Combining Aspects of Reactive Systems

Leonid Kof and Bernhard Schätz

Technische Universität München, Fakultät für Informatik, Boltzmannstr. 3, D-85748 Garching bei München, Germany
{kof|schaetz}@in.tum.de

Abstract. For reactive systems, a large collection of formal models has been developed. While the formal relationship between those models is often carefully analyzed, the methodical implications for selecting or constructing appropriate models for specific application domains are rarely addressed. We classify and compare different specification methods for distributed systems concerning communication, behavior, and causality. We discuss the implications of these dimensions, especially concerning the combination of their properties.

1 Introduction

In the last thirty years, a variety of different formalisms for specifying distributed and reactive systems were introduced, like Owicki/Gries [1], CSP [2], CCS [3], UNITY [4], Esterel [5], Focus [6], to name a few. In general, each of them draws from a different foundation and is therefore exhibiting its own strengths and weaknesses. As a consequence, often pragmatic description formalisms like Statecharts end up with a large set of formal models [7] differing in essential aspects. However, for the engineer it is not always obvious which model to select for a specific application domain.

In this paper we identify three essential aspects of those formalisms by stripping away the more technical details:

- models of **communication** and **compositionality** as a related issue
- model of **behavior** and **hiding internal structure** as a related issue
- models of **causality** and **action refinement** as a related issue,

each aspect offering different variations to choose from. By classifying formalisms accordingly, we show that these variations can be chosen independently. More importantly, there also is a methodological dimension to identify combinations that are useful concerning the modeling of reactive systems. In Section 2 we introduce our classification dimensions and different characteristics of each dimension. In Section 3 we start off with the classification of some prominent formalisms and sketch how less prominent combination could look like. Furthermore, we discuss methodical implications. Finally, in Section 4 we sum up the results of the previous sections.

Development Step	Modeling Aspect	Abstraction
Composition ('Observations about components still hold after combing them')	**Communication** ('How does the behavior of one component influence the behavior of others?')	Interference by environment
Modularization ('Observations about components still hold after hiding internal structure')	**Behavior** ('How are observations combined to describe a behavior?')	Scheduling of actions
Action Refinement ('Relation between actions still hold after refining actions')	**Timing** ('How are actions combined to describe an observation?')	Delays between actions

Table 1. Aspects of Reactive Systems

2 Classification

In the following subsections we consider three different aspects of reactive systems: *communication*, *behavior*, and *causality*. Since in the following we compare those models, we give a short informal list of those common concepts that are used for comparison:
A *component* is a unit of a system capsuling a state and supplying an interface. The *interface* of a component describes the part of the component which can be accessed by other components or the environment, e.g., by means of communication. *Behavior* relates components to (sets of) observations. Using *composition*, components are combined into a new component. An *observation* is a set of events of a component related by a causality relation, describing some form of sequence of actions. An *event* is an observable interaction occurring at the interface of a component, e.g. the communication of a message. *Causality* defines a relation between events of a component, inducing observations in form of executions. By adding *time*, it is possible to describe behavior which is influenced by the fact that no communication takes place by modeling time has passed without communication. *Refinement* relates different behaviors; e.g., behavioral refinement (relating a behavior of a component to a more restricted form of behavior) and structural refinement (relating a component to a network of subcomponents).

As shown in Table 1, the above aspects are related to principles of system description: **Communication** is related to the *compositionality* of a model. **Behavior** is related to the *abstraction* from implementation aspects (e.g., continuity) simplifying the description of a system behavior (e.g., modeling fairness). **Causality** is related to *refining* the interaction of a system, e.g. when abstracting from internal structure or when breaking up an atomic interaction.

For each of these aspects, we describe different variation classes of models. Each variation describes a different level of abstraction from concrete implementations as found in models of reactive systems. The corresponding classes are ordered concerning their capability to support these aspects. For sake of brevity,

we only distinguish three classes in each aspect - of course, when considering fine-grained mathematical classifications more complex orders are needed, as, e.g., [8] shows for behavioral models. Since however we focus here on the methodical principles behind these formalisms we restrict these aspects to basic classes and concentrate on the aspect of combining them.

2.1 Modes of Communication

The aspect of communication deals with modes of synchronization between reactive systems, including the ways of exchanging information. As mentioned above, the corresponding methodical aspect is the issue of compositionality, i.e., the capability to deduce the observations of a composed system from the observations of its components. In this dimension, we consider the following range:

Implicit communication: This mode of communication corresponds to implicit communication, e.g. by using (undirected) shared variables. Since no explicit communication mechanism is used, the environment can change the (shared) variables unnoticed by the component. Therefore, compositionality is dependent on the behavior of the environment. This approach is used, e.g., in co-routine approaches like [1] or UNITY [4].

Explicit event based synchronous communication: While this model offers an explicit communication mechanism, synchronization is undirected (there is no designated sender and receiver); synchronization between components takes place by the components agreeing on an event they are all ready to accept. Therefore, when composing components, in this model the possibility of blocking has to be considered. This model of communication is found in TCSP [2] or CCS [3].

Explicit message based asynchronous communication: In this model there is an explicit distinction between sender and receiver; furthermore, the receiver of a message is *input enabled*, i.e. always ready to accept a message. Examples for this model are semantics for asynchronous circuits like [9], reactive modules [10], or stream processing functions like [6].

Note that increasing modularity is related to increasing abstraction from restrictions concerning compositionality: from compositionality with respect to freedom of interference (implicit), via compositionality with respect to deadlock (synchronous), to unrestricted compositionality (asynchronous). The corresponding refinement steps successively add these restrictions (cf., [11]).

2.2 Types of Behavior Modeling

The behavioral aspect focuses on how observations are combined to form a description of the behavior of a system. In denotational models (e.g., traces [9], failures [2], or stream processing functions [6]), observations are defined by the elements of the domain; in algebraic ([12]) or operational models (e.g., CCS [3], I/O-Automata [13]) observations are defined in terms of states and possible transitions from these states. Since these formalisms describe (potentially)

nonterminating systems, infinite observations are included in the behavior of a system. Accordingly, this aspect is related to the abstraction from an operational view of the system including issues like explicit parallelism/true concurrency of events as well as fairness of observations. Consequently, depending on the level of abstraction supplied by the model, phenomena like divergence in TCSP [2] can occur. Combining the issues of fairness and explicit parallelism, we obtain the following range:

Finite: These models essentially support only the description of finite behavior; they do not include a distinction between arbitrary sequentialization and parallel execution. No notion of fairness is supported; divergence can occur in these models. Examples are TCSP [2] or receptive processes [14].

Weak Fairness: While this model also only supports admissible behavior for sequential executions, it also supports explicit parallelism or weak fairness avoiding the treatment of divergence. Examples for this model are continuous stream processing functions [6] or CCS [3].

General Fairness: These systems support general non-admissible behavior or fairness. Examples are TLA [15], or general trace-based descriptions [12].

The methodical aspect of this dimension is the increasing abstraction from scheduling details: from scheduling parallel systems analogously to sequential ones (finite), via a fair scheduling of parallel systems (weak), to a fair scheduling independent of the kind of systems (general). The corresponding refinement steps ensure these implicit fairness assumptions.

2.3 Causality and Time Modeling

Models of causality describe possible relations between the (inter)actions of a component or system. Since those relations describe the unfolding of the communication and behavioral actions over the time, those models always contain some (explicit or implicit) aspect of time. The causality relation should allow for *structural refinement*, i.e., we should be able to replace an abstract component by a network of sub-components often requires augmentation of the causality relation by internal actions of the network. Concerning causality we obtain the following range of this dimension:

Metric Models: These models introduce an explicit labeling of events with (real-valued) time stamps. Non-operational properties like Zeno behavior[1] can arise. Examples for this model are Timed CSP, Timed or Densely-Timed Focus.

Strict Sequentialization: Here, a linear order of events is imposed, making absence of events implicit or explicit. Accordingly, the model excludes causal loops and either imposes an interleaving semantics or introduces implicit timing constraints. Examples for this model are trace-based (e.g., asynchronous circuits [9]) or state-based history semantics (TLA [15]).

[1] By Zeno-behavior we characterize those models that do not exclude the occurrence of infinitely many (inter)action in a finite amount of time; see, e.g., [16].

Unrestricted Causality: Here, no restrictions are imposed on the interaction relation of a system, since an input received by the system can immediately stimulate the production of some output without any delay (*perfect synchrony hypothesis*). These model is used in Esterel [5] or some Statecharts variants [7].

Note that this dimension in ordered with respect to abstraction from timing aspects: from metric models explicitly dealing with real time, via sequentialization abstracting from the passing of time between events, to unrestricted causality abstracting from the introduction of delays. The corresponding refinement steps introduce explicit delays or durations (cf., [17]).

3 Putting Different Classifications Together

To illustrate the classification, we classify formalisms as shown in Table 2: TCSP [2], CCS [3], I/O-Automata [13], Asynchronous Traces [9] Esterel [5], Focus [6], TLA [15], Co-routines [1], fair process algebra [18] and Receptive Processes [14]. We also consider timed variants of some formalisms: Timed CSP [19], Timed Focus[6] and Timed TLA [20]. In the following, we consider Esterel, Focus, and TCSP more thoroughly to clarify the classification principles. The rest of the classification is treated analogously. Then the address the issue of combing aspects to from specific variants of models of reactive systems.

3.1 Classification of Existing Techniques

To clarify the construction of Table 2, we use Esterel, Focus, and TCSP for illustration.

Esterel uses signals to describe system states. An execution of a system consists of a sequence of computing rounds. Within each round all the signals (inputs *and* outputs) are considered as perfectly synchronous, leading to a "unrestricted causal" model. Furthermore, communication is "message asynchronous", as the sender does not have to wait for the receiver. As Esterel is a state transition system with (possibly) infinite runs and without any explicit fairness conditions, it belongs to the class of weak fair systems. *Focus* defines each component as a stream processing function. Such a stream implies a linear order on messages, so we classify the causality modeling as "strict sequentialization". Message delivery is not influenced by the receiver and therefore the communication mode is asynchronous. As Focus includes infinite histories of interaction to describe the behavior of a system, it is classified as a weak fair system. In *TCSP* message exchange is realized by general synchronizing events; thus it is classified as using synchronous event-based communication. Since additionally TCSP considers only finite runs, it is classified as finite.

3.2 Combining Aspects

When combining aspects to form a specific modeling formalism, the question arises whether these combinations lead to reasonable models. Therefore, from a

Behavior	Communication	Causality		
		Metric	Sequential	Unrestricted
Finite	Implicit		Co-routines	
	Synchronous	Timed CSP	TCSP	
	Asynchronous		Receptive Processes	
Weak	Implicit			
	Synchronous	Timed CCS	CCS	
	Asynchronous		Focus, Traces	Esterel
General	Implicit	Timed TLA	TLA	
	Synchronous		Fair Process Alg.	
	Asynchronous	Timed Focus	I/O-Automata	

Table 2. Classification according to communication, behavior, and causality modeling.

technical point of view we are interested in independent dimensions, supporting arbitrary combinations. From a methodical point of view, we are furthermore interested in selecting the right class for each dimension for a particular application domain. As each dimension is ordered according to its degree of abstraction, the selection depends on the levels of abstraction needed in the application domain.

To show the independence of the dimensions, each possible combination is investigated; For example, for 'finite/asynchronous/real-time', we can derive a timed variant from the receptive processes along the lines of Timed CSP. For 'weak/asynchronous/real-time', suitable variants of Statecharts can be identified. Appropriate real-time variants are discussed, among others, in the Statecharts classification of [7]. For 'general/synchronous/real-time', timed variants of fair process algebras can be used.

The methodical aspect of combinations is especially relevant if several models are combined in a development approach to support views of a system with decreasing levels of abstraction. While an abstract, service-oriented view of embedded hardware is supported best by a model including asynchronous communication, general fairness, and unlimited causality, a low-level, task-oriented view may require synchronous communication, weak fairness, and metric time. Here, the ordering within the dimensions can help to find appropriate models and corresponding refinement steps stating which properties must be considered explicitly (e.g., buffer sizes when moving from asynchronous to synchronous communication, causal loops when moving from unrestricted causality to sequentialization).

4 Conclusion

The introduced classification scheme supports the analysis of reactive models from a methodical as well as a technical point of view. Such classification is useful when selecting a formalism best suited to capture the aspects of systems for a given application domain. Technically, the classification helps by structuring the choice; methodically, the classification helps by addressing the strengths

and weaknesses arising from certain combinations. The independence of these aspects help when 'switching on and off' of certain properties to produce possible formalisms. The ordering within the domains helps to support the relation between different models arranged in a development process.

References

1. Owicki, S., Gries, D.: An Axiomatic Proof Technique for Parallel Programs. Acta Informatica **14** (1976)
2. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall International (1985)
3. Milner, R.: Communication and Concurrency. Series in Computer Science. Prentice Hall (1989)
4. Chandy, K.M., Misra, J.: Parallel Program Design - A Foundation. 2 edn. Addison-Wesley (1989)
5. Berry, G.: The Esterel v5 Language Primer. Technical report, INRIA (2000) http://www-sop.inria.fr/meije/esterel/esterel-eng.html; accessed August 19, 2002.
6. Broy, M., Stølen, K.: Specification and Development of Interactive Systems: FOCUS on Streams, Interfaces, and Refinement. Springer (2001) Texts and Monographs in Computer Science.
7. von der Beeck, M.: Comparison of Statecharts Variants. In: Proceedings of FTRTFT94. (1995) LNCS 863.
8. van Glabbeek, R.J.H.: Comparative concurrency semantics and refinement of actions. Technical Report 109, Centrum voor Wiskunden en Informatica (1996) CWI Tracts.
9. Dill, D.L.: Trace Theory for Automatic Hierarchical Verification of Speed Independent Circuits. ACM Distinguished Dissertations. The MIT Press (1989)
10. Alur, R., Henzinger, T.A.: Reactive modules. Formal Methods in System Design: An International Journal **15** (1999) 7–48
11. Schätz, B.: Ein methodischer Übergang von asynchronen zu synchronen Systemen. PhD thesis, Technische Universität München (1998)
12. Bergstra, J.A., Ponse, A., Smolka, S.A.: Handbook of Process Algebra. Elsevier (2001)
13. Lynch, N., Tuttle, M.: An Introduction to Input/Output Automata. CWI Quarterly **2** (1989) 219–246
14. Josephs, M.B.: Receptive process theory. Acta Informatica **29** (1992) 17–31
15. Lamport, L.: Verification and Specification of Concurrent Programs. In Bakker, J., Roever, W.P., G.Rozenberg, eds.: A Decade of Concurrency - Reflexions and Perspectives, Springer Verlag (1993) 347–374 LNCS 803.
16. Shields, M.W.: Semantics of Parallelism. Springer (1997)
17. Scholz, P.: Design of reactive Systgems and their Distributed Implementaion with Statecharts. PhD thesis, Technische Universität München (1998)
18. Parrow, J.: Fairness properties in process algebra with applications in communication protocol verification. PhD thesis, Uppsala University (1985)
19. Davis, J., Schneider, S.: An Introduction to Timed CSP. PRG- 75, PRG Programming Research Group Oxford (1989)
20. Abadi, M., Lamport, L.: An old-fashioned recipe for real time. ACM Transactions on Programming Languages and Systems **16** (1994) 1543–1571