Project Description — Follow-Up Project Proposal

Gerald Lüttgen, Professor (W3/permanent), University of Bamberg
Walter Vogler, Professor (C3/permanent), University of Augsburg

Foundations of Heterogeneous Specifications Using
State Machines and Temporal Logic

1 State of the Art and Preliminary Work

This is a follow-up grant proposal (renewal) to DFG project LU 1748/3-1 / VO 615/12-1 of the same title, of which approx. 75% (full-time equivalent) has been completed to date and whose funds are expected to be depleted at the end of March 2017 (Augsburg) and November 2016 (Bamberg), resp. The granted current support pays for one PhD student (Research Assistant) each at Augsburg (Dipl.-Inform. Ference Bujtor) and Bamberg (Dipl.-Math. Sascha Fendrich). This section reports on what we have achieved so far and reviews some new related work that is relevant for the follow-up research proposed here.

According to the original grant proposal, the current "project's aim [has been] to significantly enhance the theoretical foundations and practical utility of those heterogeneous specification formalisms for reactive systems that mix operational and logic styles of specification. Our particular interest is in formalisms that add logic elements to state machine notations, such as Caillaud et al.'s *Modal Interfaces* (MI) [71], which emphasise logic conjunction for composing specifications, and Paige et al.'s *Contractual State Machines* (CSM) [45], which extend a simple hierarchical state machine notation by temporal-logic operators."

Because the original proposal was devised for 36 months but has only been funded for 24 months, we had to narrow its scope. We decided to leave out the theory and tool support for *Contractual Statecharts* (CSC), which eliminated work packages BA5 ("Defining a formal semantics for CSM/CSC"), BA6 ("Adapting the existing CSM toolset to CSC"), BA7 ("Developing refinement checkers for CSC") and BA8 ("Evaluating the CSC framework via case studies"), and shifted the analogous work packages A6–A8 for modal interface theories from Augsburg to Bamberg.

Given this narrowing of the project aim to interface theories, the research project has fully delivered so far. While the implementation of tool support and its evaluation are ongoing, the project's outcome to date has already significantly advanced the theoretical foundations of and the degree of heterogeneity within interface specifications, as described below. This is evidenced by 11 peer-reviewed publications in the proceedings of international conferences (7 papers) and in leading international journals (4 articles), while 2 further journal articles have been submitted and are currently under review (cf. Sec. 1.1). In addition, 3 Bachelor's theses [50, 74, 76] and 2 Master's theses [40, 42] have been successfully completed at the Universities of Augsburg and Bamberg on topics closely related to this research project.

Interface Theories, Interface Automata & Modal Transition Systems. *Interface theories* support the component-based design of concurrent systems and offer a semantic framework for, e.g., software contracts [3], web services [8] and the Internet of Things [56]. Technically, interface theories are concurrency-theoretic formalisms that allow one to specify component interactions by means of nondeterministic transition systems (LTSs) labelled with input and output actions.

These LTS may be combined via parallel composition, conjunction and quotient operators. They are equipped with compositional refinement preorders, which allow one to relate component interfaces to each other as well as to relate interfaces to component implementations.

The focus of our investigations have been interface theories combining de Alfaro and Henzinger's *Interface Automata* (IA) [30] and Larsen's *Modal Transition Systems* (MTS) [51]. This has resulted, as we argue below, in the most elegant and general such interface theory to date – *Modal Interface Automata* (MIA) – and has been found very problematic in the past [5, 52, 71], due to a conflict between unspecified inputs being forbidden in MTS but allowed in IA. To be able to describe our achievements we first provide some background on the IA and MTS settings.

The distinguishing feature of IA is a parallel composition operator on system components, where receiving an unexpected input is regarded as an error, i.e., as a communication mismatch. In so-called *pessimistic* interface theories [5], a parallel composition of components is not defined, if such a mismatch occurs. In *optimistic* theories [18, 19, 52, 71], such as the ones considered in this research project, a communication mismatch is acceptable as long as the system environment can prevent it from being reached; in this case, the components are called *compatible*. Technically, all those states of the parallel composition are pruned from which entering an error state cannot be prevented by the environment. The refinement relation on IA is an *alternating simulation* not unlike bisimulation [64], where a refined interface simulates a specifications' inputs, while the specification simulates the interface's outputs.

IA suffers from the fact that outputs cannot be required, because any interface may be implemented by a component that accepts all inputs and does not offer any output, thereby avoiding errors altogether. This is undesired in practice and has led researchers to combine IA and MTS; the latter features may- and must-transitions to express allowed and required behaviour, resp. According to *modal refinement*, all required behaviour must be preserved and no disallowed behaviour may be added when refining an interface. Consequently, whereas outputs are optional in IA, they may now be enforced in theories combining IA and MTS, such as our *Modal Interface Automata* (MIA). In order to express conjunction we combined – already in the conference version of [60] – IA with dMTS. The latter is a restricted form of Disjunctive MTS [54], where a disjunctive must-transition has several target states but only a single action.

Results of work packages. We now provide a summary of the results of the work packages of the original proposal, excluding the rather obvious work package WP 1 devoted to the orientation of the two RAs at the start of the project and the work packages deleted due to the narrowing of scope.

Work packages WP 2–5 deal with "Studying compatibility & consistency notions" (WP 2), "Developing fully-abstract semantic theories" (WP 3), "Comparing our theories to related work" (WP 4) and "Extending MI by contracts" (WP 5). Since the revised project goal was to further develop our version MIA of modal interfaces [60], which builds upon both IA and MTS, we have started to investigate the above issues for these two basic models, and only then we returned to the continued development of MIA. Comparisons to related work have been carried out throughout. In the following, we therefore structure the presentation of our obtained research results accordingly. We finish with a brief report on the currently ongoing work packages WP 6–8 on "Implementing an editor & a simulator for MIA" (WP 6), "Developing refinement checkers for MIA" (WP 7) and "Evaluating the MIA framework via case studies" (WP 8).

Basic models & full abstraction. We developed fully-abstract refinement preorders in [11] for IA and in [10, 12] for (dMTS), i.e., coarsest precongruences for parallel composition that preserve compatibility and deadlock-freedom, resp., as the basic property.

In [11], the studied model is called EIO, a variant of IA with an additional explicit set of error states. For an unprejudiced study, parallel composition is just an unpruned parallel product, where the states with a mismatch are marked as errors. The fully-abstract precongruence is characterized as component-wise inclusion of a pair of trace sets. This corresponds to the independent work

in [18] and the older, more denotational approach of Dill [33]. A pruning on the parallel product is then defined by us in [11] and proved correct in our setting where, notably, inputs may be nondeterministic; surprisingly we discovered that, in the corresponding IA-setting [29], the definition of pruning is inadequate since parallel composition is not associative.

In this context, two orthogonal, minor aspects are explored in the context of two student theses. The Master's thesis [42] considers the approach of [29], where a simulation – with an additional condition for the set of enabled inputs – instead of alternating simulation is used. The student studied several variations of this and compared them to alternating simulation [30, 61]. The second thesis, a Bachelor's thesis [74], works out our results of [11] with *multicasts* instead of binary handshakes. Multicast allows an output to synchronize with several inputs, instead of only one input, and is employed in related work [71].

Analogously, the aim in [12] is to justify a semantics for MTS by a full-abstraction result, where the basic requirement is that a refinement step must not introduce a deadlock. Note that MTS does not distinguish between input and output actions, so that compatibility is not an issue, and employs a CSP-like parallel composition. An interesting aspect in [12] is that we avoid prejudicing our results by a disputable deadlock notion for MTS: we associate with each MTS its LTS implementations using standard modal refinement; whether a deadlock is introduced then depends only on these LTSs. We characterized the according fully-abstract precongruence by a modality-sensitive form of failure semantics and proved it to be thorough. Note as an aside that MIA-refinement, as described below, inherits non-thoroughness from (d)MTS and modal refinement. In addition, we considered in [12] may- and must-testing in the sense of De Nicola and Hennessy [32] where, in the latter case, a failure/divergence semantics arises. In this context, we showed how to construct a conjunction wrt. failure- and failure/divergence-semantics, which is an MTS in both settings, whereas no MTS is a conjunction wrt. modal refinement.

We continued our approach of [12] in [10] for dMTS. Again, a testing preorder based on deadlock avoidance can be characterised by failure semantics, but the latter is difficult to determine and the preorder fails to be a precongruence. This shortcoming disappears if one aims to avoid deadlock and divergence instead. Here, a failure/divergence semantics arises that is coarser, hence arguably better, than the standard one, and which can also be justified by a natural variant of must-testing. This semantics was only mentioned once before in [82]. Some preparatory results for [10] can be found in the Bachelor's thesis [76].

Modal Interface Automata & contracts. Whereas the first MIA-refinement preorder in [60], on which the original grant proposal was based, inherited features mainly from IA, we improved this in [61] by a proper treatment of τ-must-transitions and by allowing standard weak transitions when matching output-may-transitions. Although we still have essentially only must-transitions for inputs to avoid the compositionality problem in [52], we loosened input-determinism by allowing disjunctive must-transitions for inputs. As desired, MIA-refinement is a precongruence for parallel composition $\|$, disjunction and, by its nature, conjunction. To support the use of conjunction for perspective-based specification, a refinement notion that allows one to extend a MIA's input and output alphabets is needed. This does not succeed in the optimistic approach of [61], but we developed a suitable refinement notion for the pessimistic approach of MIO [5].

Whereas the compositionality problem of [52] is only partly solved in [71], we proposed a complete solution in a fully nondeterministic setting with multicasts, true input-may-transitions and conjunction supporting alphabet extension [13]. The key idea for this full combination of IA and dMTS is to add a *universal state* e to the MIA structure. An input-may-transition from some state s to e expresses that, as in IA, one may add to s the resp. input-transition with arbitrary subsequent behaviour. An alternative representation of this feature is studied in a Bachelor's thesis [50]. Most importantly, there is now also a quotient operator in [13], i.e., an adjoined operation to $\|$: given a global specification P and an already implemented component D, the quotient $P/\!\!/D$ is the coarsest specification Q such that $Q\|D$ refines P, i.e., $P/\!\!/D$ is the most flexible completion of D wrt. P. Quotienting is important for interface theories, because it can be employed for decomposing con-

current specifications stepwise, for specifying contracts and for reusing components. In contrast to [71], our quotient operator for MIA permits *nondeterministic* specifications and complements $\|$ rather than a simpler parallel product without pruning.

So far, errors e in MIA model *unknown* behaviour for which no guarantees can be made and which satisfy the law $e \sqsubseteq p \Rightarrow p = e$, capturing that an error cannot be introduced when refining an ordinary state. We also investigated an error-aware variant EMIA of MIA in [38], whereby errors model *unwanted* behaviour that must not be implemented. Hence, EMIA satisfies the additional law $p \sqsubseteq e \Rightarrow p = e$, i.e., refining cannot redefine an erroneous situation to be non-erroneous. This leads to a finer interface semantics that has several desirable properties, including a more natural treatment of variability inherent, e.g., in software product lines [52] and the fact that EMIA is itself an assembly theory [46], i.e., pairwise compatibility of multiple components already guarantees their overall compatibility.

According to the original project proposal we extended MIA by contracts [14]. To do so, we introduced, analogously to [59], a temporal logics for safety properties and an intuitive satisfaction relation between MIA and formulae. For each logical operator, we defined a corresponding operator on MIA such that a MIA satisfies a formula if and only if it refines the translation of the formula into a MIA. This way and as promised in the original proposal, we developed a truly heterogeneous and more practical interface theory that supports concise interface specifications.

Tool support & case studies. Work on tool support has started in Summer 2015 in the context of a Master's project at Bamberg [40] and is ongoing. In contrast to what we envisaged in the original project proposal, we have not implemented the MIA theory within the *MIO Workbench* [5] but in Google's parallel programming language *Go* [34]. This has been done in anticipation of our future use of MIA as a behavioural type theory for concurrent programming, which is one of the main objectives of our follow-up research proposal presented below. The choice of Go is justified not only by the fact that it is a modern parallel programming language that is employed for developing large concurrent systems, e.g., for *platforms as a service* and *application container engines*, but especially by its support of MIA's synchronous communication discipline.

Our Go implementation so far includes facilities to (i) specify interfaces textually, with support for the MIA operators parallel composition, conjunction, disjunction and hiding, (ii) simulate interfaces via tracing their behaviour step-by-step, (iii) export interfaces into the well known dot-format for graphically displaying MIAs by off-the-shelf viewers, and (iv) check MIA-refinement between interfaces using the QBF-theory within the SMT-solver Z3 [31]. At its core, our refinement checker adapts an existing QBF-encoding of the refinement problem in *Parameterized Modal Transition Systems* [6] to MIA. MIA differs from the former setting in that it is parameter-free but includes internal transitions and distinguishes between input and output actions.

In the next few months we will finalise our prototypic MIA implementation in Go and evaluate it via case studies and comparisons with established interface theory tools such as TICC [2], Ptolemy II [55] or the MIO Workbench [5].

Related work for the proposed follow-up research. Our research project's continuation described below will focus on extending our interface theory MIA with data as well as with mechanisms to express liveness and fairness properties, and on casting MIA as a behavioural type theory. We discuss relevant related work next.

Interface theories with data. Interface theories have also been developed for shared-variable communication, where one speaks of *Interface Modules* [16, 35]. In *Sociable Interfaces* [28], for example, interfaces synchronise via actions and change data values during output actions. Assumptions on the environment are formulated for input actions – essentially as pre- and post-conditions on data states –, and interfaces are deterministic for inputs. Variables can be written by concurrently executing interfaces. To enhance compositional reasoning, Sociable Interfaces enforces that an interface is informed about value changes of variables of interest. A similar approach to handling data has also been adopted by Hennicker et al. [28] for MIO [4].

Moreover, Doyen et al. [35] study a conjunction operator for another interface theory involving shared-variable communication in order to be able to reason about *combined* specifications, i.e., when a component implements several interfaces, and component reuse. However, parallel composition in [35] permits only limited communication, if at all, and no full-abstraction result is proved. As an aside, the conjoining of specifications has also been investigated in TLA, a temporal logic of actions with shared-memory communication [1].

Less studied is value-passing communication in the context of interface theories with data. A recent extension of IA with value passing can be found in Holík et al. [47], where each process has local variables whose values can be communicated. Transitions have data-dependent guards that are required to be mutually exclusive; hence, only deterministic processes are considered. Refinement is stronger than in IA because, if the larger process offers several outputs, at least one of them has to be preserved. Similar to our MIA theory, also a quotient operator is studied and applied to mediator synthesis.

Liveness & fairness. So far our heterogeneous MIA theory only deals with safety properties but not with progress or more general liveness properties. In practice, liveness is only satisfied under some fairness assumptions; see [39] for a classic survey or the recent survey [81] that focuses on concurrent systems. Weak and strong fairness has also been studied in process algebras; see, e.g., [27] and also the dissertation of Parrow [68].

Fairness is frequently specified by adding Büchi states or counters to automata-based specification and verification frameworks; when doing so, often compositionality is ignored. One early approach on how to reason about fairness properties with Büchi states in a compositional, heterogeneous process algebra is [22]. More recently, Siirtola et al. [75] utilizes unidirectional counters and the chaos-free failures/divergence-semantics (CFFD) [79], so that liveness and fairness can be compositionally decided using the popular FDR2 [73] and NuSMV [20] model checkers. Another approach to fairness employing CFFD semantics can be found in [70], while a compositional semantics for weakly fair traces is presented in [83].

A way to avoid the technical complications of fairness assumptions is to treat them implicitly, e.g., as in fair testing [72]. Fair-testing refinement preserves properties of the form 'action a remains always possible so that the occurrence of a may be assumed to be guaranteed, i.e., a is live. This approach is employed for formal verification in combination with partial order reduction in [80].

Behavioural types. Behavioural type systems aim to ensure various properties of concurrent programs, such as race freedom, so as to guarantee well-defined interactions between components [41]. For example, IA has been employed as a foundation for a rich type theory within the Ptolemy II software framework [55]. Similar lines of research utilizing process-algebraic ideas have been developed in the context of contracts, e.g., for mobile processes [15]; the relation of contracts to interfaces has also been investigated [67]. More importantly, however, a simple yet practical version of 'processes as types' has recently attracted much attention: multi-party *session types* [26, 48], which are based on ideas of the π-calculus [65] and are used for formally reasoning about communication protocols and, in particular, web services.

Session types allow one to statically check for simple communication properties between processes that cooperate according to a globally agreed protocol. Of particular interest are compatibility and progress properties, which ensures that communication sessions are always successfully conducted to the end. A session type results from decomposing the globally agreed property by projecting it onto each protocol participant. Hence, session type theories deal with closed systems and refer to a pessimistic notion of compatibility. This is in contrast to IA-based interface and other behavioural type and refinement theories, which consider open components and employ an optimistic notion of compatibility.

The most important point of departure between session types on the one hand and behavioural type and refinement theories on the other hand is that the former allow one to reason about properties of a concurrent system's communication structure only – and this only for restricted communication topologies –, while the latter consider semantically deeper properties and general

communication structures. For example, most session type theories do not support reasoning about complex temporal properties or about protocols such as the sliding window protocol [17].

In our follow-up proposal we plan to apply MIA's refinement preorder to behavioural type checking. However, MIA-refinement is likely to be too fine for this purpose, because it is compositional wrt. all contexts. By restricting compositionality to contexts satisfying a specific communication structure that is specified by a session type, we will coarsen MIA-refinement and thereby permit more implementations. Note that, while most session theories employ an asynchronous communication mechanism, also variants with synchronous multicast communication, as adopted by MIA, have been studied [49].

1.1 Project-Related Publications

1.1.1 Articles Published by Outlets with Scientific Quality Assurance

[PSP1] G. Lüttgen and W. Vogler. *Modal Interface Automata.* Logical Methods in Computer Science, 9(3:4), 2013.

[PSP2] G. Lüttgen, W. Vogler, and S. Fendrich. *Richer interface automata with optimistic and pessimistic compatibility.* Acta Informatica, 52(4-5):305–336, 2015. An extended abstract appeared in 13th Intl. Workshop on Automated Verification of Critical Systems (AVoCS 2013), vol. 66 of ECEASST, Europ. Assoc. of Software Science and Technology, 2013.

[PSP3] F. Bujtor, S. Fendrich, G. Lüttgen, and W. Vogler. *Nondeterministic Modal Interfaces.* In 41st Intl. Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM 2015), vol. 8939 of LNCS, pp. 152–163, Springer, 2015. A full version has been submitted to the Theoretical Computer Science journal.

[PSP4] F. Bujtor and W. Vogler. *Error-pruning in Interface Automata.* Theoretical Computer Science, 597:18–39, 2015. An extended abstract appeared in 40th Intl. Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM 2014), vol. 8327 of LNCS, pp. 162–173, Springer, 2014.

[PSP5] F. Bujtor and W. Vogler. *Failure Semantics for Modal Transition Systems.* ACM Trans. on Embedded Computing Systems, 14(4):67, 2015. An extended abstract appeared in 14th Intl. Conf. on Application of Concurrency in System Design (ACSD 2014), pp. 42–51, IEEE, 2014.

[PSP6] F. Bujtor, L. Sorokin, and W. Vogler. *Testing preorders for dMTS: Deadlock and the new deadlock/divergence-testing.* In 15th Intl. Conf. on Application of Concurrency in System Design (ACSD 2015), pp. 60–69, IEEE, 2015. A full version has been submitted to the ACM Trans. on Embedded Computing Systems journal.

[PSP7] S. Fendrich and G. Lüttgen. *A generalised theory of Interface Automata, component compatibility and error.* In 12th Intl. Conf. on Integrated Formal Methods (IFM 2016), LNCS, Springer, 2016. Accepted for publication after peer-reviewing.

[PSP8] F. Bujtor and W. Vogler. *ACTL for Modal Interface Automata.* In 16th Intl. Conf. on Application of Concurrency in System Design (ACSD 2016), IEEE. Accepted for publication after peer-reviewing.

1.1.2 Other Publications & Patents

None

2 Objectives and Work Programme

2.1 Anticipated Total Duration of the Project

This is a follow-up proposal (renewal) to DFG project *Foundations of Heterogeneous Specifications Using State Machines and Temporal Logic*, which has been funded for 24 months (grant no. LU 1748/3-1 and, resp., VO 615/12-1) and of which approx. 75% (full-time equivalent) has been completed to date. Further DFG funding is now sought for extending this project by 36 months.

2.2 Objectives

Aim. The project's aim is to significantly enhance the theoretical foundations and practical utility of those *heterogeneous specification formalisms for concurrent systems*, that mix operational and logic styles of specification. Our particular interest is in interface theories on the basis of de Alfaro and Henzinger's Interface Automata (IA) and Larsen's Modal Transition Systems (MTS), which emphasise parallel composition and logic conjunction for composing specifications. The project so far has introduced the novel *Modal Interface Automata* (MIA) theory, which stands apart from related work in that it fully supports compositional, nondeterministic specifications and features temporal-logic operators for specifying safety properties. The distinct advantage of this are more compact, concise and reusable specifications.

The follow-up project, for whose funding we apply here, shall deliver substantial extensions to our MIA theory along two axes: *heterogeneity* and *applicability*. Firstly, MIA's heterogeneity shall be greatly expanded by adding further logic operators and refining MIA's semantics, so that also liveness and fairness properties can be expressed. This will permit the specification of significant additional classes of reactive-system and concurrent-software requirements within MIA, namely those capturing behavioural guarantees. Secondly, MIA's application scope shall be widened significantly from solely being a refinement theory for compositional specification and design, to also lending itself as a behavioural type theory. This will allow for advanced typing support for parallel programming, which will ultimately lead to more reliable concurrent software. As a first but important step towards technology transfer, the project shall produce and evaluate novel, prototypic tools for MIA within the parallel programming language Go.

Objectives. The concrete objectives of the follow-up project are as follows:

1. *Extending MIA with data (WP BA1 & A2).* Practical specification and programming languages obviously deal with data. In parallel programs, data is exchanged in intricate ways via shared variables or value passing. Reasoning about data becomes nontrivial, because data must be guarded against interferences caused by data races. Here, we wish to extend MIA with concepts for data, and to revise our MIA framework accordingly.

2. *Expanding MIA's heterogeneity (WP A3, A4 & BA5).* We wish to add further propositional and temporal logic operators, in particular implication and operators for expressing liveness properties. In this context, we will explore the trade-off between relaxing MIA-refinement, e.g., in the spirit of fair testing, and enriching MIAs, e.g., by counters or Büchi states.

3. *Developing MIA to a behavioural type theory (WP A+BA6, A+BA7, A8 & BA11).* In order to utilise MIA for parallel programming, we wish to study MIA as a type theory, where MIAs describe the behaviour of processes and communication channels. This involves adapting MIA-refinement for type checking as well as inferring behavioural types from parallel programs. The former will employ session types for describing the communication structure of the partner processes with which a process communicates.

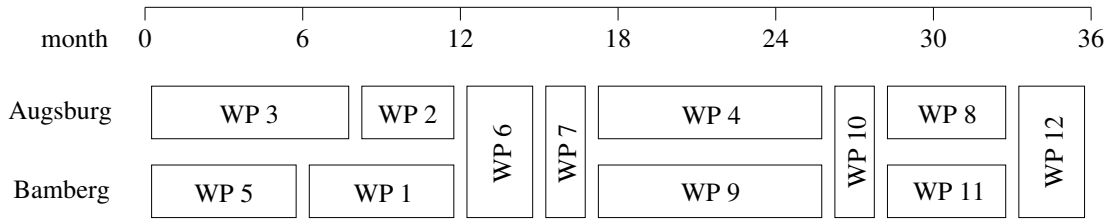

Figure 1: Proposed scheduling of work packages.

4. *Tool support & case studies (WP BA9, A+BA10 & A+BA12).* Interface theories and behavioural types will only be adopted in the practice of developing reactive systems and concurrent programs, if they are accompanied by tools. Starting from our first prototype tool of MIA in Google's parallel programming language Go, we wish to optimise the SMT-based refinement checker and add facilities for providing diagnostic information to the programmer. We will further incorporate our temporal logic operators as well as data. The resulting tool shall be evaluated by means of several case studies and made available publicly.

Outcome. In the last 18 months (full-time equivalent), the project has developed the theoretical foundations for novel, expressive and heterogeneous interface theories. The project's outcome to date are 11 peer-reviewed publications in international conferences and high-quality journals. To make our results useful for software development practice, the project extension for which we apply here, will add to our theory essential features for concurrent programming wrt. data, heterogeneity and behavioural types. A sophisticated tool prototype accompanied by case studies will lay the basis for a future technology transfer to the software industry.

2.3 Work Programme incl. Proposed Research Methods

The proposed programme of work involves 12 work packages, in which our international and national collaborators participate as follows: Prof. Cleaveland in work packages WP BA9, A+BA10 and BA11, Prof. Valmari in WP A4, and Prof. Nestmann and Dr. Peters in WP A+BA6, A+BA7 and A8. Work packages whose name starts with letter 'A' and letters 'BA' will be led by the project team at Augsburg and Bamberg, resp., and those starting with 'A+BA' will be conducted jointly throughout. The proposed scheduling of all work packages is depicted in Fig. 1.

WP BA1: Shared-variable communication. This work package shall investigate a different communication mechanism to MIA's multicast, namely shared-variable communication. In particular, processes will communicate via *global* variables, which can be read in transition guards and written by actions associated with transitions. A particular emphasis of our study will be on the impact of data races, i.e., write/write and read/write conflicts, on compatibility.

Our reference semantics for share-variable MIA will be a ground semantics that simply encodes data values into states, but which is impractical due to state explosion. Instead, we will employ *pre- and post-conditions* at transitions in the form of logic formulas over atomic data propositions, such as value comparisons. This is inspired by Hennicker et al.'s handling of data for MIO [4] and also by Sociable Interfaces [28], and naturally supports compositional and symbolic reasoning. The resulting MIA transitions may be symbolic in that data values, also for write operations, might not necessarily refer to specific values but to ranges of values; these ranges may be expressed by basic data types such as *integer* or *bool*. This aspect will be taken into account by our notion of MIA-refinement in the shared-variable setting.

WP A2: Value-passing communication. Using similar ideas as above, this work package shall extend MIA's multicast communication by value-passing. In departure from shared-variable

communication, processes will have *local* variables, which again can be used in transition guards and actions. Following the lines of [47] we will study, in our more general nondeterministic setting, the influence of value passing on compatibility and the construction of quotients.

WP A3: Revisiting MIA semantics. Also as a preparation for WP A4, we shall investigate whether there are natural precongruences lying strictly between MIA-refinement and the trace- and failure-based preorders studied by us in [11] for IA and in [10, 12] for (d)MTS, resp. The motivation is that the latter preorders are fully justified by intuitive notions of observation, in contrast to the finer bisimulation-like MIA-refinement. At the same time, the interplay of the transition modalities is currently represented poorly in the (d)MTS failure-semantics.

A first step shall be to combine the approaches of [10, 11, 12] and to determine the coarsest precongruence wrt. parallel composition that preserves compatibility and deadlock-freedom. Another candidate is the coarsest precongruence wrt. parallel composition and $\wedge$, which preserves compatibility and consistency; here, $\wedge$ is the conjunction operator we developed for MIA-refinement [13]. This second idea is taken from Logic LTS [58].

WP A4: Expressing liveness in MIA. So far, operators for expressing liveness properties are missing in our MIA-logic. It seems unlikely that such connectives can be translated into resp. operators on MIAs in a way that the main results of our heterogeneous framework are preserved. Typically, the treatment of liveness properties needs a preorder based on fairness notions, possibly for a modified model, e.g., with Büchi states. This usually leads to complications wrt. compositionality. Therefore, it might be more promising to study preorders that are slightly stricter than the failure preorder, such as the CFFD-preorder of Valmari et al. [70] or the fair-testing preorder [72]. As explained in the related work section above, the fair-testing preorder preserves properties of the form 'action a will be performed eventually' without the need of fairness notions or Büchi states.

WP BA5: Implication for MIA. Like conjunction, implication $\rightarrow$ is a very important operator for specification, which can be characterised in MIA as the adjoint of conjunction. It is unlikely that implication on general MIAs exists, because this would yield a negation operator on MIAs via $P \rightarrow \mathrm{ff}$. Therefore, we wish to explore restricted forms of implication, e.g., by adapting the independence relation of Larsen et al. [53]. Also negative results showing that implication does not exist for certain MIAs would be of interest. In this context, it might also be worthwhile to study the impossibility of a finite axiomatisation of MIA-refinement in the presence of implication.

WP A+BA6: Comparing interface theories and session types. 'Processes as types' is a wide field, which is addressed by largely disjoint research groups within the concurrency theory community. Since the next work package aims at enhancing the MIA interface theory with contextual information expressed by session types, it is necessary to understand in technical detail the commonalities and differences between interface theories and session types. Session types are syntactic representations of communication structures, whereas MIA is an operational model focusing on a richer semantics and with fewer restrictions. We expect that the typing relation of session types can be understood as an abstraction of MIA-refinement. A challenging question is what role the transition modalities in MIA play in this context.

WP A+BA7: Coarsening MIA-refinement. A MIA specifying a system component can be implemented by more MIAs than those permitted by MIA-refinement, if contextual information is taken into account. The current MIA-refinement is a precongruence and, thus, respects composition with arbitrary contexts, i.e., component environments. In practice, however, a system designer or programmer often has a class of contexts satisfying a specific communication structure in mind. Such classes may be captured by session types, which in essence coarsens the refinement relation. Technically, this will correspond to quotienting the specification-side MIA by session types, thereby eliminating behaviour not respecting the given communication structure. For this purpose we will investigate and adapt our quotienting operator for MIA.

WP A8: Extending MIA with name-passing. A strength of session types is the possibility to delegate a session from one component to another by name-passing as in the π-calculus [65]. Similarly, channel names can be passed along channels in parallel programming languages including Go. As an aside, note that channels in Go are finite FIFO buffers, which can be modelled in MIA. Because we wish to use MIAs as behavioural types for Go, we will enrich value-passing MIA by the capability to pass names as values. To do so, we will built upon the vast work on π-calculi theory. Of particular interest to us will be how the presence of name-passing effects compatibility and consistency within MIA.

WP BA9: Implementing heterogeneous MIA with data. Our existing initial tool for MIA programmed in Go will be extended to handle data on the basis of the first two work packages on shared-variable and value-passing communication, resp. In the first place, this involves the implementation of MIA's operators, which now deal with symbolic transitions and with variable values associated with states. A second challenge is the according extension of our SMT-based refinement checker, for which we will employ a *counterexample-guided abstraction refinement* (CEGAR) approach [21] including *predicate abstraction* [43], as is standard in modern automated verification [36]. Obviously, we will have to restrict ourselves to decidable data theories such as integer linear arithmetic, so that refinement checking can again be reduced to SMT-checking. If time permits, we will also study how the refinement checker could be extended to handle the name-passing described in the previous work package.

WP A+BA10: Providing diagnostic information. So far, our prototypic tool does not provide any feedback to the user if a refinement check fails. The aim of this work package is to enhance the tool with the ability to provide according diagnostic information. Since MIA-refinement is a co-inductively defined behavioural relation, we envisage that this information will have the form of a strategy for a two-player game [77]. To achieve our goal, we first need a game-theoretic characterisation of MIA-refinement in the presence of data and the resp. pre- and post-conditions. The second step is then to extract the strategy from the satisfying assignment that is output by the SMT-solver; recall that our refinement checker employs a QBF-encoding.

WP BA11: Behavioural type checking and type interference. In this work package we will use MIA as a behavioural type theory for the parallel programming language Go. We envisage that the programmer annotates every Go process in a program with a MIA that describes the process' use of channels when communicating with the other processes in the program. Such a MIA will typically be the result of the refinement-based design methodology supported by our MIA theory. Essentially, type checking is MIA-refinement checking as discussed in WP BA9, where the abstract MIA (i.e., the implemented type) to be checked against the given MIA (i.e., the desired type) is inferred from the Go process. This will take place inside the CEGAR mentioned above. Part of the work package will also be the design of a nice textual syntax for MIA types, which is suitable for programmers; in particular, one can think of a notation that decomposes MIA types onto each process and channel of the considered Go program separately.

WP A+BA12: Case studies. Our heterogeneous interface theory MIA and its use as behavioral type theory in the Go programming language will be evaluated by means of case studies, which will employ our prototypic tool support for MIA.

We will focus on at least two of the five following case studies taken from the rich literature on interface theories and related fields, some of which we will suitably detail with data aspects in order to be able to analyse also data-dependent communication behaviour: (a) a *car parking system*, which has been studied by Benvensite et al. [7] and which utilises Modal Interfaces [71] and its quotienting operator for expressing contracts; (b) a *mode logic within an aircraft control system* [57, 62], which is typically designed in a stepwise, top-down manner and which monitors an aircraft's sensors and selects the control laws that in turn compute values for adjusting the aircraft's actuators; (c) a *harbour docking system* that manages ships waiting to dock at a harbour,

tracks the entering of ships into the port to the quays and prevents the double allocation of ships to quays, whose design has been investigated by Harbird [44] using a model transformation approach that combines operational and declarative design language features; (d) an *autonomous rendezvous and docking system*, which describes a spacecraft's capability of locating and docking with another spacecraft and which has been studied by Emmi et al. [37] in the context of Interface Automata; (e) the *remote procedure call* case study of Broy and Lamport [9], which has been employed by Larsen et al. for evaluating Modal Transition Systems with conjunction [53].

In each case, we will design the system in a refinement-based fashion using MIA as a specification and design language, starting from a global specification that conjunctively composes declarative behavioural requirements and which we expect to heavily employ MIA's temporal-logic operators. Then, we will successively detail much of the declarative content using operational constructs, and each such refinement step shall be verified by our refinement checker. At the end of the design phase we will have obtained a collection of parallely composed MIAs, each of which may be interpreted as a behavioural type. The system under study shall then be implemented in Go as a collection of processes, one for each behavioural type. Our type checker will be exercised to verify that each process satisfies its type, so that our compositional theory guarantees that the implemented parallel system indeed satisfies its global specification.

Using this approach to specification, design and verification, we will systematically evaluate our interface and behavioural type theories as well as our tool prototype. Our focus will be on several orthogonal aspects. Firstly, we will explore the flexibility of MIA as a compositional, heterogeneous refinement theory, as well as the utility of the behavioural type system for parallel programming. Secondly, the benefit of heterogeneity, i.e., the mixing of declarative and operational styles of specification, will be investigated. In this context, we wish to establish whether heterogeneity does indeed yield more concise specifications and whether our MIA interface theory adopts the right set of operational, propositional-logic and temporal-logic operators. Thirdly, the scalability of our algorithms for checking MIA-refinement, inferring behavioural types from Go processes and providing diagnostic information to designers and programmers will be evaluated and, if necessary, improved. Although the above case studies are below industrial size, they share many characteristics of industrial software designs. The conduct of these case studies will help us transferring our technology to software engineers who can then build tool-supported and industrial-strength interface methodologies on top of it.

2.4 – 2.6 Not Applicable

2.4 Data handling; 2.5 Other information; 2.6 Descriptions of proposed investigations involving experiments on humans, human materials or animals.

2.7 Information on Scientific and Financial Involvement of International Cooperation Partners

Not applicable; see Sec. 5.4.1 for information on international (and national) cooperation partners with whom we will be working together informally. These partners neither have applied nor will apply for funding with the DFG or a partner organization in the context of this project proposal.

3 Bibliography

[1] M. Abadi and L. Lamport. Conjoining specifications. *ACM TOPLAS*, 1(3):507–534, 1995.

[2] B.T. Adler, L. de Alfaro, L.D. da Silva, M. Faella, A. Legay, V. Raman, and P. Roy. TICC: A tool for interface compatibility and composition. In *CAV 2006*, vol. 4144 of *LNCS*, pp. 59–62. Springer, 2006.

[3] S. Bauer, A. David, R. Hennicker, K.G. Larsen, A. Legay, U. Nyman, and A. Wasowski. Moving from specifications to contracts in component-based design. In *FASE 2012*, vol. 7212 of *LNCS*, pp. 43–58. Springer, 2012.

[4] S. Bauer, R. Hennicker, and M. Wirsing. Interface theories for concurrency and data. *Theoret. Comp. Sc.*, 412(28):3101–3121, 2011.

[5] S. Bauer, P. Mayer, A. Schroeder, and R. Hennicker. On weak modal compatibility, refinement, and the MIO Workbench. In *TACAS 2010*, vol. 6015 of *LNCS*, pp. 175–189. Springer, 2010.

[6] N. Benes, J. Kretínský, K.G. Larsen, M.H. Møller, S. Sickert, and J. Srba. Refinement checking on Parametric Modal Transition Systems. *Acta Inform.*, 52(2-3):269–297, 2015.

[7] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T.A. Henzinger, and K.G. Larsen. Contracts for system design. Techn. Rep. RR-8147, INRIA, France, 2012.

[8] D. Beyer, A. Chakrabarti, T.A. Henzinger, and S.A. Seshia. An application of web-service interfaces. In *ICWS 2007*, pp. 831–838. IEEE, 2007.

[9] M. Broy, S. Merz, and K. Spies, eds. *Formal Systems Specification: The RPC-Memory Specification Case Study*, vol. 1169 of *LNCS*. Springer, 1996.

[10] F. Bujtor, L. Sorokin, and W. Vogler. Testing preorders for dMTS: Deadlock and the new deadlock/divergence-testing. In *ACSD 2015*, pp. 60–69. IEEE, 2015.

[11] F. Bujtor and W. Vogler. Error-pruning in Interface Automata. *Theoret. Comp. Sc.*, 597:18–39, 2015.

[12] F. Bujtor and W. Vogler. Failure semantics for Modal Transition Systems. *ACM Trans. on Embedded Computing Systems*, 14(4):67, 2015.

[13] F. Bujtor and W. Vogler. Nondeterministic modal interfaces. In *SOFSEM 2015*, vol. 8939 of *LNCS*, pp. 152–163. Springer, 2015.

[14] F. Bujtor and W. Vogler. ACTL for Modal Interface Automata. In *ACSD 2016*. IEEE, 2016. Accepted for publication.

[15] G. Castagna and L. Padovani. Contracts for mobile processes. In *CONCUR 2009*, vol. 5710 of *LNCS*, pp. 211–228. Springer, 2009.

[16] A. Chakrabarti, L. de Alfaro, T.A. Henzinger, and F.Y.C. Mang. Synchronous and bidirectional component interfaces. In *CAV 2002*, vol. 2404 of *LNCS*, pp. 414–427. Springer, 2002.

[17] M. Charalambides, P. Dinges, and G. Agha. Parameterized concurrent multi-party session types. In *FOCLASA 2012*, vol. 91 of *EPTCS*, pp. 16–30, 2012.

[18] T. Chen, C. Chilton, B. Jonsson, and M. Kwiatkowska. A compositional specification theory for component behaviours. In *ESOP 2012*, vol. 7211 of *LNCS*, pp. 148–168. Springer, 2012.

[19] C. Chilton. *An Algebraic Theory of Componentised Interaction*. PhD thesis, Univ. of Oxford, England, 1990.

[20] A. Cimatti, E.M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An open source tool for symbolic model checking. In *CAV 2002*, vol. 2404 of *LNCS*, pp. 241–268. Springer, 2002.

[21] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In *CAV 2000*, vol. 1855 of *LNCS*, pp. 154–169. Springer, 2000.

[22] R. Cleaveland and G. Lüttgen. A semantic theory for heterogeneous system design. In *FSTTCS 2000*, vol. 1974 of *LNCS*, pp. 312–324. Springer, 2000.

[23] R. Cleaveland and G. Lüttgen. A logical process calculus. In *EXPRESS 2002*, vol. 68,2 of *ENTCS*. Elsevier, 2002.

[24] R. Cleaveland, J. Parrow, and B. Steffen. The Concurrency Workbench: A semantics-based tool for the verification of concurrent systems. *ACM TOPLAS*, 15(1):36–72, 1993.

[25] R. Cleaveland and D. Yankelevich. An operational framework for value-passing processes. In *POPL '94*, pp. 326–338. ACM, 1994.

[26] M. Coppo, M. Dezani-Ciancaglini, L. Padovani, and N. Yoshida. A gentle introduction to multiparty asynchronous session types. In *SFM 2015*, vol. 9104 of *LNCS*, pp. 146–178. Springer, 2015.

[27] G. Costa and C. Stirling. Weak and strong fairness in CCS. *Inform. and Comput.*, 73(3):207–244, 1987.

[28] L. de Alfaro, L.D. da Silva, M. Faella, A. Legay, P. Roy, and M. Sorea. Sociable interfaces. In *FroCoS 2005*, vol. 3717 of *LNAI*, pp. 81–105. Springer, 2005.

[29] L. de Alfaro and T.A. Henzinger. Interface automata. In *FSE 2001*, pp. 109–120. ACM, 2001.

[30] L. de Alfaro and T.A. Henzinger. Interface-based design. In *Engineering Theories of Software-Intensive Systems*, vol. 195 of *NATO Science Series*. Springer, 2005.

[31] L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS 2008*, vol. 4963 of *LNCS*, pp. 337–340. Springer, 2008.

[32] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoret. Comp. Sc.*, 34:83–133, 1984.

[33] D.L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*. MIT Press, 1989. ACM Distinguished Dissertation.

[34] A.A. Donovan and B.W. Kernighan. *The Go Programming Language*. Addison-Wesley, 2015.

[35] L. Doyen, T.A. Henzinger, B. Jobstmann, and T. Petrov. Interface theories with component reuse. In *EMSOFT 2008*, pp. 79–88. ACM, 2008.

[36] V. D'Silva, D. Kroening, and G. Weissenbacher. A survey of automated techniques for formal software verification. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(7):1165–1178, 2008.

[37] M. Emmi, D. Giannakopoulou, and C.S. Păsăreanu. Assume-guarantee verification for Interface Automata. In *FM 2008*, vol. 5014 of *LNCS*, pp. 116–131. Springer, 2008.

[38] S. Fendrich and G. Lüttgen. A generalised theory of Interface Automata, component compatibility and error. In *IFM 2016*, LNCS. Springer, 2016. Accepted for publication.

[39] N. Francez. *Fairness*. Texts and Monographs in Computer Science. Springer, 1986.

[40] J. Gareis. Prototypical integration of the Modal Interface Automata theory in Google Go. Master's thesis, Univ. Bamberg, Germany, 2015.

[41] S.J. Gay, N. Gesbert, and A. Ravara. Session types as generic process types. In *EXPRESS/SOS 2014*, vol. 160 of *EPTCS*, pp. 94–110, 2014.

[42] S. Göhrle. Input-Verweigerungs-Simulation für Interface-Automaten. Master's thesis, Univ. Augsburg, Germany, 2014.

[43] S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In *CAV '97*, vol. 1254 of *LNCS*, pp. 72–83. Springer, 1997.

[44] L. Harbird. *Model-Based Refinement of Contractual State Machines*. PhD thesis, Univ. York, England, 2012.

[45] L. Harbird, A. Galloway, and R.F. Paige. Towards a model-based refinement process for Contractual State Machines. In *IEEE Symp. on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pp. 108–115. IEEE, 2010.

[46] R. Hennicker and A. Knapp. Moving from interface theories to assembly theories. *Acta Informatica*, 52(2-3):235–268, 2015.

[47] L. Holík, M. Isberner, and B. Jonsson. Mediator synthesis in a component algebra with data. In *Correct System Design*, vol. 9276 of *LNCS*, pp. 238–259. Springer, 2015.

[48] K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In *POPL 2008*, pp. 273–284. ACM, 2008.

[49] D. Kouzapas, R. Gutkovas, and S.J. Gay. Session types for broadcasting. In *PLACES 2014*, vol. 155 of *EPTCS*, pp. 25–31, 2014.

[50] C. Kühbacher. Implizite und verbotene Eingaben bei modalen Interface-Automaten. Bachelor's thesis, Univ. Augsburg, Germany, 2015.

[51] K.G. Larsen. Modal specifications. In *Automatic Verification Methods for Finite State Systems*, vol. 407 of *LNCS*, pp. 232–246. Springer, 1989.

[52] K.G. Larsen, U. Nyman, and A. Wasowski. Modal I/O automata for interface and product line theories. In *ESOP 2007*, vol. 4421 of *LNCS*, pp. 64–79. Springer, 2007.

[53] K.G. Larsen, B. Steffen, and C. Weise. A constraint oriented proof methodology based on Modal Transition Systems. In *TACAS '95*, vol. 1019 of *LNCS*, pp. 17–40. Springer, 1995.

[54] K.G. Larsen and L. Xinxin. Equation solving using Modal Transition Systems. In *LICS '90*, pp. 108–117. IEEE, 1990.

[55] E.A. Lee and Y. Xiong. A behavioral type system and its application in Ptolemy II. *Formal Aspects of Computing*, 16(3):210–237, 2004.

[56] M. Lohstroh and E.A. Lee. An interface theory for the Internet of Things. In *SEFM 2015*, vol. 9276 of *LNCS*, pp. 20–34. Springer, 2015.

[57] G. Lüttgen and V. Carreño. Analyzing mode confusion via model checking. In *SPIN '99*, vol. 1680 of *LNCS*, pp. 120–135. Springer, 1999.

[58] G. Lüttgen and W. Vogler. Ready simulation for concurrency: It's logical! *Inform. and Comput.*, 208:845–867, 2010. An extended abstract appeared in ICALP 2007, vol. 4596 of LNCS, pp. 752-763, Springer, 2007.

[59] G. Lüttgen and W. Vogler. Safe reasoning with Logic LTS. *Theoret. Comp. Sc.*, 412(28):3337–3357, 2011. An extended abstract appeared in SOFSEM 2009, vol. 5404 of LNCS, pp. 376-387, Springer, 2009.

[60] G. Lüttgen and W. Vogler. Modal interface automata. *Logical Methods in Computer Science*, 9(3:4), 2013. An extended abstract appeared in TCS 2012, vol. 7604 of LNCS, pp. 265-279, Springer, 2012.

[61] G. Lüttgen, W. Vogler, and S. Fendrich. Richer interface automata with optimistic and pessimistic compatibility. *Acta Inform.*, 52(4-5):305–336, 2015. An extended abstract appeared in AVoCS 2013, vol. 66 of ECEASST, EASST, 2013.

[62] Mathworks, Inc. Control design: Designing an aircraft elevator control system. URL http:// www.mathworks.de/control-systems/demos.html?file=/products/demos/stateflow/fdir/FDIR_overview.html. Last accessed on 26th July 2012.

[63] M. Merro, J. Kleist, and U. Nestmann. Mobile objects as mobile processes. *Inform. and Comput.*, 177(2):195–241, 2002.

[64] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[65] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I & II. *Inform. and Comput.*, 100(1):1–77, 1992.

[66] U. Nestmann. What is a good encoding of guarded choice? *Inform. and Comput.*, 156(1-2):287–319, 2000.

[67] P. Nuzzo, A. Iannopollo, S. Tripakis, and A.L. Sangiovanni-Vincentelli. Are interface theories equivalent to contract theories? In *MEMOCODE 2014*, pp. 104–113. IEEE, 2014.

[68] J. Parrow. *Fairness Properties in Process Algebra*. PhD thesis, Uppsala Univ., Sweden, 1986.

[69] K. Peters. *Translational Expressiveness – Comparing Process Calculi Using Encodings*. PhD thesis, Techn. Univ. Berlin, Germany, 2012.

[70] A. Puhakka and A. Valmari. Liveness and fairness in process-algebraic verification. In *CONCUR 2001*, vol. 2154 of *LNCS*, pp. 202–217. Springer, 2001.

[71] J.-B. Raclet, E. Badouel, A. Benveniste, B. Caillaud, A. Legay, and R. Passerone. A modal interface theory for component-based design. *Fundamenta Informaticae*, 107:1–32, 2011.

[72] A. Rensink and W. Vogler. Fair testing. *Inform. and Comput.*, 205(2):125–198, 2007.

[73] A.W. Roscoe. *Understanding Concurrent Systems*. Springer, 2010.

[74] C. Schlosser. EIO-Automaten mit Parallelkomposition ohne Internalisierung. Bachelor's thesis, Univ. Augsburg, Germany, 2012.

[75] A. Siirtola, A. Puhakka, and G. Lüttgen. Introducing fairness into compositional verification via unidirectional counters. In *ACSD 2012*, pp. 32–41. IEEE, 2012.

[76] L. Sorokin. F-Semantik für disjunktive Modale Transitionssysteme. Bachelor's thesis, Univ. Augsburg, Germany, 2014.

[77] C. Stirling. Bisimulation, modal logic and model checking games. *Logic Journal of the IGPL*, 7(1):103–124, 1999.

[78] A. Valmari. A stubborn attack on state explosion. *Formal Methods in System Design*, 1(4):297–322, 1992.

[79] A. Valmari and M. Tienari. An improved failures equivalence for finite-state systems with a reduction algorithm. In *PSTV '91*, pp. 3–18. North-Holland, 1991.

[80] A. Valmari and W. Vogler. Fair testing and stubborn sets. In *SPIN 2016*, LNCS. Springer, 2016. Accepted for publication.

[81] R. van Glabbeek and P. Höfner. CCS: it's not fair! – Fair schedulers cannot be implemented in CCS-like languages even under progress and certain fairness assumptions. *Acta Inform.*, 52(2-3):175–205, 2015.

[82] W. Vogler. Failures semantics and deadlocking of modular Petri nets. *Acta Inform.*, 26(4):333–348, 1989.

[83] W. Vogler. Efficiency of asynchronous systems, read arcs, and the MUTEX-problem. *Theoret. Comp. Sc.*, 275(1-2):589–631, 2002.

[84] D. Zhang and R. Cleaveland. Fast generic model-checking for data-based systems. In *FORTE 2005*, vol. 3731 of *LNCS*, pp. 83–97. Springer, 2005.

4 Requested Modules/Funds

Funds are requested for the full application period of 36 months. All funds other than for staff are to be split 50:50 between Augsburg and Bamberg.

4.1 Basic Module

4.1.1 Funding for Staff

At each site, Augsburg and Bamberg:

- *1 Research Assistant, RA* (PhD student) for 36 months full-time (Doktorandin/Doktorand und Vergleichbare, 100% der regelmäßigen Arbeitszeit)

  This RA will be the main researcher contributing to the project from Augsburg's and Bamberg's end, resp. That this post is full-time does not only match the volume of work proposed, but is also necessary to attract suitable candidates, given the excellent job prospects and the high starting salaries offered in the IT sector.

  It is envisaged that, initially, the RAs currently working on the project, Dipl.-Inform. Ferenc Bujtor (Augsburg) and Dipl.-Math. Sascha Fendrich will stay on; however, they are likely to leave in Spring 2017 when they are expected to have completed their doctoral studies. In Augsburg, the RA post will likely be filled with a student who conducts their Master's thesis on a concurrency-theoretic topic with Prof. Vogler.

  In Bamberg, Johannes Gareis, M.Sc., shall be employed as RA after the leaving of Sascha Fendrich. For his Master's thesis, Mr. Gareis has built an initial prototype of a tool implementing MIA for the programming language Go [40].

- *1 Student Assistant, SA* (Studentische Hilfskraft) at the standard rate.
  - *At Augsburg:* 13.44 Euro per hour for 44 hours per month for 24 months, for a total of 14,192.64 Euro.
  - *At Bamberg:* 12.16 Euro per hour for 44 hours per month for 36 months, for a total of 19,261.44 Euro.

  After a familiarisation period with the project of three months, the SA will assist the RA with tool development (mainly with the programming tasks of WP BA9, A+BA10 & BA11) and evaluation (with the conduct of case studies in WP A+BA10). A student at Augsburg who has taken one or more of Prof. Vogler's modules on automata theory, process algebra and distributed algorithms, would be most suitable for the SA position at Augsburg. The SA at Bamberg will likely be recruited among the students who take Prof. Lüttgen's modules on software engineering, software analysis and software verification.

4.1.2 Direct Project Costs

4.1.2.1 Equipment up to 10,000 Euro, Software and Consumables. Specialised equipment for carrying out the proposed project is not required. All staff on the project will be equipped by the Universities of Augsburg and Bamberg with

- *State-of-the-art PCs and/or laptops*. All software necessary for carrying out the project is available free of charge.
- *Headsets and webcams*. These will enable the communication (via Skype or similar services) between the project sites Augsburg and Bamberg, and with the project partners at Maryland, USA, Tampere, Finland, and Berlin, Germany.

No funds for consumables are requested.

4.1.2.2 Travel Expenses. Funds are requested to cover the cost of travel that is required for carrying out the project and presenting its results to the international scientific community:

- *Conferences/workshops*

  Each site is expected to actively participate in two international conferences/workshops during each project year, with one travel being within Europe and one overseas. Examples of targeted meetings are, in alphabetical order, the international conferences on *Application of Concurrency to System Design* (ACSD), *Concurrency Theory* (CONCUR), *European Symposium on Programming* (ESOP), *Fundamental Approaches to Software Engineering* (FASE), *Foundations of Software Science and Computation Structures* (FOSSACS), *Foundations of Software Engineering* (FSE), *Automata, Languages and Programming* (ICALP) and *Current Trends in Theory and Practice of Computer Science* (SOFSEM).

  The cost of a European trip is estimated at 1,500 Euro (including registration fees, accommodation and travel expenses), and the one for an overseas trip at 2,200 Euro. Therefore, 22,200 Euro is requested in total for all conference/workshop travel.

- *Augsburg–Bamberg cooperation*

  The cooperation between Augsburg and Bamberg will require of the RAs mutual, quarterly short visits of three days each for the duration of the project. In addition, each applicant will travel to the partner site once per year for three days. This totals 18 three-day trips, estimated at 400 Euro per trip. Thus, 7,200 Euro is requested to support the Augsburg–Bamberg cooperation throughout the application period.

- *Further cooperations*

  The two international project partners are located in Maryland, USA (Prof. Rance Cleaveland) and Tampere, Finland (Prof. Antti Valmari). For the project duration, one one-week visit to each of the partners by one member of the project team is planned. The travel expenses for the Maryland trip and the Tampere trip will be approx. 1,900 Euro and 1,200 Euro, resp.

  Semi-annual three-day visits to the national project partner in Berlin, Germany (Prof. Uwe Nestmann and Dr. Kirstin Peters), are envisaged by one member of the project team, estimated at 500 Euro per trip. Hence, the expected expenses for these six trips will total 3,000 Euros.

  The overall costs of visits to cooperation partners are thus 6,100 Euro.

- *Summer schools*

  Each of the two RAs working on the project will apply to a distinguished international summer school, such as the Marktoberdorf Summer School, related to the topics Formal Specification, Concurrency Theory and Automated Verification that are relevant to this proposal.

The projected costs are 2,000 Euro per RA for travel, accommodation and participation fees, thus, totalling 4,000 Euro.

In summary, the requested funds for travel are 39,500 Euro overall.

4.1.2.3 Visiting Researchers. In addition to the funds for visiting the two international project partners as explained under bullet point "*Further cooperation*" above, we request funds for one one-week visit of each of these project partners to Augsburg or Bamberg. Analogously to above, these costs total 3,100 Euro.

We also request funds for our national partner to visit Augsburg or Bamberg every six months for three days each. These six visits are estimated at 400 Euro each, totalling 2,400 Euro.

Thus, we request 5,500 Euro overall for visiting researchers.

4.1.2.4 & 4.1.2.5 Not Applicable. 4.1.2.4 Expenses for laboratory animals; 4.1.2.5 Other costs.

4.1.2.6 Project-Related Publication Expenses. 250 Euro p.a. are requested for each site, Augsburg and Bamberg, totalling 1,500 Euro over the application period.

This fund will enable us to publish in some of the increasing number of high-quality open-access journals and conference/workshop proceedings that impose moderate charges, such as those appearing in the *Leibniz International Proceedings in Informatics* (LIPIcs) and *Electronic Proceedings in Theoretical Computer Science* (EPTCS) series.

4.1.3 Instrumentation

Funds for equipment exceeding 10,000 Euro or major instrumentation exceeding 100,000 Euro are not requested.

4.2 – 4.7 Not Applicable

4.2 Module temporary position for funding; 4.3 Module replacement funding; 4.4 Module temporary clinician substitute; 4.5 Module Mercator fellows; 4.6 Module workshop funding; 4.7 Module public relations funding.

5 Project Requirements

5.1 Employment Status Information

(a) Lüttgen, Gerald, Universitätsprofessor (W3), University of Bamberg (lifelong civil servant)
(b) Vogler, Walter, Universitätsprofessor (C3), University of Augsburg (lifelong civil servant)

5.2 First-Time Proposal Data

Not applicable

5.3 Composition of the Project Group

Both applicants will be the only people working on the proposed project, who will *not* be paid out of the DFG's funds.

5.4 Cooperation with Other Researchers

5.4.1 Researchers Cooperating on this Project

Past & present cooperations. International collaborators on the original project have been Dr. Benoît Caillaud, INRIA Rennes / IRISA, France, Prof. Rance Cleaveland, Univ. Maryland, USA, Prof. Keijo Heljanko, Aalto Univ., Finland, and members of their research groups. The envisaged collaboration with Prof. Richard Paige, Univ. York, England, did not materialize due to the project's funding of only 24 instead of 36 months, which resulted in eliminating the work packages related to *Contractual Statecharts* (CSC) on which Prof. Paige had intended to cooperate.

Much input from the collaborators was gained at a two-day workshop on the topic of interface theories at the University of Augsburg on 27–28th February 2014. This workshop was attended by Dr. Caillaud and Dr. Jean-Baptiste Raclet (France), Dr. Antti Siirtola (Finland), as well as two national researchers working on interface theories – Prof. Rolf Hennicker (LMU Munich, Germany) and Prof. Alexander Knapp (Univ. Augsburg, Germany) – and the project team, i.e., Mr. Bujtor, Mr. Fendrich, Prof. Lüttgen and Prof. Vogler. The workshop included presentations on current research activities and very stimulating discussions on the topics (i) deterministic vs. nondeterministic interface theories, (ii) optimistic vs. pessimistic notions of compatibility, (iii) the importance of the quotienting operator and (iv) the complexities of interface operators.

The implementation and practical evaluation of the MIA interface theory developed within this project has recently been started at Bamberg [40] and is the focus for the approx. 6 months (full-time equivalent) that remain in the original project. In this context, Prof. Cleaveland will provide advise on various issues on tool building and, in particular, on ways how to efficiently implement the MIA refinement preorder. A visit of Prof. Cleaveland to Bamberg is currently being planned for Spring/Summer 2016.

Future cooperations. In the follow-up project proposed here, the project team at Augsburg and Bamberg will continue the collaboration with Prof. Cleaveland and initiate a new collaboration with Prof. Antti Valmari, Tampere Univ. Technology, Finland, and Prof. Uwe Nestmann, Technical Univ. Berlin, Germany:

Prof. Rance Cleaveland, Univ. Maryland, USA, is Full Professor within UMD's Department of Computer Science, is Chairman and Co-founder of Reactive Systems, Inc., and has been the Executive and Scientific Director of the U.S. Fraunhofer Center for Experimental Software Engineering between 2005 and 2015. He has collaborated with Prof. Lüttgen since 1995, including joint research on the topic of heterogeneous specification formalisms [22, 23].

Prof. Cleaveland has much experience in providing industrial-strength tool support based on firm mathematical groundings in concurrency theory and automated verification, and in conducting industrial case studies. In addition to his knowledge in implementing process-algebraic theories, which he gained as one of the chief designers and implementors of the pioneering *Concurrency Workbench* [24], he is a leading expert in automatically checking data-based systems [25, 84]. Integrating data into our MIA interface theory using shared-variable and value-passing communication is a main research objective of the follow-up project proposed here.

Prof. Antti Valmari, Tampere Univ. Technology, Finland, is Full Professor within Tampere's Department of Mathematics and is currently collaborating with Prof. Vogler on algorithmic aspects of fair testing [80].

Prof. Valmari's research concentrates on verification algorithms, methods and tools for reactive and concurrent systems, which has led to many process-algebraic semantic models and to verification-motivated work in data structures and algorithms, regarding which he is best known for his partial-order reduction methods [78]. He is an internationally recognised expert in integrating aspects of liveness and fairness in compositional, concurrency-theoretic settings [70, 80], which is a main objective of the research proposed here.

Prof. Uwe Nestmann, Techn. Univ. Berlin, Germany, is Full Professor within the Institute for Software Engineering and Theoretical Computer Science. We also expect to cooperate with Dr. Kirstin Peters who is a Senior Staff Scientist (Habilitandin) in Prof. Nestmann's research group [69].

Prof. Nestmann and Dr. Peters are leading experts in process algebras and their use for reasoning about distributed systems; in particular, they have much experience in type systems and π-calculi [63, 66]. Prof. Nestmann and Dr. Peters will advise us on how best to extend MIA with name passing, and will also assist us in investigating typing techniques employed in π-calculi and session types for our envisaged MIA-based behavioural type theory.

5.4.2 Past Collaborators

Within the past three years, the applicants have collaborated scientifically with the following national and international researchers: Prof. Blai Bonet, Univ. Simón Bolívar, Venezuela; Dr. Benoît Caillaud, INRIA Rennes / IRISA, France; Prof. Gianfranco Ciardo, Univ. Iowa, USA; Prof. Rance Cleaveland, Univ. Maryland, USA; Prof. Flavio Corradini, Univ. Camerino, Italy; Dr. Mike Dodds, Univ. York, England; Dr. Patrik Haslum, Australian National Univ., Australia; Dr. Victor Khomenko, Univ. Newcastle, England; Norman Kluge, Hasso Plattner Institut, Univ. Postdam, Germany, Dr. Jan Tobias Mühlberg, K.U. Leuven, Belgium; Dr. Richard Müller, Humboldt Univ. Berlin, Germany; Prof. Richard Paige, Univ. York, England; Prof. Frank Piessens, Univ. Leuven, Belgium; Dr. Christian Stahl, Techn. Univ. Eindhoven, The Netherlands; Prof. Sylvie Thiébaux, Australian National Univ., Australia; Prof. Antti Valmari, Tampere Univ. Technology, Finland.

5.5 Scientific Equipment

All equipment necessary for carrying out the proposed project, i.e., PCs/laptops with standard software, headsets and webcams, will be provided by the Universities of Augsburg and Bamberg.

5.6 – 5.7 Not Applicable

5.6 Project-relevant cooperation with commercial enterprises; 5.7 Project-relevant participation in commercial enterprises.

6 Additional Information

None