Early Verification of Software Behavior in a Time Interval Framework

Paulo Sérgio Muniz Silva

Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, 158, trav. 3 05508-900 São Paulo SP, Brazil
e-mail: paulo.muniz@poli.usp.br

Abstract

Verification of software behavior in early moments of the development process is not an easy task. Typically, software engineers draw diagrams to reason about the software functional behavior, for example, drawing Message Sequence Charts (MSCs). Not always does the intended behavior described by MSCs correspond to their actual behavior. To help the verification of the actual behavior of MSCs, this paper describes an interpretation of (basic) MSCs in a temporal framework that formally represents the qualitative, and possibly imprecise, temporal information conveyed in MSCs. The framework is an algebra of binary relations on time intervals, and provides rules of reasoning about the temporal information. The interpretation provides a polynomial-time solution to the verification problem and lays the foundations of a model checking tool.

Keywords: formal verification and validation, requirements model checking, temporal reasoning.

1. Introduction

Verification of the software system behavior when software engineers work on the requirements or analysis viewpoints is not an easy task. In early moments of software development, software engineers largely draw diagrams to reason about the behavior of the software that will be built. For example, when working with use case models [12], from the analysis viewpoint, software engineers typically draw sequence diagrams [24] to depict the realization of scenarios for the use cases flows of events, which show interactions between domain objects. With the exception of real-time software engineers, the majority of software engineers do not accomplish the verification of the temporal properties of these diagrams, even if the solution for the software system that will be built requires distributed and concurrent architectural styles. An important fact that contributes to this situation is that traditional verification and validation tools are not of much help when faced with the partial information available at these early stages [9]. How to rigorously verify the temporal behavior of software systems initially defined in these intuitive diagrams?

The aforementioned sequence diagram is based on the Message Sequence Chart (MSC) artifact [24], which is a well-known visual and intuitive tool largely used to model the behavior of systems, representing sequences of events exchanged between system abstractions. Nowadays, it is adopted within a wide range of system and software developing methods. The International Telecommunication Union (ITU) formally defined an MSC standard [11]. We are interested in a particular interpretation of MSCs that analyzes their temporal behavior in a pure qualitative and intuitive manner, without assigning any timing

values or considering any underlying architectural constraints, while preserving their intuitive simple drawings. Our motivation stems from the fact that when intuitively describing the behavior of the problem domain events, we often reason qualitatively. For example, we usually say: "*X* sends the message *M2* to *Y* **after** receiving the message *M1* from *Z*", instead of assigning particular time values to the sending and the receiving events. Also important is that these kinds of events last a certain time, allowing us to say: "*X* sends the message *M4* to *W* **during** the reception of the message *M3* from *Z*". In other words, we are solely interested in the purely qualitative temporal analysis of the software behavior and would like to have a straight semantics to capture this qualitative, and often imprecise, temporal information conveyed in the diagrams.

An appropriate temporal ontology that directly supports these temporal notions is Allen's time interval theory [1]. Allen's theory was developed in the context of the so-called 'Naive Physics' with its common-sense representations, by taking the notion of the interval of time as a primitive one. It has been popular in natural language understanding [2], planning [3], knowledge representation [30], and in other fields of AI research. Allen's theory is an algebra of binary relations on intervals, carrying qualitative temporal information and allowing a formal reasoning about such information. Our interpretation of MSCs is defined within such a time framework.

On the other hand, software requirements can be viewed as a descriptive theory of an application domain [25] and MSCs are a useful and simple tool to model the temporal properties of this descriptive theory. Our proposal is to present a particular interpretation of MSCs that supports the generation of consequences of their temporal properties, in such a manner that we can formally check the potential conflicts between the intended and the actual behavior in Allen's temporal framework, from the requirements or analysis viewpoints. The verification process shall completely disregard the architectural constraints on the computational solution that will be built, to be consistent with the basic assumption that the verification is performed from the requirements or analysis views, and not from the design view. Lastly and importantly, the presented formal model lays the foundation of a model checking tool. This verification tool will help the verification process, automatically checking for conflicts between the actual semantics of the temporal behavior of an MSC and its intended behavior modeled by a software engineer.

A preliminary version of these ideas appeared in [19]. The present paper completely reformulates the original interpretation of MSCs, simplifies the interpretation and, most important, proposes a tractable solution for the verification problem, overcoming the complexity limitations of the former model. Section 2 summarizes the verification methodology. Section 3 presents the proposed interpretation of MSCs. Section 4 presents two simple worked examples. Section 5 presents some conclusions and our current related research.

Related Work. MSCs have been extensively analyzed in the last years. Current trends in the analysis of MSCs take various approaches: process algebra [10, 16, 17] and varieties of model checking [4, 5, 6, 7, 13], to mention just a few. Studies working with partially ordered event structures derived from MSCs [20, 21] are of special interest. However, these partially ordered structures are studied from the design view. Our encoding in the form of Allen's theory is another model for the semantics of the MSC that makes no assumption about architectural or design constraints (for example, it does not make any assumption about the semantics of message passing, e.g. first in first out). The complexity of the solution for the verification problem is similar to that later related works, but it presents the outcomes in the

form of qualitative temporal relations, as the first intuitions about the functional requirements behavior are used to be.

2. The Verification Methodology

Firstly, we will present a brief overview of MSCs, quoting [5]. MSCs are a graphical representation that shows message exchanges between concurrent process abstractions within a system. Figure 1(a) shows a *basic* MSC [11], the MSC type we will use as a model for the intended software system behavior.

Each vertical line has a start and an end symbol, and represents processes or autonomous agents (*P1*, *P2* and *P3*). Each horizontal arrow describes a message sent from one process to another (*a*, *b* and *c*). The tail of an arrow corresponds to the event of sending a message, whereas the head corresponds to its receipt. Communication is one-to-one and asynchronous, and control flows independently within each process from the start symbol to the end symbol. In each process, the events are temporally ordered from top to bottom. The system terminates when all processes have terminated.

The behavior of an MSC is the set of sequences of sent and received messages, i.e., MSCs represent the intended behavior by the order of the exchanged messages between processes. The intended order does not necessarily represent the actual semantics of the MSC. Conflicts are likely to happen. For example, it is not hard to see that there is a scenario for the MSC of Figure 1 in which message *c* arrives earlier than message *a* at process *P1*, conflicting with the intended order.

Allen's temporal structure [1] captures two aspects of particular interest: the strict relative temporal knowledge (e.g. “X happens before Y”, “X happens during Y”, etc.) and the uncertainties of the information about the relationship between two events in time. The temporal structure is a simple and linear model of time. The original theory has the time interval as a primitive. Five axioms of the temporal structure and a complete set of thirteen intuitive binary relations between intervals - the Allen's basic relations - are defined. Figure 1(b) depicts these relations.

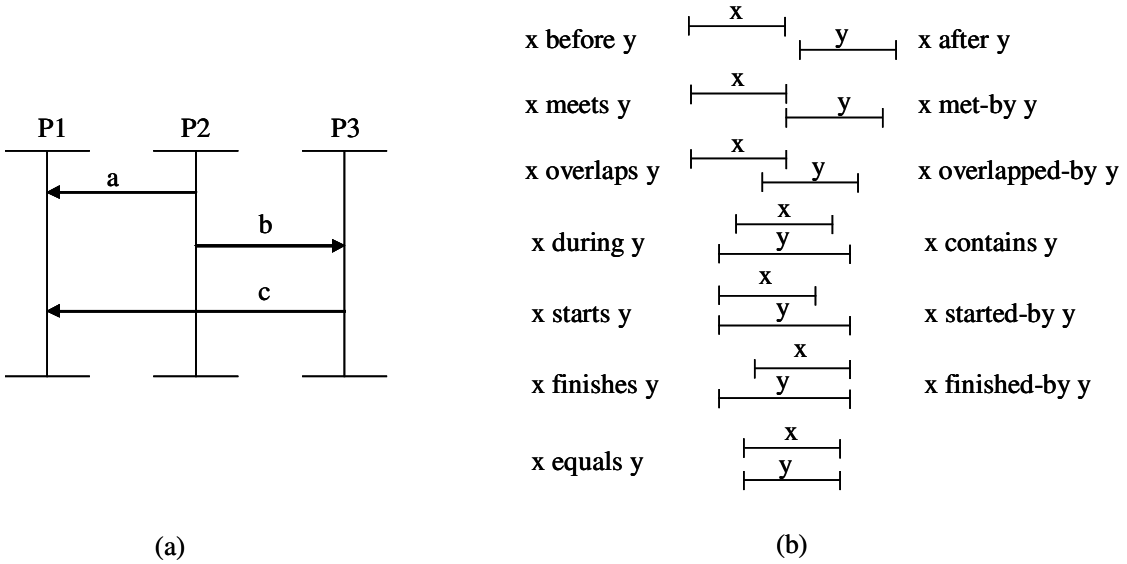

Figure 1: A Basic MSC and Allen's Basic Interval Relations

The verification methodology for a basic MSC is based on the following steps:

1. Produce the basic Message Sequence Chart that models an application domain intended behavior. The model results from the analyst's effort, and an adequate available tool can support MSCs drawings.
2. Translate the MSC into a *labeled MSC* [4], defined in section 3. A model checking tool may carry out this translation automatically.
3. Provide an interpretation based on a time interval framework for the resulting (step 2) structure. In section 3 we present the interpretation model. The interpretation builds a network structure with nodes denoting time intervals and edges denoting temporal relations between time intervals. The network structure constitutes the descriptive theory of the intended behavior. A model checking may carry out this step automatically.
4. Analyze the resulting interpretation. The network built in phase 3 is a generative system[1]. Its solution provides all the consequent temporal scenarios, which may show hidden and occasionally undesirable behaviors unforeseeable upon the making of the original MSCs. In other words, a model checking tool solves the network automatically and the analyst can verify if there are conflicts between the actual semantics of an MSC and its visual order. The analysis uses only the intuitive Allen's basic relations. Finally, the detected undesirable behavior needs to be validated with stakeholders to evaluate if it actually is a problem to the future system that will be built.

3. The Interpretation of MSCs

We will define a formal structure, the *basic labeled MSC*, which captures the essential properties of the basic MSC [11]. We borrow the definition of the labeled MSC from [4], and define some required additional details[2].

Definition 1 *Let* $P = \{P_1,\ldots,P_n\}$ *be the set of* processes*, and M be the set of* messages*. Let the label* $!(i, j, a)$ *denote the event "process* P_i *sends the message* a *to process* P_j*". Let the label* $?(i, j, a)$ *denote the event "process* P_i *receives the message* a *from process* P_j*". Define the set* $L_S = \{!(i, j, a) \mid i,j \in \{1,\ldots,n\} \wedge a \in M\}$ *of* send labels*, the set* $L_C = \{?(i, j, a) \mid i, j \in \{1,\ldots,n\} \wedge a \in M\}$ *of* receive labels*, the set* $L = L_S \cup L_C$ *as the set of* event labels*, and the set* L_N *of* next process event labels*. A basic* labeled MSC *over processes P is defined by:*

- *A set E of events partitioned into a set S of sending events and a set C of receiving events.*
- *A mapping* $p : E \mapsto \{1,\ldots,n\}$ *that maps each event to a process on which it occurs.*
- *A bijective mapping* $f : S \mapsto C$ *between sending and receiving events, matching each sending event with its corresponding receiving event.*

---

[1] A generative system [25] is a theory consisting of a set of axioms and rules of inference capable of generating consequences of the theory.

[2] In [19] we took a completely different approach defining an intermediate structure to describe an MSC, the Message Flow Graph (MFG) [13]. We abandoned that approach since the MFG introduces a further complexity and requires the modeling of inessencial elements for our purposes.

- *A bijective mapping ne : $E \mapsto E$ that maps each event on a process to its consecutive event on the same process. Each process event is connected to a unique consecutive event in the same process. This mapping is called* next process event.
- *A mapping $l : E \mapsto L$ that labels each event such that $l(S) \subseteq L_S$ and $l(C) \subseteq L_C$. For consistency of labels, for all $s \in S$, if $l(s) = !(i,j,a)$ then $p(s) = i$ and $l(f(s)) = ?(i, j, a)$ and $p(f(s))=j$.*
- *A mapping $h : E \times E \mapsto L_C$, which labels the next process events on each process P_i.*

For each $i \in \{1,\ldots,n\}$, there is a total order $\preccurlyeq_i$ on the events of process P_i, that is, on the elements of $p^{-1}(i)$, such that the transitive closure of the relation $\preccurlyeq \doteq \cup_{i\in\{1,\ldots,n\}}\preccurlyeq_i \cup \{(s, f(s) \mid s \in S\}$ is a partial order on E. This partial order is called visual order.

The total order $\preccurlyeq_i$ denotes the temporal order of the events of process P_i. The partial order $\preccurlyeq$ denotes the visual order *of the MSC, enforcing the notion that "messages cannot travel back in time", and expresses the intended temporal behavior of the MSC.*

The partial order corresponding to the MSC of Figure 1 is depicted in Figure 2, where the nodes are sending and receiving events, and the edges are messages and next process events.

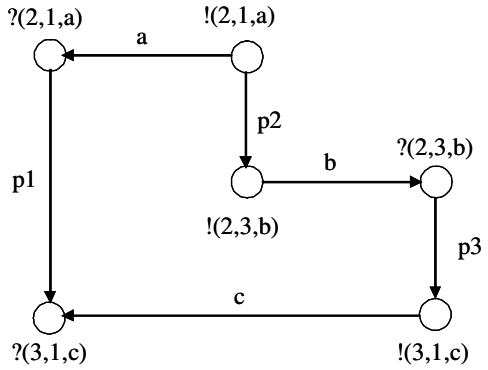

Figure 2: The partial order of the MSC of Figure 1

The interpretation of a basic labeled MSC is realized in a time interval framework.

Definition 2 *A* time interval *X is represented as a tuple (x^-, x^+), such that $x^- < x^+$, where x^- and x^+ are interpreted as points in a real line*[3]*. An interval interpretation* I-interpretation *is the mapping of time intervals to pairs of distinct real numbers such that the beginning of an interval is strictly before the end of the interval [22].*

Definition 3 *Let I be the set of all mutually exclusive basic relations {b, bi, m, mi, o, oi, s, si, d, di, f, fi, eq}, where b stands for before, bi for after, m for meets, mi for met-by, o for overlaps, oi for overlapped-by, s for starts, si for started-by, d for during, di for contains, f for finishes, fi for finished-by, and eq for equals. The relation between two time intervals is any*

[3] In Allen's original ontology [1], the intervals are primitive objects. They are not conceived in terms of their endpoints. For reasons that will become apparent right away, we will not follow that interpretation as we did in [19] but the classical approach which defines the timeinterval by their endpoints.

subset of I, representing a disjunction of the basic relations. The disjunction of all basic relations is denoted by $\top$ *and the empty relation is denoted by* $\bot$.

Allen's framework constitutes an algebra: the Allen's interval algebra. The algebra is based on the notion of relations between pairs of intervals. Under the I-interpretation, we can express the basic relations in terms of *endpoint relations*. For example, the relation *X overlaps Y* is equivalent to $x^- < y^-$, $x^- < y^+$, $x^+ > y^-$, $x^+ < y^+$. In order to define our interpretation, we have to consider the properties of Allen's interval algebra and of interval relations' networks.

Definition 4 *Allen's Interval Algebra* ***IA*** *is the algebra with underlying set* 2^I *(the power set of I), unary operator converse (denoted by* $^\cup$*), binary operator intersection (denoted by* $\cap$*) and binary operator composition (denoted by* $\circ$*). The intersection can be expressed as the set-theoretic intersection of the sets of basic relations. The composition is the union of the component-wise composition of the basic relations [22]. Allen provides a composition table for the basic relations [1]. Let* r *and* s *be relations between two intervals. The following expressions hold:*

$\forall X, Y\ (X\ r^\cup\ Y \Leftrightarrow Y\ r\ X)$

$\forall X, Y\ (X\ (r \cap s)\ Y \Leftrightarrow X\ r\ Y \wedge X\ s\ Y)$

$\forall X, Y\ (X\ (r \circ s)\ Y \Leftrightarrow \exists\ Z(X\ r\ Z \wedge Z\ s\ Y))$.

For example, we write $\{d, o, s\}$ to denote the disjunction of basic relations d, o and s. Therefore, $\{d, o\} \subset \{d, o, s\}$. Also, if X and Y are intervals and $X\ \{d, o, s\}\ Y$, then $Y\ \{di, oi, si\}\ X$. The proposed semantics of a labeled MSC interprets messages and next process event edges as time intervals, called *message* and *process intervals*, respectively. They are naturally defined in terms of the *I*-interpretation, i.e., by their endpoints. Sending and receiving events end a message interval, whereas a process interval is ended by any two consecutive events in a process. For the sake of simplicity, we will express the endpoint relations of these time intervals in terms of basic relations.

Definition 5 *In a basic labeled MSC we have two types of time intervals: the* message interval *and the* process interval.

- *Define the mapping* $u : E \mapsto \Re$*, where* $\Re$ *is the set of real numbers. The message interval is the tuple* (a, b)*, such that* $a < b$*, where* $a, b \in \Re$*, and for an* $s \in S$*, if* $u(s) = a$ *then* $u(f(s)) = b$.
- *Define the mapping* $v : E \mapsto \Re$*. The process interval is the tuple* (a, b)*, such that* $a < b$*, where* $a, b \in \Re$*,, and for* $e_1, e_2 \in E$ *and* $(e_1, e_2) \in ne$*, if* $v(e_1) = a$ *then* $v(e_2) = b$.

The visual order, which expresses the intended temporal behavior of a basic MSC, relates messages with each other through processes. Messages are sent and received by processes. Therefore, the relative occurrence of messages in the time dimension is determined by the order in which they are performed in processes. We can define the following fundamental patterns of relationships between a message interval and a process interval, depicted in Figure 3, where the symbol '*' denotes any process, the symbol '1' denotes the process of reference ("this process"), and the symbol '*ip*' denotes the label of the next process event on the process of reference.

Definition 6 *In the visual order of every basic labeled MSC there are six patterns of relationships between message and process intervals, depicted in Figure 3. A fundamental*

reasonable assumption is that a sending event is a controlled event in a process, which is only issued when a preceding event has occurred [5]. As a consequence, the visual order is not supposed to be guaranteed between pairs of receiving events, since they are not controlled events in a process. Patterns 5a and 5b (Figure 3) are a consequence of this assumption that prevents their merging into patterns 1 and 3, respectively.

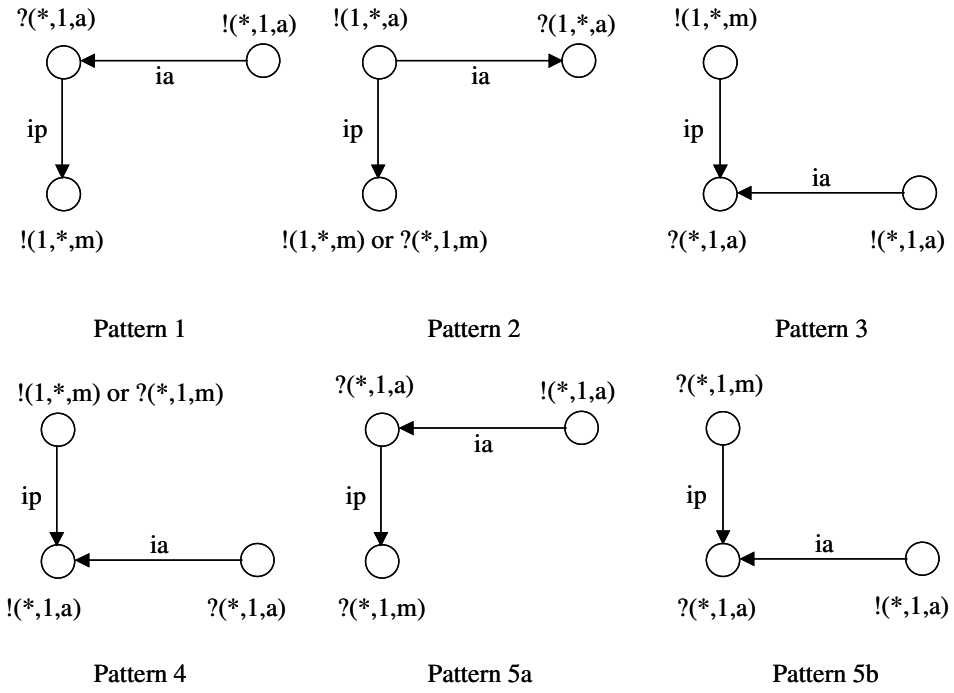

Figure 3: Basic patterns of a labeled MSC

The interpretation of the basic patterns means the evaluation of the relative positions between a message interval and the process interval. The essence of the behavior representation in an MSC are messages, therefore we are primarily interested in the relative positions of messages intervals. We proceed with the interpretation as follows: we fix the position of the process interval and vary the relative position of the message interval with respect to the process interval. Obviously, the mutual positions shall respect the patterns configurations. The comparison of the resulting mutual positions between the intervals with Allen's basic interval relations of Figure 1b gives the interval relations. Let *ia* be the message interval delimited by the sending and the receiving events of message *a*, and let *ip* be the process interval delimited by events of the next process event *p*. It is easy to see that:

- Pattern 1. The message interval *ia meets* the process interval *ip*, i.e., *ia* {*m*} *ip*.
- Pattern 2. The message interval *ia starts*, or *equals*, or *is-started-by* the process interval *ip*, i.e., *ia* {*s*, *si*, *eq*} *ip*.
- Pattern 3. The message interval *ia finishes*, or *equals*, or *is-finished-by* the process interval *ip*, i.e., *ia* {*f*, *fi*, *eq*} *ip*.
- Pattern 4. The message interval *ia is-met-by* the process interval *ip*, i.e., *ia* {*mi*} *ip*.
- Patterns 5a and 5b. As a consequence of definition 6, they present unknown temporal information, i.e. they may contain any interval relation, or *ia* ⊤ *ip*.

The set of message and process intervals from a labeled MSC, and the binary relations between each pair of them, constitute a network where the nodes are messages or process intervals and the edges are the interval relations corresponding to the basic pattern with which the message and process intervals match. We call this network the *Interval Calculus Network* (*ICN*). Figure 4 illustrates the derivation of an ICN from the labeled MSC of Figure 1, where *ia*, *ib* and *ic*, denote the message intervals to the messages *a*, *b* and *c*, respectively; and *ip1*, *ip2* and *ip3* denote the process intervals to the next process events *p1*, *p2* and *p3* of processes *P1*, *P2* and *P3*, respectively. Walking through all pairs of relations in the visual order of the MSC, we derive the following interval relations in the resulting ICN: *ia* $\top$ *ip*1, from pattern 5a; *ia* {*s*, *si*, *eq*} *ip2*, from pattern 2; *ib* {*mi*} *ip2*, from pattern 4; *ib* {*m*} *ip3*, from pattern 1; *ic* {*mi*} *ip3*, from pattern 4; *ic* $\top$ *ip1*, from pattern 5b.

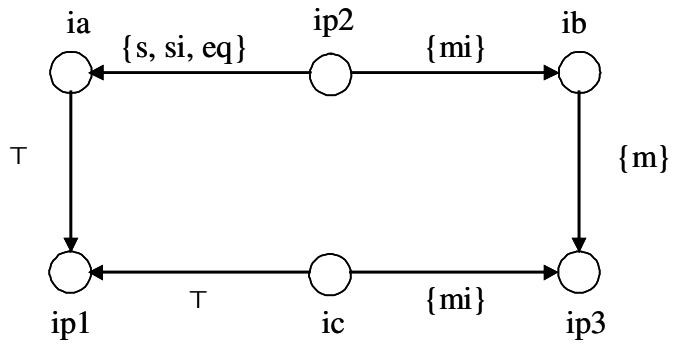

Figure 4: The resulting ICN

The ICN is a network of binary constraints whose edges are temporal relations between nodes. Let us define the *Interval Algebra Network* (***IA** network*). The following definition is quoted from [28].

Definition 7 *A network of binary constraints [18] is defined as a set X of n variables* $\{x_1, x_2, \ldots, x_n\}$*, a domain* D_i *of possible values for each variable, and binary constraints between variables. A binary constraint* C_{ij}*, between variables* x_i *and* x_j*, is a subset* $C_{ij} \subseteq D_i \times D_j$*. For networks built on Allen's framework, it is required that* $(x_j, x_i) \in C_{ji} \Leftrightarrow (x_i, x_j) \in C_{ij}$*. An* instantiation *of the variables in X is an n-tuple* $(X_1, X_2, \ldots, X_n)$*, representing the assignment of* $X_i \in D_i$ *to* x_i*. A* consistent instantiation *of a network is an instantiation of the variables such that the constraints between variables are satisfied. A network is* inconsistent *if no consistent instantiation exists.*

*An **IA** network is a network of binary constraints where the variables represent time intervals and the binary constraints between variables are represented implicitly by disjunctions of the basic relations.*

Proposition 1 *The ICN derived from a labeled MSC is an **IA** network.*

Proof. Let *M* and *N* denote the sets of message intervals and process intervals, respectively. Definition 5 states that their elements are time intervals. The ICN is a set of variables $\{x_1, x_2, \ldots, x_n\}$ with possible values taken from the domain $D = M \cup N$, that is, the values of the x_i are time intervals. The relations between the ICN variables are disjunctions of basic relations, since the derivation process of the ICN relations is carried out in accordance with the basic patterns, delivering interval relations. Consequently, the ICN is an **IA** network. ■

The fundamental reasoning problems in an **IA** network are [27]: find a scenario that is consistent with the information provided, and find the feasible relations between all pairs of intervals. We are interested in finding the feasible relations between all pairs of intervals mostly, that is, in finding the deductive consequences of the temporal knowledge represented in an MSC.

Definition 8 *A basic relation $r \in W$ is feasible with respect to a network W if and only if there exists a consistent instantiation of the network where r is satisfied [27]. Given an **IA** network W, the set of feasible relations between two variables x_i and x_j in the network is the set consisting of all and only the $r \in W$ that are feasible. The minimal network representation w, of a network W, is the network for which w_{ij} is the set of feasible relations between variables x_i and x_j in W, for every $i, j = 1,\ldots, n$. Determining the feasible relations in W can be viewed as determining the deductive consequences of the temporal knowledge.*

Constraint satisfaction techniques are used to solve the reasoning problem. For the **IA** network, finding a consistent scenario and finding the feasible relations are NP-complete problems and intractable in the worst case [27, 28]. In fact, Allen's algorithm [1] is an approximation solution for the network in the full algebra (**IA** Algebra). Subsequent research has focused on designing more efficient algorithms or identifying tractable special cases of the full algebra for which there are exact solutions [8, 14, 28]. The full **IA** algebra contains $2^{13} = 8192$ possible relations between intervals. Subclasses of the **IA** algebra are obtained by considering their subsets giving 2^{8192} subclasses [8]. In many applications the full algebra is not necessary and restricted classes of interval algebras have an exact solution to the temporal network[4].

We claim that the binary constraints of solved ICNs are a tractable subset of the **IA** algebra underlying set. In other words, we claim that the solution for the reasoning problems of the ICN is exact and tractable in polynomial time. In order to prove it we need some other definitions.

Definition 9 *Quoting [22], let a formula X $\{r_1,\ldots, r_n\}$ Y be called an* interval formula*. Such a formula is satisfied by an* I-interpretation $\Im$ *for some i, $1 < i < n$. Finite sets of interval formulas are denoted by* Θ*. The set* Θ *is* I-satisfiable *if and only if there exists an I-interpretation* $\Im$ *that satisfies every formula of* Θ*. Such a satisfying I-interpretation* $\Im$ *is called an* I-model *of* Θ*. If an interval formula* ϕ *is satisfied by every I-model of* Θ*,* ϕ *is logically implied by* Θ*, written* $\Theta \models_i \phi$*.*

The verification task requires the finding of the set of feasible relations between the intervals in an ICN. Finding a consistent scenario in an **IA** network is another way of saying whether there exists an I-model of Θ (abbreviated ISAT). Finding the feasible relations between all pairs of intervals is another way to say whether there exists the *strongest implied* relation between each pair of intervals *X*, *Y*, i.e., the smallest set *R* such that $\Theta \models_i X\ R\ Y$ (abbreviated ISI). It is a known fact [29] that ISAT(**IA**) is NP-complete, and also that ISAT and ISI are equivalent under polynomial Turing-reductions. This equivalence also extends to subclasses of **IA**, provided they contain all basic relations [22]. We are concerned about polynomial-time ISI mostly. Let us investigate if all possible binary constraints of solved ICN networks are a special tractable case of the **IA** algebra.

---

[4] The approach taken in [19] was intractable, since we worked in the full algebra.

Given a set Θ, we can generate the least subalgebra containing the relations r such that $r \in \Theta$ and which is closed under converse, intersection and composition, computing the closure with respect to conjunction and transitivity, for the set Θ of generating relations [22]. Closures can be computed using, for example, the `aclose` tool [23]. The resulting subalgebra is polynomially equivalent to the original class with respect to satisfiability.

Definition 10 *Let* $\mathbf{SA_B} \subseteq \mathbf{IA}$. *The* I-closure *of* $\mathbf{SA_B}$, *denoted by* $C_I(\mathbf{SA_B})$, *is the least subalgebra of* **IA** *containing* $\mathbf{SA_B}$ *and which is closed under converse, intersection and composition.*

Proposition 2 *Let* $\mathbf{SA_B} \subseteq \mathbf{IA}$. *The* $ISAT(\mathbf{SA_B})$ *is polynomial if and only if* $ISAT(C_I(\mathbf{SA_B}))$ *is polynomial.*

Proof. See [22]. ■

Definition 11 *Let* $\mathbf{SA_C} \subseteq \mathbf{IA}$ *be the algebra that can be translated into relations between the endpoints of the intervals using only the relations comprised by the* Continuous Point Algebra ($\mathbf{PA_C}$) *[28].* $\mathbf{SA_C}$ *has the underlying set* $\{\perp, \{<\}, \{<, =\}, \{=\}, \{>\}, \{>, =\}, \{<, =, >\}\}$, *and operators converse, intersection and composition.* $\mathbf{SA_C}$ *is also known as* Continuous Pointizable Interval Algebra.

Proposition 3 *Any path consistency algorithm [1, 15] finds* $ISAT(\mathbf{SA_C})$ *exactly and consequently* $ISI(\mathbf{SA_C})$ *in polynomial time.*

Proof. See [28].■

Now we can define the subalgebra $\mathbf{SA_{ICN}}$, which meets all the requirements for the binary constraints of the ICN, and prove that it is a tractable special case of the **IA** algebra.

Definition 12 *Let* $\mathbf{SA_B} \subseteq \mathbf{IA}$ *be the subset of relations* $\{\{m\}, \{mi\}, \{s, si, eq\}, \{f, fi, eq\}, \top\}$ *which are allowed to occur in the translation of a labeled MSC into an ICN. We define the subalgebra* $\mathbf{SA_{ICN}}$, *the least subalgebra of* ***IA*** *containing* $\boldsymbol{SA_B}$ *computed by the closure* $C_I(\mathbf{SA_B})$, *and which is closed under operators converse, intersection and composition.*

The underlying set of $\mathbf{SA_{ICN}}$, generated by the `aclose` tool [23], is enumerated in the appendix and contains all possible relations that occur in the binary constraints of solved ICN networks. We claim that our verification problem of an ICN (the ISI problem) is exact and tractable.

Proposition 4 *Any path consistency algorithm exactly finds* $ISI(\mathbf{SA_{ICN}})$ *in polynomial time.*

Proof. Let Θ be the set of interval formulas of $\mathbf{SA_{ICN}}$. It can be shown by inspection that $\Theta \subset \mathbf{SA_C}$ (the underlying set of $\mathbf{SA_C}$ is enumerated in [28]). As Θ contains all the basic relations, ISAT and ISI are equivalent under polynomial reduction. By Propositions 2 and 3 our claim immediately follows. ■

Now, we can define the ICN network formally.

Definition 13 *Let T be the set of intervals. Let* L_T *be the set of interval labels. Let R be the set of the interval relations of* $\mathbf{SA_B}$. *Let* L_R *be the set of ICN interval relation labels, whose members denote the interval relations of* $\mathbf{SA_B}$. *The ICN is a network defined by:*

- *A set M of* message intervals *and a set N of* next process event intervals *that partition the set T.*
- *A bijective mapping* $h : M \mapsto N$ *between message and next process event intervals.*

- *A mapping $z : T \mapsto L_T$ that labels each interval.*
- *A mapping $r : T \times T \mapsto L_R$ that labels each relation between intervals. r is called an* interval relation*.*
- *If there is an interval relation between two nodes with a particular set of relations, its* converse interval relation *is the set whose elements are the converse of each element from the former set.*

4. Simple worked examples

We will present two simple worked examples: the verification of the familiar MSC of Figure 1 and an example taken from [26].

The MSC of Figure 1. By applying our methodology to the MSC of Figure 1 we have:

- Step 1 The produced MSC is depicted in Figure 1.
- Step 2 The partial order of the labeled MSC is depicted in Figure 2.
- Step 3 The resulting ICN is depicted in Figure 4.
- Step 4 Solve the resulting ICN.

In the step 4, we are interested in computing the strongest implied relation between all intervals (ISI). To solve the ICN, we apply a path consistency algorithm described in the literature, for example in [14, 28], which usually requires $O(n^2)$ time (n is the number of intervals) for the computation of the Continuous Pointizable Interval Algebra ($\mathbf{SA_C}$). Here, we used the path consistency routines of the software tool `solve` [23] and found the following relations between messages *a*, *b* and *c*: R_1 is *ia* {*b*, *di*, *o*, *m*, *fi*} *ib*; R_2 is *ia* {*b*, *di*, *o*, *m*, *fi*} *ic*; R_3 is *ib* {*b*} *ic*.

For the sake of simplicity, let *!message* denote a sending event of a message in a process and *?message* denote a receiving event of a message in a process. The visual order of the MSC requires that: *?a* < *?c*, *!a* < *!b* and *?b* < *!c*. Expressing the interval relations in terms of endpoint relations, the interval relations R_1 and R_3 guarantee *!a* < *!b* and *?b* < *!c*, respectively. However, interval relation R_2 does not guarantee *?a* < *?c*. In fact, the basic interval relation $di \in R_2$ says that *?c* < *?a*, which contradicts the intended behavior expressed in the visual order. Of course, a stakeholder should evaluate if *?a* coming after *?c* is a problem for the application. If he or she concludes that this scenario must be avoided, the analysis could be carried further and investigate how we could guarantee the intended visual order in the system specification, for example, by specifying an adequate queuing model to the underlying architecture of the system that will be built.

Another example. A more interesting example is taken from [26]. Figure 5(a) depicts an MSC that specifies the general intended behavior for a simple system that controls employees entering a secure building. If we assume that the *Door*, the *Security System* and the *Camera* are autonomous systems, and the communication between them is asynchronous, we have a specification flaw. Figure 5(b) shows the correspondent partial order, without the event labels for the sake of space economy, and Figure 5(c) shows the resulting ICN.

Solving the ICN with the path consistency algorithm, we find the following interesting relations: *istartRecording* {*b*, *di*, *o*, *m*, *fi*} *iunlock*, and *istopRecording* {*bi*, *d*, *oi*, *mi*, *f*} *istartRecording*. The interval relation *istartRecording* {*di*} *iunlock* does not guarantee that the camera records when an employee enters the building (the reception of the *unlock* message

occurs before the reception of the *startRecording* message). The interval relation *istopRecording* {*d*} *istartRecording* says that the *Camera* stops recording before it starts recording! To overcome this situation, while preserving the asynchronous model requirement, we could modify the MSC by adding a message sent by the *Camera* that signals the *Security System* upon the receipt of the *startRecording* message. Only after that does the *Security System* send the unlock message to the *Door*. If we solve the modified MSC, we find *istartRecording* {*b*} *iunlock* and *istopRecording* {*bi*} *istartRecording* that guarantee the intended behavior.

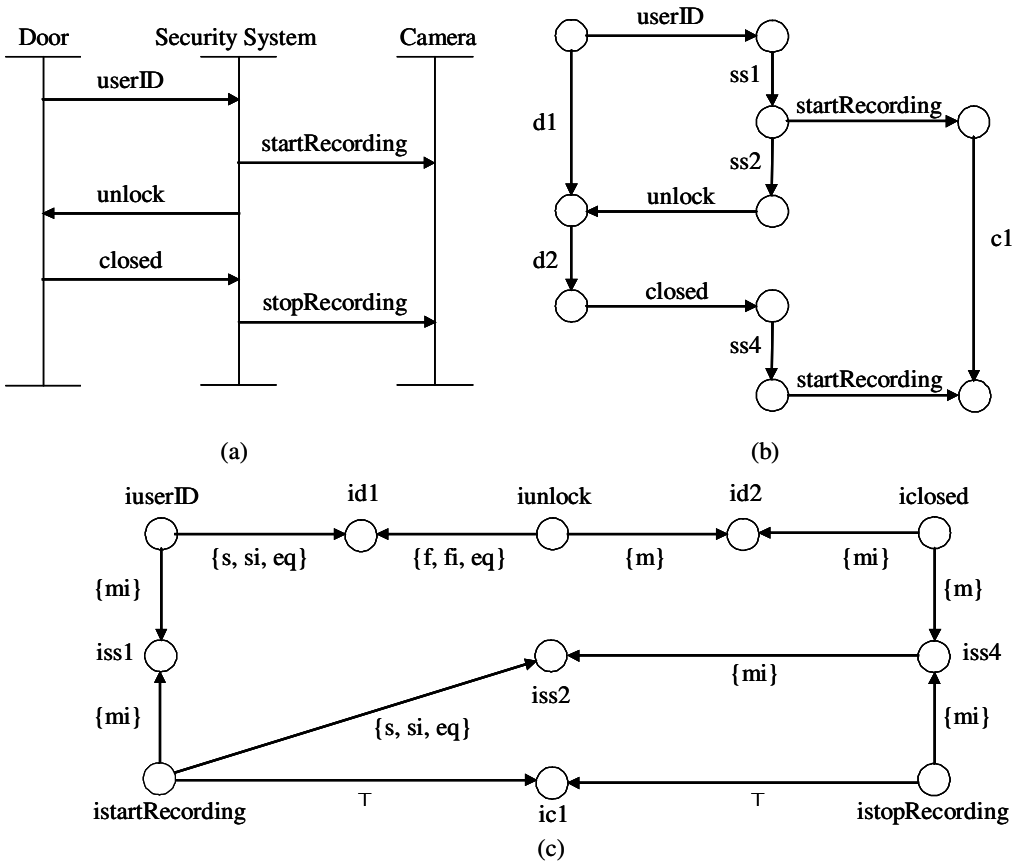

Figure 5: A simple secure building control MSC, its visual order and ICN

5. Conclusions

In this paper, we have presented a framework to formally verify the behavior of a software system from the requirements view, where traditional verification and validation tools work under great difficulties. Our approach focused on the temporal properties of simple and traditional MSCs in a purely qualitative manner, requiring neither extensions to the charts nor consideration to the underlying architecture of the system that will be built. The latter is of particular importance, since we are placed in the requirements viewpoint. Based on the verification results, software engineers have a rich set of alternatives to be considered in the design of the most appropriate underlying architecture that guarantees the software intended behavior. We proved that the present interpretation is made in a tractable subclass of the Allen's full algebra and that the verification task is exact and is carried out in polynomial time. The present interpretation is a guide to the making of a model checking tool.

Currently, we are working on two goals: (1) the integration of metric temporal reasoning (quantitative timing constraints) in our qualitative model to benefit from the expressive power of both approaches, also worked out in polynomial time; (2) the managing of MSC composition (multiple MSCs) through High-Level MSCs [11], used to describe more complex behaviors.

Appendix

The underlying set of subalgebra $\mathbf{SA_{ICN}}$ with its 30 elements is:

$\{\bot, \{eq\}, \{b\}, \{bi\}, \{d\}, \{di\}, \{o\}, \{oi\}, \{m\}, \{mi\}, \{s\}, \{si\}, \{f\}, \{fi\}, \{b, o, m\}, \{bi, oi, mi\}, \{d, o, s\}, \{di, oi, si\}, \{b, d, o, m, s\}, \{bi, di, oi, mi, si\}, \{eq, s, si\}, \{d, oi, f\}, \{di, o, fi\}, \{bi, d, oi, mi, f\}, \{b, di, o, m, fi\}, \{eq, f, fi\}, \{eq, d, di, o, oi, s, si, f, fi\}, \{eq, bi, d, di, o, oi, m, s, si, f, fi\}, \{eq, bi, d, di, o, oi, mi, s, si, f, fi\}, \top\}$.

Acknowledgments. Thanks to Nebel and Bürckert [23] for making available the software tools used in this paper.

References

[1] Allen, J. F. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832-843, 1983.

[2] Allen, J. F. Towards a general theory of action and time. *Artificial Intelligence*, 23(2), 123-154.

[3] Allen, J. F. Temporal reasoning and planning. In Allen, J. F. and Kautz, H. A. and Pelavin, R. N. and Teneberg, J. D., editors, *Reasoning about Plans*, pages 1-67. Morgan Kaufmann, 1991.

[4] Alur, R., Etessami, K. and Yannakakis, M. Inference of message sequence charts. In *22nd International Conference on Software Engineering*, pages 304-313, 2000.

[5] Alur, R., Holzmann, G. J. and Peled, D. An analyzer for message sequence charts. In Margaria, T. and Steffen, B., editors, *Tools and Algorithms for the Construction and Analysis of Systems*, number 1055 in LNCS, pages 35-48. Springer-Verlag, 1996.

[6] Alur, R. and Yannakakis, M. Model checking of message sequence charts. In *CONCUR '99: Concurrency Theory, Tenth International Conference*, number 1664 in LNCS, pages 114-129. Springer-Verlag, 1999.

[7] Ben-Abdallah, H. and Leue, S. Expressing and analyzing timing constraints in message sequence chart specifications. Technical Report 97-04, Department of Electrical and Computer Engineering, University of Waterloo, 1997.

[8] Drakengren, T. and Jonsson, P. Eight maximal tractable subclasses of Allen's algebra with metric time. *Journal of Artificial Intelligence Research*, 7:25-45, 1997.

[9] Holzmann, G. Early fault detection tools. In Margaria, T. and Steffen, B., editors, *Tools and Algorithms for the Construction and Analysis of Systems*, number 1055 in LNCS, pages 1-13. Springer-Verlag, 1996.

[10] ITU-T. *Recommendation Z.120 Annex B: Algebraic semantics of message sequence charts*. ITU-TS, Geneva, Swiss, April 1995.

[11] ITU-T. *Recommendation Z.120. Message sequence charts (MSC'96)*. ITU-TS, Geneva, Swiss, April 1996.

[12] Jacobson, I. et al. *Object-Oriented Software Engineering - a Use Case Driven Approach*. Addison-Wesley Publishing Co., London, 1992.

[13] Ladkin, P. B. and Leue, S. Interpreting message flow graphs. *Formal Aspects of Computing*, 7(5):473-509, 1995.

[14] Ladkin, P. B. and Maddux, R. On binary constraint problems. *Journal of the ACM*, 41(3):435-469, 1994.

[15] Mackworth, A. K. Consistency in networks of relations. *Artificial Intelligence*, 8:99-118, 1997.

[16] Mauw, S. and Reniers, M. A. An algebraic semantics of basic message sequence charts. *The Computer Journal*, 37(4):269-277, 1994.

[17] Mauw, S. and Reniers, M. A. Operational semantics for MSC'96. *Computer Networks and ISDN Systems*, 31(17):1785-1799, 1999.

[18] Montanari, U. Networks of constraints: fundamental properties and applications to picture processing. *Information Science*, 7:95-132, 1994.

[19] Muniz Silva, P. S. Extended message sequence charts with time-interval semantics. In *Proceedings of the Fifth International Workshop on Temporal Representation and Reasoning*, pages 37-44, Sanibel Island, FL, USA, 1998.

[20] Muscholl, A. and Peled, D. Message sequence graphs and decision problems in Mazurkiewicz traces. In *Proceedings of MFCS*, number 1672 in LNCS, pages 81-91. Springer-Verlag, 1999.

[21] Muscholl, A. and Peled, D. and Su, Z. Deciding properties for message sequence charts. In *Proceedings of FoSSaCS'98*, number 1378 in LNCS, pages 226-242. Springer-Verlag, 1998.

[22] Nebel, B. and Bürckert, H-J. Reasoning about temporal relations: a maximal tractable subclass of Allen's interval algebra. *Journal of the ACM*, 42(1):43-66, 1995.

[23] Nebel, B. and Bürckert, H-J. Software for machine assisted analysis of Allen's interval algebra. Available from `ftp.informatik.uni-freiburg/documents/papers/ki/allen-csp-solving.programs.tar.gz`, 1995.

[24] OMG. *OMG Unified Modeling Language Specification - version 1.4*. Object Management Group, Inc., USA, September 2001.

[25] Turski, W. M. and Maibaum, T. S. *The specification of computer programs*. Addison-Wesley Publishing Co., London, 1987.

[26] Uchitel, S. *Incremental Elaboration of Scenario-Based Specifications and Behaviour Models Using Implied Scenarios*. PhD thesis, Imperial College of Science, Technology and Medicine. University of London, Department of Computing, 2003.

[27] van Beek, P. G. Reasoning about qualitative temporal information. *Artificial Intelligence*, 58:297-326, 1992.

[28] van Beek, P. G. and Cohen, R. Exact and approximate reasoning about temporal relations. *Computational Intelligence*, 6(3):132-144, 1990.

[29] Vilain, M. B. and Kautz, H. A. Constraint propagation algorithms for temporal reasoning. In *Proceedings of the Fifth National Conference of the American Association for Artificial Intelligence*, pages 366-382, Philadelphia, PA, USA, 1986.

[30] Weida, R. and Litman, D. Terminological reasoning with constraint networks and an application to plan recognition. In Nebel, B., Swartout, W. and Rich, C., editors, *Principles of knowledge representation and reasoning: Proceedings of the Third International Conference*, pages 282-293, Cambridge, MA, USA, 1992.