Safe and Reliable Interoperability of Medical Devices using Data-Dependent Controller Synthesis

Franziska Bathelt-Tok*
*Software Engineering for Embedded Systems
Technische Universität Berlin
Email:franziska.bathelt-tok@tu-berlin.de

In the medical sector, there is a high demand for innovative methods to ensure a vendor independent, safe, and reliable communication between heterogeneous medical devices when patients should be provided with the highest possible level of care. Due to the increasing amount of medical devices that should interact, the complexity of such systems grows continuously. To accommodate this trend and ensure a correct functionality, we aim at providing an automated and correct composition of the involved components. The concepts in the field of service composition developed for service-oriented architectures (SOAs) are highly promising to reach such an automation, while ensuring reliability and safety of the system.

The core idea of SOAs is to develop services with sophisticated functionality by composing simpler services appropriately. Services that are independently developed by different providers often cannot communicate with each other directly due to interface incompatibilities. To overcome these incompatibilities an adaption or unification of the interfaces becomes necessary. But, the internal adaption of their interfaces is not realizable due to the provision as closed-source services. To enable their interaction nevertheless, it is necessary to develop a connector that communicates with each service to be composed and, by that, overcomes the arising incompatibilities. This component, called controller, is responsible for the routing and modification of messages as well as for the compliance of behavioral requirements.

As medical devices have similar characteristics, they can be understood as services. Thus, the problem of enabling the interoperability of medical devices can be shifted to the more general problem of service composition.

However, existing approaches dealing with automated service composition, like [3], do not provide a formal data-treatment nor consider data-dependent behavior. Thus, they are insufficient or much manual effort is necessary to enable a safe interoperability of medical devices using service composition.

In our work, we address the problem of providing a controller synthesis process under consideration of a formal data-treatment to enable its application in the medical area. Our approach, allows to

1) express and ensure complex, data-dependent specifications the composed system has to fulfill,
2) detect and consider data-dependent behavior,
3) synthesize a controller automatically,
4) ensure correctness-by-construction w.r.t. data-dependent, functional and safety-critical requirements

As base for our approach, we assume that a formal representation as algebraic Petri net (APN) [5] of each service is known. This is not a restrictive assumption, because ongoing research in the field of data mining focuses on the extraction of formal models out of textual descriptions. As medical devices have to pass a certification process, they are always available. Furthermore we assume that the interface matching as well as the set of requirements are expressed by a subset of the computation tree logic (CTL). These requirements specify the interaction between the single devices and define the behavior of the composed system, respectively. Our main target is to synthesize a correct controller, so that the composed system fulfills all given requirements. For this, we have developed a three-step approach, which is shown in Figure 1.

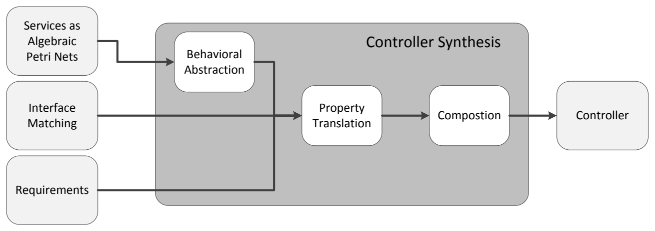

Fig. 1. Basic Idea of the Controller Synthesis Approach [1]

Based on the inputs, we firstly determine restrictions of the data domains during the *behavioral abstraction* step. In this step, the behavior of each service is reduced to the observable behavior at the interface ports. The idea is shown in Figure 2.

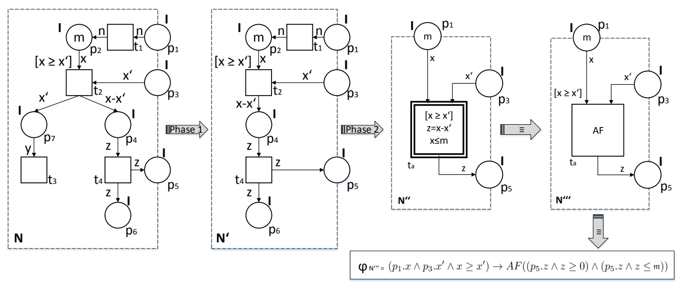
m
n
t1
p1
p2
x
[x ≥ x′]
x′
t2
p3
x-x′
p7
p4
y
z
t3
t4
p5
p6
N
Phase 1
N′
Phase 2
ta
[x ≥ x′] z=x-x′ x≤m
N″
≡
AF
N‴
$\varphi_{N'''} = (p_1.x \land p_3.x' \land x \geq x') \rightarrow AF((p_5.z \land z \geq 0) \land (p_5.z \land z \leq m))$

Fig. 2. Basic Idea of the Behavioral Abstraction

For each net the reduction process is done in two phases: Starting from a net representing the behavior of a service, all parts of the net that are not involved in the communication that is realized by interface places (p_1, p_3, p_5) are omitted. As result of this elimination step, we get a reduced net (N'), which is the second net in Figure 2. Based on this net, the rest of the behavior is automatically analyzed to determine restrictions of the data domains at the (output) interface ports. For this, we summarize all internal states and transitions into one single hierarchical transition [4]. Data-dependencies occurring along this abstraction step must be extracted and stored. As result we get a CTL-formula that must be conjunct with the CTL-formulae which are representing the interface matching and the requirements. For this purpose, we have defined and proven transformation rules that describe the mapping of atomic and more complex elements of the formulae to corresponding elements of APNs. To enable this mapping, in the *property translation* step, the second step of our approach, we have extended the syntax and semantics of APNs by path operators the CTL-formulae comprise. In Figure 3 the transformation of a formula into an extended version of APN is exemplified.

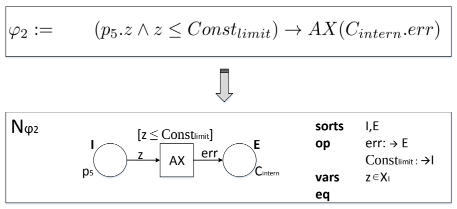

Fig. 3. Basic Idea of the Transformation from CTL-formulae into APNs

Atomic elements of the formulae are transformed into elements of APNs, e.g. $p_5.z$ is transformed into a place with label p_5 and an arc with inscription z. Propositions that comprise relational operators on the left-hand side of the implication, e.g. $z \leq Const_{limit}$, are translated into guards. Variables, like z, and constants, like err and $Const_{limit}$, become part of the algebraic specification as variables (**vars**) and operations (**op**), respectively. Additionally, CTL-specific operators, e.g. AX, are represented as label for the transition, which exemplifies the extension of APN. In this way, the set of CTL-formulae is transformed into a set of extended APNs.

Afterwards, in the *composition* step, we compose these nets to an overall Petri net in which all properties are fulfilled. This is due to the fact that the composition of the extended APNs is equivalent to the conjunction of the CTL-formulae which are represented by the corresponding nets. Figure 4 visualizes the functionality of our composition algorithm.

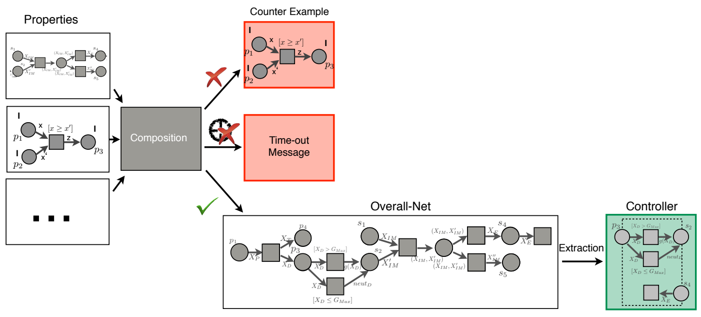

Fig. 4. Functionality of the Composition

Nevertheless, the composition is not always successful. This can have two reasons: First, it is possible that formulae are contradictory. In this case, the composition algorithm discloses a counterexample to the user to clarify which requirement is not compatible with the others. The user can decide whether this requirement is negligible and whether the controller synthesis process should be continued without this requirement. Second, there might be an infinite run time of the composition process. This is caused by undecidable problems that can occur during the analysis of the reachability graphs of the nets to be composed. To avoid this, we have defined a time interval depending on the complexity of the system. If the composition process does not terminate within this time limit, a timeout notification is sent to the user, who can change the specification and give it another try. If the composition succeeds, we get an overall net that comprises all properties, i.e., all extended APNs that represent the corresponding CTL-formulae.

In the last step of our synthesis process, the requested controller can be extracted from the resulting overall net, which includes the behavior of all services. As the purpose of the controller is to ensure the correct communication between the services, we have to extract all parts of the overall net that are responsible for that. For this, we use the interface matching to mark the interface places and separate the net structure between these places. The extended APN that comprises the separated net structures is the controller. After extracting the controller, we substitute the extended transitions, i.e., that are labeled with CTL-specific operators, by transitions an original APN consists of. The main advantage of reverting the extension is the usability of existing tools developed for original APN. The resulting controller, which is represented by an (original) APN, will be combined with the APNs that represent the service behavior and that have been used as input for our approach. Because it is correct-by-construction, it ensures that the composed system, consisting of the services and the synthesized controller, fulfills all requirements that have been specified by the customer.

As each of these three steps is fully automatic and proven as correct, we get an automated synthesis of service controllers that are correct w.r.t. data-dependent, functional and safety-critical requirements. Currently, we are evaluating this approach using a case study which has been provided by Dr. Oliver Blankenstein (Endocrinology, Charité Berlin). This case study deals with the development of an artificial pancreas, where a glucose sensor and an insulin pump have to interact. A sketch of this can be found in [2]. In future work, we aim to introduce a priority of the requirements so that a controller can be synthesized that ensures at least the most important properties. This means, properties that are absolutely necessary are combined first. Thus, a first draft of the controller is provided which can be extended by less important properties. This may reduce the risk of a time-out.

References

[1] F. Bathelt-Tok and S. Glesner. Towards the automated synthesis of data dependent service controllers. In *Service-Oriented Computing ICSOC 2013 Workshops*, Lecture Notes in Computer Science. Springer, 2014.

[2] F. Bathelt-Tok, S. Glesner, and O. Blankenstein. Data-dependent controller synthesis to enable reliable and safe interoperability of medical devices. PervasiveHealth '14, 2014.

[3] C. Gierds, A. J. Mooij, and K. Wolf. Reducing adapter synthesis to controller synthesis. *IEEE T. Services Computing*, 5(1):72–85, 2012.

[4] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume Volume 1. Springer-Verlag, 1997.

[5] W. Reisig. Petri nets and algebraic specifications. *Theoretical Computer Science*, 80:1–34, 1991.