. Li and their colleagues
- X. Xie, M. Jeng and their colleagues
- S. Lafortune, Y. Wang and their colleagues
- L. Piroddi and R. Cordone
- A. Giua and his colleagues
- P. Antsaklis, J. Moody and M. Iordache
- K. Barkaoui and his colleagues

- X. Cao and C. Cassandras
- D. Bertsekas, J. Tsitsiklis and their colleagues

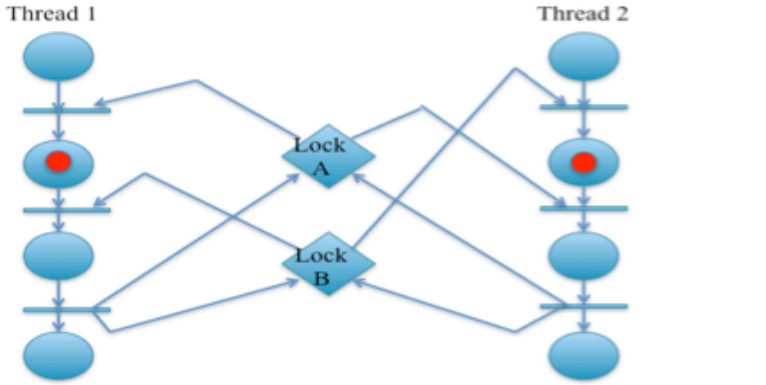
Some motivating applications: Sequential resource allocation in flexible automation



Workflow management in FMS

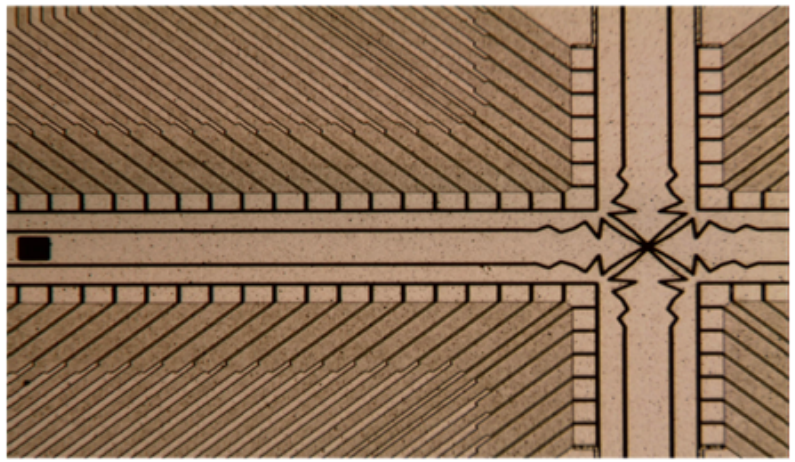


Traffic management in AGV systems



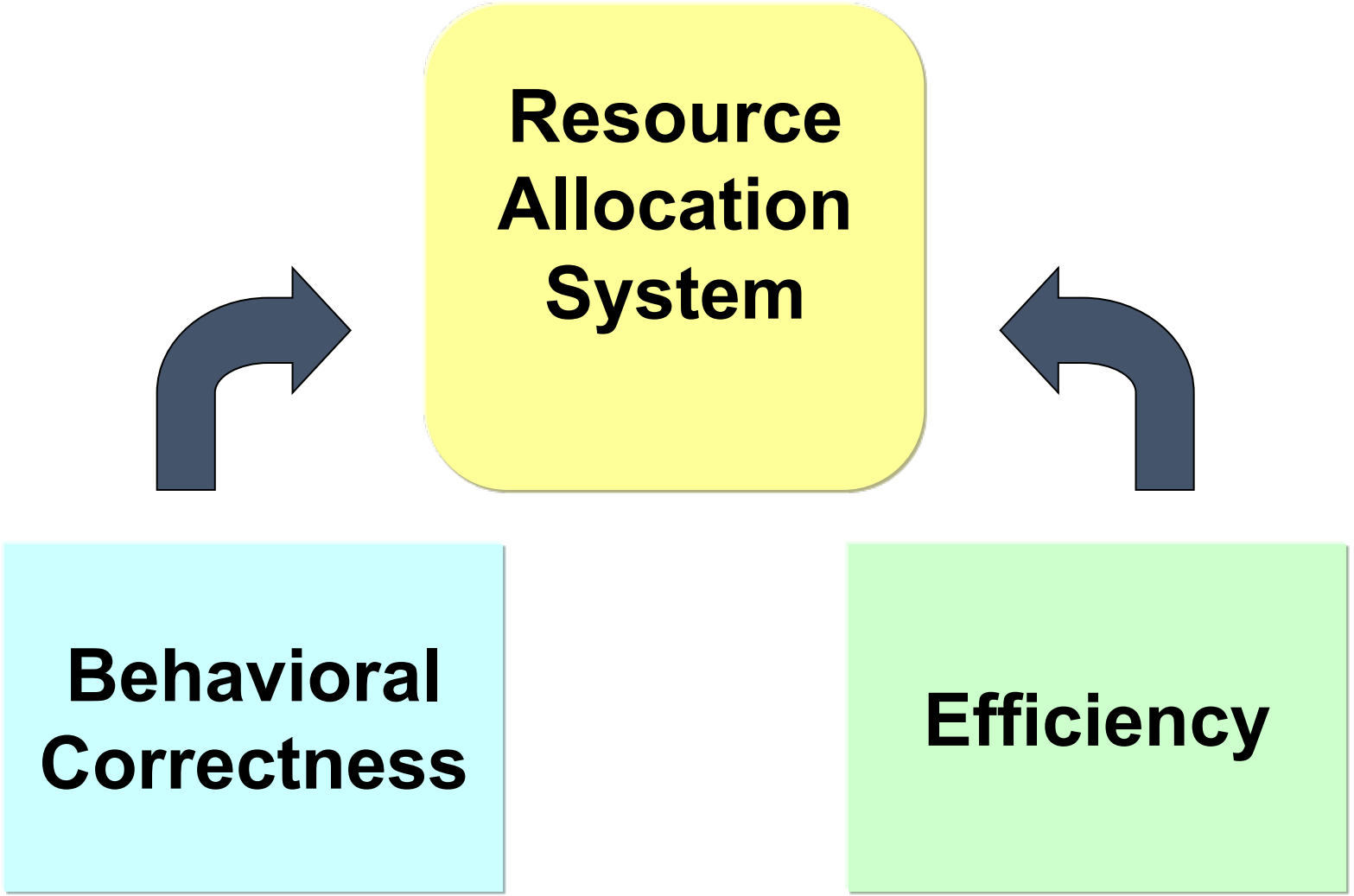
Dijkstra, 1965; Havender, 1968; Habermann, 1969; Coffman *et al*, 1971; Holt, 1972

Semaphore allocation in multi-threaded software

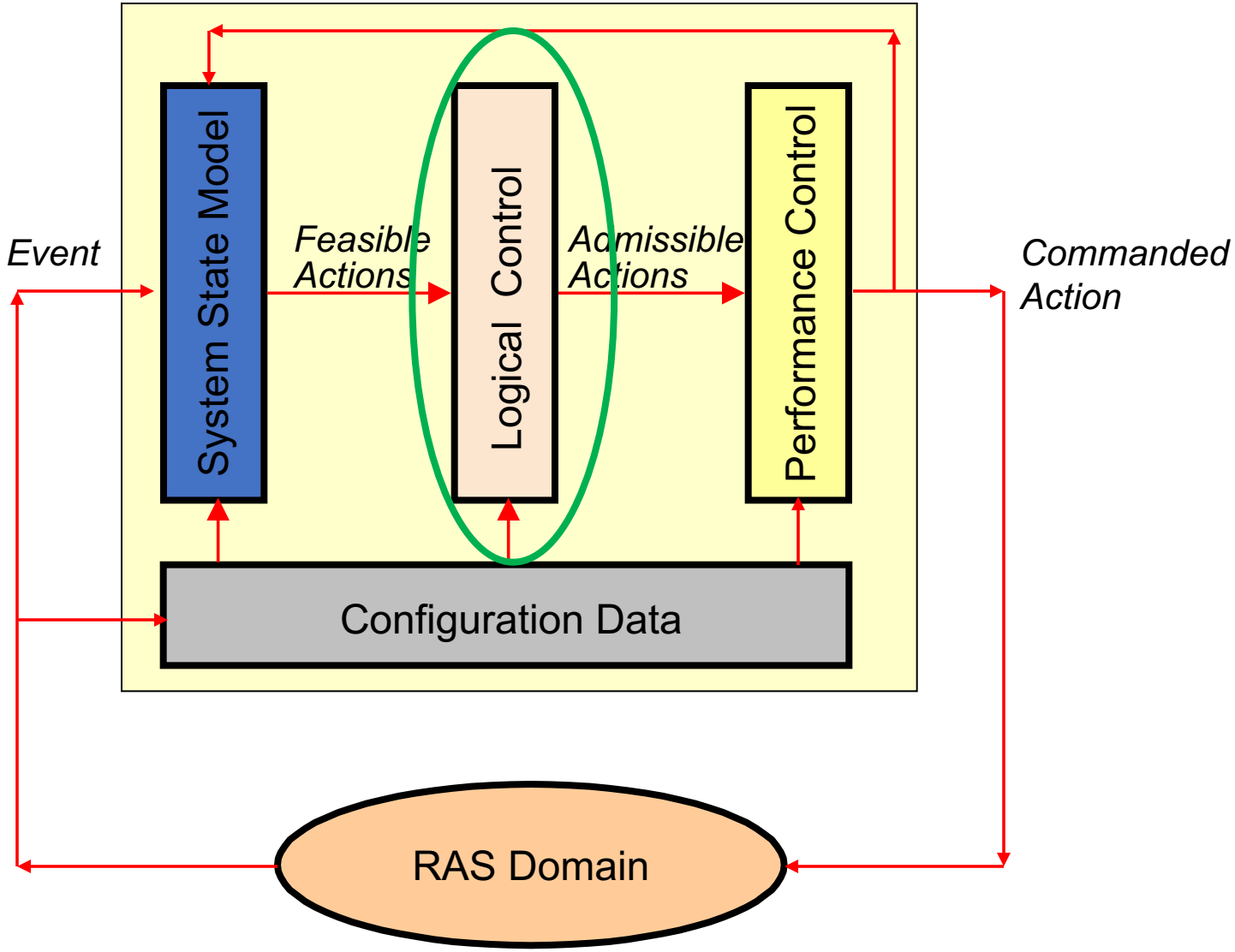


Allocation of ion traps in quantum computing

Control of Sequential RAS: An integrated framework for Behavioral and Performance-oriented Control

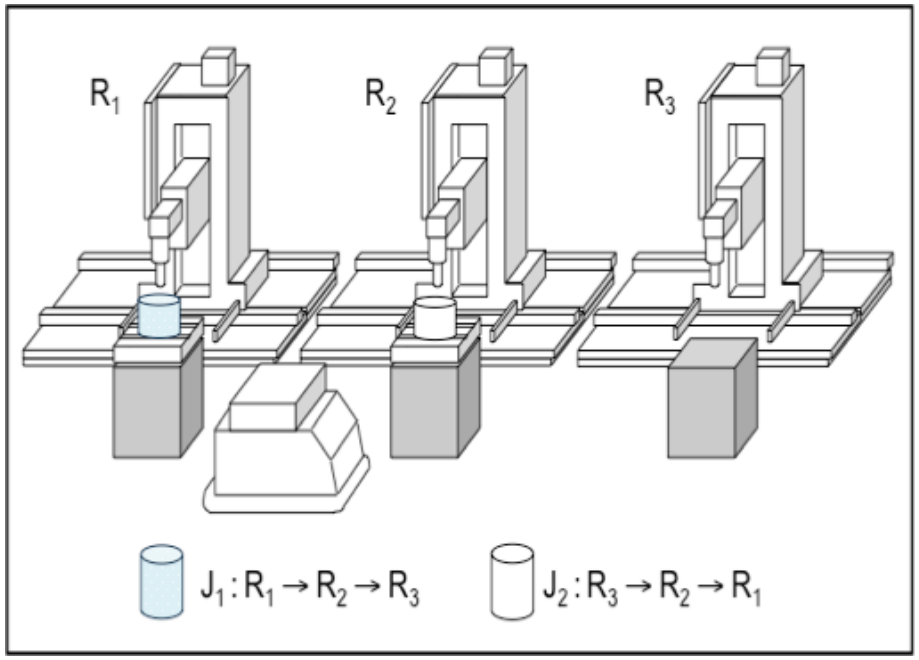


An Event-Driven RAS Control Scheme



The employed modeling abstraction: Sequential Resource Allocation Systems (RAS)

- A set of (re-usable) resource types $R = \{R_i, i = 1,...,m\}$.
- Finite capacity C_i for each resource type R_i.
- A set of process types $J = \{J_j, j = 1,...,n\}$.
- A set of process stages for each job type, $\{p_{jk}, k = 1,...,\lambda_j\}$, organized according to some sequential logic.
- A resource requirements vector for each process stage p, $a_p[i]$, i = 1,...,m.
- Resource Allocation Protocol: In order to advance from stage p to stage q, a process must be allocated the resource differential $(a_q\text{-}a_p)^+$ and only then it releases the unnecessary resource set expressed by $(a_p\text{-}a_q)^+$.
- A timing distribution D(p) characterizing the execution time of each process stage p.



A RAS taxonomy

Structure of the process sequential logic

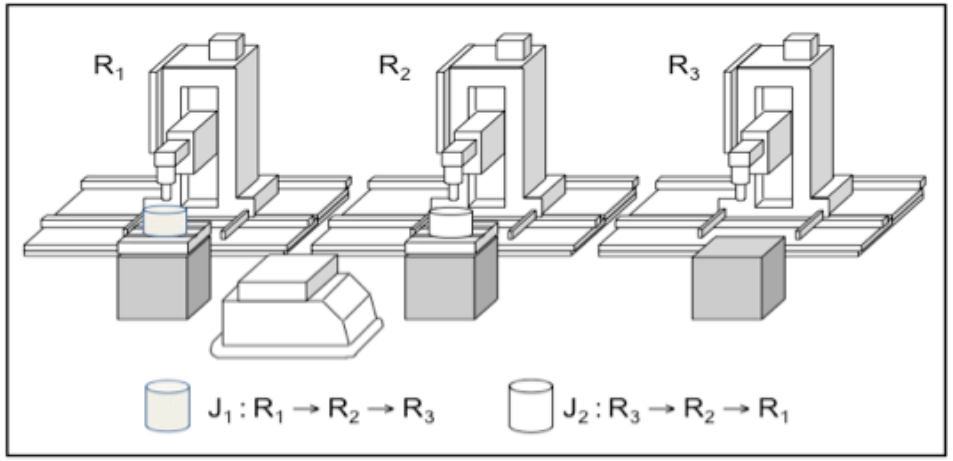
- Linear: each process is defined by a linear sequence of stages
- Disjunctive: A number of alternative process plans encoded by an acyclic digraph
- Merge-Split or Fork-Join: each process is a fork-join network
- Complex: a combination of the above behaviors

Structure of the stage resource requirement vectors

- Single-unit: each stage requires a single unit from a single resource
- Single-type: each stage requires an arbitrary number of units, but all from a single resource
- Conjunctive: Arbitrary number of units from different resources

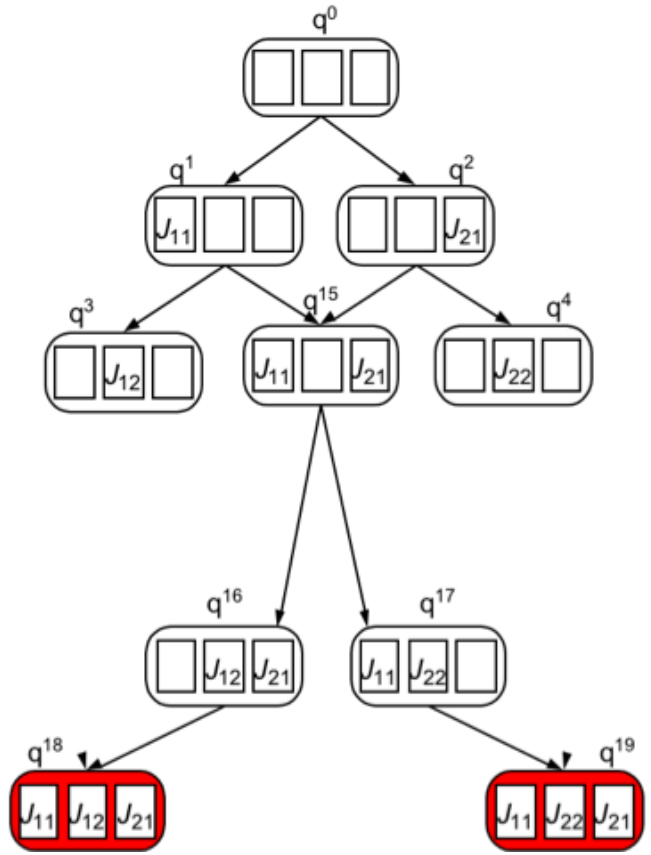
Some additional interesting features: ***uncontrollable*** transitions w.r.t. timing and/or routing, ***unit*** resource capacities, the class of ***timing distributions*** allowed, etc.

An FSA-based characterization of the RAS logical control problem

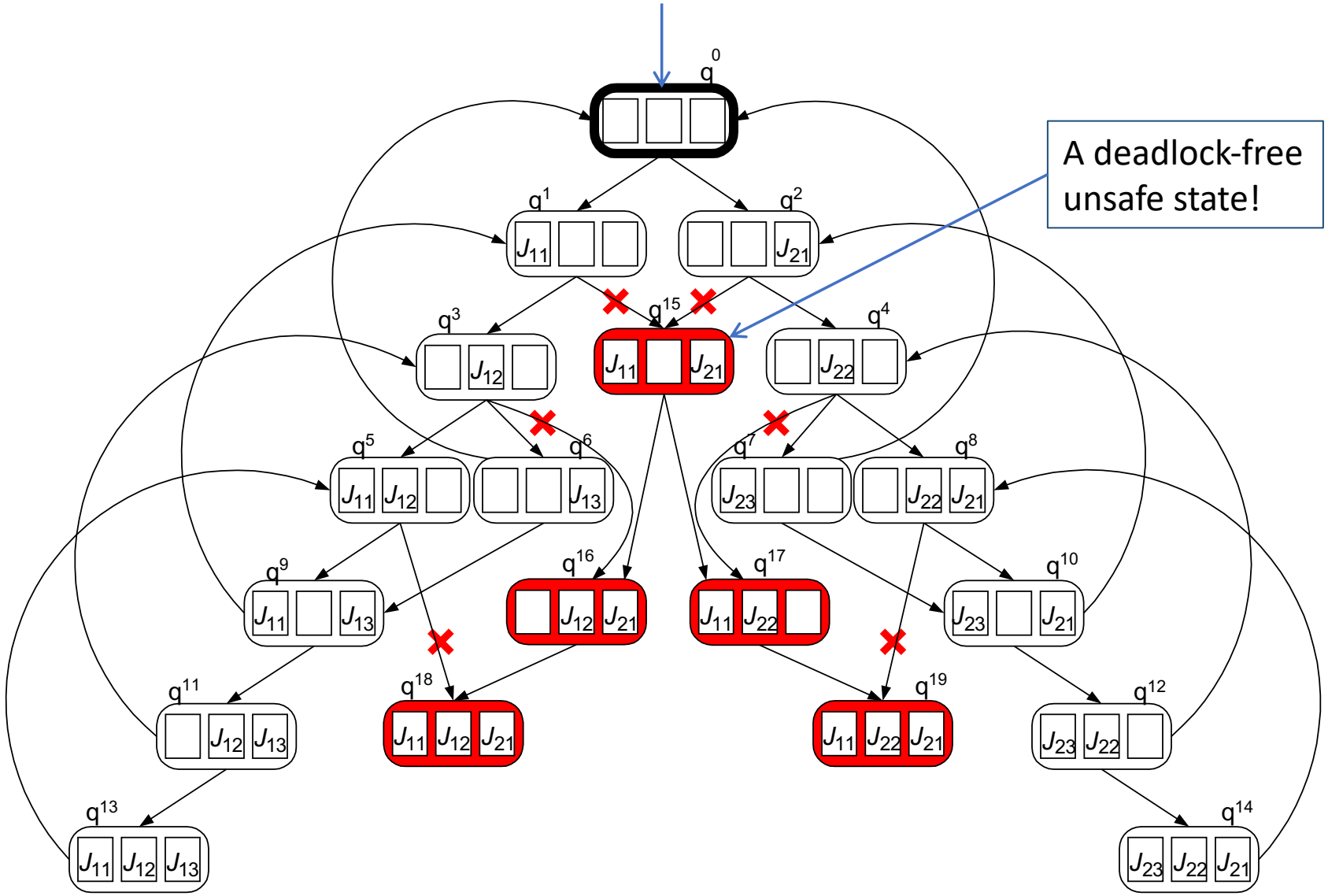


- The RAS state is defined by the number of active process instances at each processing stage.
- The RAS events correspond to
  - loading,
  - unloading, or
  - advancing a single process instance by one processing stage.

An FSA modeling the RAS behavior



Safe vs. Unsafe Region and the Optimal Deadlock Avoidance Policy



Complexity Considerations

- State Safety is an NP-complete problem even for linear, single-unit RAS.

  (by reduction of the SAT or 3SAT problem)

- Size of the State Transition Diagram (STD) of SU-RAS:

$$O\left(\binom{C+Q}{Q}^{m}\right)$$

where:

- C = max resource capacity
- Q = max number of stages supported by a resource
- m = number of resource types

Dealing with the *non-polynomial* complexity of RAS state safety

- ***Sub-optimal*** one-step-lookahead policies based on state properties that are polynomially verifiable and identify a strongly connected component of the RAS state space containing the empty state (known as Polynomial-Kernel policies) , e.g.,
  - Banker's algorithm (Dijkstra 1965)
  - RUN (Resource Upstream Neighborhood)
  - RO (Resource Ordering)
- Special RAS structure admitting optimal DAP of polynomial complexity w.r.t. the RAS size.
- Automated liveness verification tools that are derived from structural characterizations of deadlock and take the form of mathematical programming formulations of polynomial size w.r.t. the RAS structure. For certain RAS classes, the availability of these tools also enables an incremental computation even of the optimal DAP, that forgoes the explicit enumeration of the underlying state space.

- A methodology for implementing the maximally permissive DAP as a classifier of the RAS state space into its safe and unsafe subspaces.

S. Reveliotis, "Logical Control of Complex Resource Allocation Systems", NOW Series on Foundations and Trends in Systems and Control, vol. 4, no. 1-2, pgs 1-223, 2017.

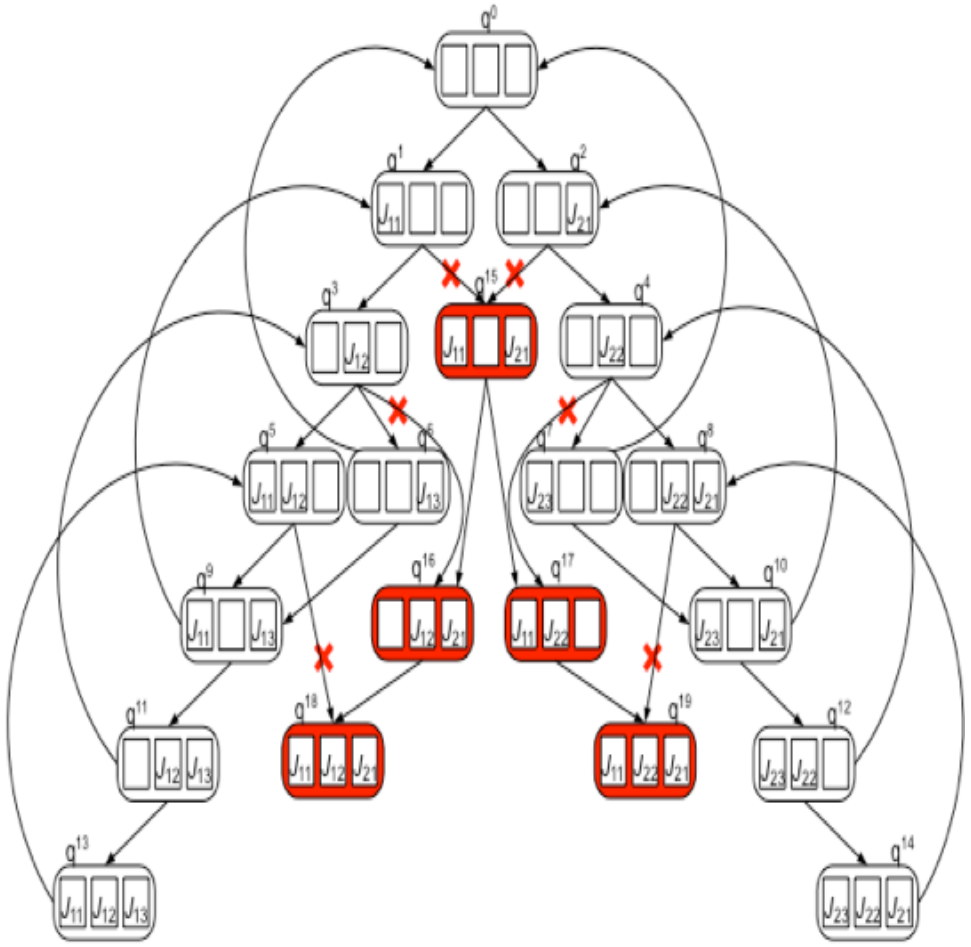
Designing and implementing the maximally permissive DAP through classification theory

Compute the reachable state space of the considered RAS.

Use standard supervisory control theory to classify the various states into safe and unsafe.

Design a compact classifier that effects the aforestated dichotomy of the reachable state space.

Use this classifier on-line in order to accept or reject tentative transitions.



The defining “ingredients” of the presented method

- **Representations:**
  - “Parametric” classifiers: based on a *parameterized classification mechanism* (or a *parameterized ``architecture”*) that will enable the effective and efficient classification of a given (reachable) RAS state as “safe” or “unsafe”.
  - “Non-parametric” classifiers: based on the *explicit but efficient storage* of transitions leading from the safe to the unsafe region.
  - “Completeness”: The ability of the employed representations to provide an effective characterization of the target state-space dichotomy for any given RAS instance from the considered RAS class.

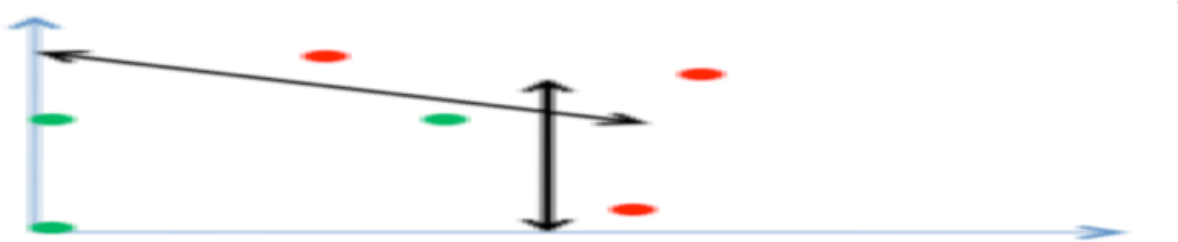
- **On-line classification algorithms**
  - “Structural minimality” (w.r.t. a certain representation): Perform the sought classification of any given state from the underlying state space, using a classifier of the specified representation that minimizes the involved computational cost.

- **Classifier construction process**
  - Need to cope with the explosive size of the safe and unsafe state spaces.
  - For parametric representations, we need to address the “parameter selection” problem:
    - an optimization problem in the employed “representation space”.

Some enabling ideas and results

1. The "monotonicity" property of (D/C-) RAS state safety:

s is safe $\wedge$ s' $\leq$ s => s' is safe.



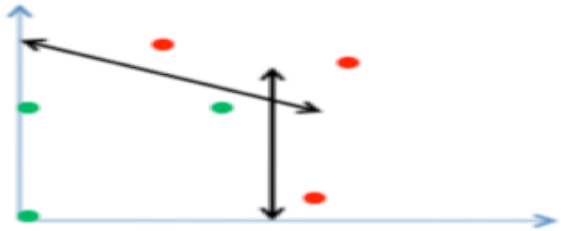
Focus the classifier construction only on the correct classification of maximal safe and minimal unsafe states.

2. Focus on "boundary" unsafe states, i.e. states that can be reached in one transition from some safe state.

3. Ignore state components corresponding to stages that cannot be entangled in deadlock (this is more relevant for "parametric classification" methods).

"Parametric" representations: *"Linear"* classifiers

- The classifier is a system of linear inequalities, $A \cdot s \leq b$, such that:
  - every safe state *s* satisfies all inequalities; and
  - every unsafe state *u* violates at least one inequality.



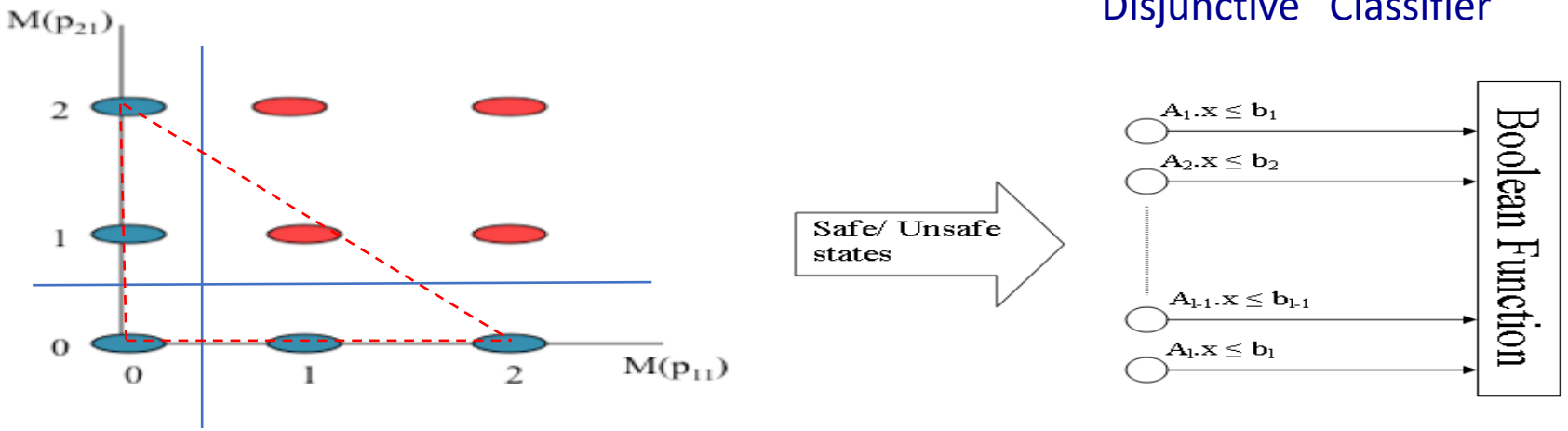
- The monotonicity of state safety implies the existence of linear classifiers with: $A \geq 0 \wedge b \geq 0$

- A linear classifier is *structurally minimal iff* it employs the smallest possible number of linear inequalities.

- The design of a structurally minimal classifier for the representation of the optimal DAP of a given RAS can be formulated as a Mixed Integer Program (MIP).

- Alternatively, it can be perceived as a "minimal set covering" problem where each employed inequality is associated with the subset of RAS states that it recognizes as unsafe. ➔ Efficient heuristics from the corresponding optimization theory and customized combinatorial optimization algorithms.

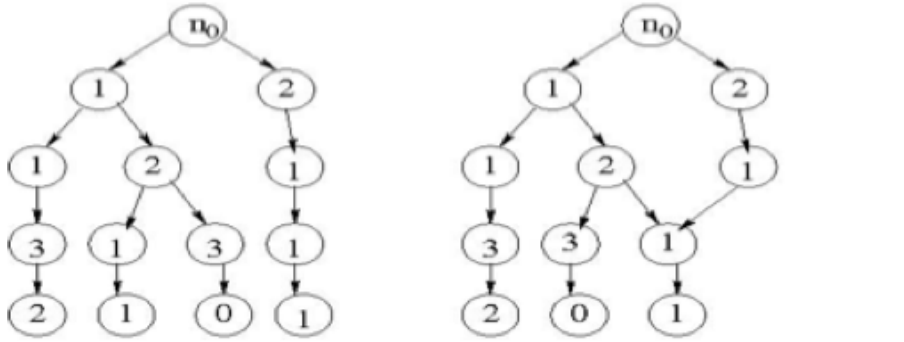
“Parametric” representations: *Complete* classifiers



- Linear classifiers are ***not*** complete representations of the optimal DAP, even for linear, single-unit RAS.
- The identification of a complete parametric representation can be generally addressed through the corresponding classification theory of dichotomizing a finite vector set.
- For RAS classes where state safety is monotone, the optimal DAP can always can be represented by a disjunction of a number of linear classifiers. The corresponding classifiers are known as “disjunctive”.
- The design theory of (structurally minimal) linear classifiers has been extended to disjunctive classifiers.

“Non-parametric” representations

1. Identify and store only the (minimal) boundary unsafe states using an efficient (“symbolic”) representation.

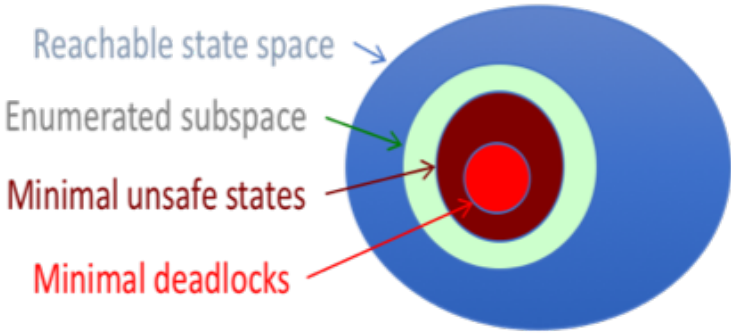


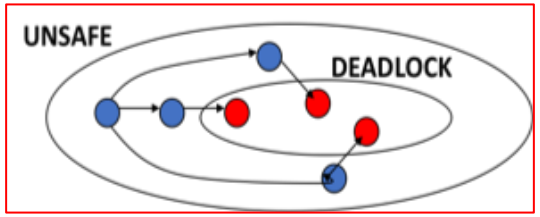
2. Search efficiently for the (minimal) boundary unsafe states, avoiding a complete enumeration of the underlying reachable state space:

a)

Symbolic computation based on Binary Decision Diagrams (BDDs)

b)





→ The implementation of the optimal DAP through an effective enumeration and efficient storage of the minimal boundary unsafe states applies even to RAS with infinite state spaces; e.g., RAS with ``Reader/Writer'' resource types.

An efficient enumeration of the minimal boundary unsafe states for Disjunctive - Conjunctive RAS

I. Enumerating the minimal deadlocks:

Theorem: Consider a minimal deadlock state s_d with active processing stages $\{q_1,\ldots,q_t\}$. Then, state s_d can be expressed as

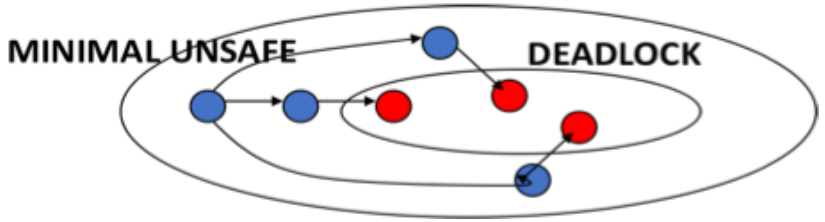
$$s_d = \max\{s_1,\ldots,s_t\}$$

where each vector s_i, $i=1,\ldots,t$, is a minimal state blocking the active instances in processing stage q_i, and the 'max' operator is applied component-wise on its vector arguments.

Remark: Minimal deadlocks can be programmatically enumerated through the result of the above theorem, by a recursive algorithm that synthesizes them from all minimal states that block single transitions.

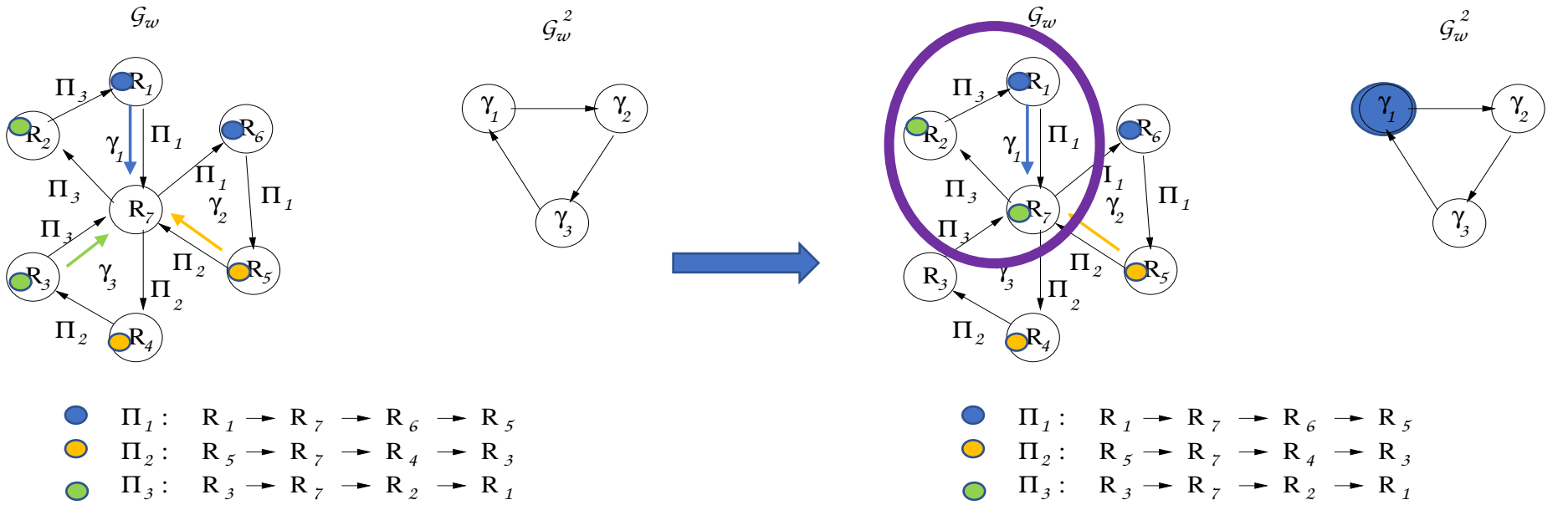
An efficient enumeration of the minimal boundary unsafe states for Disjunctive - Conjunctive RAS (cont.)

II. Enumerating the minimal boundary deadlock-free unsafe states:



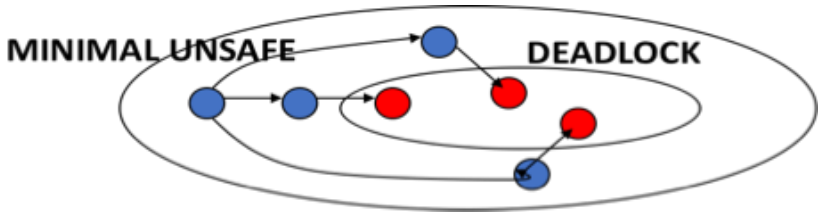
Every process advancement from a minimal deadlock-free unsafe state leads to another unsafe state or a deadlock.

But these process advancements do not preserve minimality!



An efficient enumeration of the minimal boundary unsafe states for Disjunctive - Conjunctive RAS (cont.)

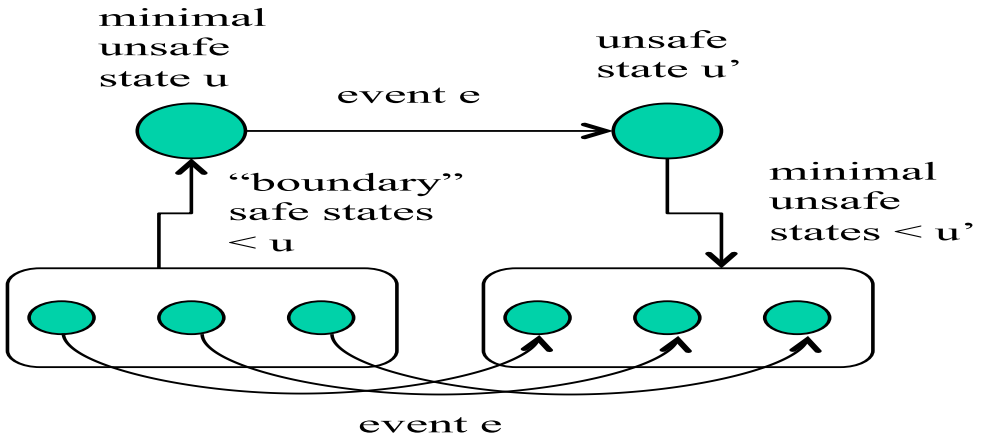
II. Enumerating the minimal boundary deadlock-free unsafe states (cont.):



Every process advancement from a minimal deadlock-free unsafe state leads to another unsafe state or a deadlock.

But these process advancements do not preserve minimality.

Theorem:



Some experimental results

Table 4.7: Some computational results regarding the efficacy of the presented algorithms for the enumeration of the target set $\bar{S}_{r\bar{s}}$, originally reported in [102]

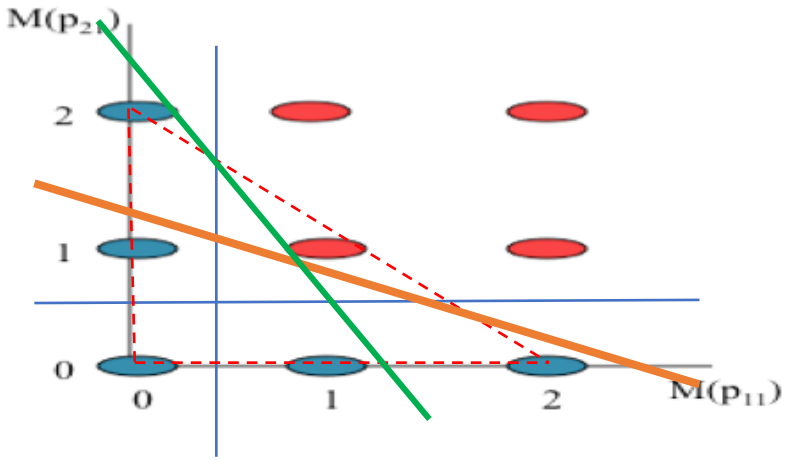
	$\lvert S_r \rvert$	$\lvert S_{r\bar{s}} \rvert$	$\lvert \bar{S}_{r\bar{s}} \rvert$	γ	α	A	t_o	t_n	t_r
Linear, Single-Type	635572	349546	1654	2807	5912	8548	1225	69	0
	1463878	458127	284	540	868	2124	1418	6	0
	404542	237591	3475	6419	12848	9210	2002	73	0
	1508301	573055	686	1076	3017	5792	2633	35	0
	799071	401974	7568	11753	20736	15534	4759	199	0
	1743534	887064	2840	4130	18516	33020	13270	555	0
	1659342	987461	10468	21410	45139	39937	16690	1447	1
	1962454	1098441	7171	11002	24859	42639	21001	989	0
	4488904	2748034	658	2088	8384	23095	29765	259	0
	3436211	1789990	36874	68317	135762	91164	105074	5863	1
	14158338	4977105	35022	54327	116337	136038	229759	31933	10
			30110	59864	169071	234629		36305	4
			33744	72935	193572	274553		44222	5
			22157	43718	129210	190378		23382	7
			20891	39738	165495	299266		51295	4
			16910	29819	82621	138687		14851	3
Linear, Conjunctive	646746	175449	779	849	1815	908	167	2	0
	1767552	406300	559	660	1050	4729	541	20	0
	915716	412871	849	1103	6065	23837	1551	509	0
	738720	644955	4162	4614	10752	8646	2251	327	0
	2939463	1328411	3655	5928	10619	11354	5060	193	0
	2430581	867647	14185	16547	29722	38765	6774	1112	0
	1962454	1098441	7171	11002	24859	42639	12087	1073	0
	1712672	977484	7214	10617	32861	54891	13027	1782	0
	3554952	2366757	2533	3659	17855	19856	21082	614	0
	6051299	2300151	5069	6837	24373	65602	34650	1865	0
	24430444	7268845	9783	11307	32410	90665	114614	5339	1
			3145	6998	20843	27747		550	0
			16910	29819	82621	138687		12649	3
			33744	72935	193572	274553		45550	5
			4046	4393	6592	7585		92	0
			428	585	2957	31425		1098	0
Disjunctive, Conjunctive	571536	0	0	0	0	0	19	0	0
	2171880	13992	28	28	32	126	48	0	0
	1229688	64968	241	271	279	373	105	0	0
	2693250	76536	79	79	87	37	130	0	0
	3416000	280000	1	1	8	1	158	0	0
	2448000	410112	9	9	9	50	580	0	0
	1663534	230889	2665	5857	6966	5030	801	173	0
	2340408	511277	1522	2620	3376	3760	4455	37	0
	7885856	605977	2323	2628	2710	4882	4871	19	0
			17394	24192	27412	25199		1001	4
			381	516	691	2729		77	0
			1215	1245	1294	3490		15	0
			24	31	31	77		747	0
			6635	8543	13382	42510		665	2
			792	1975	2000	2920		6	4
			2468	6035	7444	12777		88	19

- Col. 1: # of reachable states.
- Col. 2: # of reachable unsafe states.
- Col. 3: # of minimal reachable unsafe states.
- Col. 4: # of minimal unsafe states generated by the algorithm (without checking for reachability.)
- Col. 5: # of unsafe states visited by the algorithm.
- Col. 6: # of boundary safe states visited by the algorithm.
- Col. 7: Computing time (in secs) required for the computation of the minimal unsafe states through a complete enumeration of the underlying state space.
- Col. 8: Computing time (in secs) required for the computation of the minimal unsafe states through the discussed algorithm.
- Col. 9: Computing time (in secs) required for the the removal of unreachable states from the target set of minimal unsafe states.

Ongoing work: *Maximal linear* DAPs

Definition: Suboptimal DAPs, Δ, admitting a subset S of S_{rs} that has the following properties:

1. It contains the empty state s_0.
2. It induces a strongly connected component of the underlying STD.
3. It is separable from the remaining reachable states by means of a linear classifier.
4. It is ***maximal*** w.r.t. the sets of states satisfying properties 1-3.



A systematic enumeration of these target policies Δ can be attained by a search process through the subsets of S_{rs}, that

- starts with the state set S_{rs} itself, and
- for every considered state subset S of S_{rs}, it does the following:
  1. It assesses the linear separability of S from $S_r \setminus S$.
  2. If S is linearly separable, then it recognizes S as a state set defining a maximal linear DAP Δ.
  3. Otherwise, it generates maximal subsets of S that are obtained through
     - the removal of a maximal state $s \in S$, and
     - the further thinning of S so that it satisfies Property 2 of the above definition,

     and enters these subsets in the maintained search list for further consideration.

Performance-oriented controllers for logically controlled RAS

- Long-term throughput maximization of logically controlled RAS that operate in a repetitive setting can be formulated as an Average-Reward Continuous-Time Markov Decision Process (AR-CT-MDP); and these formulations extent to non-exponential timing distributions through the use of the method of stages.
- Thanks to the applied DAP, the AR-CT-MDP of the previous item belongs in the class of communicating MDPs, for which there exists very well developed theory and a host of solution algorithms.
- However, the practical applicability of the classical MDP algorithms is limited by the explosive size of the underlying state space. Hence, we have to resort to Approximate Dynamic Programming (ADP) methods.
- In the case that the applied DAP admits a linear representation, the timed dynamics of the controlled RAS can be conveniently modeled as a Generalized Stochastic Petri Net (GSPN). In this representation, resource allocation and routing decisions are modeled by the "untimed" transitions of the GSPN, while the execution of the various processing stages, and the corresponding delays, are modeled by the "timed" transitions of the net.
- In the GSPN modeling framework, the randomized stationary policies of the aforementioned MDP are represented by the various "random switches" of the net, i.e., the extraneously defined distributions that regulate the firing of the untimed transitions that are enabled at the various reachable markings of the net.
- The above connection of the net random switches with randomized stationary policies for the underlying MDP enables "approximation in the policy space", where the optimization in the corresponding policy spaces is performed through stochastic approximation.
- Currently, we also pursue a solution approach based on the notion of "fluid relaxation" of the original scheduling problem. This approach leverages a timed-continuous Petri net model that is abstracted from the original GSPN model.
- For RAS with non-stationary generation of their various process instances, throughput maximization can be addressed through Model Predictive Control (MDP) schemes, where the original quest for stability is substituted by a requirement for liveness preservation.

Concluding remarks

- Sequential RAS is an abstraction that finds extensive applicability in many technological domains.
- The corresponding applications must be controlled for (i) behavioral correctness and (ii) performance optimization.
- These two control problems are, respectively, very amenable to DES qualitative and quantitative control theory.
- The RAS logical control problem of deadlock avoidance / liveness-enforcing supervision has been extensively investigated by the DES community, and the current results are analytically and computationally very powerful.
- These results also demonstrate very vividly the value of identifying and exploiting “special structure” in DES supervisory control (SC) theory through (i) pertinent representations and (ii) customized methods.
- Some of the presented ideas and techniques that have been generated in the context of the RAS SC control problem, hold potential for applicability in other application domains of DES SC theory and practice.
- Finally, the presented results on RAS logical control also enable, in a substantial manner, the effective resolution of the complementary problem of the performance-oriented control of these systems.

Thank you!