Synthesis of Reliable and Cost-Effective Cyber-Physical System Architectures

Nikunj Bajaj

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-41
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-41.html

May 1, 2015

Synthesis of Reliable and Cost-Effective Cyber-Physical System Architectures

by Nikunj Bajaj

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Alberto L. Sangiovanni Vincentelli
Research Advisor

(Date)

* * * * * * *

Professor Sanjit Seshia
Second Reader

(Date)

Abstract

Cyber physical system (CPS) are interconnection of computation, networking and physical elements. Modern CPS are distributed, networked and safety-critical systems and architectural design of such systems with fault tolerance and performance constraints is a challenging task. In this thesis, we address the problem of synthesizing safety-critical cyber-physical system architectures to minimize a cost function while guaranteeing the desired reliability. We cast it as an optimization problem with the component cost as the objective and the performance and reliability requirements as the constraints. The challenge is to generate symbolic reliability constraints, which is exponential in the size of the system due to exhaustive enumeration of failure cases on all possible system configuration. We propose two algorithms to overcome this problem that we refer to as Integer-Linear Programming Modulo Reliability (ILP-MR) and Integer-Linear Programming with Approximate Reliability (ILP-AR). ILP-MR solves an easier optimization problem with performance constraints and iteratively introduces redundancy in the system with a background reliability analysis routine. Conversely, ILP-AR solves the problem in one iteration by symbolically representing the reliability constraints computed using an in house developed approximate algebra. We compare the two approaches and demonstrate their effectiveness on the design of aircraft electrical power system architectures.

Acknowledgements

I offer my regards and blessings to all those who supported me in any aspect during the completion of this thesis. Foremost, I would like to express my sincere gratitude to my advisor Prof. Alberto Sangiovanni Vincentelli for offering me the opportunity to work on my Master Thesis at the University of California, Berkeley, under his guidance. I thank him for his continuous support, encouragement and advice throughout my stay at Berkeley.

I owe my deep gratitude to Pierluigi Nuzzo, for his patient guidance, continuous encouragement, insightful comments and constructive criticism. I really enjoyed working under his supervision on this compelling project. I thank Safa Messaoud and Mohammad Mozumdar for their contributions in the development of the design example. I thank Prof. Sanjit Seshia, Prof. Edward Lee, Prof. Jaijeet Roychowdhury, Prof. Stavros Tripakis and Prof. El Ghaoui for the kind transfer of knowledge and inspiration through lectures and meetings.

I am very grateful to our sponsors- the iCyPhy consortium, The Terraswarm Research Center, IBM and UTC. I am thankful to our industrial partners especially Michael Masin and Eelco Scholte for the productive discussions and their valuable feedback on the work.

I am thankful to my labmates John Finn, Liangpeng Guo, Alberto Puggelli, Mehdi Maasoumy, Baihong Jin, Ankush Desai and especially Shromona Ghosh, Gautam Kumar, Anurag Khandelwal, Antonio Iannopollo for their constant support during this work and to all my friends and colleagues for making my stay at Berkeley so amazing. Last but not the least, I would like to thank my father, my mother, my sisters and brothers-in-law, who have always been a constant source of love and support for me in every possible way.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

In this chapter, we give some background about Cyber Physical Systems (CPS) and the challenges involved in their design. We conclude with a summary of the contribution of our work.

1.1 Cyber Physical Systems

CPS imply a tight synergy of the physical world with the computation and communication worlds. A typical CPS is modeled as a distributed network of several interacting elements with a physical interface [2] [3]. For instance, a wireless sensor network, as shown in Fig. 1 a, monitoring certain aspect of the environment and sending data to a central server could be characterized as a typical CPS. The sensors reflect the physical part of the system that monitors properties of the environment (like temperature, pressure, humdidity). On the other hand, sensors might communicate with a central server where computations can take place to convert the local data into global information. The central node might communicate back to the distributed sensor network, for instance, to increase or decrease the data polling rate. This constitutes the cyber part of the system. This scenario would typically entail an ensemble of sensors distributed at different geographical locations. Other examples of CPS include but not limited to, distributed robotics, automotive systems, smart grids.

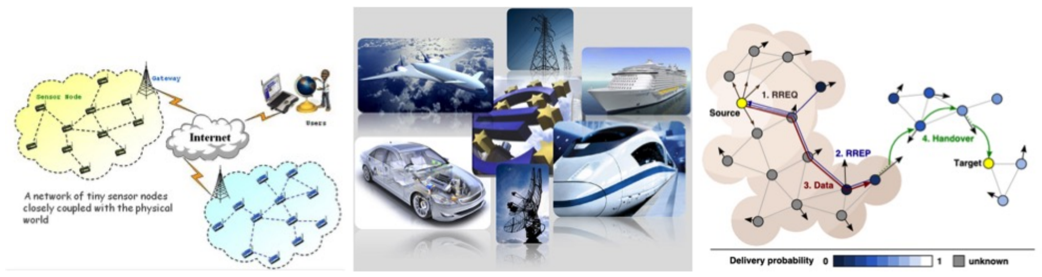


Figure 1: (a)Sensor Network (b) Automotive Systems (c) Communication Network

It is easy to notice from the above example, that a CPS architecture is an interconnection of heterogeneous components assembled to perform a specified function where embedded software components are connected in feedback with physical processes to form a large, distributed, networked control system subject to tight cost, safety and reliability constraints.

1.2 Design Challenges

The design of such systems places several challenges in terms of their analysis, testing and verification because of the large scale, performance constraints and heterogeneity of components and requirements. A derivative approach is often followed in the design process where minor additions are made over pre-existing implementations. This process hardly scales to the complexity of the system. Heuristic approaches fail to guarantee the desired reliability. The safety critical nature of the CPS entails design algorithms based on sound mathematical techniques whereas the complexity of the system necessitates scalable algorithms. It is, therefore, highly desirable to devise effective abstractions that enable co-design and optimization of CPS architectures under several, possibly conflicting concerns, while guaranteeing design correctness and fault tolerance.

Specifically, this work contributes towards architecture synthesis of systems with a focus on certain key aspects like physical properties of components, distributed nature of system, scale and reliability. We focus on a class of CPS that can be represented as a distributed network of components serving a common functionality modeled as a *flow*. Common examples of such systems include a communication network that represents a *flow of message* (Fig. 1 c.) from source to sink nodes, or a sensor network that represents a *flow of data* (Fig. 1 a.) from sensors to a central server, or a swarm of robots (serving a purpose such as surveillance) that represents a *flow of actuation commands* from a central server to the robots or amongst the robots themselves, or a power network that represents a *flow of power* from generator to load. These large systems are often composed of components that are subject to failure, still, used in safety-critical applications, which entails introducing redundancy in the system. Introducing redundancy implies increasing system cost and size, therefore, it is not possible to introduce redundancy arbitrarily in an attempt to make the system fault-tolerant.

1.3 Contributions

As mentioned above, introducing optimal redundancy in the system entails quantifying failure and a rigorous analysis of how component failure propagates to system level. We address this problem by using an optimization-based methodology for the synthesis of CPS architectures whose reliability is a function of the interconnection structure. Our goal is to minimize the overall system cost (e.g. number and weight of components) while guaranteeing that an upper bound on system failure probability is met. Our contributions can be summarized as follows:

- We provide a general graph representation of an architecture that allows an efficient casting of the design problem as an integer linear program (ILP), capable of modeling a variety of system requirements, such as connectivity, safety, reliability and energy balance;
- We propose two algorithms to decrease the complexity of exhaustively enumerating all failure cases on all possible graph configurations, i.e. Integer-Linear Programming Modulo Reliability (ILP-MR) and Integer-Linear Programming with Approximate Reliability (ILP-AR). ILP-MR *lazily* combines an ILP solver with a background *exact* reliability analysis routine. The solver *iteratively* provides candidate configurations that are analyzed and accordingly modified, only when needed, to satisfy the reliability requirements. Conversely, ILP-AR *eagerly* generates *monolithic* problem instances using *approximate* reliability computations that can still generate estimations to the correct order of magnitude, and with an explicit theoretical bound on the approximation error;
- We develop a tool ARCHEX that implements the two approaches and provides an easy interface to customize the system specifications and properties.
- We compare the performance of the two approaches and demonstrate their effectiveness on the design of safety-critical industrial-scale architectures for aircraft electrical power distribution.

The research in this thesis is the result of a collaborative effort. In particular, the presented work has been carried out in close collaboration with the co-authors of the paper [4], on which this thesis is largely based. While it is difficult to draw sharp boundaries among the distinct contributions, we observe that the contribution of the author of this thesis was instrumental in the development of Algorithm 2 jointly with P. Nuzzo, the development of the ILP-AR algorithm, jointly with P. Nuzzo, and under the guidance of M. Masin, and the implementation of the ILP-MR and ILP-AR algorithms under the guidance of P. Nuzzo.

The remainder of the thesis is organized as follows. After giving a brief survey of the previous work in CPS design in Chapter 2 we present the system model used in this thesis and mathematical formulation of the synthesis problem in Chapter 4. Then we detail the ILP-MR and ILP-AR algorithms, respectively, in Chapter 4 and Chapter 5. We compare both algorithms and validate their effectiveness on the design of aircraft electrical power system (EPS) architectures in Chapter 6. Finally, we draw conclusions and present some future work in Chapter 7.

Chapter 2
Previous Work

In this chapter we discuss the high-level challenges in CPS design and the methodologies to address these challenges, proposed in the literature. We then provide a brief analysis of the methods explored for the design of fault tolerant systems. We also compare how our design methodology and algorithms compare to the ones existing in the literature.

2.1 CPS Design Methodology

Although, the advent of CPS is a fairly recent phenomenon in both academia and industry, its economic and social potential has been realized and major investments are being made on this technology [5]. In turn, design of CPS has become a key focus area and researchers are well aware of the challenges in deploying the full potential of CPS in real world applications. As mentioned in [6] "Cyber Physical Systems will not be operating in a controlled environment and must be robust to unexpected conditions and adaptable to subsystem failures". A well-explored solution in this regard is model based design technique, which entails mathematical modeling of the physical systems, testing and simulating offline, putting the system together with a formal model of communication layer and repeating the steps at a higher level of abstraction. An interesting analysis has been provided in [7] that illustrates how these steps are inter dependent and a design methodology is proposed that facilitates correct coevolution of the model with system realization. Other key concerns pointed out in the literature regarding the design of CPS are the need of a strong semantic foundation and compositional reasoning. Carolyn Talcott [8] proposes an event-based semantics for CPS design. Event- based semantics has the potential to reason locally about individual components and globally about system properties. Lee [6] appropriately raises the concern of how to design predictable and reliable systems, given, components that might deviate from expected behavior under certain operating conditions. Our methodology addresses both these concerns. We work on event based semantics in designing CPS architecture, because we analyse how failure event of a component propagates to the system and enable the system to respond to combination of such events, whereas our platform-based-design (PBD) approach encapsulates compositional and heirarchical reasonsing.

Platform-based design(PBD) has been successfully adopted in the automotive and consumer electronics [9] domains to overcome similar challenges, by formalizing the design flow as a sequence of refinement steps from the original specification to the final implementation. A basic principle of PBD is the distinction between the *function* (what the system is supposed to do, i.e. the specifications) and the *architecture* (how specifications are realized, i.e. the components implementing the function together with their interconnections), which allows for automatic design space exploration. At each refinement step, the design is regarded as a platform instance, i.e. a valid composition of library elements that are pre-characterized by their cost and performance metrics. The objective is therefore to select a platform instance that correctly implements a given specification. The mapping of such a specification onto an architecture can be formalized by an optimization problem whose solution represents the functional specification to be implemented by the subsequent refinement step. This process repeats recursively until an implementation is reached.

2.2 Fault Tolerant Systems

The approach presented above allows us to evaluate reliability directly from the system structure, by associating a reliability model to each system component and interconnection, which is one of the key contributions of the work. Fault tolerance is a widely discussed topic in the literature, however in this chapter, we restrict the disucssion to methods pertaining to compositional reasoning. For instance, Kaiser et al. [10] proposed compositional fault trees that allows to construct fault trees of the system, from the fault trees of its components. Many traditional safety and reliability assessment is based on a set of methods that are hard to incorporate into automatic design space exploration and optimization frameworks. In addition to the complexity of exact network reliability analysis, which is an NP-hard problem [11], techniques such as Fault Tree Analysis (FTA) or Reliability Block Diagrams (RBD) often rely on a set of system abstractions, which are hardly interoperable with system design flows [10]. For instance, in FTA, causal chains leading to some failure are depicted as a tree, inherently describing a hierarchical breakdown. However, in FTA, decomposition into modules mostly relates to the hierarchy of failure influences rather than to the actual system architecture. Therefore, the integration of fault trees with other system design models, or the automatic generation of fault trees from design artifacts, if at all sufficient to model failure propagation paths, is not directly possible.

In contrast, our *compositional* approach and simplistic system representation allows us to optimize for reliability, performance and cost simultaneously. We are able to formulate two concrete algorithms- ILP-MR and ILP-AR that trades between domain knowledge and computation time respectively, to synthesize CPS archiectures. Instead of formulating a single, “flat” optimization problem, the ILP-MR algorithm avoids the expensive generation of symbolic reliability constraints via an iterative approach inspired by the *ILP Modulo Theory* paradigm [1, 12]. An instance of this approach has already shown to be promising when applied to topology design of aircraft power systems [12, 13] On the other hand, the ILP-AR algorithm aims to efficiently solve a *single optimization* problem, albeit of a larger size, to provide the optimal solution without repetitive calls to the exact reliability analysis function, which may also be expensive. ILP-AR exploits an approximate “algebra” for reliability calculation, inspired by the work of Helle et al. [14]. However, our algebra is *richer* than the previously proposed one, since it accounts for the number of redundant paths implementing a certain function as well as the number of components of the same *type* that are actually used in these paths. Our reasoning on individual components allows us to relax the simplifying assumption that any used component is either maximally redundant (i.e. participates in just one path) or essential (i.e. participates in all paths) because at this level of granularity we can account for the exact degree of redundancy of each component, instead of entire path which may share componnets. We also generate failure probability expressions that can be used in an ILP formulation, provide estimations to the correct order of magnitude, and with well-defined theoretical bounds on their potential “optimism”. Our algebra provides a better bound than the one proposed before along with proof based on a sound mathematical foundation. Finally, while [14] focuses on the analysis and combination of different failure cases, we assess, for the first time, the effectiveness of using an approximate reliability algebra on a *synthesis* problem.

2.3 Summary

This chapter endows a review of the challenges, methodologies and algorithms for CPS design proposed in the literature. We also give a brief overview of our approach. In the next chapter we provide a detailed mathematical formulation of our design problem.

Chapter 3
Problem Formulation

In this chapter, we formally introduce some key concepts and terminology about the system and give intuitive explanation drawn from two examples- Wireless Sensor Network (WSN) and Electric Power System (EPS). We also provide the mathematical formulation of architecture synthesis as an optimization problem.

3.1 System Modeling

In the context of Platform Based Design, we assume that a design is assembled out of a *library* (collection) $\mathcal{L}$ of *components* and *composition rules*. For instance, in a WSN the library might be composed of sensors, routers, base stations, gateways, servers or in case of EPS, components like generators, buses, rectifiers, loads might constitute the library. Each component is associated with a set of *attributes*, which are used to capture both its functional and non-functional properties, such as energy, performance, and cost. Sensors can be characterized by properties like battery life, communication range, size; generators can be characterized by power rating; almost any component will have an associated cost parameter. We also assume in this problem, that tentative failure probability is known for every component in the network which could be derived statistically or by reliability analysis of its components.

Components can be connected via *terminals* and terminal *variables*. At this level of abstraction, terminals are *logic* in nature. *Input* terminals are used to receive a signal or the value of a terminal variable; *output* terminals are used to send a signal or assign a value of a terminal variable. Composition rules define how terminals are connected and terminal variables are assigned between components. For the purpose of this report, $\mathcal{L}$ is parameterized by a set of vectors, including terminal variables $\boldsymbol{w}$, component costs $\boldsymbol{c}$, and failure probabilities $\boldsymbol{p}$.

In the context of synthesis, it becomes extremely important to distinguish between the notion of a template and an architecture, which can be formalized as below.

Definition 3.1 (Template). *A system* template $\mathcal{T}$ *consists of a finite set of* components $(\mathcal{X})$ *in which the set of nodes is fixed, while the interconnection structure is variable and can be reconfigured.*

Definition 3.2 (Architecture). *A system* architecture *consists of a finite set of* components *and their* interconnections*, and can be modeled as a directed graph* $\mathcal{G} =$

(V, E)*, where each node* $v_i \in V \subseteq \mathcal{X}$ *represents a component and each edge* $e_{ij} \in E$ *represents the interconnection from* v_i *to* v_j *(*$i, j \in \{1, \ldots, |V|\}$*,* $|V|$ *being the cardinality of* V*).*

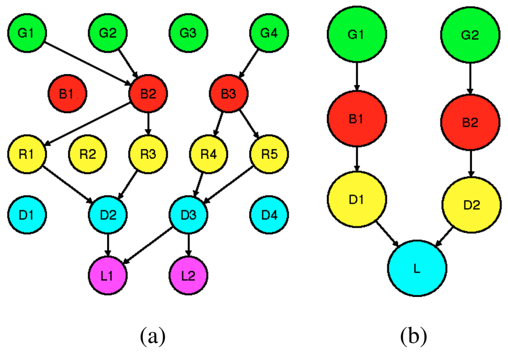


Figure 2: (a) Architecture template configuration example; (b) Architecture analyzed in Example 5.1.

An assignment over E is a *configuration*. As shown in Fig. 2 (a) which could represent an EPS or a WSN, the *template* $\mathcal{T}$ is the entire set of nodes (cardinality 18), architecture is the set of *"used"* (cardinality 13) components connected in the given configuration. Therefore, synthesis of an architecture refers to an assignment over edges or choosing a configuration over a subset of nodes in a given template while satisfying the composition rules. It is important to note that it is desired to synthesize architecture with minimal number of components. In practice, every node in a template can be mapped to an element of a physical architecture. In a reconfigurable architecture, edges can also be conveniently associated with *switches* to denote interconnections that are selectively activated. This is because the system might respond to component failure by changing the truth assignments over switches dynamically. Based on the attributes of our library, both nodes and edges can be labeled with the terminal variables $\boldsymbol{w}$, costs $\boldsymbol{c}$ and failure probabilities $\boldsymbol{p}$.

Finally, we assume that an external control unit can react to component failures, by modifying the link (switch) configuration to activate alternative paths from sources to sinks. The reliability of an architecture is then determined by its topological structure and the redundancy of the paths available to perform a critical function, associated to a *functional link*. To define a functional link, we assume that each component can also be labelled with a *type*, defining its functionality (role or task) in a system. We first introduce a partition on $\mathcal{G}$, to which we link the notion of component type as follows.

Definition 3.3 (Graph Partition and Component Type). *A* partition $\mathit{\Pi} = \{\mathit{\Pi}_1, \mathit{\Pi}_2, \ldots, \mathit{\Pi}_n\}$ *over the set of nodes* V *of* $\mathcal{G}$ *is a set of nonempty subsets of* V *such that* V *is a disjoint union of these subsets. We say that two nodes* a *and* b *have the same* type*, written* $a \sim b$*, when they belong to the same set in* $\mathit{\Pi}$*. If* a *is in* $\mathit{\Pi}_i$*, then we say that its type is* i*.*

In this regard, we mean to augment the template $\mathcal{T}$ with an additional parameter called *type* with each component in the set. For instance, in case of power system the type of components might refer to different electrical components that serve a specific function: Generators - supplies power, AC buses- carries AC Power, Rectifiers- converts AC power to DC power, DC Buses- carries DC power or Loads- converts the electrical energy to different usable forms. This entire system together is performing a functionality of carrying power from the generator to the load.

We also recall that a *walk* $\mu(v_a, v_b)$ of $\mathcal{G}$ is a sequence of nodes $\{n_0, \ldots, n_k\}$ such that $n_0 = v_a$, $n_k = v_b$ and $e_{n_i n_{i+1}} \in E$ for each i. When all nodes in μ are distinct, we say that μ is a (simple) *path*, and write $|\mu|$ to denote the length of μ. Let $\mathit{\Pi}_1$ and $\mathit{\Pi}_n$ be the subsets of V including, respectively, all sources and sinks. Then, a *functional link* F_i is the set of paths from any source in $\mathit{\Pi}_1$ to a sink $v_i \in \mathit{\Pi}_n$ that are used to perform an essential system function, on which a reliability requirement is given. For instance, in Fig. 2 functional link corresponding to sink $L1$ is given by $\{G1 \rightarrow B1 \rightarrow R1 \rightarrow D1 \rightarrow L1; G2 \rightarrow B2 \rightarrow R3 \rightarrow D2 \rightarrow L1; G4 \rightarrow B3 \rightarrow R4 \rightarrow D3 \rightarrow L1\}$.

In practice, such a function may consist in transferring data or energy from a source to the sink through a sequence of input-output links. For example, in case of WSN different components might serve different functionalities in the communication stack. For instance, sensors might characterize the physical layer, while gateways might serve as the network layer. Switches might represent communication between physical and data link layer while routers might be used for communication between data link and network layer. The web servers could represent the application layer. Again, any subset of components that includes atleast one component of each type in serving a function of carrying the sensor data to the server for further processing may be regarded as a functional link. We should note here, that to have one component of each type is not a restrictive assumption because our framework makes no assumptions on the number of functional links. If there exists one or more paths performing a required function and including different types of components then a new functional link can be defined to represent the set.

3.2 Optimization Problem

Based on the definitions above, we cast the architecture synthesis problem as an optimization problem. *Given a library $\mathcal{L}$ and a template $\mathcal{T}$, our goal is to derive a configuration that satisfies a set of interconnection and reliability requirements, while minimizing the cost and the complexity (number of components) of the overall network.* The set of Boolean variables E will then include our decision variables. Based on the final assignment over E, some of the edges and nodes in $\mathcal{T}$ will be selected to generate an optimal architecture; unnecessary nodes and edges will instead be pruned away to minimize the overall cost. In the following, we provide example formulations for the objective function and the requirements in terms of Boolean arithmetic constraints.

Objective Function Let e be the adjacency matrix of $\mathcal{T}$, i.e.

$$e_{ij} = \begin{cases} 1 & \text{if there is one connection from } v_i \text{ to } v_j \\ 0 & \text{otherwise} \end{cases}$$

Then, the *objective function* can be expressed as the sum of the costs of all components (associated with nodes) and switches (associated with edges) used in the template, i.e.

$$\sum_{i=1}^{|V|} \delta_i c_i + \sum_{i=1}^{|V|} \sum_{j=i+1}^{|V|} (e_{ij} \vee e_{ji}) \tilde{c}_{ij} \tag{1}$$

where c_i is the cost of component i, $\tilde{c}_{ij}$ is the cost of the switch on edge e_{ij}, and δ_i is a binary variable equal to one if the component is instantiated in a configuration and zero otherwise. We express δ_i in terms of the edge variables as

$$\delta_i = \bigvee_{j=1}^{|V|} (e_{ij} \vee e_{ji}) \tag{2}$$

meaning that δ_i is one if there exists at least an edge (incoming or outgoing) between v_i and any other node in the graph (we assume $e_{ii} = 0$ for all i). Moreover, we use $(e_{ij} \vee e_{ji})$ when computing the cost of the switches, to avoid double counting the contribution of a switch associated to a bidirectional interconnection.

Interconnection Constraints *Interconnection* requirements originate from the composition rules in $\mathcal{L}$ and are used to enforce legal connections among components.

For example, let D, L, and B be subsets of V. Then, we can prescribe that there exists at least (most) one connection from a node in L and a node in D as follows:

$$\sum_{i=1}^{|D|} e_{l_j d_i} \geq (\leq)\, 1 \quad \forall\, j \in \mathbb{N} :\ 1 \leq j \leq |L|, \tag{3}$$

where $e_{l_j d_i}$ is the edge from node l_j to node d_i, and the inequality turns into an equality when one and only one connection is admitted. The above pattern of constraint is often observed in networked architectures because a component can communicate with the rest of the network through another specific component only and it must be connected to the same.

For instance, suppose in a WSN L represents a set of sensors and D represent a set of gateways. For the sensors to communicate with the server, data has to pass through a router. This could be formulated as an interconnection constraint of the form of equation Eqn. (4). The textual translation of the above statement is that there exists at least one path from every sensor to a router. Similarly, in case of an EPS a typical connectivity constraint that forbid parallel connection of AC Sources can be easily formulated in the form of above equation. Suppose we denote the set of AC generators by G and the set of Buses by B, then the constraint can be formulated as

$$\sum_{i=1}^{|G|} e_{b_j g_i} \leq\ 1 \quad \forall\, j \in \mathbb{N} :\ 1 \leq j \leq |B|, \tag{4}$$

Moreover, we can state that if there exists an interconnection from any node in L to a node d_j in D, then d_j must be connected to at least one node in B, using a constraint of the form:

$$\bigvee_{1 \leq i \leq |L|} e_{l_i d_j} \leq \bigvee_{1 \leq k \leq |B|} e_{d_j b_k} \quad \forall\, j \in \mathbb{N} :\ 1 \leq j \leq |D|. \tag{5}$$

Referring to the WSN, if L represents the set of sensors, D represents the set of routers and B represents the set of gateways then a constraint can be imposed in the form of equation 5. It states that if a router is connected to a sensor then it must be connected to at least one gateway.

Another set of interconnection constraints can be used to enforce conservation laws or balance equations in physical systems, e.g. an EPS requiring that the maximum power provided by a source in $\mathcal{T}$ is greater than or equal to the maximum power required by the connected sinks. Let d be a node in the graph, which is neither a source nor a sink. Let B be the set of direct predecessors of d, and L be the set of its direct successors. Then, the balance equation at the terminals of d can be written as

$$\sum_{i=1}^{|B|} w_{b_i} e_{b_i d} \geq \sum_{j=1}^{|L|} w_{l_j} e_{d l_j}. \tag{6}$$

Interconnection requirements as the ones above originate linear arithmetic constraints in the decision variables, or include logical operations (conjunctions and disjunctions) that can be linearized with standard techniques [15].

Reliability Constraints A typical reliability requirement prescribes that the failure probability of a sink, i.e. the probability that a sink gets disconnected from a source because of failures, should be less than a desired threshold. Therefore, to formulate a reliability constraint, we need to compute the probability of composite failure events in the system, starting from the failure probabilities of the components. Specifically, we denote as *system failure* R_i an event in which there is no possible connection between any of the available sources and a sink i, i.e. when the functional link F_i breaks, and as *reliability level* r_i the probability of R_i. Practically, the above notion of failure models the interruption of any information or energy transfer to an essential portion of the system. We assume that when a component fails it cannot be recovered, and the adjacent links are no longer usable. Moreover, failures in different components are independent.

Let P_i be the event that component i fails (self-induced failure). Then, the event R_i of a system failure affecting component i can be recursively computed as follows

$$R_i = P_i \cup \bigcap_{1 \leq j \leq |V|, e_{ji} \neq 0} R_j, \tag{7}$$

where e_{ji} is j^{th}-row, i^{th}-column element of the adjacency matrix $\mathbf{e}$ of $\mathcal{T}$. In other words, component i fails when either a failure is generated in itself, or when failures are induced through its predecessors. A symbolic constraint for r_i can then be generated using Eqn. (7) to enumerate all possible failure events while traversing $\mathcal{T}$ from node i to the sources. However, such an exact computation, based on the enumeration of all possible component failure events, has exponential complexity on a fixed graph configuration [11]. The problem is further exacerbated when compiling a symbolic expression for a reconfigurable graph, since, in general, enumerating all possible configurations results in exponential number of constraints. To overcome this issue, we propose two approaches to the solution of the optimal architecture selection problem, namely, ILP-MR and ILP-AR, which we detail in the following chapters.

3.3 Summary

In this chapter we provided details on how we model a Cyber Physical System for architecture synthesis. We introduce our formalism and key terminology and explain them with some examples. We also frame architecture synthesis as an optimization problem and provide a method to obtain the objective function (cost of system) from the system representation. We also provide formulation of some generic constraints and show how they can be used to represent typical consraints on CPS architecture.

Chapter 4
ILP Modulo Reliability

We pointed out in that architecture synthesis of CPS typically involves interconnection, energy balance and reliability constraints. Out of these, it is relatively simpler to formulate the interconnection and energy balance requirements as integer linear constraints. However, reliability analysis of the network is an NP Hard problem and results in an exponential number of constraints. Therefore, in our ILP- MR algorithm we solve a simpler optimization problem without introducing the reliability constraint at all. The optimizer therefore results in a minimal redundancy topology satisfying the inter-connection and energy balance constraints. If the system does not meet reliability requirements then the optimizer is called iteratively (with a reliability analysis module in loop) with added constraints till sufficient redundancy is introduced in the system.

4.1 Algorithm

As mentioned above, the ILP Modulo Reliability (ILP-MR) algorithm avoids the expensive generation of symbolic reliability constraints by adapting the ILP Modulo Theory approach [12] to reliability computations, as summarized in Algorithm 1. ILP-MR receives as inputs: the library of components $\mathcal{L}$, together with their attributes $\boldsymbol{c}$, $\boldsymbol{w}$ and $\boldsymbol{p}$, the template $\mathcal{T}$, and the set of requirements, including interconnection and reliability constraints. To simplify, we assume that r is the worst case failure probability over a set of nodes of interest, for which the same reliability requirement r^* must be satisfied.

An ILP is solved in a loop with a reliability analysis routine. SOLVEILP generates minimum cost architectures for the given set of interconnection constraints. The RELANALYSIS routine computes the probability of composite failure events at critical nodes, starting from the failure probabilities of the components, a problem known as K-terminal reliability problem in the literature [11]. To do so, we implement a modified depth-first search algorithm to traverse the graph $\mathcal{G}$, representing the architecture, from node i (root) to the source nodes (leaves), by applying a path enumeration method, and by turning event relations as in Eqn. (7) into probability expressions. However, any other exact reliability analysis method for directed graph can also be used [11]. Although the K-terminal reliability problem is NP-hard, the key idea is to solve it only when needed, i.e. a small number of times, and possibly on smaller graph instances.

Algorithm 1 *ILP Modulo Reliability (ILP-MR)*

Input: Architecture template $\mathcal{T}$, component variables $\boldsymbol{w}$, costs $\boldsymbol{c}$ and failure probabilities $\boldsymbol{p}$, reliability requirement r^*

Output: Adjacency matrix $\mathbf{e}^*$ of the final architecture $\mathcal{G}^*$

```
$r \leftarrow 2r^*$
$(Cost, Cons) \leftarrow$ GenILP($\mathcal{T}$, $\boldsymbol{w}$, $\boldsymbol{c}$)
while $r > r^*$ do                                          ▷ failure probability
    $\mathbf{e}^* \leftarrow$ SolveILP($Cost$, $Cons$)
    if $\mathbf{e}^* = [\,]$ then return UNFEASIBLE
    $r \leftarrow$ RelAnalysis($\mathbf{e}^*$, $\boldsymbol{p}$)
    if $r > r^*$ then
        $Cons \leftarrow$ LearnCons($Cons$, $r$, $r^*$, $\mathbf{e}^*$)
    if $Cons = [\,]$ then return UNFEASIBLE
return $\mathbf{e}^*$
```

At each iteration of ILP-MR, if the optimal architecture satisfies the reliability constraints, it is returned as the final solution. Otherwise, LearnCons estimates the number of redundant paths needed to achieve the desired reliability and suggests a set of strategies to implement the required paths by augmenting the original optimization problem with a set of interconnection constraints. This constraint learning function is, therefore, instrumental to efficiently converge towards a reliable architecture, while minimizing the number of calls to RelAnalysis. We provide details about this function in the following section.

4.2 Learning Constraints to Improve Reliability

When no reliability constraints are enforced in the ILP, the solver attempts to use the minimum number of components and interconnections to perform a specific function at minimum cost. Typically, such a "minimal" architecture has also minimal redundancy, hence minimal reliability. Based on this intuition, we develop strategies that increase the reliability of the solution, albeit at a higher cost, by enforcing a larger number of redundant components and interconnections. An iterative approach similar to ILP-MR has been recently reported, based on a set of *ad hoc* strategies, customized for the specific problem at hand [1] [16]. In Algorithm 2, we introduce instead a generic template that can directly apply to the general problem formulation in Chapter 4, hence to a broader set of applications.

The idea is to add more components and connections to increase the reliability. The challenge is to determine the granularity of how much redundancy should be added. On the two extremes exist: Firstly, adding all the components available in the template and create a maximally redundant architecture in the first iteration; Secondly, adding only one component or one connection each time and iterate till reliability requirement is met or the template is exhausted. While the second approach may become computationally expensive and take a long time to converge, the first approach may result in an over redundant architecture. Our approach described in Algorithm 2 is incremental i.e. it increases reliability only when required, still, it ensures fast convergence by minimizing the number of calls to the optimizer. It increases the reliability of the system in two stages which we refer to as Coarse Tuning and Fine Tuning. The first stage operates when the reliability of the obtained architecture is far from the requirement while the second stage initiates when the obtained reliability is in the proximity of requirement. We measure the closeness of the obtained reliability to required reliability by introducing the notion of a reliability of a path calculated from reliability of components. If the obtained reliability differs from the required reliability by more than the reliability of single independent path from source to sink then we introduce more paths which increases the reliability by orders of magnitude otherwise we fine tune reliability by incrementally adding one component/ connection at a time.

Algorithm 2 LEARNCONS

Input: Current constraints $Cons$, reliability r, reliability requirement r^*, adjacency matrix $\mathbf{e}^*$
Output: Final constraints $Cons$
Based on the current reliability level r, LEARNCONS estimates the number of additional redundant paths k required to satisfy the desired reliability r^* (function ESTPATH). As an example, under the assumption that all the paths in a functional link F are independent, then k can be computed as

$$k = \lfloor \log(r^*/r)/\log\rho \rfloor, \tag{8}$$

where ρ is the failure probability of a single path in F, and $\lfloor x \rfloor$ denotes the integer part of x.

```
$k \leftarrow$ ESTPATH($r, r^*, \mathbf{e}^*$)
$NewCons \leftarrow [\,]$
$S \leftarrow$ GETSINKS($\mathbf{e}^*$)
$(T_1, \ldots, T_n) \leftarrow$ GETTYPES($\mathbf{e}^*$)
for all $v \in S$ do
    if $k \geq 1$ then
        for all $i \in (T_{n-1}, T_{n-2}, \ldots, T_1)$ do
            $NewCons \leftarrow$ ADDPATH($v$, $i$, $k$, $NewCons$, $\mathbf{e}^*$)
    else
        $i \leftarrow$ FINDMINREDTYPE($v$, $\mathbf{e}^*$)
        $NewCons \leftarrow$ ADDPATH($v$, $i$, 1, $NewCons$, $\mathbf{e}^*$)
    if $NewCons = [\,]$ then return UNFEASIBLE
$Cons \leftarrow (Cons, NewCons)$
return $Cons$
```

Since, in reality, the paths in F are not independent, this is a conservative estimation which avoids over-design. Then, if at least one additional path is required, for a sink node and component types implementing F and used in the current architecture, ADDPATH generates new constraints to enforce that at least k additional components of each type have a path to the sink. These constraints do not necessarily translate into instantiating more components, as far as additional paths to the sink can be obtained by just increasing the number of interconnections. If k additional paths cannot be obtained with the current template, ADDPATH attempts to enforce the maximum available number of paths. Conversely, if the estimated number of paths is zero, LEARNCONS attempts to still improve the overall reliability by enforcing one additional path between the sink and a component whose type has minimum redundancy in the current architecture, i.e. for which the total number of paths to the sink is minimum (as obtained from FINDMINREDTYPE). If no additional paths can be added between a sink and a component of any type, LEARNCONS terminates with `UNFEASIBLE`.

To enforce additional paths, ADDPATH uses the walk indicator matrix of $\mathcal{T}$, defined below.

Lemma 4.1 (Walk Indicator Matrix). *Let* $\mathbf{e}$ *be the adjacency matrix of a graph* $\mathcal{T}$*; let* $\mathbf{a} \odot \mathbf{b}$ *the* logical product *of two logical matrices* $\mathbf{a}$ *and* $\mathbf{b}$ *in* $\mathbb{B}^{m \times m}$*, defined as* $(\mathbf{a} \odot \mathbf{b})_{ij} = \bigvee_{k=1}^{m} a_{ik} \wedge b_{kj}$*; let* $\mathbf{e}^k = \underbrace{\mathbf{e} \odot \ldots \odot \mathbf{e}}_{k \text{ times}}$ *be the* k*-th* logical power *of* $\mathbf{e}$*. Then, the entry in row* i *and column* j*,* $\eta_{n_{ij}}$*, of the* walk indicator *matrix* $\boldsymbol{\eta}_n = \bigvee_{k=1}^{n} \mathbf{e}^k$ *is* 1 *if and only if there exists a directed walk of length less or equal* n *from vertex* v_i *to vertex* v_j*.*

We can then require at least k additional connections of components of type T_i, belonging to the set Π_i of the partition Π of $\mathcal{T}$, to a sink v via at least one path of length $n - i + 1$ by enforcing

$$\sum_{w \in \Pi_i} \eta_{n-i+1_{w,v}} \geq k + \sum_{w \in \Pi_i} \eta^*_{n-i+1_{w,v}}, \tag{9}$$

where η_{n-i+1} and η^*_{n-i+1} are the walk indicator matrices, respectively, for $\mathcal{T}$ (decision variables) and the current architecture $\mathcal{G}^*$. The constraint (9) can be converted into an equivalent set of linear constraints in the elements of $\mathbf{e}$ (edge variables) by using standard linearization techniques. The following result summarizes the properties of the ILP-MR approach.

Similarly, FINDMINREDTYPE determines the type j with minimum redundancy as

$$j = \arg\min_{\Pi_i \in \Pi} \sum_{w \in \Pi_i} \eta^*_{n-i+1_{w,v}}. \tag{10}$$

Theorem 4.1 (Soundness and Correctness of ILP-MR). *For a given template $\mathcal{T}$ and within the approximation error ϵ of the* SOLVEILP *and* RELANALYSIS *routines, ILP-MR (Algorithm 1) is sound and complete.*

4.3 Soundness and Completeness of ILP-MR

To prove the above theorem we introduce an architecture transformation function δ that takes an architecture A and produces another architecture A'. This is a representation of a conjunction of the above two algorithms because Algorithm 2 requests change in architecture (to increase redundancy) and Algorithm 1 implements the same. Here we assume architecture A represents the functional link corresponding to one sink node. The same analysis can be repeated for other functional links without loss of generality. Let us also denote the architecture A' obtained from the i^{th} call to the function δ by A_i. Clearly A_0 is the first minimal redundancy architecture obtained from the first call to Algorithm 1. We also denote by q_i as the failure probability of the sink node of architecture A_i. Let us denote $\mathcal{A} = A_0, A_1, A_2 \ldots$ as the sequence of architectures obtained from the repeated application of function δ and we define a set ordering as $A_i \leq A_j$ if $q_j \geq q_i$. We know that the function δ always add more components or connections on existing architecture $\therefore$ $q_0 \geq q_1 \geq q_2 \ldots$ and $\mathcal{A}$ is a totally ordered set.

We know that the algorithm terminates when either $q_i \geq r^*$for i $\in 0, 1, 2 \ldots$, (where $r*$ denotes the reliability requirement) i.e. the obtained architecture meets the reliability requirement or the function δ reaches a fixed point, $\delta(A_i) = A_i$, i.e. all components and connections in the given template are exhausted. We observe that, since the number of components in $\mathcal{T}$ is finite, the ILP-MR routine, based on Algorithms 1 and 2, will terminate.

Moreover, if the algorithm terminates because of the condition $q_i \geq r^*$, we know that architecture A_i will satisfy all the requirements, because RELANALYSIS implements an exact reliability analysis method, being only subject to the rounding error ϵ due to the ILP solver and to RELANALYSIS. On the other hand, because all available components will be eventually used to increase the reliability, if ILP-MR terminates with `UNFEASIBLE`, then there is no architecture, obtained from the given template, which is able to satisfy all the constraints.

4.4 Summary

In this chapter we introduced our ILP-MR algorithm that solves the synthesis problem in loop with a reliability analysis routine. We show efficient and generic ways to iterate between the optimizer and reliability analysis routine and prove the termination, soundness and completeness of the algorithm.

Chapter 5
ILP with Approximate Reliability

As pointed in Section 3.2 exhaustive enumeration of all failure cases and generating exact reliability expression can result in exponential number of constraints. The ILP-AR algorithm replaces exact reliability computations with an approximate algebra that allows encoding a reliability requirement into a number of linear constraints on Boolean variables. In Sec. 5.1 we introduce the approximate reliability algebra, which estimates the correct order of magnitude for the failure probabilities of all the components in an architecture by leveraging the fact that components with the highest failure probability tend to dominate the overall failure probability. Then, in Sec. 5.3 we report the main results on correctness and complexity of ILP-AR.

5.1 Approximate Reliability Algebra

In the approximate algebra, components contribute to the system failure probability based on their *degree of redundancy*, which is defined based on the notions of functional link and component type defined in Chapter 4. We recall that any path in a functional link F_i consists of an interconnection of components, each having a role or performing a sub-task defined by its type. Components of the same type can be interchanged and introduce redundancy in the system architecture. We say that a component type j, associated to a partition Π of $\mathcal{G}$, *jointly implements* a functional link F_i, written $\Pi_j \vdash F_i$, if all paths in the functional link F_i include at least one node in Π_j, i.e.

$$\Pi_j \vdash F_i \text{ iff } \forall \mu \in F_i : \mu \cap \Pi_j \neq \emptyset. \tag{11}$$

Trivially, we have $\Pi_1 \vdash F_i$ and $\Pi_n \vdash F_i$ for all i. Moreover, multiple nodes of the same type are allowed in a path as far as they are adjacent to each other. This is not a restrictive assumption because if same type of components occur at different levels of the path then they can just be named as a different type of component and the calculations do not change. Given a path μ, possibly including multiple instances of the same type, we denote as $\hat{\mu}$ the *reduced path* obtained from μ after replacing all the instances of the same type with a single node, still of the same type.

We can then define the degree of redundancy h_{ij} associated with type j and link F_i as the number of components of type j used in at least one reduced path of

F_i, i.e. $h_{ij} = |(\cup_{\mu \in F_i} \hat{\mu}) \cap \Pi_j|$. Finally, we approximate the failure probability r_i of a functional link F_i by

$$\tilde{r}_i = \sum_{j \in I_i} h_{ij} p_j^{h_{ij}} \tag{12}$$

where $I_i = \{j | \Pi_j \vdash F_i\}$ is the set of all component types that jointly implement F_i, p_j is the probability of failure of any of the components of type j, and h_{ij} is the degree of redundancy associated with type j and link F_i.

Example 5.1. We illustrate the application of (12) to the architecture $\mathcal{E}$ represented in Fig. 2 (b). We consider a partition where $\Pi_1 = \{G_1, G_2\}$, $\Pi_2 = \{B_1, B_2\}$, $\Pi_3 = \{D_1, D_2\}$ and $\Pi_4 = \{L\}$. Sink L is connected to sources G_1 and G_2 via two (reduced) paths, each using components of all the four types listed above. Then, the approximate expression for the failure probability of L would be $\tilde{r}_L = p_L + 2p_D^2 + 2p_B^2 + 2p_G^2$, while exact calculations lead to

$$r_L = p_L + (1 - p_L)\{p_D + (1 - p_D)[p_B + (1 - p_B)p_G]\}^2.$$

When all components are assumed to fail with the same probability p, we obtain $\tilde{r}_L = p + 6p^2$ and $r_L = p + 9p^2 + O(p^3)$.

5.2 Bound on Approximation

The estimation in Example 5.1 has the same order of the exact calculation, and the error becomes negligible for small p. In general, it is possible to state the following theorem, providing a bound on the error of the approximate reliability algebra.

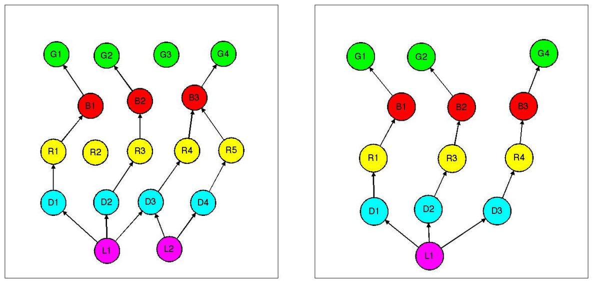
Theorem 5.1. *Given a graph $\mathcal{G}$ and a partition Π, let $\tilde{r}$ and r be, respectively, the approximate and exact failure probability for a functional link $F = \{\mu_1, \dots, \mu_f\}$, denoted by its set of f independent paths ($f = |F|$). Let $I = \{j | \Pi_j \vdash F\}$ the set of component types jointly implementing F. Then, the following inequality holds:*

$$\frac{\tilde{r}}{r} \geq \frac{mf}{M_f}, \tag{13}$$

where $M_f = \prod_{j=1}^{j=f} |\mu_j|$ and $m = |I|$.

Based on Theorem 5.1, (12) can provide "optimistic" estimations, but the bound to such optimism can be explicitly estimated for a given graph template $\mathcal{T}$.

The above theorem establishes a bound on the approximation in calculation of failure probability. The bound is subject to the constraint on the architecture of the system that restrains connection to exist only between subsequent layers, which is characterised by type as defined in Chapter 4. Also, we recall here that a functional link is the set of paths from any source in Π_1 to a sink $v_i \in \Pi_n$.



(a) (b)

Figure 3: (a) Architecture template configuration for multiple loads; (b) Architecture template configuration for single load

For instance, in figure 3 nodes of different colors represent different types (interchangeably called layers). Green color(top layer) represent source nodes while cyan(bottom layer) represents the sink layer. The set of paths from source nodes to a particular sink node is called a functional link. For instance, functional link corresponding to sink $L1$ is given by $\{G1 \rightarrow B1 \rightarrow R1 \rightarrow D1 \rightarrow L1; G2 \rightarrow B2 \rightarrow R3 \rightarrow D2 \rightarrow L1; G4 \rightarrow B3 \rightarrow R4 \rightarrow D3 \rightarrow L1\}$.

Without loss of generality we can perform the analysis on one sink node because the same calculation can be carried out for other sink nodes. We represent functional link corresponding to sink node $L1$ in figure 3. One such generic template can involve n types of components, represented by the set $\{T_1, T_2, ..., T_n\}$, jointly implementing the function. We denote the number of instances of components of the respective layers used in a given functional link by $c_1, c_2, ...c_n$. In Figure 3, $n = 4$, $c_1 = c_2 = c_3 = c_4 = 3$.

Now suppose we denote the event of failure of a component X by $E(X)$. Here, we assume that the sink node failure is induced i.e it does not fail itself, but the failure happens when all paths connecting the sink node to the source nodes fail. For all other nodes in the architecture we assume that the failure could result due to self failure or induced failure. Therefore for any component X

$$E(X) = E(X_s) \cup E(X_i), \tag{14}$$

where, $E(X_s)$ and $E(X_i)$ denotes self and induced failure of component X respectively.

Suppose we intend to find an expression for the event of failure of a sink node, in a maximally redundant architecture represented by the above template. Clearly, under our connectivity constraints (connection is allowed only between subsequent layers) maximally redundant architecture is obtained if every component is connected to all nodes in the layer immediately above. Trivially, the top layer represents the source and bottom layer represents sink. We can now state that for a sink node L_1 the failure event can be represented as

$$\begin{aligned}
E(L_1) &= E(L_{1_i}), \text{ assuming } E(X_s) \text{ is a zero-probability event} \\
&= \bigcap_{k=1}^{c_1} E(T_{1k}) \\
&\quad \text{where, } E(T_{jk}) \text{ represents event of failure of } k^{th} \text{ component in } j^{th} \text{ layer} \\
&= \bigcap_{k=1}^{c_1} \left[E(T_{1k_s}) \bigcup E(T_{1k_i}) \right] \\
&= \bigcap_{k=1}^{c_1} \left[E(T_{1k_s}) \bigcup \left[\bigcap_{j=1}^{c_2} E(T_{2j}) \right] \right] \\
&= \left[\bigcap_{k=1}^{c_1} E(T_{1k_s}) \right] \bigcup \left[\bigcap_{j=1}^{c_2} E(T_{2j}) \right]
\end{aligned} \tag{15}$$

Let us now denote the event of failure of sink node L_1, when k layers of components are present by V_k. Clearly from Equation 15

$$\begin{aligned}
V_1 &= \bigcap_{k=1}^{c_1} E(T_{1k}) \\
&= \bigcap_{k=1}^{c_1} E(T_{1k_s}), \text{ since there is only one layer, } E(T_{1k}) = E(T_{1k_s}) \\
V_2 &= \left[\bigcap_{k=1}^{c_1} E(T_{1k_s}) \right] \bigcup \left[\bigcap_{j=1}^{c_2} E(T_{2j}) \right] \\
&= \left[\bigcap_{k=1}^{c_1} E(T_{1k_s}) \right] \bigcup \left[\bigcap_{j=1}^{c_2} E(T_{2j_s}) \right] \\
&\quad \text{since there are only two layers, } E(T_{2j}) = E(T_{2j_s})
\end{aligned} \tag{16}$$

Following the trend as shown in Equation 16 let us assume

$$\begin{aligned}V_m &= \bigcup_{j=1}^{m-1}\left[\bigcap_{k=1}^{c_j} E(T_{jk_s})\right] \bigcup \left[\bigcap_{k=1}^{c_m} E(T_{mk})\right] \\ &= \bigcup_{j=1}^{m}\left[\bigcap_{k=1}^{c_j} E(T_{jk_s})\right] \text{, since there are only } m \text{ layers, } E(T_{mj}) = E(T_{mj_s})\end{aligned} \tag{17}$$

If we can prove that V_{m+1} can also be expressed in the form above, then it concludes the inductive step of the proof.

$$\begin{aligned}V_{m+1} &= \bigcup_{j=1}^{m-1}\left[\bigcap_{k=1}^{c_j} E(T_{jk_s})\right] \bigcup \left[\bigcap_{k=1}^{c_m} E(T_{mk})\right] \\ &= \bigcup_{j=1}^{m-1}\left[\bigcap_{k=1}^{c_j} E(T_{jk_s})\right] \bigcup \left[\bigcap_{k=1}^{c_m} \left[E(T_{mk_s}) \cup E(T_{mk_i})\right]\right] \\ &= \bigcup_{j=1}^{m-1}\left[\bigcap_{k=1}^{c_j} E(T_{jk_s})\right] \bigcup \left[\bigcap_{k=1}^{c_m} \left[E(T_{mk_s}) \cup \left[\bigcap_{r=1}^{c_{m+1}} E(T_{m+1}, r)\right]\right]\right] \\ &= \bigcup_{j=1}^{m}\left[\bigcap_{k=1}^{c_j} E(T_{jk_s})\right] \bigcup \left[\bigcap_{r=1}^{c_{m+1}} E(T_{m+1,r})\right] \\ &= \bigcup_{j=1}^{m+1}\left[\bigcap_{k=1}^{c_j} E(T_{jk_s})\right] \\ &\qquad \text{since there are only } m+1 \text{ layers, } E(T_{m+1,j}) = E(T_{m+1,j_s})\end{aligned} \tag{18}$$

The above derivation establishes that for a maximally redundant functional link comprising n types of components the failure event of sink node can be represented in terms of failure of other components in the architecture as

$$E(L_1) = \bigcup_{j=1}^{n}\left[\bigcap_{k=1}^{c_j} E(T_{jk_s})\right] \tag{19}$$

If all the paths in the functional link have to be independent then $c_1 = c_2 = \ldots = c_n = c$, thus,

$$E(L_1) = \bigcup_{j=1}^{n}\left[\bigcap_{k=1}^{c} E(T_{jk_s})\right] \tag{20}$$

Therefore, we know that for any architecture that can be represented by using the same set of components as above, the Reliability Algebra approximates it to the same configuration. Clearly, if the obtained architecture is also maximally redundant then the approximation is exact and this is the best case. If more and more paths are removed the appproximation worsens and the worst case occurs when under the connectivity constraints the obtained architecture is minimally redundant or functional link F contains a set of parallel paths from source to sink. Therefore, if we can establish a bound on the approximation in the worst case, that would hold for all other cases in a given template. Formally, the minimally redundant architecture with n types of components and c independent paths would have c paths from a source node to the sink node under consideration, each path consisting of n components. The failure event of the sink node under this configuration denoted by $E(\hat{L_1})$ can be given as

$$E(\hat{L_1}) = \bigcap_{k=1}^{c} \left[\bigcup_{j=1}^{n} E(T_{jk_s}) \right] \tag{21}$$

The formula means that the load would fail when each of the c paths connecting it to a source node fail and a path would fail if any component in the path fails. The next step is to prove a bound on probability of failure of sink node in maximally redundant and minimally redundant configurations, represented in equations 20 and 21 respectively. It is easy to show that the deviation in the probability expression of the events $E(L_1)$ and $E(\hat{L_1})$ is worst when we consider all components to have the failure probability p, where

$$p = max_{v \in V}\, p_v, \text{where } p_v \text{ is the failure probability of node v} \tag{22}$$

Under the above assumtions, probability of failure of load under maximally redundant configuration can be given by

$$P\big(E(L_1)\big) = P\left(\bigcup_{j=1}^{n} \left[\bigcap_{k=1}^{c} E(T_{jk_s}) \right] \right) \tag{23}$$

Let us denote the event of failure of all components at level j by

$$E(T_j) = \bigcap_{k=1}^{c} E(T_{jk_s}),\ \forall j \in 1, 2, \ldots, n \tag{24}$$

$$\begin{aligned}
P\big(E(T_j)\big) &= \prod_{k=1}^{c} P\big(E(T_{jk_s})\big) \text{ since these are independent events} \\
P\big(E(T_j)\big) &= p^c, \text{ since all failure probabilities are equal} \qquad (25) \\
\therefore P(E_1) &= \sum_{r=1}^{n} (-1)^{r-1} \binom{n}{r} p^{cr}
\end{aligned}$$

Again, probability of failure of load under minimally redundant configuration can be given by

$$P\big(E(\hat{L}_1)\big) = P\left(\bigcap_{k=1}^{c}\left[\bigcup_{j=1}^{n} E(T_{jk_s})\right]\right) \tag{26}$$

Let us denote the event of failure of a path k by

$$E(C_k) = \bigcup_{j=1}^{n} E\big(T_{jk_s}\big),\ \forall k \in 1, 2, \ldots, c \tag{27}$$

$$\begin{aligned}
P\big(E(C_k)\big) &= \sum_{r=1}^{n} (-1)^{r-1} \binom{n}{r} p^{r} \\
\therefore P\big(E(\hat{L}_1)\big) &= \prod_{k=1}^{c} P(C_k) \text{ since all paths are independent} \\
&= \left(\sum_{r=1}^{n} (-1)^{r-1} \binom{n}{r} p^{r}\right)^{c} \text{ since all paths have same probability}
\end{aligned} \tag{28}$$

Now, we have the failure probability of the sink node in both maximally and minimally redundant. We provide a bound for the ratio $P(E(L_1))/P(E(\hat{L}_1))$ below:

$$\begin{aligned}
P(E(\hat{L_1})) &= \left(\sum_{r=1}^{n}(-1)^{r-1}\binom{n}{r}p^r\right)^c \\
&= \left(1-\sum_{r=0}^{n}(-1)^{r}\binom{n}{r}p^r\right)^c \\
&= (1-(1-p)^n)^c \\
&= p^c\left(1+(1-p)+(1-p)^2+\cdots+(1-p)^{n-1}\right)^c, \\
&\text{using } a^n-b^n=(a-b)(a^{n-1}+a^{n-2}b+\cdots+b^{n-1})
\end{aligned} \tag{29}$$

$$\begin{aligned}
P(E(L_1)) &= \sum_{r=1}^{n}(-1)^{r-1}\binom{n}{r}p^{rc} \\
&= 1-\sum_{r=0}^{n}(-1)^{r}\binom{n}{r}(p^c)^r \\
&= 1-(1-p^c)^n \\
&= p^c\left(1+(1-p^c)+(1-p^c)^2+\cdots+(1-p^c)^{n-1}\right)
\end{aligned} \tag{30}$$

Since $p \in [0,1]$ and c is a positive integer, $p^c \leq p$, which implies that $1-p^c \geq 1-p$. Thus, $P(E(L_1)) \geq p^c\,(1+(1-p)+(1-p)^2+\cdots+(1-p)^{n-1})$.

$$\begin{aligned}
P(E(L_1)) &\geq p^c\left(1+(1-p)+(1-p)^2+\cdots+(1-p)^{n-1}\right) \\
\Rightarrow \frac{P(E(L_1))}{P(E(\hat{L_1}))} &\geq \frac{p^c\,(1+(1-p)+(1-p)^2+\cdots+(1-p)^{n-1})}{p^c\,(1+(1-p)+(1-p)^2+\cdots+(1-p)^{n-1})^c} \\
&= \frac{1}{(1+(1-p)+(1-p)^2+\cdots+(1-p)^{n-1})^{c-1}} \\
&\geq \frac{1}{n^{c-1}},\ \text{since } 1 \geq (1-p) \geq (1-p)^2 \geq \cdots \geq (1-p)^{n-1}
\end{aligned} \tag{31}$$

For the same configuration failure probability of the sink node using approximate reliability algebra is given by $P_{ara}(E(L_1)) = cnp^c \geq cP(E(L_1))$

$$\therefore \frac{P_{ara}(E(L_1))}{P(E(\hat{L_1}))} \geq c\frac{P(E(L_1))}{P(E(\hat{L_1}))} \geq \frac{c}{n^{c-1}} \tag{32}$$

Therefore, we see that Equation 32 proves the Theorem 5.1, where $c = f$ represents the number of independent paths and $n = m$ represents the number of types.

5.3 Correctness and Complexity of ILP-AR

The overall ILP-AR approach is illustrated in Algorithm 3. To implement GENILP-AR, we use the approximate algebra to capture all the reliability requirements. While (12) is a nonlinear expression, a linear encoding of the same constraint can be obtained as follows

$$\sum_{k \in \{1,\ldots,k_{max}\}, j \in \{1,\ldots,n\}} k \cdot x_{ijk} \cdot p_j^k \leq r_i^*, \tag{33}$$

where r_i^* is the required failure probability, x_{ijk} is an auxiliary binary variable equal to 1 if $j \in I_i$ and $h_{ij} = k$, and 0 otherwise, and k_{max} is the maximum possible value for h_{ij} in the given template, i.e. $k_{max} = \max_{1 \leq j \leq n} |\Pi_j|$.

Algorithm 3 *ILP With Approximate Reliability (ILP-AR)*

```
Input: Architecture template $\mathcal{T}$, component variables $\boldsymbol{w}$, costs $\boldsymbol{c}$ and failure probabilities $\boldsymbol{p}$, reliability requirement $r^*$
Output: Adjacency matrix $\mathbf{e}^*$ of the final architecture $\mathcal{G}^*$
  $(Cost, Cons) \leftarrow$ GENILP-AR($\mathcal{T}$, $\boldsymbol{w}$, $\boldsymbol{c}$, $\boldsymbol{p}$, $r^*$)
  $\mathbf{e}^* \leftarrow$ SOLVEILP($Cost$, $Cons$)
  if $\mathbf{e}^* = [\,]$ then return UNFEASIBLE
  return $\mathbf{e}^*$
```

Additional constraints are needed to express the auxiliary variables x_{ijk} in terms of our decision variables. We assume that the reference template $\mathcal{T}$ only includes reduced paths. This is not a restrictive assumption, since multiple instances of adjacent nodes of the same type can be added by refining $\mathcal{T}$ in a second step of the selection process. Then, by lemma 4.1, we link the indicator variables x_{ijk} to the decision variables by adding the following constraints for each type j in $\{1, \ldots, n\}$:

$$\sum_{k=0}^{k_{max}} x_{ijk} \leq 1, \tag{34}$$

and, $\forall k \in \mathbb{N} : 0 \leq k \leq k_{max}$,

$$\sum_{w \in \Pi_j} \left(\eta_{nw,v_i} \wedge \left(\bigvee_{s \in \Pi_1} \eta_{ns,w} \right) \right) = k \rightarrow (x_{ijk} = 1). \tag{35}$$

The constraint in (35) counts the number of components of type j which are connected by at least one path to v_i and to any source in Π_1. The indicator variable x_{ijk} is then set to 1 if the number of such components is k. Constraint (34) enforces that only one of the x_{ijk} variables be set to one. The implication in (35) can be easily converted into a linear constraint using standard techniques [15]. Overall, it can be shown that the number of constraints (and auxiliary variables) generated by the computations in (33)-(35) is $O(|V|^3 n)$, where $n = |\Pi|$. This amounts to a polynomial complexity in the number of nodes and partitions in $\mathcal{T}$, which contrasts with the exponential complexity of the exact computations in Chapter 4 and Chapter 5. Finally, the following result holds for the ILP-AR approach.

Theorem 5.2 (Correctness of ILP-AR). *For a given template $\mathcal{T}$ and within the error bound provided in Theorem 5.1, ILP-AR (Algorithm 3) is sound and complete.*

Informally, the result follows from the fact that, for each type of components, ILP-AR attempts to determine the degree of redundancy needed to meet the reliability requirement. Therefore, if ILP-AR returns `UNFEASIBLE`, assuming that the interconnection constraints are feasible, then we can conclude that $\mathcal{T}$ does not provide enough redundancy to satisfy the reliability constraints. On the other hand, when ILP-AR provides an optimal topology, the solution will satisfy the reliability requirement with an approximation error which is worst case bounded by (13).

5.4 Summary

In this chapter we introduced our ILP-AR algorithm that solves the synthesis problem using an approximate reliability algebra. We show how the ILP-AR algorithm generates polynomial number of constraints in the size of the system as opposed to the exponential number of constraints in case of exhaustive enumeration of failure cases. We also establish rigorous bounds on the approximation, given a template, and show that the approximation always produces failure probability to the correct order of magnitude.

Chapter 6
Case Study and Results

In this chapter we present in detail the requirements and specifications for the design of a typical Aircraft Electric Power System. We also introduce our prototype tool ArchEx, demonstrate the application of both algorithms on the case study and present the results and scalability of the algorithms.

6.1 Aircraft Electric Power System

We apply our algorithms to the selection of optimal architectures for power generation and distribution in a passenger aircraft. Fig. 4 illustrates a sample architecture in the form of a single-line diagram, a simplified notation for three-phase power systems [1]. Typically, aircraft EPS components include power sources, such as the left and right generators (L/R-GEN) and the auxiliary power units (APU) in Fig. 4. The generators power the buses and their loads (not shown in Fig. 4). AC power is converted to DC power by rectifier units (TRU). A bus power control unit monitors the availability of power sources and configures a set of switches, denoted as contactors, such that essential buses remain powered even in the presence of failures.

We aim to generate an EPS architecture that satisfies a set of connectivity, power flow and reliability requirements, while minimizing the total cost. We then model the architecture as a directed graph, where each node represents a component (with the exception of contactors, which are associated with edges) and each edge represents an interconnection. We assume a template $\mathcal{T}$ consisting of the following component types: generators (LG/RG), AC buses (LB/RB), rectifiers (LR/RR), DC buses (LD/RD), loads (LL/RL), two on each side, and one APU. The platform library attributes include generator power ratings $\boldsymbol{g}$, load power requirements $\boldsymbol{l}$, component costs $\boldsymbol{c}$ and failure probabilities $\boldsymbol{p}$, as summarized in Table 1. In our examples, we assume that only generators, buses and rectifiers fail with a probability of 2×10^{-4}.

Connectivity properties can be expressed by using constraints as the ones in (4) and (5). For instance, we can prescribe that any rectifier must be directly connected to only one AC bus, and that all DC buses that are connected to a load or another DC bus must be connected to at least one rectifier to receive power from an AC bus. Power-flow constraints are used to enforce that the total power provided by the generators in each operating condition is greater than or equal to the total power

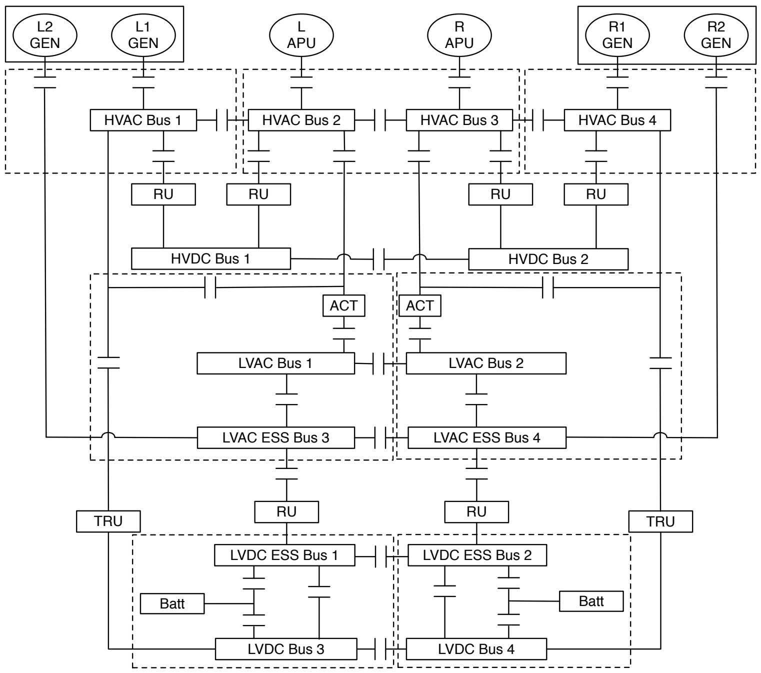


Figure 4: Sample single-line diagram of an aircraft electric power system from [1]. Contactors are represented by double bars.

Table 1: Components and attributes used in the EPS example.

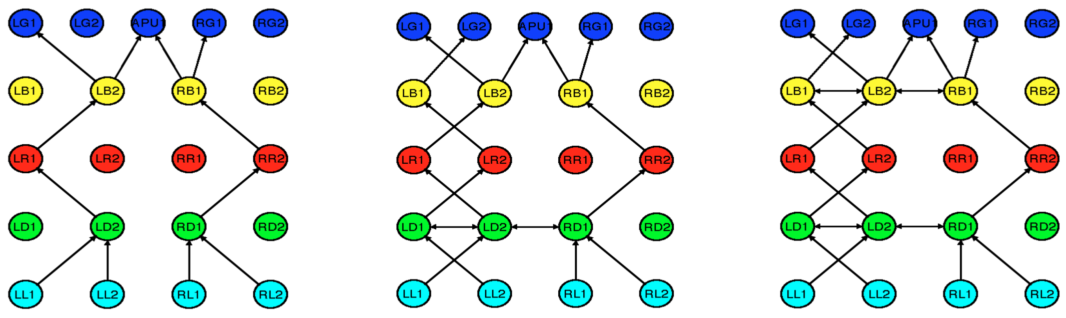
Generators	g (kW)	Loads	l (kW)	Components	c
LG1	70	LL1	30	Generator	g/10
LG2	50	LL2	10	Bus	2000
RG1	80	RL1	10	Rectifier	2000
RG2	30	RL2	20	Contactor	1000
APU	100				

required by the connected loads, by using expressions as in (6). Finally, a reliability constraint prescribes that the probability that a load gets unpowered because of failures should be less than a desired threshold. A functional link will then consist of the set of paths from any generator to the load. Moreover, since our template supports only reduced paths, we use an edge between two nodes of the same type as a shorthand notation to indicate two redundant components: if v_i and v_j, with $v_i \sim v_j$, are connected by an edge, then any direct predecessor of v_i is also a direct predecessor of v_j and vice versa.

6.2 Results

We have developed ArchEx, a prototype framework for system architecture exploration and synthesis, implementing both the ILP-MR and ILP-AR algorithms.

ArchEx leverages Yalmip [17] and Cplex [18] to, respectively, formulate and solve ILP problems. All the numerical experiments were performed on an Intel Core i7 2.8-GHz processor with 8-GB memory.

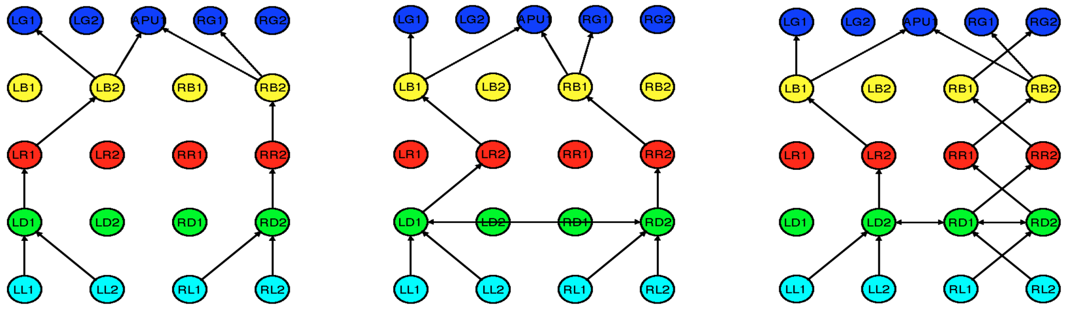


(a) Architecture 1 (b) Architecture 2 (c) Architecture 3

Figure 5: EPS architectures and reliability as obtained at each iteration of an ILP-MR run with $r^* = 2 \times 10^{-10}$: (a) $r = 6 \times 10^{-4}$; (b) $r = 2.8 \times 10^{-10}$; (c) $r = 0.79 \times 10^{-10}$.

ILP-MR Fig. 5 shows the architectures obtained at each iteration of the ILP-MR algorithm for a load failure probability requirement $r^* = 2 \times 10^{-10}$. By solving for just the connectivity and power flow constraints, we obtain the simplest possible architecture (Fig. 5a), which only provides a single path from a load to a generator (or APU), thus showing the highest failure probability. Based on the parameters in Table 1 we obtain $\rho = 8 \times 10^{-4}$, which leads to $k = 2$, as discussed in Sec. 4.2. Therefore, at the second iteration, two additional paths are enforced between each load and a generator, as shown in Fig. 5b. Since the requirement is not yet satisfied, a third iteration is used to fine tune the reliability, by adding one more path between each load and an AC bus. The total computation time to generate the architectures in Fig. 5 was about 38 s.

ILP-AR Three architectures obtained using the ILP-AR algorithm for different load failure probability requirements are instead shown in Fig. 6. The lower the required failure probability, the higher the number of redundant paths and components instantiated from the original template, and the higher the associated cost. For each architecture, the approximate algebra provides an estimation $\tilde{r}$ of the failure probability which is extremely close to the actual value r obtained by exact computations. While the failure probability of the architecture in Fig. 6c exceeds the requirement,



(a) Architecture 1 (b) Architecture 2 (c) Architecture 3

Figure 6: EPS architectures synthesized using ILP-AR for different reliability requirements: (a) $r^* = 2 \times 10^{-3}, \tilde{r} = 6.0 \times 10^{-4}, r = 6 \times 10^{-4}$; (b) $r^* = 2 \times 10^{-6}, \tilde{r} = 2.4 \times 10^{-7}, r = 3.5 \times 10^{-7}$; (c) $r^* = 2 \times 10^{-10}, \tilde{r} = 7.2 \times 10^{-11}, r = 2.8 \times 10^{-10}$.

the error is well within the bound predicted by Theorem 5.1. The execution time of each optimization run in Fig. 6 was also approximately 38 s; however, about 70% of the computation time was used to generate the optimization constraints, which can also be performed off-line for a given template.

6.3 Scalability and Discussion

To test the scalability of both the approaches, we designed EPS architectures with an increasing number of components. In Table 2, we report on the execution time of the ILP-MR approach using Algorithm 2 (at the top) in comparison with the one obtained by a lazier approach, which only adds one additional path at each iteration between the load and a component with a minimal degree of redundancy. The dramatic reduction in time spent for reliability analysis (e.g., more than one day versus 3 min for a 50-node architecture) shows the advantage of using the analysis results to infer the number of required redundant paths, as proposed in Algorithm 2. When this inference is feasible, ILP-MR outperforms ILP-AR (see solver times in Table 3) for architectures with more than 40 nodes. On the other hand, as evident from Table 3, once the optimization problem is generated for a given template, ILP-AR is more competitive for smaller architectures. Yet, problems with several thousands of constraints, and including a realistic number of generators (normally less than 10), can still be formulated and solved in a few hours. We also observe that, because of the sparsity of the EPS adjacency matrix, in this case study, it was possible to reduce the number of generated constraints, which is always smaller than the asymptotic estimation in Sec. 5.

Table 2: Number of iterations, reliability analysis and solver time for different EPS architecture sizes ($r^* = 10^{-11}$, $n = 5$) using ILP-MR with LEARNCONS (top) and with a lazier strategy, adding only one path at each iteration (bottom).

$\lvert V \rvert$ (# Generators)	#Iterations	Analysis time (s)	Solver time (s)
20 (4)	3	34	4.3
30 (6)	3	78	9
40 (8)	3	106	14
50 (10)	3	181	18
20 (4)	4	72	13
30 (6)	7	852	28
40 (8)	10	9118	58
50 (10)	14	39563	114

Table 3: Number of constraints, problem generation (setup) and solver time for different EPS architecture sizes ($r^* = 10^{-11}$, $n = 5$) using ILP-AR.

$\lvert V \rvert$ (# Generators)	# Constraints	Setup time (s)	Solver time (s)
20 (4)	5290	27	11
30 (6)	24514	402	77
40 (8)	74258	3341	494
50 (10)	176794	18902	5059

6.4 Summary

In this chapter we introduced the Aircraft Electric Power System case study and demonstrated the application of both our algorithms. We show the results and also showed that the algorithms scale to the complexity of industrial systems. We infer that ILP-AR turns out to be preferable when we aim to a coarser estimation of the capability (and limitations) of an architecture template and a platform library in terms of reliability. On the other hand, ILP-MR makes it easier to incorporate domain-specific knowledge, since a designer can customize the techniques adopted to improve reliability at each iteration. Moreover, ILP-MR becomes the preferred choice, especially for larger problem instances, when we can estimate the number of redundant paths needed to satisfy the requirement as early as possible, or when we are willing to pay for a longer execution time to incrementally fine tune the reliability of the design.

Chapter 7
Conclusions and Future Work

We have introduced, characterized and implemented two efficient ILP-based algorithms for the optimal selection of cyber-physical system architectures subject to safety and reliability constraints. We have also demonstrated the scalability of the algorithms on a case study of industrial significance, an Aircraft Electric Power System. For the ILP-MR algorithm we provide generic strategies to improve the reliability of the system in a loop with a reliability analysis routine. On the other hand for ILP-AR algorithm we have provided explicit theoretical bounds on the approximation which provides the confidence to use the proposed method for the design of safety critical Cyber Physical Systems.

As a future work, we plan to further investigate the generalization of the presented approaches to support a broader category of systems (e.g. power grids, communication networks) and design concerns (e.g. impact of system dynamics and transients). We also wish to augment the tool with an interface that facilitates requirement specification i.e. supports automated translation of requirements given in textual language (composed out of a library patterns) to integer linear constraints.

References

1. P. Nuzzo, H. Xu, N. Ozay, J. Finn, A. Sangiovanni-Vincentelli, R. Murray, A. Donze, and S. Seshia, "A contract-based methodology for aircraft electric power system design," *IEEE Access*, vol. 2, pp. 1–25, 2014.
2. P. Derler, E. A. Lee, M. Torngren, and S. Tripakis, "Cyber-physical system design contracts," in *ICCPS '13: ACM/IEEE 4th International Conference on Cyber-Physical Systems*, April 2013. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/959.html
3. P. Derler, E. A. Lee, and A. Sangiovanni-Vincentelli, "Modeling cyber-physical systems," *Proceedings of the IEEE (special issue on CPS)*, vol. 100, no. 1, pp. 13 – 28, January 2012. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/843.html
4. B. N., P. Nuzzo, M. Masin, and A. Sangiovanni-Vincentelli., "Optimized selection of reliable and cost-effective cyber-physical system architectures," in *in Proc. Design, Automation and Test in Europe, to appear*, 2015.
5. A. Fisher, C. Jacobson, E. A. Lee, R. Murray, A. Sangiovanni-Vincentelli, and E. Scholte, "Industrial cyber-physical systems - icyphy," in *Proc. Complex Systems Design & Management (CSD&M)*. Springer, December 2013, pp. 21–37, paris, France. [Online]. Available: http://icyphy.org/pubs/34.html
6. E. A. Lee, "Cyber physical systems: Design challenges," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-8, Jan 2008. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-8.html
7. J. Jensen, D. Chang, and E. Lee, "A model-based design methodology for cyber-physical systems," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, July 2011, pp. 1666–1671.
8. C. Talcott, "Cyber-physical systems and events," in *Software-Intensive Systems and New Computing Paradigms*, ser. Lecture Notes in Computer Science, M. Wirsing, J.-P. Bantre, M. Hlzl, and A. Rauschmayer, Eds. Springer Berlin Heidelberg, 2008, vol. 5380, pp. 101–115. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-89437-7_6
9. P. Nuzzo, F. De Bernardinis, and A. Sangiovanni Vincentelli, "Platform-based mixed signal design: Optimizing a high-performance pipelined ADC," *Analog Integr. Circuits Signal Process.*, vol. 49, no. 3, pp. 343–358, 2006.
10. B. Kaiser, P. Liggesmeyer, and O. Mäckel, "A new component concept for fault trees," in *Proc. Australian Workshop on Safety Critical Systems and Software*, 2003.
11. C. Lucet and J.-F. Manouvrier, "Exact methods to compute network reliability," in *Proc. Int. Conf. on Mathematical Methods in Reliability*, 1997.

12. C. Hang, P. Manolios, and V. Papavasileiou, "Synthesizing cyber-physical architectural models with real-time constraints," in *Proc. Int. Conf. Comput.-Aided Verification*, Dec. 2011.
13. C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Satisfiability*, A. Biere, H. van Maaren, and T. Walsh, Eds. IOS Press, 2009, vol. 4, ch. 8.
14. P. Helle, M. Masin, and L. Greenberg, "Approximate reliability algebra for architecture optimization," in *Proc. Int. Conf. on Computer Safety, Reliability, and Security*, 2012, pp. 279–290.
15. W. L. Winston, *Operations Research: Applications and Algorithms, 4th Edition*. Independence, KY: Cengage Learning, 2004.
16. S. Messaoud, "Optimal Architecture Synthesis for Aircraft Electrical Power Systems," Master's thesis, T.U. Munich – U.C. Berkeley, 2013.
17. J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Int. Symp. Computer Aided Control Systems Design*, 2004, pp. 284–289.
18. (2012, Feb.) IBM ILOG CPLEX Optimizer. [Online]. Available: www.ibm.com/software/integration/optimization/cplex-optimizer/