in: I. Rozman, M. Pivka (eds.), Int. Conf. on Software Quality (ICSQ'95), Maribor (Slovenia), University of Maribor Press, pp. 55-64, 1995.

Compositional Object Specification and Verification*

Stefan Conrad

University of Magdeburg
Institute for Technical Information Systems
P.O.Box 4120, D-39016 Magdeburg, Germany
conrad@iti.cs.uni-magdeburg.de

Abstract

Formal specification and verification of software systems have gained more and more importance in the software development process by striving for better software quality, i.e., by guaranteeing correctness and reliability. In the area of information systems temporal logic is widely used for describing the dynamic behaviour of the system. Because information systems are state-based systems, the use of next operators for temporal specification is often required.

However, using a classical semantics of temporal logic we get problems in composing several object specifications to form a system. In order to overcome these problems we present a semantics for linear temporal logic allowing local stuttering. Thereby, compositional specification (and verification) is possible and follows the specifier's intuition, i.e., respecting for each object its own local next-time. On the one hand, compositionality is an indispensable prerequisite for reusing object specifications. On the other hand, we need compositionality for making formal verification feasible and easier to manage.

Keywords:

formal object specification, object verification, compositional specification, compositional verification, object-oriented modelling of information systems, temporal logic for objects.

1 Introduction

Temporal logic has become a frequently used and widely accepted means for specifying and verifying properties of concurrent programs and reactive systems (e.g., see [8, 19]). Starting about a decade ago, first approaches towards compositional specification in the area of temporal logic were made. At the time being, there are only few approaches which really provide compositionality. But often they impose unsatisfactory restrictions or problems which hinder software designers in describing large system in an intuitive way.

Considering the approaches given in the literature, it becomes obvious that composition of temporal logic specifications is a difficult problem. The first approaches towards compositional temporal logic allow only a very restricted form of composition. In [2] an explicit idle action was introduced to allow the environment of a process to have state transitions without the considered object itself being concerned. However, this approach requires that we have to explicitly deal with idle transitions when specifying a system. Instead, idle transitions should be an implicit concept not visible at the syntactical level of temporal logic. Taking idle transitions into account for all properties to be specified leads to a very complex and complicated specification which often is not easily readable.

A more recent approach to the temporal specification of object societies is the *Object Specification Logic* OSL [24]. OSL offers the possibility of specifying inheritance between objects. But a particular point w.r.t. compositional specification is that

*This research is partially supported by CEC as Esprit Basic Research Action IS-CORE (working group 6071) and ModelAge (working group 8319).

OSL uses a global time model in which the global-next instant of time is always also the local-next instant. Thereby, the possible global development of a system must be explicitly taken into account when specifying components of the system. In consequence the local behaviour of objects cannot be specified in an intuitive way.

In order to overcome these problems, we present a special interpretation of linear temporal logic for the (local) specification of objects (or sub-systems) in this paper. This interpretation allows to specify the local behaviour of an object without making assumptions about possible global behaviour of a whole object society.

Thereby, compositional specification (and verification) becomes possible in an intuitive and pragmatic way. Compositionality is an essential prerequisite for achieving correctness and reliability of large systems. Moreover, reusability of specifications for parts as well as for whole systems is enhanced by this kind of compositionality.

In addition, especially for formal verification of large systems compositionality is needed. Following the structure of the specification, properties can be proved as locally as possible. If a part of the specification has to be modified only the proofs of those properties have to be checked which are local properties of that part of the specified system or are more global properties based on such local properties.

Our approach seems to be a promising step towards a fully compositional temporal logic. The price we have to pay for these properties is a little bit more complicated formalization. In order not to reinvent the wheel we embed this temporal logic into a framework for the specification of object systems similar to that proposed in [24]. Here, however, we abstain from dealing with certain object-oriented concepts like inheritance.

The remainder of this paper is organized as follows. In Sect. 2 we give an example for a problem in composing (and reusing) temporal logic specifications. In Sect. 3 we first present a new interpretation of linear temporal logic with `next` operator allowing models with idle transitions. Then we provide a framework for specifying systems consisting of several objects. At the end of this section we briefly point out properties of our logic. Finally, we conclude by discussing related work and sketching future work.

2 The Dilemma of "Next"

First of all we have to explore a fundamental problem in composing and reusing temporal logic specifications. We will see that the usual semantics of the `next` operator $\bigcirc$ is the focus of this problem which has already been discovered by several other authors, e.g., in [2, 1]. However, we need a `next` operator for expressing changes of states in a natural way. In this section we describe this problem by means of a simple example. Beforehand, we briefly introduce a classical version of linear temporal logic (LTL) which we will use in that example.

Def. 1 (Syntax of LTL)

Let $AP = \{P, Q, \ldots\}$ be a set of atomic propositions. The set of formulas of LTL is the least set of formulas generated by the following rules:

1. each atomic proposition $P \in AP$ is a formula;
2. if p and q are formulas, then $p \wedge q$ and $\neg p$ are formulas;
3. if p and q are formulas, then $\Box p$ and $\bigcirc p$ are formulas.

Other propositional connectives (e.g., $\vee$, $\rightarrow$) can be introduced as abbreviations in the usual way. The temporal operator $\Diamond$ can also be introduced as abbreviation: $\Diamond p \equiv \neg\Box\neg p$. $\lhd$

The temporal operators introduced by this definition are $\Box$ and $\Diamond$ standing for "always" and "eventually" (or "sometime"), respectively. These operators are future-directed.

Def. 2 (Semantics of LTL)

The semantics of a formula p of LTL is given with respect to a linear-time structure $\mathcal{M} = (\sigma, \rho)$ where

- $\sigma = \langle 1, 2, 3, \ldots \rangle$ is a (finite or infinite) ordered sequence of natural numbers (representing the instants of time), and
- $\rho : \mathbb{N} \rightarrow 2^{AP}$ is a function labeling each instant by the set of atomic propositions in AP which are true at that state.

By $\langle \mathcal{M}, i\rangle \models p$ we denote that "in structure $\mathcal{M}$ formula p is true at instant i". $\models$ is inductively defined by:

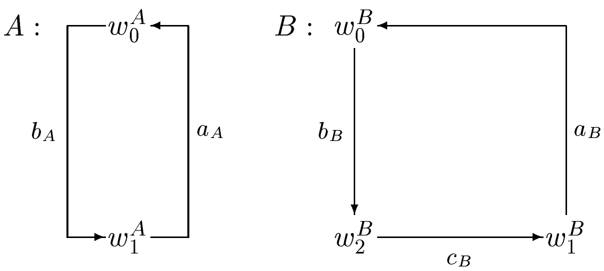
1. for atomic proposition P:
   $\langle \mathcal{M}, i\rangle \models P \quad$ iff $\; i \leq |\sigma|$ and $P \in \rho(i)$;
2. $\langle \mathcal{M}, i\rangle \models p \wedge q$ iff $\langle \mathcal{M}, i\rangle \models p$ and $\langle \mathcal{M}, i\rangle \models q$,
   $\langle \mathcal{M}, i\rangle \models \neg p \quad$ iff $\; i \leq |\sigma|$ and not $\langle \mathcal{M}, i\rangle \models p$;
3. $\langle \mathcal{M}, i\rangle \models \Box p \quad$ iff
   $$\forall j \in I\!\!N : i \leq j \leq |\sigma| \Longrightarrow (\langle \mathcal{M}, j\rangle \models p),$$
   $\langle \mathcal{M}, i\rangle \models \bigcirc p \quad$ iff $\; \langle \mathcal{M}, i+1\rangle \models p$.

A formula p is *valid* in $\mathcal{M}$ (or $\mathcal{M}$ is a *model* for p; written $\mathcal{M} \models p$) iff $\langle \mathcal{M}, i\rangle \models p$ for all $i = 1..|\sigma|$. $\lhd$

For models we allow finite as well as infinite sequences of instants. This is due to the fact that there may be an only finite number of state transitions in the life of an object in an information system. Therefore, we have to take the length of the sequence into account for the satisfaction of formulas.

Example 1

Assume two objects (or processes) A and B to be given which can be described by the following diagrams:



In order to describe to behaviour of these processes in an adequate way, we could use the following temporal logic formulas:

$$\begin{array}{llcll} A: & a_A \to \bigcirc b_A & \qquad & B: & a_B \to \bigcirc c_B \\ & b_A \to \bigcirc a_A & & & c_B \to \bigcirc b_B \\ & & & & b_B \to \bigcirc a_B \end{array}$$

Assuming w_0^A and w_0^B to be the initial states of these processes we can state two models $\mathcal{M}_A = (\sigma^A, \rho^A)$ and $\mathcal{M}_B = (\sigma^B, \rho^B)$ depicted as follows (by unwinding the diagrams given above):

$$\mathcal{M}_A: \quad 1 \xrightarrow{a_A} 2 \xrightarrow{b_A} 3 \xrightarrow{a_A} 4 \xrightarrow{b_A} 5 \xrightarrow{a_A} 6 \xrightarrow{b_A} \cdots$$

$$\mathcal{M}_B: \quad 1 \xrightarrow{a_B} 2 \xrightarrow{c_B} 3 \xrightarrow{b_B} 4 \xrightarrow{a_B} 5 \xrightarrow{c_B} 6 \xrightarrow{b_B} \cdots$$

where

$$\begin{array}{lcl} \rho^A(1) & = & \{\, a_A \,\} \\ \rho^A(2) & = & \{\, b_A \,\} \end{array} \qquad \begin{array}{lcl} \rho^B(1) & = & \{\, a_B \,\} \\ \rho^B(2) & = & \{\, c_B \,\} \\ \rho^B(3) & = & \{\, b_B \,\} \end{array}$$

and

$$\rho^A(i+2) = \rho^A(i)$$
$$\rho^B(i+3) = \rho^B(i)$$

for every $i \in I\!\!N$.

But, what does happen when we bring these two processes together to form one system $A|B$ and then impose additional synchronization conditions? Let us consider the restriction that the events a_A and a_B must occur simultaneously as well as b_A and b_B. Intuitively, we would expect that the following diagram determines a model for this system:

$$\mathcal{M}_{A|B}: \quad 1 \xrightarrow[a_B]{a_A} 2 \xrightarrow[c_B]{} 3 \xrightarrow[b_B]{b_A} 4 \xrightarrow[a_B]{a_A} 5 \xrightarrow[c_B]{} 6 \xrightarrow[b_B]{b_A} \cdots$$

where

$$\begin{array}{lcl} \rho(1) & = & \{\, a_A,\ a_B \,\} \\ \rho(2) & = & \{\, c_B \,\} \\ \rho(3) & = & \{\, b_A,\ b_B \,\} \\ & \vdots & \end{array}$$

However, with regard to the corresponding temporal logic specifications of A and B, $\mathcal{M}_{A|B}$ is not a model. This can easily be seen by considering the formula $a_A \to \bigcirc b_A$. This formula says that for each instant i along a life cycle σ, if a_A is true at i then b_A is true at instant $i+1$. Regarding $\mathcal{M}_{A|B}$ we find that a_A is true at instant 1, but b_A is not true at instant 2. $\bowtie$

The problem shown in the example is that "next-time" for process A, "next-time" for process B, and "next-time" for the composed system $A|B$ are, in general, not the same. This is due to the fact that in the system $A|B$ the environment of process A, i.e., the process B, may execute state transitions while the process A does not change its state.

As far as we know, this problem occurs in every linear discrete temporal logic with `next` operator. Because the `next` operator appears to be a natural means in specifying properties of state-based systems like information systems, we propose a more subtle interpretation for linear temporal logic in the next section.

3 A Linear Temporal Logic Allowing Local Stuttering

In this section we present the formal framework for a compositional object specification based on temporal logic. First, we introduce (local) object signatures and local temporal languages for which we define interpretation structures and satisfaction of local formulas. Afterwards, we present system signatures in which local object signatures are combined to form more complex systems. For system signatures we define system languages and system interpretation structures. We conclude the formal definition by giving the notion of satisfaction of system formulas. Finally, we briefly point out essential properties of our logic.

3.1 Local Signatures and Interpretations

Object signatures are local signatures, i.e., the symbols given by an object signature are private to an object described by that object signature.

Def. 3 (Object Signature)

An object signature Σ is a pair (S, A) where S and A are disjoint sets of symbols (state symbols and action symbols, respectively). $\lhd$

We distinguish between state and action symbols because they stand for conceptually different things. This also causes a different treatment when we define the satisfaction of formulas.

Def. 4 (Local Language)

The local language $\mathcal{OLL}_\Sigma$ over an object signature $\Sigma = (S, A)$ is the least set given inductively by:

- $p \in \mathcal{OLL}_\Sigma$ for each $p \in (S \cup A)$;
- if $\phi, \psi \in \mathcal{OLL}_\Sigma$ then $(\neg\phi), (\phi \vee \psi), \bigcirc\phi, \Box\phi \in \mathcal{OLL}_\Sigma$, too.

As usual, $\wedge$, $\rightarrow$, $\Diamond$, etc. can be introduced as abbreviations. $\lhd$

State symbols in S provide state propositions and action symbols in A provide action propositions. An action proposition shall denote the happening of an action. For simplicity we restrict our considerations to linear temporal logic. In the following definition the natural numbers are used as underlying time domain. Thereby, we assume time to be discrete, linear, and unbounded to the future.

Def. 5 (Local Interpretation Structure)

A local interpretation structure $\mathcal{IS}_\Sigma$ for an object signature $\Sigma = (S, A)$ is a pair $(\mathcal{V}_S, \mathcal{V}_A)$ where

- $\mathcal{V}_S : I\!N \rightarrow 2^S$ is a function giving for each instant of time a set of state propositions and
- $\mathcal{V}_A : I\!N \rightarrow 2^A$ is a function giving for each instant of time a set of action propositions

and for which the following condition holds:

$$\forall i \in I\!N : \mathcal{V}_A(i) = \emptyset \implies \mathcal{V}_S(i+1) = \mathcal{V}_S(i)$$

$\lhd$

The functions $\mathcal{V}_S$ and $\mathcal{V}_A$ assign to each instant of time the set of state propositions and action propositions (respectively) fulfilled at that instant. The given condition expresses the frame property that there is no state change without an occurrence of an action.

By means of a small example we demonstrate our comprehension of typical object behaviour with regard to time.

Example 2

Let us consider a typical object with an object signature $\Sigma = (\{s_1, s_2\}, \{a_1, a_2, a_3\})$. In the life of this object there are several actions happening at different instants of time. For simplicity we assume that at each instant there may be at most one action occurring. So a typical life could be depicted by:

$$1 \xrightarrow{\{a_1\}} 2 \xrightarrow{\{a_2\}} 3 \xrightarrow{\emptyset} 4 \xrightarrow{\{a_3\}} 5 \xrightarrow{\emptyset} 6 \xrightarrow{\{a_1\}} 7 \xrightarrow{\{a_1\}} \cdots$$

$$\{s_1\} \quad \emptyset \quad \{s_2\} \quad \{s_2\} \quad \{s_1\} \quad \{s_1\}\{s_1, s_2\} \quad \cdots$$

For each instant i of time we have placed the value of $\mathcal{V}_A(i)$ above the arrow to the subsequent instant and the value of $\mathcal{V}_S(i)$ in the row below. Here, we see that at instants 3 and 5 there are no actions occurring in this object. In consequence, there is no state change between instants 3 and 4 and between instants 5 and 6. However, these (i.e., instants 3 and 5) are moments in which the environment may have state transitions caused by the occurrence of action outside this object. Of course there may also be occurrences of actions in the environment when there is an occurrence of an action in the considered object at the same time. $\bowtie$

This example shows that we assume open semantic structures in the sense that each object may be embedded into a larger system (as well as the more general principle that each object society may be embedded into a larger object society). In [2] and [11] a similar view of systems is assumed. In consequence, we have to take care of independent state transitions in different objects by allowing *idle transitions* for each object.

Before we proceed to the definition of satisfaction of local formulas, we have to clarify our notion of "local next-time". Considering only the state propositions it can be seen that idle transitions do not affect the state of an object. Thus, the states connected by idle transitions can be regarded as identical for the considered object. However, the environment may distinguish between any two of these states in case of a non-idle transition in the environment. With regard to the example above the local states at instants 3 and 4 are in fact one state from the object's local point of view. From the global point of view they are separated because at instant 4 there is an action occurring in that object causing a state transition. Thereby, we are modeling the real world situation that an object in an information system can stay unchanged for some time until something is happening to it. We consider the time in which the object remains unchanged as one local moment for this object.

Def. 6 (Local Satisfaction)

The satisfaction of a formula $\phi \in \mathcal{OLL}_\Sigma$ by an local interpretation structure $\mathcal{IS}_\Sigma$ for an object signature $\Sigma = (S, A)$ at an instant $i \in I\!N$ (written $[\mathcal{IS}_\Sigma, i] \models \phi$) is defined by:

- for $s \in S$:
  $[\mathcal{IS}_\Sigma, i] \models s$ iff $s \in \mathcal{V}_S(i)$;
- for $a \in A$:
  $[\mathcal{IS}_\Sigma, i] \models a$ iff
  $$\begin{array}{l} \exists j \in I\!N: \\ \quad i \leq j \ \wedge \ a \in \mathcal{V}_A(j) \\ \quad \wedge \ (\forall k \in I\!N: i \leq k < j \Longrightarrow \mathcal{V}_A(k) = \emptyset); \end{array}$$
- $[\mathcal{IS}_\Sigma, i] \models \neg\phi$ iff not $[\mathcal{IS}_\Sigma, i] \models \phi$;
- $[\mathcal{IS}_\Sigma, i] \models \phi \vee \psi$ iff
  $$[\mathcal{IS}_\Sigma, i] \models \phi \ \text{ or } \ [\mathcal{IS}_\Sigma, i] \models \psi;$$
- $[\mathcal{IS}_\Sigma, i] \models \bigcirc\phi$ iff
  $$\begin{array}{l} \exists j \in I\!N: \\ \quad i \leq j \ \wedge \ \mathcal{V}_A(j) \neq \emptyset \ \wedge \ [\mathcal{IS}_\Sigma, j+1] \models \phi \\ \quad \wedge \ (\forall k \in I\!N: i \leq k < j \Longrightarrow \mathcal{V}_A(k) = \emptyset) \ ; \end{array}$$
- $[\mathcal{IS}_\Sigma, i] \models \Box\phi$ iff
  $$\forall j \in I\!N: \ i \leq j \Longrightarrow [\mathcal{IS}_\Sigma, j] \models \phi \ ;$$

A formula $\phi \in \mathcal{OLL}_\Sigma$ is valid in $\mathcal{IS}_\Sigma$ iff $[\mathcal{IS}_\Sigma, i] \models \phi$ for each instant $i \in I\!N$. $\phi \in \mathcal{OLL}_\Sigma$ is called a Σ-tautology iff ϕ is valid in every local interpretation structure for Σ. $\triangleleft$

In explanation some remarks are necessary:

- At first glance the interpretation of action symbols may appear confusing and cumbersome. But this definition of satisfaction for action symbols reflects our intuition that instants between which there are only idle transitions cannot be distinguished from the local point of view of the object itself. In consequence we have to inspect the next instant in which some actions are occurring if we want to determine whether a formula consisting of an action symbol $a \in A$ is fulfilled at some instant.

- In a similar way we have given an interpretation of $\bigcirc$. Thus, $\bigcirc$ refers to the instant immediately following the next action occurrence for the considered object.

Up to now we have only considered single objects, i.e., their (local) signatures and local languages. The next step is to put a collection of single objects together to form a system.

3.2 System Signatures and Interpretations

When we are embedding objects into a larger system consisting of several objects we have to provide for each object a unique value to identify that object. Therefore, we introduce a set ID as identification space for objects in a system.

Def. 7 (System Signature)
A system signature $\widehat{\Sigma}$ is a pair $(ID, \langle \Sigma^I \rangle_{I \in ID})$ where

- ID is a set of object identifiers and
- $\langle \Sigma^I \rangle_{I \in ID}$ is an ID-indexed family of object signatures $\Sigma^I = (S^I, A^I)$.

$\triangleleft$

By choosing an arbitrary identity I we can "lift" each object signature $\Sigma = (S, A)$ to a primitive system signature $(\{I\}, \langle \Sigma^I \rangle)$ with $\Sigma^I = \Sigma$.

Def. 8 (I-Translation)
Given an object signature $\Sigma = (S, A)$, the I-translation $I(\phi)$ of a formula $\phi \in \mathcal{OLL}_\Sigma$ is obtained by inductively applying the following rules:

- $I(a) = I.a \quad \text{for} \quad a \in A$;
- $I(s) = I.s \quad \text{for} \quad s \in S$;
- $I(\neg\phi) = \neg\, I(\phi)$;
- $I(\phi \vee \psi) = I(\phi) \vee I(\psi)$;
- $I(\bigcirc\phi) = \bigcirc^I I(\phi)$;
- $I(\Box\phi) = \Box I(\phi)$.

$\triangleleft$

This means that the I-translation of a formula ϕ can be obtained by replacing each symbol $p \in (S \cup A)$ by $I.p$ and $\bigcirc$ by $\bigcirc^I$. The operator $\bigcirc^I$ is a local operator (for each $I \in ID$). This is due to the fact that in the context of a system consisting of several objects we have to refer to the local notion of next-time for each object. Please note that $\Box$ is not replaced by a local variant because Def. 6 allows us to consider $\Box$ as a local operator as well as a global one.

Def. 9 (System Language)
The system language $\mathcal{S}ys\mathcal{L}_{\widehat{\Sigma}}$ for a system signature $\widehat{\Sigma} = (ID, \langle \Sigma^I \rangle_{I \in ID})$ is the least set of formulas constructed by:

- for each $I \in ID$ the I-translation of every formula in $\mathcal{OLL}_{\Sigma^I}$ is a formula in $\mathcal{S}ys\mathcal{L}_{\widehat{\Sigma}}$;
- for all $I, J \in ID$ with $I \neq J$: if $p \in A^I$ and $q \in A^J$ then $(I.p \to J.q) \in \mathcal{S}ys\mathcal{L}_{\widehat{\Sigma}}$.

$\triangleleft$

By this definition the local languages for the single objects identified by elements of ID are embedded into the system language. Furthermore simple interaction formulas are allowed. However, it should be noted that over and above that we do not allow to mix the different local languages to form new formulas. Before we could allow this, we will have to investigate the more complicated interaction between different local time operators, i.e., $\bigcirc^I$ and $\bigcirc^J$ for $I \neq J$. We will glance at this problem in the conclusion.

Def. 10 (System Interpretation Structure)

A system interpretation structure $\mathcal{S}ys\mathcal{IS}$ for a system signature $\widehat{\Sigma} = (ID, \langle \Sigma^I \rangle_{I \in ID})$ is an ID-indexed family $\langle \mathcal{IS}_{\Sigma^I} \rangle_{I \in ID}$ where $\mathcal{IS}_{\Sigma^I}$ is a local interpretation structure for the object identified by I (for each $I \in ID$). $\lhd$

Def. 11 (System Satisfaction)

A system interpretation structure $\mathcal{S}ys\mathcal{IS} = \langle \mathcal{IS}_{\Sigma^I} \rangle_{I \in ID}$ for a system signature $\widehat{\Sigma} = (ID, \langle \Sigma^I \rangle_{I \in ID})$ satisfies a formula $\psi \in \mathcal{S}ys\mathcal{L}_{\widehat{\Sigma}}$ at instant $i \in I\!N$ (written $[\mathcal{S}ys\mathcal{IS}, i] \models_{\widehat{\Sigma}} \psi$) iff:

- if $\psi = I(\phi)$ is the I-translation of a formula $\phi \in \mathcal{OLL}_{\Sigma^I}$ for some $I \in ID$, then $[\mathcal{IS}_{\Sigma^I}, i] \models \phi$; or
- if $\psi = (I.p \rightarrow J.q)$ for $I, J \in ID$, $p \in A^I$, and $q \in A^J$, then $p \notin \mathcal{V}_A^I(i) \vee q \in \mathcal{V}_A^J(i)$.

A formula $\psi \in \mathcal{S}ys\mathcal{L}_{\widehat{\Sigma}}$ is satisfied by a system interpretation structure $\mathcal{S}ys\mathcal{IS}$ ($\mathcal{S}ys\mathcal{IS} \models_{\widehat{\Sigma}} \psi$) iff $[\mathcal{S}ys\mathcal{IS}, i] \models_{\widehat{\Sigma}} \psi$ holds for every instant $i \in I\!N$. $\lhd$

Please note the difference in the satisfaction of I-translated local formulas and of interaction formulas. On the one hand the system satisfaction (or global satisfaction — in contrast to local satisfaction) of an I-translation of a local formula is based on the local satisfaction in the local interpretation structure (as given in Def. 6). On the other hand the global satisfaction of an interaction formula $(I.p \rightarrow J.q)$ is not defined as $[\mathcal{IS}_{\Sigma^I}, i] \models p \implies [\mathcal{IS}_{\Sigma^J}, i] \models q$. Such a definition would make different instants the same on the global level. Because p and q are action symbols it may happen that they are locally satisfied although $p \notin \mathcal{V}_A^I(i)$ or $q \notin \mathcal{V}_A^J(i)$. In contrast the definition given above really refers to the global instant.

3.3 Properties of the Logic

The logic presented before has two main properties:

- It can be shown that there is an institution (in the sense of [13]) for this logic.
- We can prove that there is a correspondence between the local language for specifying objects and a classical temporal logic.

The former property is necessary for compositionality of temporal logic specifications. The latter one implies that we can adopt an existing proof system of the classical temporal logic for the local language of our logic (cf. discussion in the next section).

A more detailed consideration of these properties together with formal proofs can be found in [5].

4 Discussion and Conclusions

In this paper we have presented a temporal logic with an unusual but not unintuitive interpretation of the temporal operators. Although this interpretation causes a more costly formalization, we think this expenditure is justified. Our logic seems to meet compositionality and reusability requirements to a high degree because it can be used within an institutional framework and because it is insensitive to finite stuttering although we allow (local) `next` operators.

The properties of our temporal logic framework, namely compositionality and better reusability of object specifications, are an important improvement with regard to the use of temporal specification of object behaviour in real applications. The development of specifications for large systems becomes feasible due to the intuitive approach of specifying the local behaviour of objects without taking the environment into account. In our logical framework this is an implicit assumption simplifying the specification task. Thereby, this logical framework seems to be an ideal semantical basis for the development of high-quality software systems. Even the special requirements of software systems with a long lifetime (for instance information systems) can be captured in our approach. Thus, our approach can be considered as an important step towards a full applicability of formal methods in the development of complex software systems. In this way the formal framework presented here can make an important contribution to achieve a better software quality.

In this paper we have not dealt with a proof system for our logic. This is due to the fact that existing proof systems for temporal logics can easily be adapted for the local specification part of our logic because of the correspondence to classical interpretation of linear temporal logic. However, this only works for the local languages of objects. On the global level (i.e., for the system specification) we cannot state such a correspondence to some well-known logic. This is why we only allow a very restricted global language up to now (as in OSL [24]). At the global level we have the problem of finding a sensible relation between the different local temporal operators. Obviously formulas like

$$\bigcirc^I \bigcirc^J P \rightarrow \bigcirc^J \bigcirc^I P$$

cannot in general be valid (provided $I \neq J$). A promising idea is the translation of local specification into a global language which allows additional temporal logic operators (especially, an until operator is needed). However, the resulting specifications are much more complicated. We are currently looking for useful properties which can be retranslated into the more restricted logic.

The formal verification of local properties, e.g. for single objects, can be done by means of a theorem prover. To a certain degree such a theorem prover could work automatically. Nevertheless, we do not believe that proofs can in general be found in a totally automatic way. Therefore, a part of the work in proving properties must always be done by human beings. Theorem provers and similar tools can and should be employed for taking on tedious routine matters.

As far as we know, [9] is one of first approaches considering systems where each component may have its own local time. For the communication and interaction of the components explicit synchronization operators are used. Thereby, the proposed logic is much more involved. Furthermore, it is not clear whether that logic is compositional at all.

Among the different approaches to compositionality and reusability in temporal logic specification only few give a concrete logic with properties similar to our logic. Some authors (for instance [1]) only work on a semantic level without giving a concrete logic. The approach presented in [3] uses a temporal logic based on real numbers as underlying time domain and, therefore, does not have a `next` operator. Other approaches like [18, 22] are based on other computational models (they assume asynchronous communication and do not have global states). In the area of temporal specification of object societies there are several approaches (e.g., [10, 24]) which are quite related to our comprehension of objects. However, they are based on "classical" interpretations of temporal logic and, thereby, fail to be insensitive to finite stuttering. A nice approach towards insensitiveness to finite stuttering is given in [25] where a common model for Lamport's Temporal Logic of Actions [17] and the Duration Calculus [21] is presented. Another very recent approach which seems to come close to our ideas can be found in [20].

The proposal of a temporal logic given in this paper arose from the work on a simplified specification language for objects, TROLL *light* [12, 15], which was designed for the investigation of validation and verification of object specification. For this language we developed a simplified logic based on event-indexed next operators [4, 6] which has a compositional proof system. TROLL *light* is a dialect of TROLL [14, 16] which originates from the same theoretical foundations (cf. [23, 7]) as the OSL.

The work presented in this paper seems to be a promising basis for several extensions. As already mentioned a more detailed investigation of mixing the local languages on the global level is necessary. An extension to a first-order temporal logic seems to be quite natural and uncomplicated (cf. [24]). Further improvement of the expressive power of this logic could be achieved by adding further temporal operators (e.g., "until" and "precedes"). Besides this language-oriented extensions an integration of additional modelling concepts for objects like inheritance or dealing with reification of events are major topics for future work.

Acknowledgement:

This paper continues a joint work with Martin Gogolla which started in Braunschweig within the projects KorSo (= "Correct Software"; supported by the German National Ministry of Research and Technology (BMFT) under Grant No. IT 01 IS 203 D) and Compass (Esprit BRA working group 6112). Moreover, fruitful discussions with other colleagues (especially with G. Denker, H.-D. Ehrich, P.-Y. Schobbens, and G. Saake) have contributed to the progress of this work.

References

[1] M. Abadi and L. Lamport. Composing Specifications. *ACM Trans. on Programming Languages and Systems*, 15(1):73–132, 1993.

[2] H. Barringer. The Use of Temporal Logic in the Compositional Specification of Concurrent Systems. In A. Galton, editor, *Temporal Logics and Their Applications*, pages 53–90. Academic Press, 1987.

[3] H. Barringer, R. Kuiper, and A. Pnueli. A Really Abstract Concurrent Model and its Temporal Logic. In *Proc. ACM Symp. Principles of Programming Laguages*, pages 173–183, 1986.

[4] S. Conrad. On Certification of Specifications for TROLL *light* Objects. In H. Ehrig and F. Orejas, editors, *Proc. of the 9th Workshop on Abstract Data Types – 4th Compass Workshop (ADT'92)*, pages 158–172. Springer-Verlag, LNCS 785, 1994.

[5] S. Conrad. Temporal Logic Specification of Objects: An Approach to Compositionality and Reusability Allowing `next` Operators. In R. Wieringa and R. Feenstra, editors, *Working papers of the Int. Workshop on Information Systems - Correctness and Reusability*, pages 228–241. Vrije Universiteit Amsterdam, RapportNr. IR-357, 1994.

[6] S. Conrad and H.-D. Ehrich. An Elementary Logic for Object Specification and Verification. In U. Lipeck and G. Vossen, editors, *Workshop Formale Grundlagen für den Entwurf von Informationssystemen, Tutzing*, pages 197–206. Technical Report Univ. Hannover, No. 03/94, 1994.

[7] H.-D. Ehrich, G. Denker, and A. Sernadas. Constructing Systems as Object Communities. In M.-C. Gaudel and J.-P. Jouannaud, editors, *Proc. Theory and Practice of Software Development (TAPSOFT'93)*, pages 453–467. Springer, Berlin, LNCS 668, 1993.

[8] E.A. Emerson. Temporal and Modal Logic. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B*, pages 995–1072. Elsevier, North-Holland, 1990.

[9] P. Enjalbert and M. Michel. Many-Sorted Temporal Logic for Multi-Processes Systems. In M.P. Chytil and V. Koubek, editors, *Mathematical Foundations of Computer Science 1984*, pages 273–281. Springer, LNCS 176, 1984.

[10] J. Fiadeiro and T. Maibaum. Temporal Reasoning over Deontic Specifications. *Journal of Logic and Computation*, 1(3):357–395, 1991.

[11] J.L. Fiadeiro, J.F. Costa, A. Sernadas, and T.S.E Maibaum. Process Semantics of Temporal Logic Specification. In M. Bidoit and C. Choppy, editors, *Recent Trends in Data Type Specification (WADT'91)*, pages 236–253. Springer, Berlin, LNCS 655, 1993.

[12] M. Gogolla, S. Conrad, and R. Herzig. Sketching Concepts and Computational Model of TROLL *light*. In A. Miola, editor, *Proc. of the 3rd Int. Conf. Design and Implementation of Symbolic Computation Systems (DISCO'93)*, pages 17–32. Springer-Verlag, LNCS 722, 1993.

[13] J.A. Goguen and R.M. Burstall. Institutions: Abstract Model Theory for Specification and Programming. *Journal of the ACM*, 39(1):95–146, 1992.

[14] T. Hartmann, G. Saake, R. Jungclaus, P. Hartel, and J. Kusch. Revised Version of the Modelling Language TROLL (Version 2.0). Informatik-Bericht 94–03, Technische Universität Braunschweig, 1994.

[15] R. Herzig, S. Conrad, and M. Gogolla. Compositional Description of Object Communities with TROLL *light*. In C. Chrisment, editor, *Proc. of the Basque Int. Workshop on Information Technology (BIWIT'94)*, pages 183–194. Cépaduès-Éditions, Toulouse, 1994.

[16] R. Jungclaus, G. Saake, T. Hartmann, and C. Sernadas. TROLL – A Language for Object-Oriented Specification of Information Systems. *ACM Transactions on Information Systems*, 1995. *To appear.*

[17] L. Lamport. The Temporal Logic of Actions. Technical Report 79, DEC Systems Research Center, 1991.

[18] K. Lodaya and P.S. Thiagarajan. A Modal Logic for a Subclass of Event Structures. In

Th. Ottmann, editor, *Automata, Languages and Programming (ICALP'87)*, pages 290–303. Springer, LNCS 267, 1987.

[19] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems; Specification.* Springer-Verlag, New York, 1991.

[20] A. Mokkedem and D. Méry. A Stuttering Closed Temporal Logic for Modular Reasoning about Concurrent Programs. In D.M. Gabbay and H.J. Ohlbach, editors, *Proc. of the First Int. Conf. on Temporal Logic (ICTL'94)*, pages 382–397. Springer LNAI 827, 1994.

[21] A.P. Raven, H. Rischel, and K.M. Hansen. Specifying and Verifying Requirements of Real-time Systems. *IEEE Trans. on Software Engineering*, 19(1):41–55, 1993.

[22] W. Reisig. Temporal Logic and Causality in Concurrent Systems. In H. Vogt, editor, *Concurrency '88*, pages 121–139. Springer, LNCS 335, 1988.

[23] A. Sernadas and H.-D. Ehrich. What Is an Object, After All? In R. Meersman, W. Kent, and S. Khosla, editors, *Object-Oriented Databases: Analysis, Design and Construction (Proc. 4th IFIP WG 2.6 Working Conference DS-4, Windermere (UK))*, pages 39–70, Amsterdam, 1991. North-Holland.

[24] A. Sernadas, C. Sernadas, and J.F. Costa. Object Specification Logic. Internal Report, INESC, University of Lisbon, 1992. To appear in Journal of Logic and Computation.

[25] M.U. Sørensen, O.E. Hansen, and H.H. Løvengreen. Combining Temporal Specification Techniques. In D.M. Gabbay and H.J. Ohlbach, editors, *Proc. of the First Int. Conf. on Temporal Logic (ICTL'94)*, pages 1–16. Springer LNAI 827, 1994.