Safe Incremental Design of UML Architectures

Anne-Lise Courbis, Thomas Lambolais, Thanh-Hung Nguyen

Safe Incremental Design of UML Architectures

Anne-Lise Courbis[1], Thomas Lambolais[1], and Thanh-Hung Nguyen[2]

[1]LGI2P, école des mines d'Alès, Nîmes, France. firstName.lastName@mines-ales.fr
[2]Hanoi University of Science and Technology, Hanoi, Vietnam. hungnt@soict.hust.edu.vn

Abstract

IDF is an Incremental Development Framework which supports the development and the verification of UML models for concurrent and reactive systems. IDF offers refinement and extension techniques the goal of which is to check the preservation of liveness properties during the model developments. Here, we improve the framework in order to analyze models from a safety point of view. Some techniques such as refinement maintain safety properties, however, extension does not maintain such properties. For this purpose, we associate IDF with the experienced tools of safety analysis based on the BIP language by translating UML models into BIP. We demonstrate on a basic example the complementarity of liveness and safety analyses.

Keywords*: UML composite components, BIP architectures, conformance analysis, safety analysis, refinement, incremental development, transformation of UML models into BIP models.*

1 Introduction

Designing UML models of software intensive reactive systems is recognized to be a tricky and crucial task. Reactivity means that such systems must continuously react to their environment, at a speed defined by this environment. It implies *liveness properties*, stating that the system will eventually react as it must. These systems are also dependable, so that reliability, availability and robustness are of primary importance. This implies *safety properties*, stating that undesired behaviors of the system will never happen.

Despite the increasing number of conferences and researches about UML, the use of UML is not yet fully accepted by industrialists [27]. The lack of a precise semantics and the numerous ambiguities of UML could explain this defeat. Another related main explanation is the lack of support for UML designers in the process for both setting up models and evaluating them. The rule of the thumb is that an architecture assessment should be held as soon as architecture decisions begin to be made and the cost of reversing those decisions would be more than conducing software architecture evaluation [2]. However, few methods of UML model development support model evaluation during the design phases. The novelty of our proposed approach is to consider at early steps of the design both liveness and safety properties.

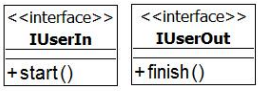
In a previous work [19, 18], we have presented our Incremental Development Framework (IDF) and its associated tool IDCM (Incremental Development of Conformance Models). IDF supports the development and the evaluation of UML models for reactive systems. It deals with UML primitive components whose behavior is represented by a state machine and UML composite components whose parts are primitive components or composite ones. IDF allows models to be developed step by step. At every step of the design, the model is verified as being consistent with the model obtained at the previous step, a step being a model evolution which can be of four kinds: *extension*, *refinement*, *increment* or *substitution*. The verification of these four kinds of model evolution is based on a conformance relation [22] which verifies that liveness properties defined by a reference model are preserved. However, this work has its own shortcomings: explicit verification of safety properties is not addressed. This article aims at enhancing IDF in order to be able to model and check explicit safety properties. That is a way to cover the verification of the complete properties of a system since it has been proved that the safety/liveness spectrum covers all linear-time properties of a system under design [29].

We present in section 2 an example pointing out an incremental development of a model whose liveness analysis is demonstrated using IDCM, but suffering from a lack of safety analysis. Section 3 presents the BIP meta-model that

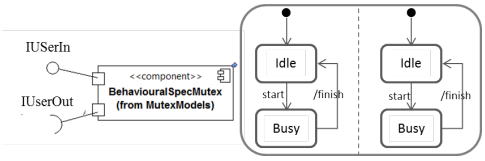
has been selected as an intermediate format to analyze models in order to use D-Finder tool to check safety analysis. The tool IDCM associated with IDF, as well as UML and BIP models presented in this article, may be downloaded on the website [1]. Section 5 presents related works. We conclude in section 6 and present our future directions.

2 Motivating example

Let us consider MUTEX, a mutual exclusion system that can process two orders of task execution in parallel: it has two ports (see Fig.1), each of them allowing the reception of a demand of task execution (operation *start* of interface *IUserIn*) and the transmission of an acknowledgement at the end (operation *finish* of interface *IUserOut*). The high level specification of MUTEX represents the system from an external point of view and does not yet represent the resource. It is modeled from a behavioral point of view by an atomic UML component whose behavior is specified by a state machine (see Fig.1b).



(a) *Interfaces.*

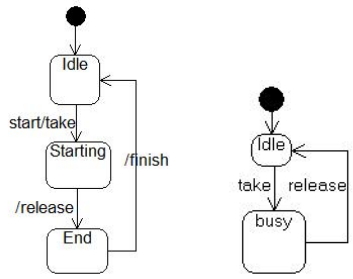


(b) *MUTEX Atomic component.*

Figure 1: MUTEX Behavioral specification.

The next step of modeling aims at representing the internal view of the system, i.e. the shared resource and the two users. The interface provided by the resource offers two actions: *take* and *release*. The designer thus specifies the components *SpecUser* and *SpecResource* and their associated behaviors (cf. Fig. 2) in order to match MUTEX high level specification. The first model set up by this way is named *MUTEXSpec*: it is a composite component which assemblies two *SpecUser* components and one *SpecResource* component (see Fig.3a).

A new modeling step is starting to set up a possible implementation of *MUTEXSpec* architecture (see Fig.3b) based on components *User* and *Resource*. *User* is not detailed since it is not required to understand the example. *Resource* is presented in Fig. 5a. Both *User* and *Resource* have been checked by IDCM as extending and thus conforming their specification (Fig.3c). The conformance relation [20] between an implementation model and a specification model guarantees that actions that are mandatory after any trace of the specification must also be accepted by the implementation after the same trace. This relation has been implemented with its variant, refines and extends, in [22]. It requires models to be transformed into LTS (Labelled Transition System) that is automatically achieved using IDCM and CADP toolbox [15] (see [19, 18] for more details about the transformation). In the following, we give an interpretation of these relations on the MUTEX models. For formal definitions, see [18]. *MUTEX* composite component conforms to *MUTEXSpec* component: it is a suitable implementation of the specification. *MUTEXSpec* architecture is a refinement of *MUTEX* behavioral specification: it has no extra traces and realizes all mandatory part of the specification. *Resource* component conforms to *ResourceSpec* and is an extension: *Resource* is able to behave like its specification but has also extra behavior which does not conflict with its specification.



(b) (a)

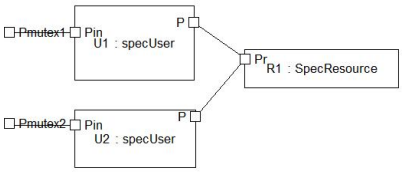
Figure 2: (a) SpecUser and (b) SpecResource state machines

This example points out that *MUTEX* is a "good" realization of the initial specification from a liveness point of view as defined by the ISO standard [16]. However, we are not able to verify the safety property stating that the resource has to be exclusive. We may interpret this property in terms of states that the components of the system may or may not reach. If the resource *R1* is exclusive, it means that *U1* and *U2*, the two users of MUTEX may not be at the same time in state *Starting*, which is the output state of the transition *start/takeResource*, while *R1* is in state *Busy*.

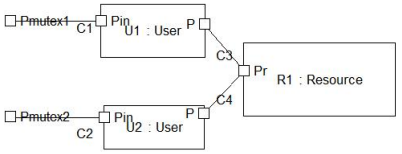
IDF relations cannot verify an explicit property such as this one. It is thus necessary to associate another tool with IDCM.

3 Safety analysis of architectures

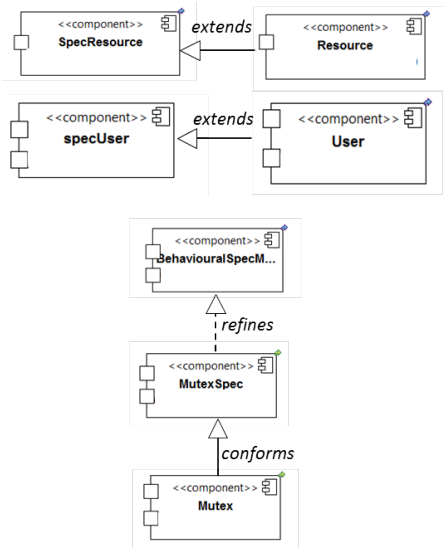
We are interested in explicitly modeling safety properties in order to complement incremental techniques. We are looking for a method which takes into account the incremental aspect of modeling in order to be integrated into



(a) *MUTEXSpec composite component*



(b) *MUTEX composite component.*



(c) *Liveness relations between MUTEX components.*

Figure 3: Incremental development of the MUTEX system.

IDF. The BIP (Behavior, Interaction and Priority) modeling and verification framework is based on several principles which match with IDF: development of correctness-by-construction model, incrementality, compositionality and composability [5]. It includes many tools for safety property analysis and model transformation from several languages such as AADL and Lustre. To the best of our knowledge, there is no available transformation tool from UML to BIP taking into account both primitive and composite component descriptions. We have thus developed a module of IDCM that may be uploaded on the IDCM website [1]. The transformation principle is presented and illustrated in section 4.

3.1 BIP language

BIP is a powerful language for modeling heterogeneous real-time systems. We only focus on specific concepts useful to understand the safety analysis purpose and the transformation from UML to BIP. As we do not consider data analysis of UML models, we will not consider data modeling in BIP. In the same way, we will not deal with priority concepts of the BIP language. Refer to [4] for detailed information about the BIP language. Fig. 4 gives an overview of the main classes of BIP meta-model corresponding to UML concepts we deal with.

BIP describes compound components by a set of interactions between atomic components whose behaviors are represented by LTS. Two kinds of interactions are possible: strong (also called rendezvous) and weak (broadcast). In this study, we restrict interactions to rendezvous which are defined on ports representing actions.

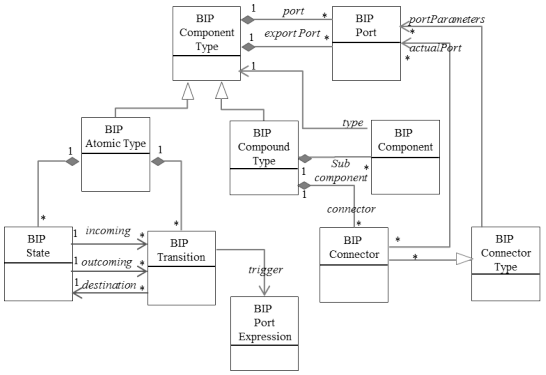


Figure 4: *Main BIP classes.*

3.2 D-Finder: a toolbox for safety analysis

D-FINDER provides methods and tools to compute invariants of BIP models. Such invariants are interesting since they preserve safety properties. There are two kinds of invariants: component invariants which are over approximation of reachable states, and interaction invariants which define global boolean constraints dealing with the synchronization of components. D-FINDER is based on an abstraction technique allowing the state space to be reduced. Its strength is to perform incremental constructions of models and incremental computations of invariants [5] allowing large-scale systems to be checked. D-Finder uses BDD library for the symbolic computation of interaction invariants, and then the SAT-solver tool Yices [13] for checking satisfiability. Verifying a safety property consists in demonstrating using Yices that the negation of the property is unsatisfiable in a context defined by the set of invariant expressions generated by D-Finder. The invariants are expressed by Boolean Behavioral Constraints [23].

3.3 Illustration of a safety property for MUTEX

Let us consider the MUTEX specification and implementation architectures (Fig. 3a and 3b). We aim at checking if the mutual exclusion property is satisfied. Equation 1 expresses the non expected property: two users U1 and U2 can be both in *Starting* state while the shared resource R1 is in *busy* state. We have to demonstrate that this property can never be satisfied.

$$(and\ R1\_busy\text{-}\ (and\ U1\_Starting\text{-}\ U2\_Starting\text{-})) \quad (1)$$

The following section focuses on UML model transformation into BIB and points out results of safefy analysis of MUTEX models after their transformation.

4 From UML to BIP architectures

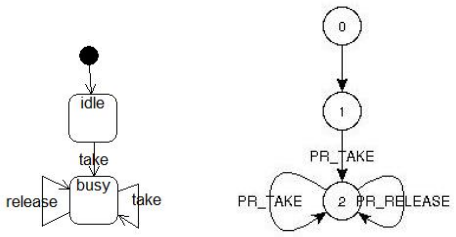
UML and BIP share the same view about component concepts: a UML component whose behavior is defined by a state machine matches the BIP atomic component concept, a UML composite component matches the BIP compound component (see Table 1). The concept of UML interface (set of visible operations) does not explicitly exist in BIP. However, it may be translated into exported ports. A matching is possible between operations belonging to interfaces associated with UML ports and the BIP ports. For instance, the port *P* of component *User* which is associated with the interface *IResource* will correspond to two BIP ports named *P_TAKE* and *P_RELEASE* since interface *IResource* contains the two operations *TAKE* and *RELEASE*. This is why the relation between UML port and BIP ports is of cardinality n in Table 1.

UML Concept	BIP Concept	card.
Atomic Component	AtomicType	1
Composite Component	CompoundType	1
Port	Port	n
Interface	–	0

Table 1: Correspondence between UML and BIP concepts

4.1 Transformation of atomic components

We have given in [18] a LTS semantics to UML state machines. BIP atomic models have a LTS semantics [4]. The mapping from UML to BIP is thus trivial. It is important to note that UML is more concrete than the LTS language and allows more complex structures to be modeled, such as actions on data or call events through ports. Hence, this semantics abstracts some concepts of UML like data, events, time, ports, guards, thanks to the intrinsic non-deterministic nature of LTS. The rules allowing the UML atomic transformation into LTS are detailed in [18]. Figure 5a represents the state machine of component Resource of MUTEX architecture and figure 6 gives the corresponding BIP model that is automatically generated by the module UMLtoBIP of IDCM.



(a) *Resource state machine.* (b) *Resource LTS.*

Figure 5: Resource component transformation into LTS.

```
package Resource
atomic type Resource
  export port Port PR_TAKE
  export port Port PR_RELEASE
  port Port i
  place Pseudostate1,idle,busy
  initial to Pseudostate1
  on i from Pseudostate1 to idle
  on PR_TAKE from idle to busy
  on PR_TAKE from busy to busy
  on PR_RELEASE from busy to busy
end
end
```

Figure 6: Resource BIP model.

4.2 Transformation of composite components

There is a direct mapping between UML composite component and BIP compound components (see Table 2): a UML composite component consists of a set of *Parts* which match BIP *Components*. A UML assembly *Connector* matches a set of *BIP connectors*. Indeed, a BIP connector is relative to the synchronization of a single operation shared by two interconnected ports, while a UML connector is relative to the synchronization of the set of operations belonging to interfaces associated with the interconnected ports. The *Port* of a *Part* belonging to a delegate connector will be exported and renamed by the name of the port of the compound component. Table 2 summarizes the matching of concepts between UML composite components and BIP compound components. To illustrate this transformation, we give in Figure 7 the BIP code of the MUTEX architecture represented in Figure 3b. This code is automatically generated by the transformation module UMLtoBIP of IDCM.

4.3 Safety analysis of MUTEX architecture

Despite MUTEX architecture conforms to MUTEXSpec, we have to verify the safety property about

UML Concept	BIP Concept	card.
Part	Component	1
Assembly connector	Connector BIP	n
Delegate connector	export Port	n

Table 2: Correspondence between concepts of UML composite components and BIP compound components

```
model Mutex
include User.bip
include Resource.bip
connector type rendezvous2(Port p1, Port p2)
        define [ p1 p2 ]
end
compound type MutexType
  component  Resource R1
  component  User U2
  component  User U2
  connector rendezvous2
    C1_release(U1.P_RELEASE, R1.PR_RELEASE)
  connector rendezvous2
    C1_take(U1.P_TAKE, R1.PR_TAKE)
  connector rendezvous2
    C2_release(U2.P_RELEASE, R1.PR_RELEASE)
  connector rendezvous2
    C2_take(U2.P_TAKE, R1.PR_TAKE)
  export port Port PMUTEX1_FINISH is U1.PIN_FINISH
  export port Port PMUTEX1_START is U1.PIN_START
  export port Port PMUTEX2_FINISH is U2.PIN_FINISH
  export port Port PMUTEX2_START is U2.PIN_START
end
component  MutexType Mutex
end
```

Figure 7: Mutex BIP model.

the mutual exclusion whose negation is expressed in equation 1 of section 3.3. For SpecMUTEX, the property is unsatisfied: it means that the resource mutual exclusion has been properly implemented in this architecture. That is not the case for the MUTEX architecture. Indeed, the state machine associated with the Resource (see Fig.5a) points out that it may be used concurrently by two users. On this example, the error is obvious, but it is not the case for large system for which components may be designed by third parties according to a high level specifications: conformance does not guarantee safety preservation and a complete specification has to define both the expected behavior (liveness properties) and unexpected states or sequence of events (safety properties). IDCM associated with BIP and its associated tools is a good way to cover all these aspects.

5 Discussion and Related work

Most of behavioral analyzes of UML models (75% according to [21]) are done by transformation of UML models into formal languages that can be handled by model checkers or theorem provers. Some of them are referenced below. However, to the best of our knowledge, no framework support the *incremental* development of UML architecture models by analyzing both *liveness and safety* behavioral aspects. In particular, no framework is able to consider abstract and non-deterministic UML models and few frameworks are able to consider UML models partially covering the requirements. Hence, even if some work addresses refinement of models, they do not focus on the reduction of non-determinism, and most of them cannot analyze models the specification of which is extended, despite it is a key action for designing complex systems and managing model evolution. Refer to [18] to have more arguments and a complete state of the art about formal verification of models.

To argue this discussion, let us take some representative works about UML/SysML architecture model verification [17, 26, 10, 24, 3]. Although [12] focuses on UML activity diagrams, we must pay attention to the sound discussion about model checking that is the base of our approach. [17] defines a UML profile to transform models into Wright for using the FDR model checker [14]. FDR focuses on liveness and safety. It deals with several refinements and provides deadlock detection. However, it does not support analysis of extended models and models are not verified under fairness assumption, which is the main drawback as pointed out in [18]: FDR can not distinguish between 'critical' livelocks (when there may exist executions where the system won't never exit some infinite internal paths) and 'false' livelocks, where under a fairness assumption, the system may leave internal infinite paths or loops. [10] defines and verifies component assemblies and performs a behavioral compatibility verification but extension and refinement techniques are not supported. [24] have extended the techniques proposed by [11] who has defined the OMEGA 2, a UML profile. Architectures are translated into IF/IFx models [8] allowing LTS models to be generated and analyzed by the CADP model checker [15]. [3] uses a notation integrating concepts of both UML and MARTE [25] and transforms models into temporal logic descriptions that can be analyzed by the model checker Zot [28]. Other approaches about AADL aims at transforming models into intermediate models such as FIACRE [9, 6] or BIP [9] to use appropriate model checkers such as TINA [7] or Yices [13]. These approaches are powerful from the safety point of view but they are not able to integrate liveness analysis for incremental development of models as it is done in IDCM.

6 Conclusion

In this article, we have pointed out the interest for incremental development of UML models and the complementarity between safety and liveness analyses. We have defined a transformation of UML models into the BIP formalism which is implemented into the tool IDCM we have developed. By this way, the liveness and safety analyses are automated. Safety properties are expressed by proposi-

tional calculus and requires designers to manipulate a specific syntax. Further steps consist in developing a support to help designers to express the safety properties in terms of UML concepts regardless the theorem prover syntax, and studying their automatic rewording when UML models are refined or extended.

Acknowledgement: This research was supported by The National Foundation for Science and Technology Development (NAFOSTED) under Grant 102.03-2013.39: Automated verification and error localization methods for component-based software.

References

[1] IDCM. `http://idcm.wp.mines-telecom.fr`. Accessed: 2017-01-30.

[2] M. A. Babar, L. Zhu, and R. Jeffery. A framework for classifying and comparing software architecture evaluation methods. In *ICSE*, pages 309 – 318, 2004.

[3] L. Baresi, G. Blohm, D. S. Kolovos, N. Matragkas, A. Motta, R. F. Paige, A. Radjenovic, and M. Rossi. Formal verification and validation of embedded systems: the UML-based MADES approach. *Software & Systems Modeling*, 14(1):343–363, 2015.

[4] A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time components in BIP. In *SEFM*, pages 3–12, 2006.

[5] S. Bensalem, M. Bozga, A. Legay, T.-H. Nguyen, J. Sifakis, and R. Yan. Incremental component-based construction and verification using invariants. In *FMCAD*, pages 257–256, 2010.

[6] B. Berthomieu and J.-P. Bodeveix. Formal Verification of AADL models with Fiacre and Tina. In *ERTS*, 2010.

[7] B. Berthomieu, P.-O. Ribet, and F. Vernadat. The tool TINA: Construction of abstract state spaces for Petri nets and time Petri nets. *International Journal of Production Research*, (14):2741–2756, 2004.

[8] M. Bozga, S. Graf, I. Ober, I. Ober, and J. Sifakis. The IF Toolset. In *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *LNCS*, pages 237–267. Springer Berlin Heidelberg, 2004.

[9] M. Y. Chkouri and M. Bozga. Prototyping of distributed embedded systems using AADL. *ACESMB*, pages 65–79, 2009.

[10] S. Chouali and A. Hammad. Formal verification of components assembly based on SysML and interface automata. *Innovations in Systems and Software Engineering*, 7(4):265–274, Oct. 2011.

[11] A. Cuccuru. Meaningful composite structures. In K. Czarnecki, I. Ober, J.-M. Bruel, A. Uhl, and M. Völter, editors, *MODELS*, volume 5301 of *LNCS*, pages 828–842. Springer Berlin Heidelberg, 2008.

[12] Z. Daw, J. Mangino, and R. Cleaveland. UML-VT: A Formal Verification Environment for UML Activity Diagrams. In *Demo.session of MoDELS*, pages 48–51, 2015.

[13] B. Dutertre. Yices 2.2. In A. Biere and R. Bloem, editors, *CAV*, volume 8559 of *LNCS*, pages 737–744. Springer, July 2014.

[14] Formal-Systems and Oxford-University-Computing-Laboratory. Failures-Divergence Refinement (FDR2 User Manual). Technical Report October, Formal System (Europe) Ltd, 2010.

[15] H. Garavel, F. Lang, R. Mateescu, and W. Serwe. CADP 2011: a toolbox for the construction and analysis of distributed processes. *International Journal on Software Tools for Technology Transfer*, 15(2):89–107, 2013.

[16] ISO/IEC9646. Information technology – open systems interconnection – conformance testing methodology and framework – part 1: General concepts, 1991.

[17] M. Kmimech, M. T. Bhiri, and P. Aniorte. Checking component assembly in ACME: an approach applied on UML 2.0 components model. In *ICSEA*, pages 494–499. IEEE, 2009.

[18] T. Lambolais, A.-L. Courbis, H.-V. Luong, and C. Percebois. IDF: A framework for the incremental development and conformance verification of UML active primitive components. *Journal of Systems and Software*, 113:275–295, 2016.

[19] T. Lambolais, A.-L. Courbis, H.-V. Luong, and T.-L. Phan. Designing and integrating complex systems: Be agile through liveness verification and abstraction. In *CSDM*, pages 69–81. Springer, 2015.

[20] G. Leduc. A framework based on implementation relations for implementing LOTOS specifications. In *Computer Networks and ISDN Systems*, volume 25, pages 23–41, 1992.

[21] F. J. Lucas, F. Molina, and A. Toval. A systematic review of UML model consistency management. *Information and Software Technology*, 51(12):1631–1645, 2009.

[22] H.-V. Luong, T. Lambolais, and A.-L. Courbis. Implementation of the Conformance Relation for Incremental Development of Behavioural Models. In K. Czarnecki, I. Ober, J.-M. Bruel, A. Uhl, and M. Völter, editors, *MoDELS*, volume 5301 of *LNCS*, pages 356–370. Springer Berlin, 2008.

[23] T.-H. Nguyen. *Constructive verification for component-based systems*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2010.

[24] I. Ober and I. Dragomir. Unambiguous UML composite structures: the OMEGA2 experience. In *SOFSEM*, pages 418–430. Springer, 2011.

[25] OMG formal/2011-06-02. UML profile for MARTE - Modeling and Analysis of Real-time Embedded Systems, 2011.

[26] J.-F. Pétin, D. Evrot, G. Morel, and P. Lamy. Combining SysML and formal models for safety requirements verification. In J. Hall, H. Kaindl, L. Lavazza, G. Buchgeher, and O. Takaki, editors, *ICSEA*, Paris, France, 2010.

[27] M. Petre. UML in practice. In *ICSE*, pages 722–731. IEEE Press, 2013.

[28] M. Pradella, A. Morzenti, and P. S. Pietro. Bounded satisfiability checking of metric temporal logic specifications. *ACM Transactions on Software Engineering and Methodology*, 22(3):20, 2013.

[29] F. B. Schneider. Decomposing Properties into Safety and Liveness using Predicate Logic. Technical report, Cornell Univ. Ithaca, NY, Dept. of Computer Science, 1987.