# A Two-Level Temporal Logic For Evolving Specifications ⋆

P.Y. Schobbens [a] G. Saake [b] A. Sernadas [c] C. Sernadas [c]

[a] *Institut d'Informatique, Rue Grandgagnage, 21, 5000 Namur, Belgium*

[b] *Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, Postfach 4120, D-39016 Magdeburg, Germany*

[c] *Secção de Ciência da Computação, Departamento de Matemática, Instituto Superior Técnico, Av. Rovisco Pais, 1049-001, Lisboa, Portugal*

**Abstract**

Traditional information system specifications are fixed: the rules of the system are frozen at specification time. In practice, most systems have to change their rules in unexpected ways during their lifetime. We present here a simple variant of a temporal logic that deals with specification evolution. It is a linear time temporal logic with two levels of time: intervals, interrupted by mutations (changes of rules), which compose lives of the system. We present a complete axiom system and complexity results, which show a large compatibility with classical linear temporal logic.

*Key words:* program specification, temporal logic, specification evolution, reconfigurable systems

## 1 Introduction

The need to formally specify the behaviour of information systems is gaining acceptance: it allows better precision during the early phases of development

⋆ This work was partially supported by the Portuguese FCT program PRAXIS XXI, and specially grant BPD/11851/97, projects SitCalc (2/2.1/MAT/262/94), LogComp (2/2.1/TIT/1658/95), ACL plus (PCEX/P/MAT/46/96) and by the European Commission Esprit WGs ModelAge (8319), Aspire (22704) and Fireworks (23531).

*Email addresses:* pys@info.fundp.ac.be (P.Y. Schobbens), saake@iti.cs.uni-magdeburg.de (G. Saake), acs@math.ist.utl.pt (A. Sernadas), css@math.ist.utl.pt (C. Sernadas).

[7], sparing thus on the important cost of errors discovered late [1]. Early specification techniques only dealt with static (e.g. data) aspects, and with individual actions of the system. However, many systems built nowadays can only be understood by considering their full behaviour over time: such systems are called *reactive systems* [13] and are nicely specified by temporal logics.

Current specification approaches use collections of communicating objects for describing information systems to combine the need for behaviour specification with structuring mechanisms developed for object systems. The structuring of communicating objects has been understood in [19] and was used as the basis of the specification languages Albert [4], OBLOG [19], GNOME [14], TROLL [8], among others.

All those specification approaches assume that the system is fully specified before operating it, so that the behaviour of the system is immutable. In practice however, most systems have to change their rules in unexpected ways during their lifetime: business rules evolve with market pressures, the legal constraints are modified, remote systems often need new capabilities [2].

To address this problem, we have investigated several approaches [15,3,21,2] based on reflection [22] in the specification logic, thus allowing programmed specification modifications. In this paper, we explore a more classical approach, for which we will be able to express most evolution problems, and yet avoid the complexity of reflection. As in [2], this is obtained by distinguishing two levels in the logic:

(1) the level of *intervals*, which comprise the time during which the system runs the same code and thus follows the same rules. An interval is ended by a *mutation* [2]. Mutations are caused by special reconfiguration orders called *mutators*. Our logic will make a specification restricted to an interval specially easy to express (using underlined operators). Mutators are usually triggered by the owner of the system to adapt it by loading new programs.
(2) the level of infinite *lives*, which is seen as a succession of intervals. In most systems, some strong rules should hold throughout its life. The transition between intervals usually also obey specific rules; some of them are permanent, others are local to the end of the current interval. The lives of the system need not be completely specified in advance: the behaviour of a mutator can be specified when the system is in operation.

Our intervals should be distinguished from the layers of [12,11], which intend to reflect a division of time of fixed length. Our application requires a division of unforeseeable length, and maybe infinite.

Another temporal logic with layers is [10]. In this work, layers model nested procedure calls. While we only have two levels, this logic can have many, even an infinite number. A small difference is that changing from an interval to the next requires to pass first to a state belonging to the level just above. Unfortunately, this logic looses the regularity properties of temporal logic.

Finally, intervals are used in [5] to model temporal refinement, i.e., the fact that a high-level action can be decomposed in a sequence of low-level steps. In our application, intervals need not be finite, but are adjacent, while in [5] they are finite and may overlap. A further difference is that [5] use two separated logical systems and models (one for each level).

In contrast, we shall see that our logic has the same theoretical properties as temporal logic. In particular, the propositional fragment of the logic makes sense, while reflection requires terms to encode formulae. In this article, we concentrate on the technical properties of the propositional fragment, since most tools for temporal logic use it. The first-order version is defined in [16], together with applications. We also refer to [2] for more motivation.

The paper is organised as follows: Section 2 gives the definition of the logic. Section 3 gives its complexity, section 4 its axiomatization, and section 5proves its completeness. Future work is sketched in section 6.

## 2 The base logic

This section presents the linear temporal logic $\mathsf{U2}$, which is a simple variant of the logic OSL [18]. It introduces two levels, but unlike *dy*OSL, the second level is not built using reflection, but a simple temporal nesting.

### *2.1 Signatures*

Intuitively, the signature contains the declarations needed for a component of the system (called an object). They are used to combine components thanks to the theory of institutions, see [17].

Here, an *object signature* is composed of a set $A$ of (boolean, time-dependent) attributes; a set $M$, the mutators, and a set $\Gamma$ of (non-mutating) actions. These sets must be disjoint. We note $\Delta = M \cup \Gamma$ the set of all actions.

An object will be specified by temporal formulae. The classical temporal connectives will be defined in two copies, one underlined and the other overlined. In the next section, we will see that the underlined connectives are evaluated in the current interval, while overlined ones will be evaluated along the sequence of intervals. A *formula* is defined inductively as:

$$\phi ::= p \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid \phi_1 \underline{\mathsf{U}} \phi_2 \mid \phi_1 \overline{\mathsf{U}} \phi_2$$

where $p$ is an attribute or an action, $\phi_1, \phi_2$ are formulae. We use the classical boolean abbreviations $\mathsf{f}, \mathsf{t}, \vee, \Rightarrow, \Leftrightarrow$. We use the strict, irreflexive, existential (strong) version of the $\mathsf{U}$ since all other temporal connectives are defined from this one (eqs. (1)–(7)). For saving space we only write one copy [1] below:

$$\mathsf{X}\,\phi = \mathsf{f}\;\mathsf{U}\;\phi \quad (1)$$

$$\mathsf{X}^?\,\phi = \neg\,\mathsf{X}\,\neg\phi \quad (2)$$

$$\phi\;\mathsf{U}_\circ\;\psi = \psi \vee (\phi \wedge \phi\;\mathsf{U}\;\psi) \quad (3)$$

$$\mathsf{F}\phi = \mathsf{t}\;\mathsf{U}\;\phi \quad (4)$$

$$\mathsf{F}_\circ\phi = \phi \vee \mathsf{F}\phi \quad (5)$$

$$\mathsf{G}\phi = \neg\mathsf{F}\neg\phi \quad (6)$$

$$\mathsf{G}_\circ\phi = \neg\mathsf{F}_\circ\neg\phi \quad (7)$$

$\overline{\mathsf{X}}$ means "at the beginning of next interval", i.e., "after next mutation", while $\underline{\mathsf{X}}$ means "at the next point in the same interval". When there is no such point, the formula is considered false. [2] In contrast, the "weak next" operator $\mathsf{X}^?$ is true in this case. The index $_\circ$ indicates that the present is included, according to Prior's convention. For instance, $\mathsf{U}_\circ$ is the reflexive (non-strict) until, which has been popularised by [13]. It allows $\psi$ to be true now. Similarly, $\underline{\mathsf{F}}$ means "sometime in the future but before next mutation", while $\underline{\mathsf{F}}_\circ$ also includes the present. Dually, $\underline{\mathsf{G}}$ means "always in the future but before next mutation", while $\underline{\mathsf{G}}_\circ$ also includes the present.

### *2.2 Semantics*

We interpret our logic $\mathsf{U2}$ in sequences of states, which model the evolution of the object. A *state* indicates the current value of attributes, and which actions are taking place (their effects will be seen in the next state): $\mathbb{S} = 2^A \times 2^\Delta$.

We will use indices $_A, _\Delta$ to indicate the first (attributes) and second component (actions) of the state. Similarly, $s_M = s_\Delta \cap M$ will be the current mutations.

Time is assumed to be discrete, and modelled by natural numbers: $\mathbb{T} = I\!N$. An *interpretation* $m$ is a total function: $\mathbb{T} \to \mathbb{S}$, i.e., an infinite sequence of states. An interval will be a subsequence of states, separated by a mutation.

[1] The convention above for writing two axioms by means of one should not be confused with the usual global temporal operators, which are defined in Section 5, equations 25–31.

[2] Both operators are defined in eq. (1), due to our convention.

Only the last interval can thus be infinite. The last state of an interval, if it exists, always contains a mutation.

For readability, we define the function "beginning of next interval" $\nu_m : I\!N \to I\!N \cup \{\infty\}$: the successor of the least $t$ such that $t \geq n$ and $m_M(t) \neq \emptyset$, or $\infty$ if it does not exist, i.e., when the current interval is infinite. Then we can define when an interpretation $m$ *satisfies* a formula $\phi$ at time $n$, noted as $m, n \Vdash_{\Sigma}^{\mathsf{U2}} \phi$ or simply $n \Vdash \phi$:

(1) $n \Vdash p$ iff $p \in m(n)$
(2) $n \Vdash \phi \wedge \psi$ iff $n \Vdash \phi$ and $n \Vdash \psi$
(3) $n \Vdash \neg\phi$ iff $n \nVdash \phi$
(4) $n \Vdash \phi \underline{\mathsf{U}} \psi$ iff $\exists n'. n < n' < \nu_m(n) \wedge n' \Vdash \psi$and $\forall n''. n < n'' < n' \Rightarrow n'' \Vdash \phi$
(5) $n \Vdash \phi \overline{\mathsf{U}} \psi$ iff $\exists n' > 0. \nu_m^{n'}(n) \Vdash \psi$ and $\forall n''. 0 < n'' < n' \Rightarrow \nu_m^{n''}(n) \Vdash \phi$

where $\nu^n$ denotes $n$ iterations (functional compositions) of the function $\nu$: $\nu^n = \nu \circ \ldots \circ \nu$. The similarity of the definitions of the two $\mathsf{U}$ can better be understood by seeing $\nu$ as an accessibility relation used for $\overline{\mathsf{U}}$. In the definition of $\underline{\mathsf{U}}$, we use implicitly the usual successor $(n+1)$ as accessibility relation, except at the end of intervals. Thus, in the definition of $\overline{\mathsf{U}}$, the iterations go through the successive intervals. We convene that when $\nu$ returns $\infty$, this state satisfies no formula, and thus $\overline{\mathsf{U}}$ cannot use it.

## 3 Complexity

This section aims to discuss the relation of propositional $\mathsf{U2}$ to $\mathsf{PTL}$. We can linearly translate any $\mathsf{PTL}$ formula into propositional $\mathsf{U2}$, preserving models and thus satisfiability. The translation of a classical signature $P$ is obtained by taking $A = P$ and $\Delta = \emptyset$. This is the intuitive translation: proposition symbols of $\mathsf{PTL}$ play the same role as our attributes. The translation of a formula is obtained by just underlining its temporal operators (overlined ones are never used, since there are no mutations declared.) This translation of $\phi$ is noted $\underline{\phi}$.

**Theorem 3.1** *For any classical model $m$: $m, n \Vdash_{P}^{\mathsf{PTL}} \phi$ iff $m, n \Vdash_{(P,\emptyset,\emptyset)}^{\mathsf{U2}} \underline{\phi}$.*

This linear reduction ensures that the complexity of deciding $\mathsf{U2}$ is Pspace-hard, since it is known [20,6] that the problem of deciding the satisfiability (and the validity) of a classical propositional temporal formula containing $\mathsf{U}$ is Pspace-complete. For the other direction, the translation of a $\mathsf{U2}$ signature is obtained by considering all symbols as propositions: $P = A \cup \Delta$. We define the special formula noted $\star$ as in [2], stating that a mutation is about to take place:

$$\star = \bigvee_{\mu \in M} \mu \tag{8}$$

The $\mathsf{U2}$ formulae can then be translated by:

$$T(\phi \underline{\mathsf{U}} \psi) = \neg \star \wedge (T(\phi) \wedge \neg\star) \,\mathsf{U}\, T(\psi) \tag{9}$$
$$T(\phi \overline{\mathsf{U}} \psi) = (\star \Rightarrow \mathsf{X}\, T(\phi)) \,\mathsf{U}_\circ\, (\star \wedge \mathsf{X}\, T(\psi)) \tag{10}$$
$$T(\phi \wedge \psi) = T(\phi) \wedge T(\psi) \tag{11}$$
$$T(\neg\phi) = \neg T(\phi) \tag{12}$$
$$T(p) = p \tag{13}$$

Again, this translation preserves models:

**Theorem 3.2** $m, n \Vdash^{\mathsf{U2}} \phi$ *iff* $m, n \Vdash^{\mathsf{PTL}} T(\phi)$

It is a linear reduction, as is seen immediately from the fact that $T$ is only used once for each subformula in the recursive definition.

Therefore the satisfiability problem of $\mathsf{U2}$ is Pspace-complete.

## 4 Axiomatization

The two $\mathsf{U}$ operators used here are classical, and thus obey the axiomatization of Pnueli [9]:

$$\mathsf{X}\,\phi \Rightarrow \mathsf{X}^?\,\phi \tag{14}$$
$$\mathsf{X}^?(\phi \Rightarrow \psi) \Rightarrow (\mathsf{X}^?\,\phi \Rightarrow \mathsf{X}^?\,\psi) \tag{15}$$
$$\mathsf{G}(\phi \Rightarrow \psi) \Rightarrow (\mathsf{G}\phi \Rightarrow \mathsf{G}\psi) \tag{16}$$
$$\mathsf{G}\phi \Rightarrow \mathsf{X}^?\,\phi \tag{17}$$
$$\mathsf{G}_\circ(\phi \Rightarrow \mathsf{X}^?\,\phi) \Rightarrow (\phi \Rightarrow \mathsf{G}_\circ\phi) \tag{18}$$
$$\phi \,\mathsf{U}\, \psi \Leftrightarrow \mathsf{X}(\phi \,\mathsf{U}_\circ\, \psi) \tag{19}$$
$$\phi \,\mathsf{U}\, \psi \Rightarrow \mathsf{F}\psi \tag{20}$$

Axiom (14) is a usual way to state that the successor is functional: there is at most one successor. Axiom 15 (Kripke's distribution) indicates that $\mathsf{X}^?$ is a universal modality (it does not imply the existence of a successor), and similarly for 16. Axioms (19), (20) have been slightly adapted to use $\mathsf{U}$; $\mathsf{U}_\circ$ stands here as a shorthand.

We include thus two copies of these axioms, one underlined and one overlined, all propositional tautologies, the modus ponens and two copies of the necessitation rule:

$$\frac{\phi}{\mathsf{G}\phi} \tag{21}$$

We only need three further axioms to relate the two levels of the logic:

$$\neg\underline{\mathsf{X}}\mathsf{t} \Leftrightarrow \star \tag{22}$$

$$\neg\overline{\mathsf{X}}\mathsf{t} \Leftrightarrow \underline{\mathsf{G}}_\circ \neg \star \tag{23}$$

$$\underline{\mathsf{F}}_\circ \overline{\mathsf{X}}\phi \Rightarrow \underline{\mathsf{G}}_\circ \overline{\mathsf{X}}\phi \tag{24}$$

The axioms express the following intuitions: (22) indicates that intervals end at mutations, (23) says that there is no next interval when the current one is infinite, (24) indicates that all states of an interval have the same next interval.

## 5 Completeness

The translations that we have used above are designed to preserve complexity. For axiomatization, we will use translations that return to the original signature since the proof system depends on the signature.

We use the following simple technique:

**Lemma 5.1** *Let $L_\Sigma, L'_\Sigma$ be two logics with translations $T_\Sigma : L_\Sigma \to L'_\Sigma$, $S_\Sigma : L'_\Sigma \to L_\Sigma$ preserving validity and assume that we have a complete inference system $I'_\Sigma$ for $L'_\Sigma$. Then any inference system for $L_\Sigma$ allowing to derive $S_\Sigma(I'_\Sigma)$, $\frac{S_\Sigma(T_\Sigma(\phi))}{\phi}$ is complete.*

**Proof.** Given a valid $\phi \in L_\Sigma$, $T_\Sigma(\phi)$ is valid. By completeness of $I'$ we have a proof $P$ of $T_\Sigma(\phi)$. $S_\Sigma$ can be applied to each element of this proof, yielding a proof $S_\Sigma(P)$ of $S_\Sigma(T_\Sigma(\phi))$. We apply the last rule to prove $\phi$. □

We define the translation $S_\Sigma$ below, omitting $\Sigma$ for readability:

$$S(\phi \mathsf{U} \psi) = \phi\underline{\mathsf{U}}\psi \vee (\underline{\mathsf{G}}\phi \wedge (\underline{\mathsf{G}}_\circ\phi)\overline{\mathsf{U}}(\phi\underline{\mathsf{U}}_\circ\psi)) \tag{25}$$

$$S(\phi \wedge \psi) = S(\phi) \wedge S(\psi) \tag{26}$$

$$S(\neg\phi) = \neg S(\phi) \tag{27}$$

$$S(p) = p \tag{28}$$

In particular:

$$S(\phi \mathsf{U}_\circ \psi) \Leftrightarrow \underline{\mathsf{G}}_\circ\phi\overline{\mathsf{U}}_\circ(\phi\underline{\mathsf{U}}_\circ\psi) \tag{29}$$

$$S(\mathsf{X}\psi) \Leftrightarrow \underline{\mathsf{X}}\psi \vee (\underline{\mathsf{G}}\mathsf{f} \wedge \overline{\mathsf{X}}\psi) \tag{30}$$

$$S(\mathsf{G}_\circ\psi) \Leftrightarrow \overline{\mathsf{G}}_\circ\underline{\mathsf{G}}_\circ\psi \tag{31}$$

**Theorem 5.2** *The rules 14–24 are complete for propositional* U2 *models.*

For the proof, we use the technique above. Most proofs are simple temporal logic proofs, and thus can be automated.

## 6 Conclusions

This paper presented a simple way to specify the evolution of specifications during the execution of a system, and the possible interferences between mutations and current operation.

Although the need for two levels of reasoning is clearly recognised for this problem, our previous attempts [2] use a powerful meta-level that makes reasoning difficult. Here, the two levels are simply distributed over time, allowing a uniform treatment by temporal reasoning. We have shown that this logic is complete, as expressive as classical temporal logic, and can be analysed with the same tools (model checkers, provers, etc.) with the same execution time.

Though more applicable than previous approaches, much remains to be done for practical applicability:

- Methodological guidelines for the planning of evolving requirements;
- Development of a full specification language (see [2], [16]);
- Systems to help in the implementation of mutations;
- Tools and theory for architectural change: here mutations occur in a single object, but the overall structure of the system is still fixed.

## References


[1] Barry W. Boehm. Improving software productivity. *Computer*, 20(9):43–57, September 1987.

[2] S. Conrad, J. Ramos, G. Saake, and C. Sernadas. Evolving Logical Specification in Information Systems. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, chapter 7, pages 199–228. Kluwer Academic Publishers, Boston, 1998.

[3] Stefan Conrad and Gunter Saake. Extending temporal logic for capturing evolving behaviour. In Zbigniew W. Ras and Andrzej Skowron, editors, *ISMIS*, volume 1325 of *LNCS*, pages 60–71. Springer, 1997.

[4] E. Dubois. ALBERT: A formal language and its supporting tools for requirements engineering. *Lecture Notes in Computer Science*, 1382:322, 1998.

[5] José Fiadeiro and Tom Maibaum. Sometimes "tomorrow" is "sometime": Action refinement in a temporal logic of objects. In D. Gabbay and H. Ohlbach, editors, *Temporal Logic*, volume 827 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 1994.

[6] D. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic*. Clarendon Press, Oxford, 1994.

[7] J. Anthony Hall. Seven myths of formal methods. *IEEE Software*, 7(5):11–19, September 1990.

[8] Ralf Jungclaus, Gunter Saake, Thorsten Hartmann, and Cristina Sernadas. TROLL—A language for object-oriented specification of information systems. *ACM Transactions on Information Systems*, 14(2):175–211, April 1996.

[9] O. Lichtenstein, A. Pnueli, and L.D. Zuck. The glory of the past. In R. Parikh, editor, *Logics of Programs*, LNCS 193, pages 196–218. Springer, 1985.

[10] Stephan Merz and Fred Kröger. Layers of temporal structures. *Fundamenta Informaticae*, 1991.

[11] Angelo Montanari, Adriano Peron, and Alberto Policriti. Theories of $\omega$-layered metric temporal structures: Expressiveness and decidability. *L. J. IGPL*, page 24, 1998.

[12] Angelo Montanari and Alberto Policriti. Decidability results for metric and layered temporal logics. *Notre Dame Journal of Formal Logic*, 37(2):260–282, Spring 1996.

[13] A. Pnueli. Specification and Development of Reactive Systems. In H.-J. Kugler, editor, *Information Processing 86*, pages 845–858. IFIP, North-Holland, 1986.

[14] J. Ramos and A. Sernadas. A brief introduction to GNOME. Research report, Section of Computer Science, Department of Mathematics, Instituto Superior Técnico, 1096 Lisboa, Portugal, 1995.

[15] G. Saake, A. Sernadas, and C. Sernadas. Evolving Object Specifications. In R. Wieringa and R. Feenstra, editors, *Information Systems — Correctness and Reusability. Selected Papers from the IS-CORE Workshop*, pages 84–99. World Scientific Publishing, Singapore, 1995.

[16] Pierre-Yves Schobbens, Amilcar Sernadas, Cristina Sernadas, and Gunter Saake. A two-level temporal logic for evolving specifications. Technical report, FUNDP and Instituto Superior Tecnico, 1998.

[17] A. Sernadas, J. F. Costa, and C. Sernadas. An institution of object behaviour. In H. Ehrig and F. Orejas, editors, *Recent Trends in Data Type Specification*, volume 785 of *Lecture Notes in Computer Science*, pages 337–350. Springer-Verlag, 1994.

[18] A. Sernadas, C. Sernadas, and J. F. Costa. Object specification logic. *Journal of Logic and Computation*, 5(5):603–630, 1995. Available as Research Report since 1992.

[19] A. Sernadas, C. Sernadas, and H.-D. Ehrich. Object-oriented specification of databases: an algebraic approach. In P. Hammersley, editor, *Very Large Data Bases 87*, pages 107–116. Morgan and Kaufmann, 1987.

[20] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logic. *Journal of the ACM*, 32(3):733–749, July 1985.

[21] Can Türker, Stefan Conrad, and Gunter Saake. Dynamically changing behavior: An agent-oriented view to modeling intelligent information systems. In Zbigniew W. Ras and Maciej Michalewicz, editors, *ISMIS*, volume 1079 of *LNCS*, pages 572–581. Springer, 1996.

[22] R. W. Weyrauch. Prologomena to a theory of mechanized formal reasoning. *Artificial Intelligence*, 13(1 and 2):133–170, April 1980.