Chapter 2

Mandatory and Potential Choice: Comparing Event-B and STAIRS

Atle Refsdal
SINTEF ICT, Norway

Ragnhild Kobro Runde
University of Oslo, Norway

Ketil Stølen
SINTEF ICT, Norway and University of Oslo, Norway

Abstract. In order to decide whether a software system fulfills a specification, or whether a detailed specification preserves the properties of a more abstract specification, we need an understanding of what it means for one specification to fulfill another specification. This is particularly important when the specification contains one or more operators for expressing choice. Operators for choice have been studied for more than three decades within the field of formal methods in general, and within methods for action-refinement in particular. In this paper we to focus on Event-B, a more recent method for action refinement. The STAIRS method belongs to another tradition. It originates from the UML community and is designed to provide an understanding of refinement and fulfillment for UML. STAIRS distinguishes between potential and mandatory choice, where only the latter is required to by preserved by refinement. This paper investigates the relationship between the operators for choice in Event-B and STAIRS.

2.1 Introduction

In order to decide whether a software system fulfills a specification, we need a clear understanding of the concept of fulfillment. Similarly, when a specification is developed further into a new more detailed (for example, platform-specific) specification, the essential properties captured by the original specification must still be present in the new specification. This requires an understanding of what it means for one specification to fulfill another specification.

STAIRS [156, 290] was designed to provide the UML community with this kind of understanding at a level of abstraction that is easily comprehensible for UML practitioners. STAIRS is inspired by formal methods and refinement theory. However, STAIRS is not really a formal method in the classical sense, as explained in the following. When formal methods are combined with more applied methods for software engineering, the resulting approaches may typically be classified according to whether:

- Artifacts of the applied method, typically specifications and models, are translated into the formal method and used for formal analysis.
- Artifacts of the applied method, again normally specifications and models, are annotated with formal expressions and used for formal analysis building on some unified underlying semantics.

STAIRS does not fit within this classification scheme since the emphasis of STAIRS is to provide a foundation for fulfillment within the conceptual universe of UML rather than supporting formal analysis of UML specifications and their relationships.

STAIRS addresses primarily sequence diagrams. Implicitely, STAIRS also defines the notion of fulfillment for the other UML notations for modeling dynamic behavior, where the behaviour may be captured by sets of sequence diagrams. In many respects, sequence diagrams are more general than other UML notations for dynamic behavior, e.g., state machines, because sequence diagrams may be used to describe examples of required behavior, rather than the complete allowed behavior. STAIRS provides this expressiveness by offering operators for potential as well as mandatory choice.

Operators for choice have been studied for more than three decades within the field of formal methods in general, and within methods for action-refinement in particular [32]. A prominent example of a method for action-refinement is Event-B [3]. The objective of this paper is to investigate the relationship between the operators for choice in Event-B and STAIRS.

A large literature exists on Event-B. There is no fixed semantics for Event-B, instead the semantics is provided implicitly by proof obligations associated with a model [152]. Nevertheless, several papers have suggested failure-divergences inspired semantics as a formal underpinning [295, 71, 307]. This paper builds on this approach. Failure-divergences semantics was originally developed for CSP [172]. In Section 2.2 we therefore start our investigation by comparing choice in CSP to choice in STAIRS. Then we conduct a comparison of Event-B and STAIRS; first at the syntactic level in Section 2.3; then at the semantic level in Section 2.4. Finally, Section 2.5 provides a summary and draws conclusions.

Preserved by refinement	**CSP**		**STAIRS**
	environment	**system**	
No		Internal choice Demonic choice	Potential choice
Yes	External choice Angelic choice		Mandatory choice

Table 2.1: Choice types in CSP and STAIRS

2.2 Kinds of Choice

In this section, we relate the kinds of choice offered by CSP [172] and STAIRS [156]. We also classify the kinds of properties that may or may not be captured depending on the available choice operators.

In order to understand the choice operators in CSP we need to understand some underlying assumptions about the involved entities, as well as the communication model. As explained by [286, p. 13], in CSP a system[1] is completely described by the way it can communicate with its environment. Hence, CSP assumes a black-box view where internal communication within the system itself is hidden. Communication is synchronous (also known as handshake communication), meaning that "events only happen when both sides agree" [286, p. 9]. This can be understood as follows: At any given point the system offers a set of events to the environment. If the environment accepts one of these events then the system moves on, otherwise a deadlock occurs.

Choices made by the environment between available alternatives are called *external* choices and represented by the $\Box$ operator in CSP, while choices made by the system are called *internal*[2] and represented by the $\sqcap$ operator. If one of the alternatives offered to the environment is removed, a deadlock will be introduced if the environment is willing to synchronize only on the removed alternative. Refinement in CSP therefore requires preservation of external choice. Internal choice, on the other hand, represents underspecification and may be reduced in a valid refinement step, as motivated by the following quote [172, p. 101–102]:

> Sometimes a process has a range of possible behaviours, but the environment of the process does not have any ability to influence or even observe the selection between the alternatives [...] The choice is made, as it were internally, by the machine itself, in an arbitrary or nondeterministic fashion [...]
>
> There is nothing mysterious about this kind of nondeterminism: it arises from a deliberate decision to ignore the factor which influence the selection [...] Thus nondeterminism is useful for maintaining a high level of abstraction in descriptions of the behaviour of physical systems and machines [...]
>
> A process specified as $(P \sqcap Q)$ can be implemented either by building P or by building Q. The choice can be made in advance by the implementor on grounds not relevant (and deliberately ignored) in the specification [...]

[1] The CSP literature typically uses the term "process".
[2] Hoare uses the term "nondeterministic or" or just "nondeterminism" for internal choice.

The term *angelic* choice (or angelic nondeterminism) is sometimes used to describe a choice that will always be made so that an undesirable result (a deadlock) is avoided if possible. Hoare explains this in terms of an implementation that, when choosing between P and Q, "minimises the risk of deadlock by delaying the choice until the environment makes it, and then selecting whichever of P and Q does *not* deadlock" [172, p. 105]. Similarly, Roscoe explains angelic choice in terms of an operator that "keeps on giving the environment the choice of action of P and Q as long as the environment picks an event they both offer" [287, p. 219]. Hence, angelic choice is a special kind of external choice. Conversely, although not used in the above references, the term *demonic choice* can be used to describe an internal choice that will (or at least can) be made so that a deadlock will occur, if possible. In [250], Morgan et al. use the terms demonic choice and internal choice interchangeably.

The two middle columns of Table 2.1 summarize the above discussion. The system column represents choices resolved by the specified system, while the environment column represents choices resolved by its environment. The column furthest to the left indicates whether choices are preserved by refinement.

According to [156], STAIRS is an approach to compositional development of UML interactions that assigns a precise interpretation to the various steps in incremental system development based on an approach to refinement known from the field of formal methods. There are a couple of ways in which STAIRS differs from CSP of immediate relevance for our discussion of choice here. First, STAIRS assumes an asynchronous communication model with infinite buffering, and is therefore not concerned with deadlock. Second, in STAIRS there is no implicit hiding of internal communication when composing specifications.

STAIRS offers two different choice operators: one for *potential* choice and one for *mandatory* choice. Along the same lines as internal choice in CSP, potential choice is motivated by the need for abstraction. This is explained by the following requirement to STAIRS stated in [156]:

> Should allow specification of potential behavior. Underspecification is a well-known feature of abstraction. In the context of interactions, "under-specification" means specifying several behaviors, each representing a potential alternative serving the same purpose, and that fulfilling only some of them (more than zero but not all) is acceptable for an implementation to be correct.

Mandatory choice, on the other hand, is motivated as follows:

> Should allow specification of mandatory behavior [...] Sometimes [...] it is essential to retain non-determinism in the implementation reflecting choice. For example, in a lottery, it is critical that every lottery ticket has the possibility to win the prizes [...] As a consequence, we need to distinguish explicit non-determinism capturing mandatory behavior from non-determinism expressing potential behavior.

Since potential choice facilitates underspecification by offering alternatives serving the same purpose, STAIRS allows potential choice to be reduced or removed by refinement. A mandatory choice, on the other hand, needs to be preserved in order to ensure that all intended behavior will be implemented. This applies regardless of whether the choice is made by the system or by the environment. The distinction between potential and mandatory choice in STAIRS is summarized in the right-hand column of Table 2.1.

Another kind of choice is probabilistic choice, meaning that each alternative should be selected according to a given probability. Probabilistic choice is beyond the scope of this

paper and has therefore not been included in Table 2.1. However, mandatory choice (as understood in STAIRS) can be understood as probabilistic choice where all of the probabilities should be higher than 0, but where nothing more is known/specificed about the probabilities.

The constructs for expressing choice offered by a specification language and its notion of refinement restrict the kinds of properties that can be captured. System properties are typically analyzed on the basis of system traces, each of which characterizes a possible run or execution. Properties can then be classified according to their means of falsification. Properties that can be falsified by a tester on the basis of a single trace are called *trace properties*, while properties that can be falsified on the basis of trace sets are called *trace set* properties [243]. The former include safety and liveness as originally investigated by Alpern and Schneider [14, 294]. The latter include information security flow properties and are what McLean referred to as possibilistic properties [243].

As an example, assume we want to specify a simulator to simulate user behavior for automatic testing of vending machines offering tea and coffee. The simulator should then be able to choose both alternatives, and the choice should be made internally by the simulator (thus reflecting a user's preference) rather than by its environment. Before using the simulator to automatically test a vending machine, we need to test the simulator itself. When testing the simulator, if we observe a single trace yielding tea, we cannot deduce that the simulator is not able to choose coffee or vice versa; such falsifications can only be made by considering all traces of the system.

Specification approaches allowing all choices made internally by the specified system (as opposed to its environment) to be reduced by refinement, have no means to ensure that trace set properties are preserved. This is referred to as the refinement paradox in [200]. In the following, we discuss the syntax and semantics of choice in Event-B and STAIRS in the light of trace properties and trace set properties.

2.3 Comparing Event-B and STAIRS at the Syntactic Level

The essence of an Event-B specification is a set of guarded events, where an event is enabled and may be chosen to occur when its guard is true. More than one event may be enabled at the same time, and the choice between enabled events is an external choice made by the environment. Internal choice made by the system itself is modeled more indirectly, using nondeterministic assignment to internal variables in order to influence the enabledness of other events.

As an example of how choice is treated in Event-B, in Fig. 2.1 we look at the two vending machine specifications given by Butler in [71]. An Event-B specification consists of a specification name, a declaration and initialization of variables and a set of named events. Each event is on the form **when** *guard* **then** *body* **end**, where the guard is a boolean statement over the variables and the body is a (possibly non-deterministic) variable assignment. An event is said to be enabled if its guard evaluates to true, otherwise it is disabled.

The difference between the two specifications in Fig. 2.1 is that in $VM1$, the internal variable $m1$ is set to *vend* after the `Coin` event has been executed, thus enabling both the `Tea` and the `Coffee` event, while in $VM2$, the internal variable $m2$ is set to either *tea* or *coffee*, thus enabling only one of `Tea` and `Coffee`. This means that in $VM1$, the choice

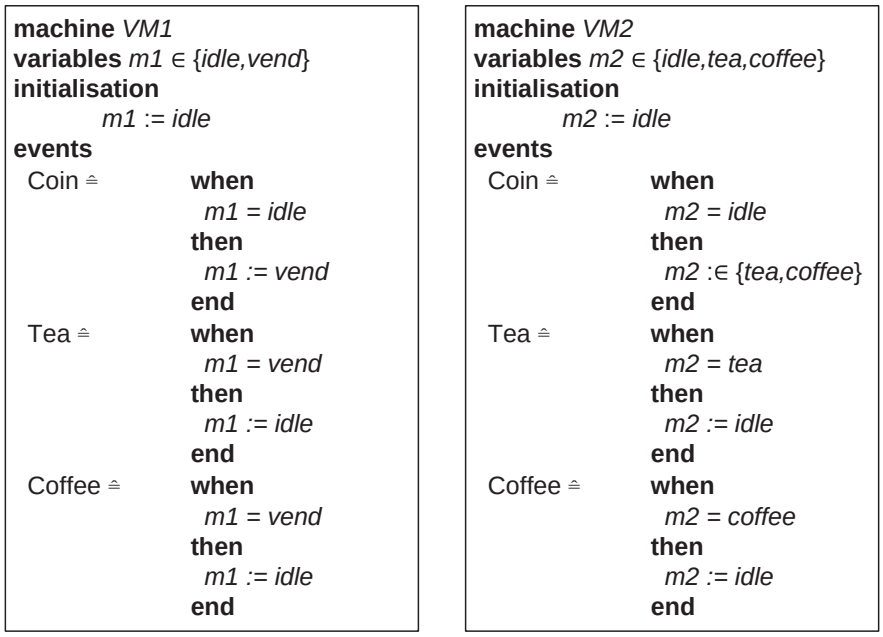

Figure 2.1: Two Event-B vending machines as specified by [71]

between `Tea` and `Coffee` is to be made by the environment, and is thus an example of external choice. In [71], it is argued that from a customer's point of view, this external choice should be preserved by refinement, meaning that $VM2$ should not be a valid refinement of $VM1$. This could be achieved for instance by requiring that a refinement should preserve the enabledness of individual events.

In $VM2$, the choice between `Tea` and `Coffee` is made by the machine itself, and this is an example of internal choice. An internal choice may be refined by an external choice, as this ensures that all events enabled in the original machine will also be enabled in the refined one. Consequently, $VM1$ should be a valid refinement of $VM2$.

As an example of potential choice in STAIRS, Fig. 2.2 gives a sequence diagram specification of a vending machine with messages that correspond to the events in $VM1$ and $VM2$ from Fig. 2.1. The main ingredients of a sequence diagram are a set of lifelines (depicted as vertical lines) and a number of messages (arrows) between the lifelines. In Fig. 2.2, the choice operator alt is used to signify that this diagram specifies two example scenarios, both starting with the vending machine receiving a coin from the environment, followed by the vending machine providing tea in one scenario, coffee in the other. As alt is used to model potential choice, a sequence diagram where only one of these scenarios is positive, and the other one is specified as negative, would be a valid refinement of `Vending Machine 1`.

In STAIRS, there is no fundamental distinction between internal and external choice. The choice between sending `Tea` or `Coffee` in Fig. 2.2 is an internal choice when seen from the sending lifeline `VM`, and an external choice when seen from the receiving lifeline `Env`. For a real vending machine, the choice between tea and coffee would be made by the user. In STAIRS, this may be modeled for instance by selection messages from `Env` to `VM` as seen in

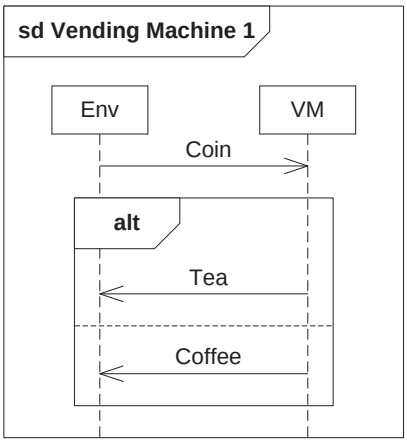

Figure 2.2: A simple vending machine with potential choice in STAIRS

Fig. 2.3. Note, however, that the choice between the two alternatives is still specified using alt, meaning that in a valid implementation, only one of the tea and coffee scenarios may be present.

Back to Event-B, Butler [71] argues that experience with Event-B modeling has demonstrated the need to be able to model both internal and external choice between enabled events more directly, in particular in situations where the guards are equal only as a result of abstraction. In reality, such a choice is not really external, but rather internal due to some condition not included in the abstract specification.

For instance, the choice between `Tea` and `Coffee` in $VM1$ in Fig. 2.1 should in some cases be seen as internal due to some condition abstracted away in $VM1$. A refinement may for instance add internal variables and guards so that coffee is always served in the morning, while tea is always served in the afternoon.

In [71], the main goal is to allow both external and internal choice to be represented directly. This is achieved by letting the specifier divide the events into groups. The intuitive interpretation is that a choice between groups of events is external, while a choice between events within a group is internal. For $VM1$ in Fig. 2.1, the specifier may state that the choice between `Tea` and `Coffee` is internal by grouping them together, giving the following event groups for $VM1$: $G_1 = \{\texttt{Coin}\}$, $G_2 = \{\texttt{Tea}, \texttt{Coffee}\}$.

In [71], the refinement relation is modified so that preservation of enabledness is preserved for event groups rather than for single events, thus ensuring that external choices are preserved (or increased) through refinement while at the same time allowing the amount of internal choice to be reduced. With the event groups G_1 and G_2 given above, this would mean that a valid refinement of $VM1$ may choose to offer only `Coffee` (or `Tea`).

Mandatory choice is not discussed in [71], and the introduction of event groups is not sufficient to capture system choices that must be preserved by refinement, i.e., trace set properties. Assume, for illustration purposes, that the specifier wants to model a machine which arbitrarily chooses between tea and coffee at run-time (similar to the user simulator described in Section 2.2). As putting `Tea` and `Coffee` in the same event group might lead to an implementation offering only one of them as seen above, the only other possibility is

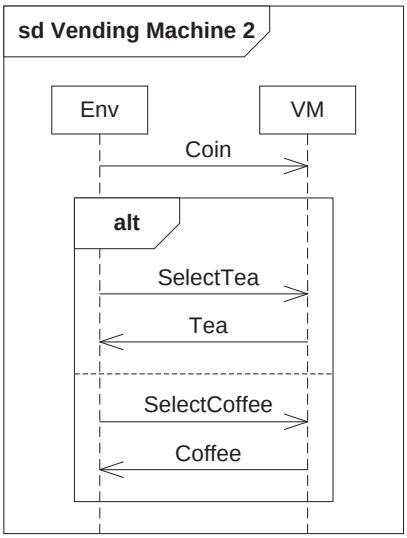

Figure 2.3: A simple vending machine with external choice in STAIRS

to put them in separate event groups. However, the semantics of such a specification would allow `Tea` (and similarly, `Coffee`) to be refused only when its guard is false, meaning that neither `Tea` nor `Coffee` could be refused after `Coin`, so that the choice between the two remains external and not internal.

A vending machine where the internal choice is made arbitrarily as described above, may be modeled in STAIRS by using the mandatory choice operator xalt as shown in Fig. 2.4. To simplify the main diagram `Vending Machine 3`, the diagram refers to two sub-diagrams `Provide tea` (also provided in Fig. 2.4) and `Provide coffee` (not shown, but symmetrical to `Provide tea`). The refuse operator is used to model that a specific alternative should be considered negative, e.g., in `Provide tea` the vending machine should serve tea and not coffee. The main diagram `Vending Machine 3` then requires the vending machine to have two mandatory behaviors, one with tea and not coffee, and one with coffee and not tea. Neither of these can be removed by refinement. Also, the mandatory choice in `Vending Machine 3` in Fig. 2.4 is a valid refinement of the potential choice in `Vending Machine 1` in Fig. 2.2, as should be expected. Further refinements may increase the mandatory behavior required by adding more xalt-operands, e.g., a third alternative providing chocolate but not tea or coffee.

2.4 Interaction-Obligations versus Failure-Divergences

In the previous section, we compared Event-B and STAIRS at the syntactic level. Semantically, an Event-B specification may be represented by a failure-divergences pair while a sequence diagram in STAIRS corresponds to a set of interaction-obligations. If the sequence

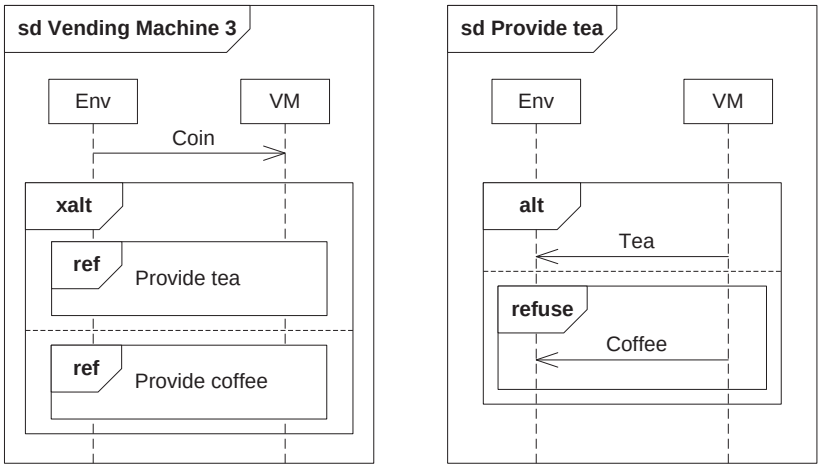

Figure 2.4: Mandatory choice in STAIRS

diagram does not contain mandatory choice, a single interaction-obligation is sufficient. In the following, we outline the intuition behind interaction-obligations and failure-divergences and how they are related. We also explain how mandatory choice is represented semantically and discuss the relationship to external choice.

2.4.1 Interaction-Obligations

A trace in STAIRS is a finite or infinite sequence of events where an event is either the sending or the reception of a message. A trace is required to fulfill certain well-formedness conditions [289]. Informally, a trace is well-formed if, for each message, the send event is ordered before the corresponding receive event.

Let $\mathcal{H}$ denote the set of all well-formed traces. An interaction-obligation is a pair (p, n) of trace-sets which classifies the elements of $\mathcal{H}$ into three categories: the positive traces p, the negative traces n, and the inconclusive traces $\mathcal{H} \backslash (p \cup n)$. The inconclusive traces are those traces that are neither specified as positive nor as negative by the sequence diagram in question.

A pre-post specification $(pre, post)$ in Hoare-logic may be used as a first approximation of the intuition behind an interaction-obligation (p, n). Roughly speaking:

- A positive trace (in p) corresponds to an execution initiated in a state fulfilling *pre* that if it terminates, does so in state fulfilling *post* (given Hoare-logic for partial correctness).
- An inconclusive trace (in $\mathcal{H} \backslash (p \cup n)$) corresponds to an execution initiated in a state fulfilling $\neg pre$.
- A negative trace (in n) corresponds to an execution initiated in a state fulfilling *pre* that terminates in state fulfilling $\neg post$.

It is worth noticing that while the inconclusive behavior corresponding to a pre-post specification is chaotic, meaning that anything is allowed, the inconclusive behavior of an

interaction-obligation is not necessarily so. For example, if the finite trace t is inconclusive then the result $t \frown t'$ of extending t with t' is not necessarily inconclusive; $t \frown t'$ may be positive ($t \notin p \cup n \wedge t \frown t' \in p$) and another extension t'' may be negative ($t \notin p \cup n \wedge t \frown t'' \in n$). In fact, an interaction-obligation may for the same environment behavior allow inconclusive, positive as well as negative behavior. A pre-post specification on the other hand, classifies executions initiated in a state (in a pre-post setting, representing the environment behavior) as either positive or negative if it fulfills *pre* and as inconclusive otherwise.

It is also worth mentioning that for interaction-obligations as for pre-post specifications, there may be environment behaviors for which no behavior is allowed. An example of a pre-post specification of this kind is

$$(\mathit{true} \ , \ x = 0 \Rightarrow \mathit{false})$$

It disallows any behavior for the initial state $x = 0$. In classical Hoare-logic, such a specification is not implementable because any real program has some kind of behavior whatever the environment does. In other words, no real program is partial. Hence, any implementable pre-post specification is total; it allows at least one system behavior for each possible initial state. The same is not true for interaction-obligations because sequence diagrams only specify example runs and not the full behavior of a real program. Hence, a sequence diagram only considering some input behaviors is unproblematic from a methodological point of view.

In Hoare-logic, refinement corresponds to weakening the pre-condition and strengthening the post-condition. Refinement of an interaction-obligation corresponds to reducing inconclusive behavior and redefining positive behavior as negative. Formally:

Definition 2.1. *An interaction-obligation* (p', n') *refines an interaction-obligation* (p, n) *if* $p \subseteq p' \cup n'$ *and* $n \subseteq n'$.

Given the mapping to pre-post specifications outlined above, reducing inconclusive behavior may be understood as weakening the pre-condition; redefining positive behavior as negative may be understood as strengthening the post-condition. Hence, refinement of interaction-obligations reflects very well refinement of pre-post specifications.

2.4.2 Failure-Divergences

In the setting of Event-B, a specification may be described by a pair (f, d) of a set of failures f and a set of divergences d. A *failure* is a pair (t, X) of a finite trace t and a set of events X that the specified system may refuse after having engaged in the external interaction corresponding to t. In other words, the specified system may deadlock after having engaged in t if offered only X or a subset of X by the environment. A *divergence* is a trace t after which the systems may diverge, meaning that it performs an infinite unbroken sequence of internal (and hence invisible) actions without any external communication happening at all, also referred to as livelock. Well-formedness constraints [172, p. 130] are imposed on failure-divergence pairs that imply that any such pair is total; it allows some behavior (possibly consisting of doing nothing) whatever the environment does.

Failures-divergences refinement corresponds to removing failures and divergences. This means set inclusion with respect to the failures and the divergences. Formally:

Definition 2.2. *A failures-divergences pair* (f', d') *refines a failures-divergences pair* (f, d) *iff* $f' \subseteq f$ *and* $d' \subseteq d$.

External choice cannot be reduced by refinement, as this would imply adding new failures in order to allow the specified system to refuse some of the events offered to the environment according to the more abstract specification.

2.4.3 Relating the Two Models

In the case of total correctness the relationship between an Event-B specification captured by (f, d) and a sequence diagram captured by the interaction-obligation (p, n) may be characterized as follows:

- The positive behavior p corresponds to $e \backslash d$, where $e = \{t \mid (t, \varnothing) \in f\}$.
- The inconclusive behavior $\mathcal{H} \backslash (p \cup n)$ corresponds to d.
- The negative behavior n corresponds to $\mathcal{H} \backslash (e \cup d)$.

Given the mapping above, reducing inconclusive behavior in STAIRS may be understood as reducing the set of divergences, while redefine positive behavior as negative in STAIRS may be understood as reducing the set of traces that are not divergences. This mapping is not information preserving since semantically different failure-divergences are mapped to the same interaction-obligation. In particular, the semantic difference between external and internal choice disappears.

Contrary to a failure-divergences pair, an interaction-obligation may be partial in the sense that there may exist environment behavior for which no positive system behavior is defined. It may be argued that a refinement should not impose additional constraints on the environment behavior and thereby increase partiality. Although this constraint may easily be imposed, it is not enforced by STAIRS because there are situations where this is not very practical. We may for example use the operator for potential choice to specify two different protocols for interaction with the environment and then leave it to the implementor to select which one to use. This choice will also restrict (or impose additional assumptions about) the behavior of the environment because the specified system will only work properly if the environment sticks to the selected protocol.

2.4.4 Sets of Interaction-Obligations

Sequence diagrams with mandatory choice is in STAIRS represented by a set of interaction-obligations. Informally speaking, each interaction-obligation represents an alternative that must be reflected in any correct implementation. In the most general case, refinement corresponds to:

Definition 2.3. *A set of interaction-obligations o' refines a set of interaction-obligations o if for each interaction-obligation $(p, n) \in o$ there is an interaction-obligation $(p', n') \in o'$ such that (p', n') refines (p, n).*

Hence, each interaction-obligation at the more abstract level must be refined by at least one interaction-obligation at the more concrete level. On the other hand, o' may have interaction-obligations that do not refine any of those is o. In the STAIRS literature this notion of refinement is called general refinement. A more restrictive version is limited refinement which also requires each of the more concrete interaction-obligations to be a refinement of at least one abstract one at the more abstract level.

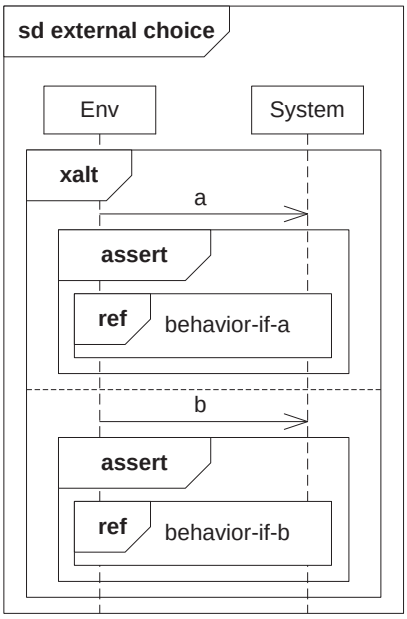

Figure 2.5: External choice represented by a sequence diagram in STAIRS

In the mapping from failure-divergences to interaction-obligations defined in the subsection above, we lost the distinction between external and internal choice. When mapping failure-divergences to sets of interaction-obligations we have the expressiveness required to keep this distinction. Roughly speaking, each external choice alternative corresponds to a separate interaction-obligation as outlined by the example in Figure 2.5. The assert, which is a standard UML 2.x operator, makes any inconclusive trace in its body negative. From the perspective of the `System` lifeline, the diagram captures an external choice between receiving either a or b. If neither behavior-if-a nor behavior-if-b contain xalt-operators, the semantics of the diagram is a pair of two interaction-obligations; one corresponding to receiving a and one corresponding to receiving b.

2.5 Conclusion

This paper compares Event-B (with a failure-divergences semantics) and STAIRS with particular focus on mandatory and potential choice. While the failure-divergences semantics gives a pure black-box interpretation of the specified system, STAIRS offers a white-box interpretation in terms of interaction-obligations and sets of interaction-obligations. Sets of interaction-obligations are required to capture mandatory choice while a single interaction-obligation is sufficient to model potential choice.

The main inspiration for writing this paper was Butler's proposal to capture external and

internal choice directly by letting the specifier divide the events into groups. The approach seemed to resemble our proposal to capture mandatory and potential choice by sets of interaction-obligations.

Our conclusion is that it does. The expressivity offered by Butler's proposal is also provided by STAIRS. In the same sense as a single event group captures internal choice, single interaction-obligations captures potential choice. When potential choice is restricted to the specified system, internal and potential choice is the same – both represent underspecification. Moreover, in the same sense as sets of event groups capture external choice, sets of interaction-obligations capture mandatory choice. When mandatory choice is restricted to the environment, external and mandatory choice is the same – both represent nondeterminism that must be preserved by refinement.

Acknowledgments. This work has been conducted as a part of the DIAMONDS (201579) project and the AGRA (236657) project, both funded by the Research Council of Norway, and the CONCERTO (232059) project funded by the Research Council of Norway and by ARTEMIS Joint Undertaking – a public private partnership in the field of embedded systems supported under the Seventh Framework Programme of the European Commission.

Bibliography

[1] M. Abadi and L. Lamport. An old-fashioned recipe for real-time. In J. W. de Bakker, C. Huizing, and W.-P. de Roever, editors, *Real-Time: Theory in Practice (REX Workshop)*, volume 600 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 1991.

[2] J.-R. Abrial. *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[3] J.-R. Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2010.

[4] J.-R. Abrial, M. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. Voisin. Rodin: an open toolset for modelling and reasoning in Event-B. *STTT*, 12(6):447–466, 2010.

[5] J.-R. Abrial, D. Cansell, and D. Mery. A mechanically proved and incremental development of ieee 1394 tree identify protocol. *Formal Aspects of Computing*, 14, 2003.

[6] J.-R. Abrial and L. Mussat. Introducing dynamic constraints in B. In *B*, volume 1393 of *LNCS*, pages 83–128. Springer, 1998.

[7] ABS tools. `http://tools.hats-project.eu`.

[8] G. A. Agha. *ACTORS: A Model of Concurrent Computations in Distributed Systems*. The MIT Press, Cambridge, Mass., 1986.

[9] W. Ahrendt and M. Dylla. A system for compositional verification of asynchronous objects. *Science of Computer Programming*, 77(12):1289–1309, Oct. 2012.

[10] M. Åkerfelt, R. I. Morimoto, and L. Sistonen. Heat shock factors: Integrators of cell stress, development and lifespan. *Nature Reviews Molecular Cell Biology*, 11(8):545–555, 2010.

[11] E. Albert, P. Arenas, A. Flores-Montoya, S. Genaim, M. Gómez-Zamalloa, E. Martin-Martin, G. Puebla, and G. Román-Díez. SACO: static analyzer for concurrent objects. In E. Ábrahám and K. Havelund, editors, *Proc. 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14)*, volume 8413 of *Lecture Notes in Computer Science*, pages 562–567. Springer, 2014.

[12] E. Albert, F. S. de Boer, R. Hähnle, E. B. Johnsen, and C. Laneve. Engineering virtualized services. In M. A. Babar and M. Dumas, editors, *2nd Nordic Symp. Cloud Computing & Internet Technologies*, pages 59–63. ACM, 2013.

[13] E. Albert, F. S. de Boer, R. Hähnle, E. B. Johnsen, R. Schlatte, S. L. T. Tarifa, and P. Y. H. Wong. Formal modeling and analysis of resource management for cloud architectures: an industrial case study using Real-Time ABS. *Service Oriented Computing and Applications*, 8(4):323–339, 2014.

[14] B. Alpern and F. B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181 – 185, 1985.

[15] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183 – 235, 1994.

[16] K. R. Apt, F. S. de Boer, and E.-R. Olderog. *Verification of Sequential and Concurrent Systems*. Texts and Monographs in Computer Science. Springer, 3rd edition, 2009.

[17] F. Arnold, A. Belinfante, F. Van der Berg, D. Guck, and M. Stoelinga. DFTCalc: A tool for efficient fault tree analysis. In F. Bitsch, J. Guiochet, and M. Kaaniche, editors, *Computer Safety, Reliability, and Security*, volume 8153 of *Lecture Notes in Computer Science*, pages 293–301. Springer Berlin Heidelberg, 2013.

[18] E. Asarin, P. Caspi, and O. Maler. Timed regular expressions. *Journal of the ACM*, 49:172–206, 2002.

[19] P. Asirelli, M. H. ter Beek, A. Fantechi, and S. Gnesi. A logical framework to deal with variability. In *Integrated Formal Methods - 8th International Conference, IFM 2010, Nancy, France, October 11-14, 2010. Proceedings*, pages 43–58, 2010.

[20] P. Asirelli, M. H. ter Beek, S. Gnesi, and A. Fantechi. Formal description of variability in product families. In *Software Product Lines - 15th International Conference, SPLC 2011, Munich, Germany, August 22-26, 2011*, pages 130–139, 2011.

[21] A. Aswani, N. Master, J. Taneja, C. D., and C. Tomlin. Reducing transient and steady state electricity consumption in hvac using learning-based model-predictive control. In *Proceedings of the IEEE*, volume 99, pages 1–14. IEEE, 2011.

[22] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.*, 1(1):11–33, Jan. 2004.

[23] R. J. Back. *On the Correctness of Refinement in Program Development*. PhD thesis, Department of computer Science, University of Helsinki, 1978.

[24] R. J. Back. Correctness preserving program refinements: Proof theory and applications. *Mathematical Center Tracts*, 131, 1980.

[25] R. J. Back. Refinement calculus ii: parallel and reactive programs. In J. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, volume 430 of *Lecture Notes in Computer Science*, pages 67–93. Springer-Verlag, 1990.

[26] R. J. Back. Atomicity refinement in a refinement calculus framework, reports on computer science and mathematics 141. Technical report, Åbo Akademi, 1993.

[27] R. J. Back and R. Kurki-Suonio. Decentralization of process nets with centralized. In *Proceedings of the 2nd ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, pages 131–142, 1983.

[28] R. J. Back and R. Kurki-Suonio. Decentralization of process nets with centralized control. *Distributed Computing*, 3(2):73–87, 1983.

[29] R. J. Back and R. Kurki-Suonio. Distributed cooperation with action systems. *ACM Trans. Program. Lang. Syst.*, 10(4):513–554, 1988.

[30] R. J. Back, L. Petre, and I. Porres. Continuous action systems as a model for hybrid systems. *Nord. J. Comput.*, 8(1):2–21, 2001.

[31] R. J. Back and K. Sere. Stepwise refinement of action systems. In *Proceedings of the international Conference on Mathematics of Program Construction, 375th Anniversary of the Groningen University*, number LNCS 375, pages 115–138. Springer-Verlag, 1989.

[32] R. J. Back and K. Sere. Stepwise refinement of action systems. *Structured Programming*, 12(1):17–30, 1991.

[33] R. J. Back and K. Sere. From action systems to modular systems. In M. Naftalin, T. Denvir, and M. Bertran, editors, *FME '94: Industrial Benefit of Formal Methods*, volume 873 of *Lecture Notes in Computer Science*, pages 1–25. Springer Berlin Heidelberg, 1994.

[34] R. J. Back and K. Sere. From action systems to modular systems. *Software - Concepts and Tools*, 17, 1996.

[35] R. J. Back and K. Sere. Superposition refinement of reactive systems. *Formal Aspects of Computing*, 8(3):324–346, 1996.

[36] R. J. Back and J. von Wright. Refinement calculus i: Sequential nondeterministic programs. In J. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, volume 430 of *LNCS*, pages 42–66. Springer-Verlag, 1990.

[37] R. J. Back and J. von Wright. Trace refinement of action systems. In B. Jonsson and J. Parrow, editors, *CONCUR-94:Concurrency Theory*, volume 836 of *Lecture Notes in Computer Science*, pages 367–384, Uppsala, Sweden, Aug 1994. Springer-Verlag.

[38] R. J. Back and J. von Wright. *Refinement Calculus: A Systematic Introduction*. Springer-Verlag, 1998.

[39] C. Baier and J. Katoen. *Principles of model checking*. MIT Press, 2008.

[40] R. Banach and M. Bozzano. The mechanical generation of fault trees for reactive systems via retrenchment I: combinational circuits. *Formal Aspects of Computing*, 25(4):573–607, 2013.

[41] R. Banach and M. Bozzano. The mechanical generation of fault trees for reactive systems via retrenchment II: clocked and feedback circuits. *Formal Aspects of Computing*, 25(4):609–657, 2013.

[42] R. Banach and M. Butler. Cruise control in hybrid event-b. In Z. Liu, J. Woodcock, and H. Zhu, editors, *Theoretical Aspects of Computing - ICTAC 2013*, pages 76–93. Springer-Verlag Berlin Heidelberg, 2013.

[43] R. Banach, H. Zhu, W. Su, and R. Huang. Continuous kaos, asm, and formal control system design across the continuous/discrete modeling interface: a simple train stopping application. *Formal Asp. Comput.*, 26(2):319–366, 2014.

[44] I. Banu-Demergian, C. Paduraru, and G. Stefanescu. A new representation of two-dimensional patterns and applications to interactive programming. In *FSEN 2013*, volume 8161 of *Lecture Notes in Computer Science*, pages 172–206. Springer, 2013.

[45] I. Banu-Demergian and G. Stefanescu. On the contour representation of two-dimensional patterns. *Carpathian Journal Mathematics*, 2014. To appear.

[46] I. Banu-Demergian and G. Stefanescu. Towards a formal representation of interactive systems. *Fundamenata Informaticae*, 131:313–336, 2014.

[47] G. Behrmann, A. David, and K. G. Larsen. A tutorial on uppaal. In M. Bernardo and F. Corradini, editors, *SFM-RT 2004*, volume 3185 of *LNCS*, pages 200–237. Springer Verlag, 2004.

[48] N. Benes and J. Kretínský. Process algebra for modal transition systemses. In *Sixth Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS 2010, Selected Papers, October 22-24, 2010, Mikulov, Czech Republic*, pages 9–18, 2010.

[49] N. Benes, J. Kretínský, K. G. Larsen, M. H. Møller, and J. Srba. Parametric modal transition systems. In *Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011. Proceedings*, pages 275–289, 2011.

[50] N. Benes, J. Kretínský, K. G. Larsen, and J. Srba. Exptime-completeness of thorough refinement on modal transition systems. *Inf. Comput.*, 218:54–68, 2012.

[51] N. Beneš. *Disjunctive Modal Transition Systems*. PhD thesis, Faculty of Informatics, Masaryk University, Brno, 2012.

[52] J. Bengtsson, K. G. Larsen, F. Larsson, P. Petterson, and W. Yi. UPPAAL — a tool-suite for the automatic verification of real-time systems. In R. Alur, T. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*. Springer, 1996.

[53] J. Bergstra, A. Ponse, and S. Smolka, editors. *Handbook of Process Algebra*. Elsevier Science Inc., New York, NY, USA, 2001.

[54] J. Berthing, P. Boström, K. Sere, L. Tsiopoulos, and J. Vain. Refinement-based development of timed systems. In J. Derrick, S. Gnesi, D. Latella, and H. Treharne, editors, *Integrated Formal Methods – 9th International Conference, IFM 2012, Pisa, Italy, June 18-21, 2012. Proceedings*, volume 7321 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2012.

[55] J. Bicarregui, A. Arenas, B. Aziz, P. Massonet, and C. Ponsard. Towards modelling obligations in Event-B. In *Abstract State Machines, B and Z*, volume 5238 of *LNCS*, pages 181–194. Springer, 2008.

[56] J. Bjørk, F. S. de Boer, E. B. Johnsen, R. Schlatte, and S. L. Tapia Tarifa. User-defined schedulers for real-time concurrent objects. *Innovations in Systems and Software Engineering*, 9(1):29–43, 2013.

[57] M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek. BioNetGen: Software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289–3291, 2004.

[58] E. Boiten and J. Derrick. Modelling divergence in relational concurrent refinement. In M. Leuschel and H. Wehrheim, editors, *IFM 2009: Integrated Formal Methods*, volume 5423 of *LNCS*, pages 183–199. Springer Verlag, February 2009.

[59] E. Boiten and J. Derrick. Incompleteness of relational simulations in the blocking paradigm. *Science of Computer Programming*, 75(12):1262–1269, 2010.

[60] E. Boiten, J. Derrick, and G. Schellhorn. Relational concurrent refinement II: Internal operations and outputs. *Formal Aspects of Computing*, 21(1-2):65–102, 2009.

[61] M. M. Bonsangue and J. N. Kok. The weakest precondition calculus: recursion and duality. *Formal Aspects of Computing*, 6A:788–800, 1994.

[62] M. M. Bonsangue, J. N. Kok, and K. Sere. An approach to object-orientation in action systems. In J. Jeuring, editor, *Mathematics of Program Construction (MPC'98)*, volume 1422 of *Lecture Notes in Computer Science*, pages 68–95. Springer, 1998.

[63] P. Bouyer, F. Laroussinie, and P.-A. Reynier. Diagonal constraints in timed automata: Forward analysis of timed systems. In P. Pettersson and W. Yi, editors, *FORMATS'05*, volume 3829 of *Lecture Notes in Computer Science*, pages 112–126. Springer- Heidelberg, 2005.

[64] J. Bowen and S. Reeves. Refinement for user interface designs. *Formal Aspects of Computing*, 21:589–612, 2009.

[65] J. Bowen and S. Reeves. Modelling safety properties of interactive medical systems. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '13, pages 91–100. ACM, 2013.

[66] M. Bozzano, A. Cimatti, and F. Tapparo. Symbolic fault tree analysis for reactive systems. In *Automated Technology for Verification and Analysis*, volume 4762 of *Lecture Notes in Computer Science*, pages 162–176. Springer Berlin Heidelberg, 2007.

[67] M. Bozzano and A. Villafiorita. The FSAP/NuSMV-SA safety analysis platform. *International Journal on Software Tools for Technology Transfer*, 9(1):5–24, 2007.

[68] M. Bravetti and G. Zavattaro. Towards a Unifying Theory for Choreography Conformance and Contract Compliance. In *Software Composition, 6th International Symposium, SC 2007, Braga, Portugal, March 24-25, 2007, Revised Selected Papers*, pages 34–50, 2007.

[69] P. Bulychev, A. David, K. G. Larsen, M. Mikučionis, D. B. Poulsen, A. Legay, and Z. Wang. UPPAAL-SMC: Statistical model checking for priced timed automata. In H. Wiklicky and M. Massink, editors, *Quantitative Aspects of Programming Languages and Systems,* Proceedings of the 10th Workshop on, volume 85 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–16. Open Publishing Association, 2012.

[70] L. Burdy, Y. Cheon, D. R. Cok, M. D. Ernst, J. R. Kiniry, G. T. Leavens, K. R. M. Leino, and E. Poll. An overview of JML tools and applications. *International Journal on Software Tools for Technology Transfer*, 7(3):212–232, 2005.

[71] M. Butler. External and internal choice with event groups in Event-B. *Formal Aspects of Computing*, 24(4-6):555–567, 2012.

[72] M. Butler and I. Maamria. Practical theory extension in event-b. In *Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday*, volume 8051 of *Lecture Notes in Computer Science*, pages 67–81. Springer, 2013.

[73] M. Butler, E. Sekerinski, and K. Sere. An Action System Approach to the Steam Boiler Problem. In *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, volume LNCS1165, pages 129–148. Springer-Verlag, 1996.

[74] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Comp. Sys.*, 25(6):599–616, 2009.

[75] D. Cansell, D. Meŕy, and J. Rehm. Time constraint patterns for Event B development. In *B 2007: Formal specification and development in B*, volume 4355 of *LNCS*, pages 140–154. Springer Heidelberg, 2007.

[76] L. Cardelli. On process rate semantics. *Theoretical Computer Science*, 391(3):190–215, 2008.

[77] Cardinal Health Inc. Alaris GP volumetric pump: directions for use. Technical report, Cardinal Health, 1180 Rolle, Switzerland, 2006.

[78] G. Castagna, N. Gesbert, and L. Padovani. A Theory of Contracts for Web Services. *ACM Trans. Program. Lang. Syst.*, 31(5):19:1–19:61, 2009.

[79] Z. Chaochen and M. Hansen. *Duration Calculus: A Formal Approach to Real-Time Systems*. Springer, Heidelberg, 2004.

[80] A. Cherubini and M. Pradella. Picture languages: From wang tiles to 2d grammars. In *Algebraic Informatics*, Lecture Notes in Computer Science, 2009.

[81] A. Cherubini, S. C. Reghizzi, M. Pradella, and P. S. Pietro. Picture languages: Tiling systems versus tile rewriting grammars. *Theoretical Computer Science*, 356:90–103, 2006.

[82] W.-N. Chin, C. David, H.-H. Nguyen, and S. Qin. Enhancing modular oo verification with separation logic. In *Proceedings of POPL '08*, pages 87–99. ACM, Jan. 2008.

[83] F. Ciocchetta and J. Hillston. Bio-pepa: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410(33–34):3065–3084, 2009.

[84] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley Professional, 2001.

[85] J. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.

[86] Council of the European Communities. Council directive 93/42/EEC of 14 June 1993 concerning medical devices. `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:1993L0042:20071011:EN:PDF`, 2007.

[87] O.-J. Dahl. Can program proving be made practical? In M. Amirchahy and D. Néel, editors, *Les Fondements de la Programmation*, pages 57–114. Institut de Recherche d'Informatique et d'Automatique, Toulouse, France, Dec. 1977.

[88] O.-J. Dahl. *Verifiable Programming*. International Series in Computer Science. Prentice Hall, New York, N.Y., 1992.

[89] O.-J. Dahl. The birth of object orientation: the simula languages. In O. Owe, S. Krogdahl, and T. Lyche, editors, *From Object-Orientation to Formal Methods, Essays in Memory of Ole-Johan Dahl*, volume 2635 of *Lecture Notes in Computer Science*, pages 15–25. Springer, 2004.

[90] O.-J. Dahl and O. Owe. Formal development with ABEL. In S. Prehn and H. Toetenel, editors, *Proc. Formal Software Development Methods (VDM'91)*, volume 552 of *Lecture Notes in Computer Science*, pages 320–362. Springer, Oct. 1991.

[91] P. H. Dalsgaard, T. L. Guilly, D. Middelhede, P. Olsen, T. Pedersen, A. P. Ravn, and A. Skou. A toolchain for home automation controller development. In *39th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2013, Santander, Spain, September 4-6, 2013*, pages 122–129, 2013.

[92] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling, symmetries, refinements. In *Formal Methods in Systems Biology*, pages 103–122. Springer, 2008.

[93] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling and model perturbation. *Transactions on Computational Systems Biology XI*, pages 116–137, 2009.

[94] V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[95] R. Darimont and A. van Lamsweerde. Formal refinement patterns for goal-driven requirements elaboration. In *Proceedings 4th ACM Symposium on the Foundations of Software Engineering (FSE'03)*, pages 179–190. ACM Press, 1996.

[96] A. David, K. G. Larsen, A. Legay, U. Nyman, and A. Wasowski. Timed I/O Automata: A complete specification theory for real-time systems. In *HSCC'10*. ACM, 2011.

[97] F. S. de Boer. Reasoning about histories in object-based distributed systems. In P. Ciancarini, A. Fantechi, and R. Gorrieri, editors, *Formal Methods for Open Object-Based Distributed Systems*, volume 10 of *IFIP - The International Federation for Information Processing*, pages 35–49. Springer US, 1999.

[98] F. S. de Boer, R. Hähnle, E. B. Johnsen, R. Schlatte, and P. Y. H. Wong. Formal modeling of resource management for cloud architectures: An industrial case study. In *Proc. European Conference on Service-Oriented and Cloud Computing (ESOCC)*, volume 7592 of *Lecture Notes in Computer Science*, pages 91–106. Springer, Sept. 2012.

[99] B. de Bono, P. Grenon, M. Helvenstijn, J. Kok, and N. Kokash. ApiNATOMY: Towards multiscale views of human anatomy. In *Processings of the 13th International Symposium on Intelligent Data Analysis (IDA)*, volume 8819 of *LNCS*, pages 72–83. Springer-Verlag, 2014.

[100] B. de Bono and P. Hunter. Integrating knowledge representation and quantitative modelling in physiology. *Biotechnology Journal*, 7(8):958–972, 2012.

[101] W.-P. de Roever and K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and their Comparison*. CUP, 1998.

[102] E. Deelman, G. Singh, M. Livny, G. B. Berriman, and J. Good. The cost of doing science on the cloud: The Montage example. In *Proceedings of the Conference on High Performance Computing (SC'08)*, pages 1–12. IEEE/ACM, 2008.

[103] J. Derrick and E. Boiten. Relational concurrent refinement. *Formal Aspects of Computing*, 15(1):182–214, November 2003.

[104] J. Derrick and E. Boiten. *Refinement in Z and Object-Z*. Springer-Verlag, 2nd edition, 2014.

[105] J. Derrick and E. Boiten. Relational concurrent refinement III: Traces, partial relations and automata. *Formal Aspects of Computing*, 26(2):407–422, 2014.

[106] J. Derrick and E. A. Boiten. Relational concurrent refinement with internal operations. In B. Aichernig, E. A. Boiten, J. Derrick, and L. Groves, editors, *BCS-FACS Refinement Workshop*, volume 187 of *ENTCS*, pages 35–53, 2006.

[107] J. Derrick and G. Smith. Temporal-logic property preservation under Z refinement. *Formal Asp. Comput.*, 24(3):393–416, 2012.

[108] D. Diaconescu, I. Leustean, L. Petre, K. Sere, and G. Stefanescu. Refinement-preserving translation from event-b to register-voice interactive systems. In *Proceedings IFM 2012*, volume 7321 of *Lecture Notes in Computer Science*, pages 221–236. Springer-Verlag, 2012.

[109] D. Diaconescu, L. Petre, K. Sere, and G. Stefanescu. Refinement of structured interactive systems. In *Proceedings ICTAC 2014*, volume 8687 of *Lecture Notes in Computer Science*, pages 133–150. Springer, 2014.

[110] H. Dierks, S. Kupfersmid, and K. G. Larsen. Automatic abstraction refinement for timed automata. In J.-F. Raskin and P. S. Thiagarajan, editors, *FORMATS'07*, volume 4763 of *LNCS*, 2007.

[111] E. W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18(8):453–457, 1975.

[112] E. W. Dijkstra. *A Discipline of Programming*. Prentice–Hall International, 1976.

[113] C. C. Din. *Verification of Asynchronously Communicating Objects*. PhD thesis, Department of informatics, University of Oslo, Norway, 2014.

[114] C. C. Din, J. Dovland, E. B. Johnsen, and O. Owe. Observable behavior of distributed systems: Component reasoning for concurrent objects. *Journal of Logic and Algebraic Programming*, 81(3):227–256, 2012.

[115] C. C. Din and O. Owe. Compositional reasoning about active objects with shared futures. *Formal Aspects of Computing*, 27:1–22, 2014.

[116] J. Dovland. *Incremental Reasoning about Distributed Object-Oriented Systems*. PhD thesis, University of Oslo, Dept. of Computer Science, 2009.

[117] J. Dovland, E. B. Johnsen, O. Owe, and M. Steffen. Lazy behavioral subtyping. Research Report 368, Dept. of Informatics, University of Oslo, Nov. 2007. Available from `heim.ifi.uio.no/creol`.

[118] J. Dovland, E. B. Johnsen, O. Owe, and M. Steffen. Lazy behavioral subtyping. *Journal of Logic and Algebraic Programming*, 79(7):578–607, 2010.

[119] J. Dovland, E. B. Johnsen, O. Owe, and M. Steffen. Incremental reasoning with lazy behavioral subtyping for multiple inheritance. *Science of Computer Programming*, 76(10):915 – 941, 2011.

[120] C. Dragoi and G. Stefanescu. Agapia v0. 1: A programming language for interactive systems and its typing system. *Electronic Notes in Theoretical Computer Science*, 203(3):69–94, 2008.

[121] C. Dragoi and G. Stefanescu. On compiling structured interactive programs with registers and voices. In *Proceedings SOFSEM 2008*, volume 4910 of *Lecture Notes in Computer Science*, pages 259–270. Springer, 2008.

[122] D. J. Duke and M. D. Harrison. Mapping user requirements to implementations. *Software Engineering Journal*, 10(1):13–20, 1995.

[123] S. Eigen, J. Navarro, and V. S. Prasad. *An aperiodic tiling using a dynamical system and Beatty sequences*, pages 223–241. Cambridge Univ. Press, 2007.

[124] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241 – 266, 1982.

[125] K. Etessami, M. Z. Kwiatkowska, M. Y. Vardi, and M. Yannakakis. Multi-objective model checking of markov decision processes. *Logical Methods in Computer Science*, 4(4), 2008.

[126] J. R. Faeder, M. L. Blinov, B. Goldstein, and W. S. Hlavacek. Rule-based modeling of biochemical networks. *Complexity*, 10(4):22–41, 2005.

[127] J. R. Faeder, M. L. Blinov, and W. S. Hlavacek. Graphical rule-based representation of signal-transduction networks. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, pages 133–140. ACM, 2005.

[128] A. Fantechi and S. Gnesi. A behavioural model for product families. In *Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2007, Dubrovnik, Croatia, September 3-7, 2007*, pages 521–524, 2007.

[129] A. Fantechi and S. Gnesi. Formal modeling for product families engineering. In *Software Product Lines, 12th International Conference, SPLC 2008, Limerick, Ireland, September 8-12, 2008, Proceedings*, pages 193–202, 2008.

[130] H. Fecher and H. Schmidt. Comparing disjunctive modal transition systems with an one-selecting variant. *J. Log. Algebr. Program.*, 77(1-2):20–39, 2008.

[131] D. Fischbein, S. Uchitel, and V. A. Braberman. A foundation for behavioural conformance in software product line architectures. In *Proceedings of the 2006 Workshop on Role of Software Architecture for Testing and Analysis, held in conjunction with the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2006), ROSATEA 2006, Portland, Maine, USA, July 17-20, 2006*, pages 39–48, 2006.

[132] A. Flores-Montoya, E. Albert, and S. Genaim. May-happen-in-parallel based deadlock analysis for concurrent objects. In D. Beyer and M. Boreale, editors, *Proc. International Conference on Formal Techniques for Distributed Systems (FMOODS/FORTE 2013)*, volume 7892 of *Lecture Notes in Computer Science*, pages 273–288. Springer, 2013.

[133] S. Fowler and A. Wellings. Formal analysis of a real-time kernel specification. In B. Jonsson and J. Parrow, editors, *Proceedings of FTRTFT'96*, volume 1135 of *Lecture Notes in Computer Science*, pages 440–458. Springer, 1996.

[134] FP6 EU project SPEEDS: SPEculative and Exploratory Design in Systems Engineering. Online at http://www.speeds.eu.com/.

[135] C. A. Furia, M. Rossi, D. Mandrioli, and A. Morzenti. Automated compositional proofs for real-time systems. *Theoretical Computer Science*, 376(3):164–184, 2007.

[136] A. Fürst. Design patterns in Event-B and their tool support. Master's thesis, ETH, Eidgenössische Technische Hochschule Zürich, Department of Computer Science, 2009.

[137] V. Garg and M. Ragunath. Concurrent regular expressions and their relationship to Petri nets. *Theoretical Computer Science*, 96:285–304, 1992.

[138] E. Gasparis. LePUS: A formal language for modeling design patterns. In *Design Pattern Formalization Techniques*, pages 357–372. IGI Global, 2007.

[139] C. Ghezzi, D. Mandrioli, and A. Morzenti. TRIO, a logic language for executable specifications of realtime systems. *Journal of Systems and Software*, 12(2):255–307, 1990.

[140] D. Giammarresi and A. Restivo. Two-dimensional languages. In *Handbook of formal languages*, pages 215–267. Springer, 1997.

[141] A. Gondal, M. Poppleton, and C. Snook. Feature composition - towards product lines of Event-B models. In *1st International Workshop on Model-Driven Product Line Engineering (MDPLE'09)*. CTIT Workshop Proceedings, 2009.

[142] M. Gordon. A classical mind: Essays in honour of C.A.R. Hoare. In A. W. Roscoe, editor, *A Classical Mind: Essays in Honour of C.A.R. Hoare*, chapter A Mechanized Hoare Logic of State Transitions, pages 143–159. Prentice Hall, 1994.

[143] C. Gratie and I. Petre. Fit-preserving data refinement of mass-action reaction networks. In A. Beckmann, E. Csuhaj-Varjú, and K. Meer, editors, *Language, Life, Limits*, volume 8493 of *Lecture Notes in Computer Science*, pages 204–213. Springer, 2014.

[144] D.-E. Gratie, B. Iancu, S. Azimi, and I. Petre. Quantitative model refinement in four different frameworks, with applications to the heat shock response. Technical Report 1067, TUCS, 2013.

[145] C. Grioli. Improvement and analysis of behavioural models with variability. Master's thesis, University of Pisa, Italy, 2013.

[146] J. Groslambert. Verification of LTL on B Event Systems. In *B 2007: Formal Specification and Development in B*, volume 4355 of *LNCS*, pages 109–124. Springer, 2006.

[147] R. Grossman, A. Nerode, A. Ravn, and H. Rischel. *Hybrid Systems*. Springer-Verlag, 1993.

[148] B. Grunbaum and G. Shephard. *Tilings and Patterns*. W.H. Freeman and Co., 1987.

[149] V. H. Haase. Real-time behavior of programs. *IEEE Transactions on Software Engineering*, 7(5):594–501, Sept. 1981.

[150] G. Haddad, F. Hussain, and G. T. Leavens. The design of SafeJML, a specification language for SCJ with support for WCET specifications. In *Proceedings of JTRES'10*, pages 155–163. ACM, 2010.

[151] R. Hähnle and E. B. Johnsen. Resource-aware applications for the cloud. *IEEE Computer*, May 2015. To appear.

[152] S. Hallerstede. On the purpose of Event-B proof obligations. *Formal Aspects of Computing*, 23(1):133–150, 2011.

[153] S. Hallerstede, M. Leuschel, and D. Plagge. Validation of formal models by refinement animation. *Science of Computer Programming*, 78(3):272 – 292, 2013.

[154] K. M. Hansen, A. P. Ravn, and V. Stavridou. From safety analysis to software requirements. *Software Engineering, IEEE Transactions on*, 24(7):573–584, Jul 1998.

[155] M. D. Harrison, P. Masci, J. C. Campos, and P. Curzon. Demonstrating that medical devices satisfy user related safety requirements. In *4th International Symposium on Foundations of Healthcare Information Engineering and Systems (FHIES2014)*, 2014.

[156] Ø. Haugen, K. E. Husa, R. K. Runde, and K. Stølen. STAIRS towards formal design with sequence diagrams. *Journal of Software and Systems Modeling*, 4:355–367, 2005.

[157] I. Hayes, M. Jackson, and C. Jones. Determining the specification of a control system from that of its environment. In K. Araki, S. Gnesi, and D. Mandrioli, editors, *FME 2003: Formal Methods, LNCS 2805*. Springer-Verlag, Berlin, 2003.

[158] He Jifeng and C. Hoare. Prespecification and data refinement. In *Data Refinement in a Categorical Setting*, Technical Monograph, number PRG-90. Oxford University Computing Laboratory, Nov. 1990.

[159] He Jifeng, C. Hoare, and J. Sanders. Data refinement refined. In B. Robinet and R. Wilhelm, editors, *Proc. ESOP 86*, volume 213 of *Lecture Notes in Computer Science*, pages 187–196. Springer-Verlag, 1986.

[160] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. In *Computational Methods in Systems Biology*, pages 32–47. Springer, 2006.

[161] G. Heineman and W. Councill (editors). *Component-Based Software Engineering: Putting the Pieces Together*. Addison-Wesley, 2001.

[162] M. Heiner, M. Herajy, F. Liu, C. Rohr, and M. Schwarick. Snoopy – a unifying Petri net tool. In S. Haddad and L. Pomello, editors, *Application and Theory of Petri Nets*, volume 7347 of *Lecture Notes in Computer Science*, pages 398–407. Springer Berlin Heidelberg, 2012.

[163] C. Heinzemann, C. Brenner, S. Dziwok, and W. Schäfer. Automata-based refinement checking for real-time systems. *Computer Science - Research and Development*, 2014.

[164] T. A. Henzinger. The theory of hybrid automata. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 278–292. IEEE Computer Society, 1996.

[165] T. A. Henzinger. The theory of hybrid automata. In *LICS 1996*, pages 278–292. IEEE, 1996.

[166] T. A. Henzinger, R. Majumbar, and J. F. Raskin. A classification of symbolic transition systems. *ACM Transactions on Computational Logic*, pages 1–32, 2005.

[167] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[168] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 441–444, 2006.

[169] T. S. Hoang and J.-R. Abrial. Reasoning about liveness properties in Event-B. In *ICFEM*, volume 6991 of *LNCS*, pages 456–471. Springer, 2011.

[170] T. S. Hoang, A. Fürst, and J. Abrial. Event-B patterns and their tool support. *Software & Systems Modeling*, 12(2):229–244, 2013.

[171] C. A. R. Hoare. *An axiomatic basis for computer programming*. Communications of the ACM, 12(10):576-580, 1969.

[172] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[173] J. Hooman. Extending Hoare logic to real-time. *Formal Aspects of Computing*, 6(6A):801–826, 1994.

[174] J. Hooman. Compositional verification of real-time applications. In W.-P. de Roever, H. Langmaack, and A. Pnueli, editors, *Compositionality: The Significant Difference (Compos '97)*, volume 1536 of *Lecture Notes in Computer Science*, pages 276–300. Springer, 1998.

[175] J. E. Hopcroft. An n log n algorithm for minimizing states in a finite automaton. *Theory of Machines and Computations*, pages 189–196, 1971.

[176] S. Hudon and T. S. Hoang. Systems design guided by progress concerns. In *Integrated Formal Methods, 10th International Conference, IFM 2013, Turku, Finland, June 10-14, 2013. Proceedings*, volume 7940 of *Lecture Notes in Computer Science*, pages 16–30, 2013.

[177] B. Iancu, E. Czeizler, E. Czeizler, and I. Petre. Quantitative refinement of reaction models. *International Journal of Unconventional Computing*, 8(5-6):529–550, 2012.

[178] B. Iancu, D.-E. Gratie, S. Azimi, and I. Petre. Computational modeling of the eukaryotic heat shock response: the BioNetGen implementation, the Petri net implementation and the PRISM implementation, 2013. Available at: http://combio.abo.fi/research/computational-modeling-of-the-eukaryotic-heat-shock-response/.

[179] B. Iancu, D.-E. Gratie, S. Azimi, and I. Petre. On the implementation of quantitative model refinement. In A.-H. Dediu, C. Martín-Vide, and B. Truthe, editors, *Algorithms for Computational Biology*, volume 8542 of *Lecture Notes in Computer Science*, pages 95–106. Springer International Publishing, 2014.

[180] K. C. II. An aperiodic set of 13 wang tiles. *Discrete Mathematics*, 160:245–251, 1996.

[181] A. Iliasov. Use Case Scenarios as Verification Conditions: Event-B/Flow Approach. In *SERENE 2011, Software Engineering for Resilient Systems*, pages 9–23. Springer-Verlag, 2011.

[182] A. Iliasov, L. Laibinis, E. Troubitsyna, A. Romanovsky, and T. Latvala. Augmenting Event-B modelling with real-time verification. Technical Report 1006, TUCS, 2011.

[183] A. Iliasov, E. Troubitsyna, L. Laibinis, A. Romanovsky, K. Varpaaniemi, D. Ilic, and T. Latvala. Supporting Reuse in Event B Development: Modularisation Approach. In *Proceedings of Abstract State Machines, Alloy, B, and Z (ABZ 2010), Lecture Notes in Computer Science*, Vol.5977, pp. 174-188, Springer, 2010.

[184] A. Iliasov, E. Troubitsyna, L. Laibinis, A. Romanovsky, K. Varpaaniemi, D. Ilic, and T. Latvala. Developing Mode-Rich Satellite Software by Refinement in Event-B. In *Science of Computer Programming 78(7)*, pages 884–905. Springer-Verlag, 2013.

[185] A. Iliasov, E. Troubitsyna, L. Laibinis, A. Romanovsky, K. Varpaaniemi, P. Väisänen, D. Ilic, and T. Latvala. Developing Mode-Rich Satellite Software by Refinement in Event B. In *FMICS 2010, The 15th International Workshop on Formal Methods for Industrial Critical Systems*, *Lecture Notes for Computer Science*, Springer, 2010.

[186] A. Iliasov, E. Troubitsyna, L. Laibinis, A. Romanovsky, K. Varpaaniemi, P. Väisänen, D. Ilic, and T. Latvala. Verifying Mode Consistency for On-Board Satellite Software. In *SAFECOMP 2010, The 29th International Conference on Computer Safety, Reliability and Security, September 2010, Vienna, Austria*, *Lecture Notes for Computer Science, Springer,* 2010.

[187] R. Jetley, S. Purushothaman Iyer, and P. L. Jones. A formal methods approach to medical device review. *Computer*, 39(4):61–67, 2006.

[188] E. B. Johnsen, J. C. Blanchette, M. Kyas, and O. Owe. Intra-object versus inter-object: Concurrency and reasoning in creol. *Electron. Notes Theor. Comput. Sci.*, 243:89–103, July 2009.

[189] E. B. Johnsen, R. Hähnle, J. Schäfer, R. Schlatte, and M. Steffen. ABS: A core language for abstract behavioral specification. In B. Aichernig, F. S. de Boer, and M. M.

Bonsangue, editors, *Proc. 9th International Symposium on Formal Methods for Components and Objects (FMCO 2010)*, volume 6957 of *Lecture Notes in Computer Science*, pages 142–164. Springer, 2011.

[190] E. B. Johnsen and O. Owe. An asynchronous communication model for distributed concurrent objects. *Software and System Modeling*, 6(1):35–58, Mar. 2007.

[191] E. B. Johnsen, O. Owe, and E. W. Axelsen. A run-time environment for concurrent objects with asynchronous method calls. In *Proc. 5th International Workshop on Rewriting Logic and its Applications (WRLA'04)*, Mar. 2004.

[192] E. B. Johnsen, O. Owe, D. Clarke, and J. Bjørk. A formal model of service-oriented dynamic object groups. *Science of Computer Programming*, pages 1–24, 2014. In press.

[193] E. B. Johnsen, O. Owe, and I. Simplot-Ryl. A dynamic class construct for asynchronous concurrent objects. In M. Steffen and G. Zavattaro, editors, *Proc. 7th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS'05)*, volume 3535 of *Lecture Notes in Computer Science*, pages 15–30. Springer, June 2005.

[194] E. B. Johnsen, O. Owe, and I. C. Yu. Creol: A type-safe object-oriented model for distributed concurrent systems. *Theoretical Computer Science*, 365(1–2):23–66, Nov. 2006.

[195] E. B. Johnsen, R. Schlatte, and S. L. Tapia Tarifa. A formal model of object mobility in resource-restricted deployment scenarios. In F. Arbab and P. Ölveczky, editors, *Proc. 8th International Symposium on Formal Aspects of Component Software (FACS 2011)*, volume 7253 of *Lecture Notes in Computer Science*, pages 185–202. Springer, 2012.

[196] E. B. Johnsen, R. Schlatte, and S. L. Tapia Tarifa. Modeling resource-aware virtualized applications for the cloud in Real-Time ABS. In *Proc. Formal Engineering Methods (ICFEM'12)*, volume 7635 of *Lecture Notes in Computer Science*, pages 71–86. Springer, Nov. 2012.

[197] E. B. Johnsen, R. Schlatte, and S. L. Tapia Tarifa. Integrating deployment architectures and resource consumption in timed object-oriented models. *Journal of Logical and Algebraic Methods in Programming*, 2014. Available online.

[198] B. Jonsson and Y.-K. Tsay. Assumption/guarantee specifications in linear-time temporal logic. *Theoretical Computer Science*, 167(2):47–72, 1996.

[199] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. In *Proceedings of the Cambridge Philosophical Society*, volume 119, pages 447–468, 1996.

[200] J. Jürjens. Secrecy-preserving refinement. In *Proceedings of Formal Methods Europe (FME'01)*, volume 2021 of *Lecture Notes in Computer Science*, pages 135–152. Springer, 2001.

[201] J. Kari. A small aperiodic set of wang tiles. *Discrete Mathematics*, 160:259–264, 1996.

[202] R. M. Keller. Formal verification of parallel programs. *Commun. ACM*, 19(7):371–384, July 1976.

[203] H. Kitano. Systems biology: a brief overview. *Science*, 295(5560):1662–1664, 2002.

[204] S. Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, 1956.

[205] M. P. Kline and R. I. Morimoto. Repression of the heat shock factor 1 transcriptional activation domain is modulated by constitutive phosphorylation. *Molecular and Cellular Biology*, 17(4):2107–2115, 1997.

[206] E. Klipp, R. Herwig, A. Kowald, C. Wierling, and H. Lehrach. *Systems Biology in Practice: Concepts, Implementation and Application*. Wiley-Vch, 2005.

[207] I. Koch, W. Reisig, and F. Schreiber. *Modeling in Systems Biology: the Petri Net Approach*. Springer, 2010.

[208] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *LICS'91*, pages 214–225. IEEE, 1991.

[209] J. Krone, W. F. Ogden, and M. Sitaraman. Modular verification of performance constraints. In *ACM OOPSLA Workshop on Specification and Verification of Component-Based Systems (SAVCBS)*, pages 60–67, 2001.

[210] J. Krone, W. F. Ogden, and M. Sitaraman. Profiles: A compositional mechanism for performance specification. Technical Report RSRG-04-03, Department of Computer Science, Clemson University, Clemson, SC 29634-0974, June 2004.

[211] W. Kuich and A. Salomaa. *Semirings, automata and languages*. Springer-Verlag, Berlin, 1985.

[212] M. Kwiatkowska, G. Norman, and D. Parker. Quantitative analysis with the probabilistic model checker PRISM. *Electronic Notes in Theoretical Computer Science*, 153(2):5–31, 2006.

[213] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic model checking for performance and reliability analysis. *SIGMETRICS Perform. Eval. Rev.*, 36(4):40–45, 2009.

[214] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

[215] L. Lamport. Hybrid systems in TLA$^+$. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Workshop on Theory of Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 77–102. LNCS 736, Springer, 1993.

[216] L. Lamport. Introduction to TLA. Technical report, SRC Research Center, Dec. 1994. Technical Note.

[217] L. Lamport. Real-time model checking is really simple. In D. Borrione and W. Paul, editors, *Correct hardware design and verification methods, CHARME2005*, volume 3725 of *LNCS*, 2005.

[218] J. Laprie. Resilience for the scalability of dependability. In *Fourth IEEE International Symposium on Network Computing and Applications*, pages 5–6, 2005.

[219] K. G. Larsen and A. Legay. Statistical model checking past, present, and future - (track introduction). In *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications - 6th International Symposium, ISoLA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part II*, pages 135–142, 2014.

[220] K. G. Larsen, U. Nyman, and A. Wasowski. Modal I/O automata for interface and product line theories. In *Programming Languages and Systems, 16th European Symposium on Programming, ESOP 2007, Held as Part of the Joint European Conferences on Theory and Practics of Software, ETAPS 2007, Braga, Portugal, March 24 - April 1, 2007, Proceedings*, pages 64–79, 2007.

[221] K. G. Larsen and B. Thomsen. A modal process logic. In *Proceedings of the Third Annual Symposium on Logic in Computer Science (LICS '88), Edinburgh, Scotland, UK, July 5-8, 1988*, pages 203–210, 1988.

[222] K. G. Larsen and L. Xinxin. Equation solving using modal transition systems. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990*, pages 108–117, 1990.

[223] R. Lassaigne and S. Peyronnet. Approximate verification of probabilistic systems. *Process Algebra and Probabilistic Methods: Performance Modeling and Verification*, 2399:277–295, 2002.

[224] M. Latteux and D. Simplot. Context-sensitive string languages and recognizable picture languages. *Information and Computation*, 138:160–169, 1997.

[225] M. Lavielle. *Mixed Effects Models for the Population Approach: Models, Tasks, Methods and Tools*. Chapman and Hall/CRC, 2014.

[226] G. T. Leavens, K. R. M. Leino, and P. Müller. Specification and verification challenges for sequential object-oriented programs. *Formal Aspects of Computing*, 19(2):159–189, 2007.

[227] K. R. M. Leino and P. Müller. A verification methodology for model fields. In P. Sestoft, editor, *Programming Languages and Systems*, volume 3924 of *Lecture Notes in Computer Science*, pages 115–130. Springer Berlin Heidelberg, 2006.

[228] M. Leuschel and M. J. Butler. ProB: an automated analysis toolset for the B method. *STTT*, 10(2):185–203, 2008.

[229] M. Leuschel, J. Falampin, F. Fritz, and D. Plagge. Automated property verification for large scale B models. In *FM*, volume 5850 of *LNCS*, pages 708–723. Springer, 2009.

[230] K. Lindgren, C. Moore, and M. Nordahl. Complexity of two-dimensional patterns. *Journal of statistical physics*, 91(5-6):909–951, 1998.

[231] B. Liskov and L. Shrira. Promises: Linguistic support for efficient asynchronous procedure calls in distributed systems. In D. S. Wise, editor, *Proc. SIGPLAN Conference on Programming Language Design and Implementation (PLDI'88)*, pages 260–267. ACM Press, June 1988.

[232] B. Liskov and J. M. Wing. A behavioral notion of subtyping. *ACMTransactions on Programming Languages and Systems*, 16(6):1811–1841, Nov. 1994.

[233] Z. Liu and M. Joseph. Specification and verification of fault-tolerance, timing, and scheduling. *ACM Trans. Program. Lang. Syst.*, 21(1):46–89, 1999.

[234] I. Lopatkin, A. Iliasov, A. Romanovsky, Y. Prokhorova, and E. Troubitsyna. Patterns for representing FMEA in formal specification of control systems. In *IEEE 13th International Symposium on High-Assurance Systems Engineering*, pages 146–151. IEEE, 2011.

[235] C. Luo and S. Qin. Separation logic for multiple inheritance. *Electronic Notes in Theoretical Computer Science*, 212:27–40, 2008.

[236] N. Lynch and F. Vaandrager. Forward and backward simulations - part ii: Timing-based systems. *Information and Computation*, 128, 1995.

[237] P. Masci, A. Ayoub, P. Curzon, M. D. Harrison, I. Lee, and H. Thimbleby. Verification of interactive software for medical devices: PCA infusion pumps and FDA regulation as an example. In *EICS 2013, Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 81–90. ACM New York, NY, USA, 2013.

[238] P. Masci, R. Rukšėnas, P. Oladimeji, A. Cauchi, A. Gimblett, Y. Li, P. Curzon, and H. Thimbleby. On formalising interactive number entry on infusion pumps. *Electronic Communications of the EASST*, 45, 2011.

[239] W. Materi and D. S. Wishart. Computational systems biology in drug discovery and development: methods and applications. *Drug Discovery Today*, 12(7):295–303, 2007.

[240] S. Matsuoka and A.Yonezawa. Analysis of inheritance anomaly in object-oriented concurrent programming languages. In G. Agha, P. Wegner, and A. Yonezawa, editors, *Research Directions in Concurrent Object-Oriented Programming*, pages 107–150. The MIT Press, 1993.

[241] C. McDowell and O. Owe. Towards a light-weight approach for concurrent active objects in java, 2015. Submitted for publication.

[242] A. McIver, C. Morgan, and E. Troubitsyna. The probabilistic steam boiler: a case study in probabilistic data refinement. In *IRW/FMP 98,* Proceedings of, 1998.

[243] J. McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 79–93. IEEE Computer Society, 1994.

[244] H. Messer, A. Zinevich, and P. Alpert. Environmental monitoring by wireless communication networks. *Science*, 312:713, 2006.

[245] B. Meyer. *Object-Oriented Software Construction*. Prentice Hall, 1988.

[246] B. Meyer. Design by contract: The Eiffel method. In *Proceedings of Tools (26)*, page 446. IEEE Computer Society, 1998.

[247] R. Milner. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence. London, UK, September 1971.*, pages 481–489, 1971.

[248] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[249] A. Mizera, E. Czeizler, and I. Petre. Self-assembly models of variable resolution. *LNBI Transactions on Computational Systems Biology*, 7625:181–203, 2011.

[250] C. Morgan, A. McIver, K. Seidel, and J. Sanders. Refinement-oriented probability for CSP. *Formal Aspects of Computing*, 8(6):617–647, 1996.

[251] C. C. Morgan. *Programming from Specifications*. Prentice Hall, 1990.

[252] J. M. Morris. A theoretical basis for stepwise refinement and the programming calculus. *Science of Computer Programming*, 9(3):287–306, Dec. 1987.

[253] E. Murphy, V. Danos, J. Feret, J. Krivine, and R. Harmer. *Elements of Computational Systems Biology*, chapter Rule Based Modelling and Model Refinement, pages 83–114. Wiley Book Series on Bioinformatics. John Wiley & Sons, Inc., 2010.

[254] H. R. Nielson. *Hoare-Logic for Run-Time Analysis of Programs*. PhD thesis, Edinburgh University, 1984.

[255] H. R. Nielson. A Hoare-like proof-system for run-time analysis of programs. *Science of Computer Programming*, 9, 1987.

[256] P. Oladimeji, H. Thimbleby, and A. Cox. Number entry and their effects on error detection. In P. Campos et al., editors, *Interact 2011*, number 6949 in Lecture Notes in Computer Science, pages 178–185. Springer Verlag, 2011.

[257] E.-R. Olderog and C. A. R. Hoare. Specification-oriented semantics for communicating processes. *Acta Informatica*, 23:9–66, 1986.

[258] P. Olsen, J. Foederer, and J. Tretmans. Model-based testing of industrial transformational systems. In *Testing Software and Systems - 23rd IFIP WG 6.1 International Conference, ICTSS 2011, Paris, France, November 7-10, 2011. Proceedings*, pages 131–145, 2011.

[259] O. Owe and I. Ryl. On combining object orientation, openness and reliability. In *Proc. of the Norwegian Informatics Conference (NIK'99)*. Tapir, Nov. 1999.

[260] O. Owe and I. Ryl. Reasoning control in presence of dynamic classes. In *12th Nordic Workshop on Programming Theory*, Bergen, 2000. On-line proceedings `http://www.ii.uib.no/~nwpt00`.

[261] M. J. Parkinson and G. M. Bierman. Separation logic, abstraction and inheritance. *SIGPLAN Not.*, 43(1):75–86, Jan. 2008.

[262] D. L. Parnas and J. Madey. Functional documents for computer systems. *Sci. Comput. Program.*, 25(1):41–61, 1995.

[263] M. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 2014.

[264] I. Pereverzeva, E. Troubitsyna, and L. Laibinis. Formal goal-oriented development of resilient MAS in Event-B. In *Proceedings of the Ada-Europe 2012 – 17th International Conference on Reliable Software Technologies*, number LNCS 7308, pages 147–161. Springer-Verlag, 2012.

[265] I. Petre, A. Mizera, C. L. Hyder, A. Meinander, A. Mikhailov, R. I. Morimoto, L. Sistonen, J. E. Eriksson, and R.-J. Back. A simple mass-action model for the eukaryotic heat shock response and its mathematical validation. *Natural Computing*, 10(1):595–612, 2011.

[266] C. Pierik and F. S. de Boer. A proof outline logic for object-oriented programming. *Theoretical Computer Science*, 343(3):413–442, 2005.

[267] D. Plagge and M. Leuschel. Seven at one stroke: LTL model checking for high-level specifications in B, Z, CSP, and more. *STTT*, 12(1):9–21, 2010.

[268] A. Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010.

[269] K. Pohl, G. Böckle, and F. J. v. d. Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[270] A. Popa, A. Sofronia, and G. Stefanescu. High-level structured interactive programs with registers and voices. *J. UCS*, 13:1722–1754, 2007.

[271] C. Priami. Stochastic pi-calculus. *Computer Journal*, 38(7):578–589, 1995.

[272] C. Priami. Algorithmic systems biology. *Communications of the ACM*, 52(5):80–88, 2009.

[273] P. Puschner and A. Burns. A review of worst-case execution time analysis (editorial). *Real-Time Systems*, 18(2/3):115–128, 2000.

[274] S. Quinton and S. Graf. Contract-based verification of hierarchical systems of components. In *Proc. of SEFM 08*, pages 377–381. IEEE Computer Society, 2008.

[275] K. Raman and N. Chandra. Systems biology. *Resonance*, 15(2):131–153, 2010.

[276] F. Redmill, M. Chudleigh, and J. Catmur. *System Safety: HAZOP and Software HAZOP*. Wiley, 1999.

[277] A. Regev and E. Shapiro. Cellular abstractions: Cells as computation. *Nature*, 419(6905), 2002.

[278] C. Rieger, D. Gertman, and M. McQueen. Resilient control systems: Next generation design research. In *Proceedings of Human System Interactions*, pages 632–636, 2009.

[279] RODIN modularisation plug-in. Documentation at http://wiki.event-b.org/index.php/Modularisation_Plug-in.

[280] M. Rönkkö, V. Kotovirta, and M. Kolehmainen. Classifying environmental monitoring systems. In *Environmental Software Systems. Fostering Information Sharing*, number IFIP Advances in Information and Communication Technology, Volume 413, pages 533–452, 2013.

[281] M. Rönkkö and X. Li. Linear hybrid action systems. *Nordic Journal of Computing*, 8(1):159–177, 2001.

[282] M. Rönkkö and A. Ravn. Action systems with continuous behavior. In P. Antsakilis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, editors, *Hybrid System V*, volume 1567 of *Lecture Notes in Computer Science*, pages 304–323. Springer, 1999.

[283] M. Rönkkö, A. Ravn, and K. Sere. Hybrid action systems. *Theoretical Computer Science*, 290:937–973, 2003.

[284] M. Rönkkö and K. Sere. Refinement and continuous behaviour. In *Proceedings of Hybrid Systems: Computation and Control*, number LNCS 1569, pages 223–237. Springer-Verlag, 1999.

[285] M. Rönkkö, M. Stocker, M. Neovius, L. Petre, and M. Kolehmainen. Designing resilience mediators for control systems. In *Proceedings of the IASTED International Conference on Modelling, Identification and Control*, pages 147–154. ACTA Press, 2014.

[286] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1998.

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

[288] R. Rukšėnas, P. Masci, M. D. Harrison, and P. Curzon. Developing and verifying user interface requirements for infusion pumps: A refinement approach. *Electronic Communications of the EASST*, 69, 2013.

[289] R. K. Runde, Ø. Haugen, and K. Stølen. Refining UML interactions with explicit and implicit nondeterminism. *Nordic Journal of Computing*, 12:157–158, 2005.

[290] R. K. Runde, A. Refsdal, and K. Stølen. Relating computer systems to sequence diagrams: the impact of underspecification and inherent nondeterminism. *Formal Aspects of Computing*, 25(2):159–187, 2013.

[291] A. Salomaa. Two complete axiom systems for the algebra of regular events. *Journal of the ACM (JACM)*, 13(1):158–169, 1966.

[292] M. R. Sarshogh and M. Butler. Specification and refinement of discrete timing properties in Event-B. Technical report, Electronic and Computer Science, University of Southampton, 2011.

[293] A. Schäfer. Combining real-time model-checking and fault tree analysis. In *FME 2003: Formal Methods*, volume 2805 of *Lecture Notes in Computer Science*, pages 522–541. Springer Berlin Heidelberg, 2003.

[294] F. B. Schneider. Enforceable security policies. *ACM Transactions on Information and Systems Security*, 3(1):30–50, Feb. 2000.

[295] S. Schneider, H. Treharne, and H. Wehrheim. A CSP account of Event-B refinement. In *Proceedings 15th International Refinement Workshop, Refine 2011, Limerick, Ireland, 20th June 2011.*, pages 139–154, 2011.

[296] S. Schneider, H. Treharne, and H. Wehrheim. The behavioural semantics of Event-B refinement. *Formal Asp. Comput.*, 26(2):251–280, 2014.

[297] S. Schneider, H. Treharne, H. Wehrheim, and D. Williams. Managing LTL properties in Event-B refinement. arXiv:1406:6622, June 2014.

[298] M. Schoeberl and R. Pedersen. WCET analysis for a Java processor. In *Proceedings of the 4th International Workshop on Java Technologies for Real-Time and Embedded Systems (JTRES'06)*, pages 202–211, 2006.

[299] P. Selinger. *A survey of graphical languages for monoidal categories*, volume Lecture Notes in Physics 813, pages 289–355. Springer, 2011.

[300] K. Sere. *Stepwise derivation of parallel algorithms*. PhD thesis, Åbo Akademi, Department of computer science, 1990.

[301] K. Sere and E. Troubitsyna. Safety analysis in formal specication. In J. M. Wing, J. Woodcock, and J. Davies, editors, *FM 99 – Formal Methods*, volume 1709 of *Lecture Notes in Computer Science*, pages 1564–1583. Springer Berlin Heidelberg, 1999.

[302] K. Sere and M. Waldén. Reverse engineering distributed algorithms. *Journal of Software maintenance: Research and Practice*, 8:117–144, 1996.

[303] L. Shargel, A. Yu, and S. Wu-Pong. *Applied Biopharmaceutics & Pharmacokinetics*. McGraw Hill Professional, 6th edition, 2012.

[304] A. C. Shaw. Reasoning about time in higher-level language software. *IEEE Transactions on Software Engineering*, 15(7):875–889, 1989.

[305] M. Shaw. A formal system for specifying and verifying program performance. Technical Report CMU-CS-79-129, Carnegie Mellon University, June 1979.

[306] R. Silva and M. Butler. Supporting reuse mechanisms for developments in event-b: Composition. Technical report, University of Southampton, 2009.

[307] R. Silva and M. Butler. Shared event composition/decomposition in Event-B. In B. K. Aichernig, F. S. de Boer, and M. M. Bonsangue, editors, *Formal Methods for Components and Objects*, volume 6957 of *Lecture Notes in Computer Science*, pages 122–141. Springer, 2012.

[308] M. Sitaraman. Compositional performance reasoning. In *Proceedings of the Fourth ICSE Workshop on Component-Based Software Engineering: Component-Certification and System Prediction*, may 2001.

[309] M. Sitaraman, G. Kulczycki, J. Krone, W. F. Ogden, and A. L. N. Reddy. Performance specification of software components. In *ACM Sigsoft Symposium on Software Reuse*, 2001.

[310] J. Skön, M. Johansson, O. Kauhanen, M. Raatikainen, K. Leiviskä, and M. Kolehmainen. Wireless building monitoring and control system. *World Academy of Science, Engineering and technology*, 65:706–711, 2012.

[311] A. M. Smith, W. Xu, Y. Sun, J. R. Faeder, and G. E. Marai. RuleBender: integrated modeling, simulation and visualization for rule-based intracellular biochemistry. *BMC Bioinformatics*, 13(Suppl 8):S3, 2012.

[312] C. Snook and M. Butler. UML-B: A plug-in for the Event-B tool set. In *Proceedings of the 1st International Conference on Abstract State Machines, B and Z, ABZ'08*, page 344. Springer-Verlag, 2008.

[313] N. Song, S. Zhang, and C. Liu. Overview of factors affecting oral drug absorption. *Asian Journal of Drug Metabolism and Pharmacokinetics*, 4:167–176, 2004.

[314] N. Soundararajan. A proof technique for parallel programs. *Theoretical Computer Science*, 31(1-2):13–29, May 1984.

[315] G. Stefanescu. *Feedback Theories (A Calculus for Isomorphism Classes of Flowchart Schemes).* Number 24 in Preprint Series in Mathematics. INCREST, 1986. Also in: *Revue Roumaine de Mathematiques Pures et Applique*, 35:73–79, 1990.

[316] G. Stefanescu. On flowchart theories: Part II. The nondeterministic case. *Theoretical Computer Science*, 52(3):307–340, 1987.

[317] G. Stefanescu. *Network algebra.* Springer Verlag, 2000.

[318] G. Stefanescu. Interactive systems: From folklore to mathematics. In *Relmics'01*, pages 197–211. LNCS 2561, Springer, 2002.

[319] G. Stefanescu. Interactive systems with registers and voices. *Draft*, 2004. National University of Singapore, 2004.

[320] G. Stefanescu. Interactive systems with registers and voices. *Fundamenta Informaticae*, 73(1):285–305, 2006.

[321] M. Stoelinga and F. Vaandrager. Root contention in ieee 1394. In J.-P. Katoen, editor, *Formal Methods for Real-Time and Probabilistic Systems*, pages 53–74. Springer Berlin Heidelberg, 1999.

[322] L. Strigini. Resilience assessment and dependability benchmarking: Challenges of prediction. In *Proceedings of DSN Workshop Resilience Assess. Depend. Benchmarking.*, 2008.

[323] W. Su, J.-R. Abrial, and H. Zhu. Formalizing hybrid systems with event-b and the rodin platform. *Science of Computer Programming*, 94:164–202, 2014.

[324] C. A. Szyperski. *Component Software.* Addison Wesley, 1998.

[325] T. Taibi and D. Ngo. Information and software technology. *Formal specification of design pattern combination using BPSL*, 45(3):157–170, 2003.

[326] A. Tarasyuk, E. Troubitsyna, and L. Laibinis. From formal specification in Event-B to probabilistic reliability assessment. In *Dependability (DEPEND), 2010 Third International Conference on*, pages 24–31, July 2010.

[327] A. Tarasyuk, E. Troubitsyna, and L. Laibinis. Formal Modelling and Verification of Service-Oriented Systems in Probabilistic Event-B. In *IFM 2012, Integrated Formal Methods*, pages 237–252. Springer, 2012.

[328] The RODIN platform. Online at http://rodin-b-sharp.sourceforge.net/.

[329] H. Thimbleby. Interaction walkthrough: evaluation of safety critical interactive systems. In G. Doherty and A. Blandford, editors, *Interactive Systems: Design, Specification and Verification*, number 4323 in Lecture Notes in Computer Science, pages 52–66. Springer Verlag, 2007.

[330] A. Thums and G. Schellhorn. Model checking FTA. In *FME 2003: Formal Methods*, volume 2805 of *Lecture Notes in Computer Science*, pages 739–757. Springer Berlin Heidelberg, 2003.

[331] K. Trivedi, K. Dong Seong, and R. Ghosh. Resilience in computer systems and networks. In *IEEE International Conference on Computer-Aided Design - Digest of Technical Papers*, pages 74–77. IEEE, 2009.

[332] E. Troubitsyna. Reliability assessment through probabilistic refinement. *Nordic Journal of Computing*, 6(3):320–342, 1999.

[333] E. Troubitsyna. *Stepwise Development of Dependable Systems*. PhD thesis, Ph.D. thesis No.29. TUCS - Turku Centre for Computer Science, 2000.

[334] E. Troubitsyna. Dependability-explicit engineering with Event-B: Overview of recent achievements. *CoRR*, abs/1210.7032, 2012.

[335] Y.-K. Tsay. Compositional verification in linear-time temporal logic. In J. Tiuryn, editor, *Proceedings of FoSSaCS 2000*, volume 1784 of *Lecture Notes in Computer Science*, pages 344–358. Springer, 2000.

[336] Safety requirements for the generic PCA pump. `http://rtg.cis.upenn.edu/gip-docs/Safety_Requirements_GPCA.doc`. Accessed: 04.04.2013.

[337] US Food and Drug Administration. Guidance for the content of premarket submissions for software contained in medical devices, May 2005.

[338] R. van Glabbeek. The linear time – branching time spectrum II; the semantics of sequential systems with silent moves (extended abstract). In E. Best, editor, Proceedings *CONCUR'93,* 4^{th} International Conference on *Concurrency Theory*, Hildesheim, Germany, August 1993, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 1993.

[339] R. van Glabbeek. The linear time - branching time spectrum I. The semantics of concrete sequential processes. In Bergstra et al. [53], pages 3–99.

[340] M. Verhoef, P. G. Larsen, and J. Hooman. Modeling and validating distributed embedded real-time systems with VDM++. In J. Misra, T. Nipkow, and E. Sekerinski, editors, *Proceedings of the 14th International Symposium on Formal Methods (FM'06)*, volume 4085 of *Lecture Notes in Computer Science*, pages 147–162. Springer, 2006.

[341] J. von Wright and K. Sere. Program transformations and refinements in hol. In *International Workshop on the HOL Theorem Proving System and Its Applications*, pages 231–239. IEEE, 1991.

[342] M. Waldén and K. Sere. Reasoning about action systems using the B-Method. *Formal Methods in System Design*, 13:5–35, 1998.

[343] A. Wang. Generalized types in high-level programming languages. Research Report in Informatics 1, Institute of Mathematics, University of Oslo, 1974. Cand. Real thesis.

[344] F. Wang. Efficient verification of timed automata with bdd-like data structures. *International Journal on Software Tools for Technology Transfer*, 6(1):77–97, 2004.

[345] R. Ward, J. Loftis, and G. McBride. The "data-rich but information-poor" syndrome in water quality monitoring. *Environmental Management*, 10(3):291–297, 1986.

[346] G. Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25(3):38–49, 1992.

[347] M. Williams, D. Cornford, L. Bastin, R. Jones, and S. Parker. Automatic processing, quality assurance and serving of real-time weather data. *Computers & Geosciences*, 37:351–362, 2011.

[348] A. Wills. Capsules and types in Fresco: Program verification in Smalltalk. In P. America, editor, *Proc. European Conference on Object-Oriented Programming (ECOOP)*, volume 512 of *Lecture Notes in Computer Science*, pages 59–76. Springer, 1991.

[349] J. Woodcock and J. Davies. *Using Z: Specification, Refinement, and Proof*. Prentice Hall, 1996.

[350] W. Xu, A. M. Smith, J. R. Faeder, and G. E. Marai. RuleBender: a visual interface for rule-based modeling. *Bioinformatics*, 27(12):1721–1722, 2011.

[351] S. Yeganefard and M. Butler. Structuring functional requirements of control systems to facilitate refinement-based formalisation. In *Proceedings of the 11th International Workshop on Automated Verification of Critical Systems (AVoCS 2011)*, volume 46. Electronic Communications of the EASST, 2011.

[352] A. Yonezawa, J.-P. Briot, and E. Shibayama. Object-oriented concurrent programming in ABCL/1. In *Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'86). Sigplan Notices*, 21(11):258–268, Nov. 1986.

[353] M. Zhang, Z. Liu, C. Morisset, and A. P. Ravn. Design and verification of fault-tolerant components. In *Methods, Models and Tools for Fault Tolerance*, volume 5454 of *Lecture Notes in Computer Science*, pages 57–84. Springer Berlin Heidelberg, 2009.

[354] Q. Zhu and T. Basar. A dynamic game-theoretic approach to resilient control system design for cascading failures. In *Proceedings of International Conference on High Confidence Networked Systems*, pages 41–46, 2012.