Conceptual Modeling in Full Computation-Tree Logic with Sequence Modal Operator*

Ken Kaneiwa[1] and Norihiro Kamide[2]

[1] Department of Electrical Engineering and Computer Science, Iwate University
4-3-5 Ueda, Morioka, Iwate 020-8551, JAPAN
kaneiwa@cis.iwate-u.ac.jp
[2] Waseda Institute for Advanced Study
1-6-1 Nishi Waseda, Shinjuku-ku, Tokyo 169-8050, JAPAN
drnkamide08@kpd.biglobe.ne.jp

Abstract. In this paper, we propose a method for modeling concepts in full computation-tree logic with sequence modal operators. An extended full computation-tree logic, CTLS*, is introduced as a Kripke semantics with a sequence modal operator. This logic can appropriately represent hierarchical tree structures in cases where sequence modal operators in CTLS* are applied to tree structures. We prove a theorem for embedding CTLS* into CTL*. The validity, satisfiability, and model-checking problems of CTLS* are shown to be decidable. An illustrative example of biological taxonomy is presented using CTLS* formulas.

1 Introduction

Full computation-tree logic, CTL*[6, 5], is known as one of the most important branching-time temporal logics using computation-trees to specify and verify concurrent systems. CTL* is sufficiently expressive for representing almost all the important temporal properties, such as liveness, fairness, and safety of concurrent systems. CTL* is more expressive than *computation-tree logic* (CTL) and *linear-time temporal logic* (LTL). CTL [2] is a useful subsystem of CTL*, but cannot express some important properties, such as strong fairness. LTL [12] is also a useful subsystem of CTL*, but cannot express the properties that verify the existence of a path. An important feature of CTL* is that the existence of paths in computation-trees can be specified and verified. A computation-tree is one representation of a non-deterministic computation or unwinding of a Kripke structure, which is a directed graph; thus it can naturally express tree structures.

However, CTL* is not suitable for representing the highly complex and informative structures of ontologies and taxonomies. This is because "normal" trees are not sufficiently expressive to represent such complex structures. *Hierarchical tree structures* (a combination of tree and hierarchical structures) are better suited for this purpose because they are useful for representing ontologies

* This paper is an extended version of [8].

and taxonomies in computer science applications. For example, *biological ontologies*, which are knowledge representation models with hierarchies of biological vocabularies, are usually represented by ISA (subconcept relation), PART-OF (inclusion relation), LOCATED-IN (spatial relation), and PROCEEDED-BY (temporal relation) using hierarchical trees or directed acyclic graphs. A biological process *pathway* is searched by finding a path in such a hierarchical tree or directed acyclic graph. In order to represent temporal properties in hierarchical tree structures, we require a highly expressive branching-time temporal logic with a new modal operator.

The aim of this study is to propose a method for modeling concepts in hierarchical tree structures (i.e., computation-trees with additional information). The concept modeling motivates us to improve CTL* with sequence modal operators. We introduce a *sequence modal operator* $[b]$, which represents a sequence b of symbols, to describe the ordered labels in a hierarchy.

The reason for using the notion of "sequences" in the new modal operator is explained below. The sequences concept is fundamental to practical reasoning in computer science, because it can appropriately represent data sequences, program-execution sequences, action sequences, time sequences, word (character or alphabet) sequences, DNA sequences etc. The notion of sequences is thus useful for representing the notions of information, attributes, trees, orders, preferences, strings, vectors, and ontologies. Additional information can be represented by sequences; which is useful because a sequence structure gives a *monoid* $\langle M, ;, \emptyset \rangle$ with *informational interpretation* [15]:

1. M is a set of pieces of (ordered or prioritized) information (i.e., a set of sequences),
2. ; is a binary operator (on M) that combines two pieces of information (i.e., a concatenation operator on sequences),
3. $\emptyset$ is the empty piece of information (i.e., the empty sequence).

The sequence modal operator $[b]$ represents labels as "additional information." A formula of the form $[b_1 \ ; \ b_2 \ ; \cdots ; \ b_n]\alpha$ intuitively means that "α is true based on a sequence $b_1 \ ; \ b_2 \ ; \cdots ; \ b_n$ of (ordered or prioritized) information pieces." Further, a formula of the form $[\emptyset]\alpha$, which coincides with α, intuitively means that "α is true without any information (i.e., it is an eternal truth in the sense of classical logic)." Simple and intuitive satisfaction relations called *sequence-indexed satisfaction relations* are required to formalize the sequence modal operator. These satisfaction relations are regarded as natural extensions of the standard two-valued satisfaction relation of classical logic. The sequence-indexed satisfaction relations, denoted as $\models^{\hat{d}}$, are indexed by a sequence $\hat{d}$, and the special case $\models^{\emptyset}$ corresponds to the classical two-valued satisfaction relation. Then, $\models^{\hat{d}} \alpha$ means that "α is true based on a sequence $\hat{d}$ of information pieces" and $\models^{\emptyset} \alpha$ means that "α is eternally true without any information."

This paper focuses on conceptual modeling by means of sequence modal formulas in an extended full computation-tree logic, CTLS*. If a sequence of concept symbols is expressed by a sequence modal operator, then each order of

conceptual modeling	sequence and formula
concept c_i isa concept c_j $(1 \leq i < j \leq n)$	$[c_1; c_2; \cdots; c_n]$
concept c has property p	$[c]p$
concept c has properties p_1 and p_2	$[c](p_1 \wedge p_2)$
concept c has property p_1 or p_2	$[c](p_1 \vee p_2)$
concept c does not have property p	$[c](\neg p)$

Table 1. Modeling of Concepts and Properties

concepts in the sequence is regarded as an ISA relation between the concepts in a hierarchy. This formulation enables us to verify some temporal properties of concepts in hierarchical tree structures.

The contents of this paper are summarized as follows: In Section 2, we provide a method for modeling concepts in sequence modal operators and hierarchical tree structures. An illustrative example of biological taxonomy using CTLS* formulas is presented. In Section 3, an extended CTL* with the sequence modal operator is introduced as a Kripke semantics with sequence-indexed satisfaction relations. In Section 4, an embedding theorem that explains the introduction of CTLS* into CTL* is described. The validity, satisfiability, and model-checking problems of CTLS* are shown to be decidable. In Section 5, some technical remarks on extensions and modifications of CTLS* are given. In Section 6, we conclude this paper.

2 Conceptual Modeling

In this section, we consider how to represent complex ontologies and taxonomies using sequence modal operators and hierarchical tree structures.

2.1 Sequence Modal Operators

In ontology representation, a concept hierarchy is constructed by ISA relations between concepts, i.e., a concept is a subconcept of another concept [9, 11, 10]. In this study, we use sequence modal operators to represent ISA relations between concepts. Let $c_1, c_2, \ldots, c_n$ be concept symbols. Then, we write a sequence of concept names by $[c_1; c_2; \cdots; c_n]$ (called a *sequence modal operator*). Each order (c_i, c_j) $(1 \leq i < j \leq n)$ of concepts in the sequence $[c_1; c_2; \cdots; c_n]$ can be used to represent the ISA relation between c_i and c_j (as shown in Table 1). For example, we declare the following order of two concepts as an ISA relation between birds and animals.

[bird;animal]

This sequence expresses that concept *bird* is a subconcept of concept *animal*, indicating that every bird is an animal.

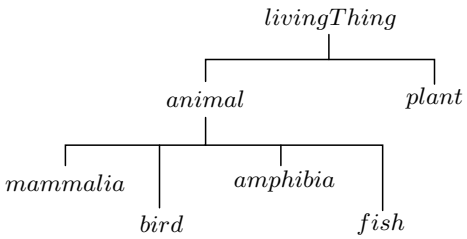

Fig. 1. A biological taxonomy

Consider the example of biological taxonomy (Linnaean taxonomy [1]) shown in Fig 1. Every concept in the taxonomy is described by the following sequence modal operators.

$[livingThing]$
$[animal; livingThing]$
$[plant; livingThing]$
$[mammalia; animal; livingThing]$
$[bird; animal; livingThing]$
$[amphibia; animal; livingThing]$
$[fish; animal; livingThing]$

For the concept hierarchy, the different meanings of concepts are characterized by some properties. Using the sequence modal operators, some properties of each concept are declared by the concept modeling as in Table 1. Let c be a concept and p be a property. Then, $[c]p$ is a sequence modal formula that implies that concept c has property p. If we write $[c_1; c_2]p$, then it means that concept c_1 is a subconcept of c_2 and has property p. For example, the following expression says that every bird is an animal and a living thing and can fly.

$$[bird; animal; livingThing]canFly$$

In addition, some complex properties of concepts are expressed by logical connectives of properties p, p_1, and p_2, such as $[c](p_1 \wedge p_2)$, $[c](p_1 \vee p_2)$, and $[c](\neg p)$ in Table 1. For example, the following sequence modal formula

$$[fish; animal; livingThing](inWater \wedge \neg canFly)$$

shows the property that every fish is an animal and a living thing in water, but it cannot fly.

2.2 Temporal Properties

For complex conceptual modeling, we consider the fact that some concepts have temporal properties; which is why the above sequence modal formulas are used in conceptual modeling. A concept can be said to be characterized by some properties but the properties may be temporally dynamic in the life cycles of instances

conceptual modeling	formula
concept c has property p eventually in all paths	$[c]\mathrm{AF}p$
concept c always has property p in all paths	$[c]\mathrm{AG}p$
concept c has property p next in all paths	$[c]\mathrm{AX}p$
concept c has property p_1 until it has property p_2 in all paths	$[c]\mathrm{A}(p_1\mathrm{U}p_2)$
concept c has property p eventually in some path	$[c]\mathrm{EF}p$
concept c always has property p in some path	$[c]\mathrm{EG}p$
concept c has property p next in some path	$[c]\mathrm{EX}p$
concept c has property p_1 until it has property p_2 in some path	$[c]\mathrm{E}(p_1\mathrm{U}p_2)$

Table 2. Modeling of Temporal Properties

of the concept. For example, the expression $[bird; animal; livingThing]canFly$ lacks temporal meaning. This is because it implies that every bird can fly but does not say that any adult bird can fly but any child bird cannot fly.

We model such a temporal property of a concept using a full computation-tree logic with sequence modal operators (called CTLS*). In the biological taxonomy, the concepts are partially ordered by sequence modal operators and the life cycle of each concept is characterized by temporal operators. Let c be a concept, p be a property, and E (some computation path), X (next), U (until), A (all computation paths), G (always), and F (eventually) be temporal operators. Table 2 shows the conceptual modeling of temporal properties in CTLS* formulas. For example, we can express some temporal properties of concept *bird* as follows.

$$[bird; animal; livingThing]\mathrm{EF}(\neg canFly \wedge (\mathrm{EF}canFly))$$

This formula implies the property that every bird cannot fly in a certain state but it can fly eventually.

Consider representing temporal properties of the concepts in the biological taxonomy. First, some temporal properties of concept *livingThing* are formalized by CTLS* formulas as follows:

$$[livingThing]\mathrm{AF}(living \wedge \mathrm{AF}(\neg living) \wedge \mathrm{A}(living\mathrm{U}(\mathrm{AG}\neg living)))$$

By combining sequence modal operators and temporal operators, this formula implies that every living thing is alive until it dies.

$$[animal; livingThing]\mathrm{AF}(motile \vee sentient)$$
$$[plant; livingThing]\mathrm{AG}\neg(motile \vee sentient)$$

These formulas classify living things into animals and plants by characterizing their life cycles. Every animal is motile or sentient, while plants are neither motile nor sentient. The orders of symbols in sequence modal operators are useful in representing a concept hierarchy. This is because the orders differentiate their validities; in other words, the two formulas $[b; c]\alpha$ and $[c; b]\alpha$ are not equivalent.

In addition, the abovementioned example includes both temporal operators and sequence modal operators; CTL* and other logics do not include both these types of operators.

Moreover, several subcategories of animals are defined by the following CTLS* formulas.

$[mammalia; animal; livingThing]\mathrm{AF}(\neg egg \wedge \mathrm{AX}child \wedge \mathrm{AXAX}adult)$
$[bird; animal; livingThing]\mathrm{AF}(egg \wedge \mathrm{AX}(child \wedge \neg canFly) \wedge \mathrm{AXAX}(adult \wedge canFly))$

Every mammal is viviparous. In the course of its life cycle, it grows from a juvenile into an adult. Every bird is oviparous and can fly when it grows into an adult. In addition to the concepts, amphibians and fish are modeled by the changes occurring during growth such that every fish is always in water, but every amphibian lives both in water and on land. We can write the following formulas where the property of birds is revised and the concepts of amphibians and fish are defined.

$[bird; animal; livingThing](\mathrm{AG}(\neg inWater) \wedge \mathrm{AF}(egg \wedge \mathrm{AX}(child \wedge \neg canFly) \wedge \mathrm{AXAX}(adult \wedge canFly)))$
$[amphibia; animal; livingThing]\mathrm{AF}((egg \wedge inWater) \wedge \mathrm{AX}(child \wedge inWater) \wedge \mathrm{AXAX}adult)$
$[fish; animal; livingThing](\mathrm{AG}(inWater) \wedge \mathrm{AF}(egg \wedge \mathrm{AX}child \wedge \mathrm{AXAX}adult))$

The differences among three concepts, *bird*, *amphibia*, and *fish*, are characterized by their living places. Every bird does not live in water but every fish always lives in water. The properties of amphibians and fish are similar to each other, but unlike fish, every amphibian can live both in the water and on land after it becomes an adult.

2.3 Hierarchical Tree Structures

The sequence modal operators in CTLS* are applied to hierarchical tree structures where each hierarchical tree structure is a specific modal of concepts in a hierarchy. Fig 2 shows a hierarchical tree structure of life cycles of concepts *mammalia* and *bird* in the biological taxonomy. We define a Kripke structure $M = \langle S, S_0, R, L^{\hat{d}} \rangle$ that corresponds to a model of the biological taxonomy as follows:

1. $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$,
2. $S_0 = \{s_0\}$,
3. $R = \{(s_0, s_1), (s_1, s_2), (s_2, s_3), (s_3, s_4), (s_4, s_4), (s_0, s_5), (s_5, s_6), (s_6, s_7), (s_7, s_4)\}$,
4. $L^{livingThing}(s_0) = L^{animal}(s_0) = \emptyset$,
5. $L^{livingThing}(s_1) = L^{animal}(s_1) = L^{mammalia}(s_1) = \{living\}$,
6. $L^{livingThing}(s_2) = L^{animal}(s_2) = L^{mammalia}(s_2) = \{child, living\}$,
7. $L^{livingThing}(s_3) = L^{animal}(s_3) = L^{mammalia}(s_3) = \{adult, living\}$,

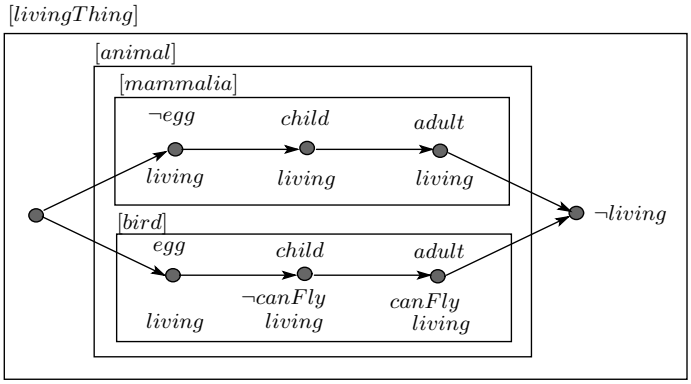

Fig. 2. Life cycles of mammalia and bird

8. $L^{livingThing}(s_4) = L^{livingThing}(s_4) = L^{mammalia}(s_4) = L^{bird}(s_4) = \emptyset$,
9. $L^{livingThing}(s_5) = L^{animal}(s_5) = L^{bird}(s_5) = \{egg, living\}$,
10. $L^{livingThing}(s_6) = L^{animal}(s_6) = L^{bird}(s_6) = \{child, living\}$,
11. $L^{livingThing}(s_7) = L^{animal}(s_7) = L^{bird}(s_7) = \{canFly, adult, living\}$,
12. $L^{mammalia}(s_0) = L^{mammalia}(s_5) = L^{mammalia}(s_6) = L^{mammalia}(s_7) = \emptyset$,
13. $L^{bird}(s_0) = L^{bird}(s_1) = L^{bird}(s_2) = L^{bird}(s_3) = \emptyset$.

We can verify the existence of a path that represents required information in the structure M. For example, we can verify: "Is there an animal that hatches from an egg and can fly?" This statement is expressed as:

$$[animal](\mathrm{EF}egg(\wedge \mathrm{EF}canFly))$$

The above statement is true because we have a path $s_0 \rightarrow s_5 \rightarrow s_6 \rightarrow s_7$ with $egg \in L^{animal}(s_5)$ and $canFly \in L^{animal}(s_7)$.

As discussed in the above, concept *animal* is further classified into concepts *bird*, *amphibia*, and *fish*. Fig 3 shows a hierarchical tree structure of their life cycles in the biological taxonomy. We define a Kripke structure $M = \langle S, S_0, R, L^{\hat{d}} \rangle$ that corresponds to a model of the biological taxonomy as follows:

1. $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}\}$,
2. $S_0 = \{s_0\}$,
3. $R = \{(s_0, s_1), (s_1, s_2), (s_2, s_3), (s_3, s_4), (s_4, s_4), (s_0, s_5), (s_5, s_6), (s_6, s_7), (s_7, s_4), (s_0, s_8), (s_8, s_9), (s_9, s_{10}), (s_{10}, s_4)\}$,
4. $L^{livingThing}(s_0) = L^{animal}(s_0) = \emptyset$,
5. $L^{livingThing}(s_1) = L^{animal}(s_1) = L^{bird}(s_1) = \{egg, living\}$,
6. $L^{livingThing}(s_2) = L^{animal}(s_2) = L^{bird}(s_2) = \{child, living\}$,
7. $L^{livingThing}(s_3) = L^{animal}(s_3) = L^{bird}(s_3) = \{adult, canFly, living\}$,
8. $L^{livingThing}(s_4) = L^{livingThing}(s_4) = \emptyset$,
9. $L^{livingThing}(s_5) = L^{animal}(s_5) = L^{amphibia}(s_5) = \{egg, inWater, living\}$,
10. $L^{livingThing}(s_6) = L^{animal}(s_6) = L^{amphibia}(s_6) = \{child, inWater, living\}$,

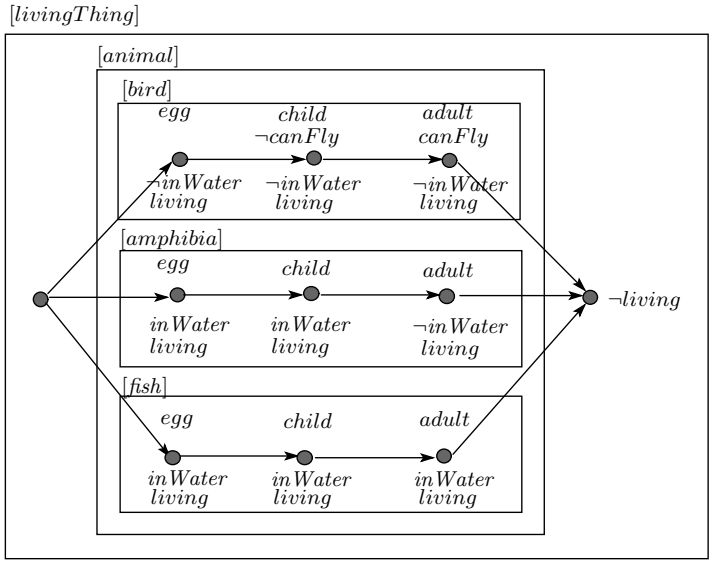

Fig. 3. Life cycles of bird, amphibias, and fish

11. $L^{livingThing}(s_7) = L^{animal}(s_7) = L^{amphibia}(s_7) = \{adult, living\}$,
12. $L^{livingThing}(s_8) = L^{animal}(s_8) = L^{amphibia}(s_8) = \{egg, inWater, living\}$,
13. $L^{livingThing}(s_9) = L^{animal}(s_9) = L^{amphibia}(s_9) = \{child, inWater, living\}$,
14. $L^{livingThing}(s_{10}) = L^{animal}(s_{10}) = L^{amphibia}(s_{10}) = \{adult, inWater, living\}$,
15. $L^{bird}(s_0) = L^{bird}(s_4) = L^{bird}(s_5) = \cdots = L^{bird}(s_{10}) = \emptyset$,
16. $L^{amphibia}(s_0) = L^{amphibia}(s_1) = \cdots = L^{amphibia}(s_4) = L^{amphibia}(s_8) = L^{amphibia}(s_9) = L^{amphibia}(s_{10}) = \emptyset$,
17. $L^{fish}(s_0) = L^{fish}(s_1) = \cdots = L^{fish}(s_7) = \emptyset$.

We can verify the existence of a path that represents required information in the structure M. For example, we can verify: "Is there an animal that hatches from an egg and is in water until it dies?" This statement is expressed as:

$$[animal](\mathrm{EF}egg \wedge (\mathrm{A}(inWater \mathrm{U} \neg living)))$$

The above statement is true because we have a path $s_0 \rightarrow s_8 \rightarrow s_9 \rightarrow s_{10} \rightarrow s_4$ with $egg \in L^{animal}(s_8)$, $\{inWater, living\} \subseteq L^{animal}(s_8) \cap L^{animal}(s_9) \cap L^{animal}(s_{10})$, and $living \notin L^{animal}(s_4)$.

Moreover, we can verify: "Is there an animal that hatches from an egg and is in water but will not be in water?" This statement is expressed as:

$$[animal](\mathrm{EF}egg \wedge inWater \wedge (\mathrm{EF}\neg inWater))$$

The above statement is true because we have a path $s_0 \rightarrow s_5 \rightarrow s_6 \rightarrow s_7$ with $egg \in L^{animal}(s_5)$, $inWater \in L^{animal}(s_5)$, and $inWater \notin L^{animal}(s_7)$.

3 Full Computation-Tree Logic with Sequence Modal Operator

Formulas of CTLS*, which are defined by combining two types of formulas *state formulas* and *path formulas*, are constructed from atomic formulas, $\top$ (truth constant), $\vee$ (disjunction), $\neg$ (negation), E (some computation path), X (next), U (until) and $[b]$ (sequence modal operator) where b is constructed from atomic sequence, the empty sequence $\emptyset$ and ; (concatenation). The other connectives $\bot$ (falsity constant), $\rightarrow$ (implication), $\wedge$ (conjunction), A (all computation paths), G (always) and F (eventually) can be defined using the connectives displayed above.

Definition 1 *Assume that the numbers of atomic formulas and atomic sequences are respectively countable. The symbol $\emptyset$ represents the empty sequence.*

Formulas α, state formulas β, path formulas γ and sequences b are defined by the following grammar, assuming p and e represent atomic formulas and atomic sequences, respectively:

$$\alpha ::= \beta \mid \gamma$$
$$\beta ::= p \mid \top \mid \beta \vee \beta \mid \neg\beta \mid [b]\beta \mid \mathrm{E}\gamma$$
$$\gamma ::= \gamma \vee \gamma \mid \neg\gamma \mid [b]\gamma \mid \mathrm{X}\gamma \mid \gamma\mathrm{U}\gamma \mid \mathrm{path}(\beta)$$
$$b ::= e \mid \emptyset \mid b \ ; \ b$$

Remark that the "path" in Definition 1 is regarded as an auxiliary function from the set of state formulas to the set of path formulas. This means that a state formula is a path formula. The set of atomic formulas is denoted as ATOM, and the set of sequences (including the empty sequence $\emptyset$) is denoted as SE. Lower-case letters $b, c, ...$ are used for sequences, lower-case letters $p, q, ...$ are used for atomic formulas, and Greek lower-case letters $\alpha, \beta, ...$ are used for (state/path) formulas. The symbol ω is used to represent the set of natural numbers. Lower-case letters i, j and k are used for any natural numbers. The symbol $\geq$ or $\leq$ is used to represent a linear order on ω, and the symbol $>$ or $<$ is used to represent a strict linear order on ω. An expression $A \equiv B$ indicates the syntactical identity between A and B. An expression $[\emptyset]\alpha$ coincides with α, and expressions $[\emptyset \ ; \ b]\alpha$ and $[b \ ; \ \emptyset]\alpha$ coincide with $[b]\alpha$. An expression $[\hat{d}]$ is used to represent $[d_0][d_1][d_2]\cdots[d_i]$ with $i \in \omega$ and $d_0 \equiv \emptyset$, i.e., $[\hat{d}]$ can be the empty sequence. Also, an expression $\hat{d}$ is used to represent $d_0 \ ; \ d_1 \ ; \ d_2 \ ; \ \cdots \ ; \ d_i$ with $i \in \omega$ and $d_0 \equiv \emptyset$.

The logic CTLS* is then defined as a Kripke structure with an infinite number of satisfaction relations.

Definition 2 *A* Kripke structure *for* CTLS* *is a structure* $\langle S, S_0, R, \{L^{\hat{d}}\}_{\hat{d}\in \mathrm{SE}}\rangle$ *such that*

1. *S is a (non-empty) set of states,*
2. *S_0 is a (non-empty) set of initial states and $S_0 \subseteq S$,*

3. *R is a binary relation on S which satisfies the condition: $\forall s \in S\ \exists s' \in S\ [(s, s') \in R]$,*
4. *$L^{\hat{d}}$ ($\hat{d} \in$ SE) are mappings from S to the power set of* AT *($\subseteq$ ATOM).*

Definition 3 *A* path *in a Kripke structure for* CTLS* *is an infinite sequence of states, $\pi = s_0, s_1, s_2, ...$ such that $\forall i \geq 0\ [(s_i, s_{i+1}) \in R]$. An expression π^i means the suffix of π starting at s_i.*

Definition 4 *Let* AT *be a nonempty subset of* ATOM. *Let α_1 and α_2 be state formulas and β_1 and β_2 be path formulas.* Sequence-indexed satisfaction relations $\models^{\hat{d}}$ *($\hat{d} \in$ SE) on a Kripke structure $M = \langle S, S_0, R, \{L^{\hat{d}}\}_{\hat{d} \in \mathrm{SE}} \rangle$ for* CTLS* *are defined as follows (π represents a path constructed from S, s represents a state in S, and e represents an atomic sequence):*

1. *for $p \in$ AT, $M, s \models^{\hat{d}} p$ iff $p \in L^{\hat{d}}(s)$,*
2. *$M, s \models^{\hat{d}} \top$ holds,*
3. *$M, s \models^{\hat{d}} \alpha_1 \vee \alpha_2$ iff $M, s \models^{\hat{d}} \alpha_1$ or $M, s \models^{\hat{d}} \alpha_2$,*
4. *$M, s \models^{\hat{d}} \neg\alpha_1$ iff not-$[M, s \models^{\hat{d}} \alpha_1]$,*
5. *for any atomic sequence e, $M, s \models^{\hat{d}} [e]\alpha_1$ iff $M, s \models^{\hat{d}\ ;\ e} \alpha_1$,*
6. *$M, s \models^{\hat{d}} [b\ ;\ c]\alpha_1$ iff $M, s \models^{\hat{d}} [b][c]\alpha_1$,*
7. *$M, s \models^{\hat{d}} \mathrm{E}\beta_1$ iff there exists a path π from s such that $M, \pi \models^{\hat{d}} \beta_1$,*
8. *$M, \pi \models^{\hat{d}} \mathrm{path}(\alpha_1)$ iff s is the first state of π and $M, s \models^{\hat{d}} \alpha_1$,*
9. *$M, \pi \models^{\hat{d}} \beta_1 \vee \beta_2$ iff $M, \pi \models^{\hat{d}} \beta_1$ or $M, \pi \models^{\hat{d}} \beta_2$,*
10. *$M, \pi \models^{\hat{d}} \neg\beta_1$ iff not-$[M, \pi \models^{\hat{d}} \beta_1]$,*
11. *for any atomic sequence e, $M, \pi \models^{\hat{d}} [e]\beta_1$ iff $M, \pi \models^{\hat{d}\ ;\ e} \beta_1$,*
12. *$M, \pi \models^{\hat{d}} [b\ ;\ c]\beta_1$ iff $M, \pi \models^{\hat{d}} [b][c]\beta_1$,*
13. *$M, \pi \models^{\hat{d}} \mathrm{X}\beta_1$ iff $M, \pi^1 \models^{\hat{d}} \beta_1$,*
14. *$M, \pi \models^{\hat{d}} \beta_1 \mathrm{U} \beta_2$ iff $\exists k \geq 0\ [(M, \pi^k \models^{\hat{d}} \beta_2)$ and $\forall j\ (0 \leq j < k$ implies $M, \pi^j \models^{\hat{d}} \beta_1)]$.*

Proposition 5 *The following clauses hold for any state formula β, any path formula γ and any sequences c and $\hat{d}$,*

1. *$M, s \models^{\hat{d}} [c]\beta$ iff $M, s \models^{\hat{d}\ ;\ c} \beta$,*
2. *$M, s \models^{\emptyset} [\hat{d}]\beta$ iff $M, s \models^{\hat{d}} \beta$,*
3. *$M, \pi \models^{\hat{d}} [c]\gamma$ iff $M, \pi \models^{\hat{d}\ ;\ c} \gamma$,*
4. *$M, \pi \models^{\emptyset} [\hat{d}]\gamma$ iff $M, \pi \models^{\hat{d}} \gamma$.*

Proof. (1) and (3) are proved by induction on c. (2) and (4) are derived using (1) and (3), respectively. We thus show only (1) below.

Case ($c \equiv \emptyset$): Obvious.

Case ($c \equiv e$ for an atomic sequence e): By the definition of $\models^{\hat{d}}$.

Case ($c \equiv b_1\ ;\ b_2$): $M, s \models^{\hat{d}} [b_1\ ;\ b_2]\beta$ iff $M, s \models^{\hat{d}} [b_1][b_2]\beta$ iff $M, s \models^{\hat{d}\ ;\ b_1} [b_2]\beta$ (by induction hypothesis) iff $M, s \models^{\hat{d}\ ;\ b_1\ ;\ b_2} \beta$ (by induction hypothesis). ∎

Definition 6 *A formula α is* valid *(*satisfiable*) if and only if one of the following clauses holds:*

1. *if α is a path formula, then $M, \pi \models^{\emptyset} \alpha$ for any (some) Kripke structure M, any (some) sequence-indexed valuations $\models^{\hat{d}}$ and any (some) path π in M,*
2. *if α is a state formula, then $M, \pi \models^{\emptyset} \mathrm{path}(\alpha)$ for any (some) Kripke structure M, any (some) sequence-indexed valuations $\models^{\hat{d}}$ and any (some) path π in M.*

An expression $\alpha \leftrightarrow \beta$ means $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$.

Proposition 7 *The following formulas are valid: for any formulas α and β, any path formulas γ_1 and γ_2, any state formula δ and any sequences b and c,*

1. $[b](\alpha \vee \beta) \leftrightarrow ([b]\alpha) \vee ([b]\beta)$,
2. $[b](\neg\alpha) \leftrightarrow \neg([b]\alpha)$,
3. $[b\,;\,c]\alpha \leftrightarrow [b][c]\alpha$,
4. $[b]\sharp\gamma_1 \leftrightarrow \sharp[b]\gamma_1$ *where* $\sharp \in \{\mathrm{E}, \mathrm{X}\}$,
5. $[b](\gamma_1 \mathrm{U} \gamma_2) \leftrightarrow ([b]\gamma_1)\mathrm{U}([b]\gamma_2)$,
6. $[b]\mathrm{path}(\delta) \leftrightarrow \mathrm{path}([b]\delta)$.

Definition 8 *Let M be a Kripke structure $\langle S, S_0, R, \{L^{\hat{d}}\}_{\hat{d} \in \mathrm{SE}} \rangle$ for* CTLS**, and $\models^{\hat{d}}$ ($\hat{d} \in$ SE*) be sequence-indexed satisfaction relations on M. Then, the* model checking problem *for* CTLS* *is defined by: for any formula α, find the set $\{s \in S \mid M, s \models^{\emptyset} \alpha\}$.*

4 Embedding and Decidability

The logic CTL* can be defined as a sublogic of CTLS*.

Definition 9 *Let* AT *be a nonempty subset of* ATOM*. A Kripke structure for* CTL* *is a structure $\langle S, S_0, R, L \rangle$ such that*

1. *L is a mapping from S to the power set of* AT*,*
2. *$\langle S, S_0, R \rangle$ is the same as that for a Kripke structure for* CTLS**.*

The satisfaction relation $\models$ on a Kripke structure for CTL* *is obtained from Definition 4 by deleting the clauses 5–6 and 11–12 and the sequence notation $\hat{d}$.*

Expressions $\models^{\hat{d}}_{\mathrm{CTLS}^*}$ and $\models_{\mathrm{CTL}^*}$ are also used for CTLS* and CTL*, respectively. Note that $\models^{\emptyset}_{\mathrm{CTLS}^*}$ includes $\models_{\mathrm{CTL}^*}$.

A translation from CTLS* into CTL* is introduced below.

Definition 10 *Let* AT *be a nonempty subset of* ATOM *and* $\mathrm{AT}^{\hat{d}}$ *be the set* $\{p^{\hat{d}} \mid p \in \mathrm{AT}\}$ ($\hat{d} \in$ SE*) of atomic formulas where $p^{\emptyset} := p$, i.e.,* $\mathrm{AT}^{\emptyset} :=$ AT*. The language $\mathcal{L}^S$ (the set of formulas) of* CTLS* *is defined using* AT, $\top, \vee, \neg, [b]$, E, path, X *and* U *by the same way as in Definition 1. The language $\mathcal{L}$ of* CTL* *is obtained from $\mathcal{L}^S$ by adding* $\bigcup_{\hat{d} \in \mathrm{SE}} \mathrm{AT}^{\hat{d}}$ *and deleting $[b]$.*

A mapping f from $\mathcal{L}^S$ to $\mathcal{L}$ is defined by:

1. $f([\hat{d}]p) := p^{\hat{d}} \in \mathrm{AT}^{\hat{d}}$ *for any* $p \in \mathrm{AT}$,
2. $f([\hat{d}]\top) := \top$,
3. $f([\hat{d}](\alpha \circ \beta)) := f([\hat{d}]\alpha) \circ f([\hat{d}]\beta)$ *where* $\circ \in \{\vee, \mathrm{U}\}$,
4. $f([\hat{d}]\sharp\alpha) := \sharp f([\hat{d}]\alpha)$ *where* $\sharp \in \{\neg, \mathrm{E}, \mathrm{X}\}$,
5. $f([\hat{d}]\mathrm{path}(\alpha)) := \mathrm{path}(f([\hat{d}]\alpha))$,
6. $f([\hat{d}][b\ ;\ c]\alpha) := f([\hat{d}][b][c]\alpha)$.

Lemma 11 *Let f be the mapping defined in Definition 10. For any Kripke structure $M = \langle S, S_0, R, \{L^{\hat{d}}\}_{\hat{d}\in \mathrm{SE}}\rangle$ for* CTLS**, any sequence-indexed satisfaction relations $\models^{\hat{d}}_{\mathrm{CTLS}^*}$ on M and any state or path s in M, we can construct a Kripke structure $N = \langle S, S_0, R, L\rangle$ for* CTL* *and a satisfaction relation $\models_{\mathrm{CTL}^*}$ on N such that for any state or path formula α in $\mathcal{L}^S$,*

$$M, s \models^{\hat{d}}_{\mathrm{CTLS}^*} \alpha \text{ iff } N, s \models_{\mathrm{CTL}^*} f([\hat{d}]\alpha).$$

Proof. Let AT be a nonempty subset of ATOM and $\mathrm{AT}^{\hat{d}}$ be the set $\{p^{\hat{d}} \mid p \in \mathrm{AT}\}$ of atomic formulas. Suppose that M is a Kripke structure $\langle S, S_0, R, \{L^{\hat{d}}\}_{\hat{d}\in \mathrm{SE}}\rangle$ for CTLS* such that

$L^{\hat{d}}$ ($\hat{d} \in \mathrm{SE}$) are mappings from S to the powerset of AT.

Suppose that N is a Kripke structure $\langle S, S_0, R, L\rangle$ for CTL* such that

L is a mapping from S to the powerset of $\bigcup_{\hat{d}\in \mathrm{SE}} \mathrm{AT}^{\hat{d}}$.

Suppose moreover that for any $s \in S$,

$p \in L^{\hat{d}}(s)$ iff $p^{\hat{d}} \in L(s)$.

The lemma is then proved by induction on the complexity of α.

• Base step:

Case ($\alpha \equiv p \in \mathrm{AT}$): $M, s \models^{\hat{d}}_{\mathrm{CTLS}^*} p$ iff $p \in L^{\hat{d}}(s)$ iff $p^{\hat{d}} \in L(s)$ iff $N, s \models_{\mathrm{CTL}^*} p^{\hat{d}}$ iff $N, s \models_{\mathrm{CTL}^*} f([\hat{d}]p)$ (by the definition of f).

• Induction step: We show some cases.

Case ($\alpha \equiv [b]\beta$): $M, s \models^{\hat{d}}_{\mathrm{CTLS}^*} [b]\beta$ iff $M, s \models^{\hat{d}\ ;\ b}_{\mathrm{CTLS}^*} \beta$ iff $N, s \models_{\mathrm{CTL}^*} f([\hat{d}\ ;\ b]\beta)$ (by induction hypothesis) iff $N, s \models_{\mathrm{CTL}^*} f([\hat{d}][b]\beta)$ by the definition of f.

Case ($\alpha \equiv \alpha_1 \vee \alpha_2$): $M, s \models^{\hat{d}}_{\mathrm{CTLS}^*} \alpha_1 \vee \alpha_2$ iff $M, s \models^{\hat{d}}_{\mathrm{CTLS}^*} \alpha_1$ or $M, s \models^{\hat{d}}_{\mathrm{CTLS}^*} \alpha_2$ iff $N, s \models_{\mathrm{CTL}^*} f([\hat{d}]\alpha_1)$ or $N, s \models_{\mathrm{CTL}^*} f([\hat{d}]\alpha_2)$ (by induction hypothesis) iff $N, s \models_{\mathrm{CTL}^*} f([\hat{d}]\alpha_1) \vee f([\hat{d}]\alpha_2)$ iff $N, s \models_{\mathrm{CTL}^*} f([\hat{d}](\alpha_1 \vee \alpha_2))$ (by the definition of f).

Case ($\alpha \equiv \mathrm{E}\beta$): $M, s \models^{\hat{d}}_{\mathrm{CTLS}^*} \mathrm{E}\beta$ iff $\exists\pi$: path starting from s ($M, \pi \models^{\hat{d}}_{\mathrm{CTLS}^*} \beta$) iff $\exists\pi$: path starting from s ($N, \pi \models_{\mathrm{CTL}^*} f([\hat{d}]\beta)$) (by induction hypothesis) iff $N, s \models_{\mathrm{CTL}^*} \mathrm{E}f([\hat{d}]\beta)$ iff $N, s \models_{\mathrm{CTL}^*} f([\hat{d}]\mathrm{E}\beta)$ (by the definition of f).

Case ($\alpha \equiv \beta_1 \mathrm{U} \beta_2$ and s is a path π): $M, \pi \models^{\hat{d}}_{\mathrm{CTLS}^*} \beta_1 \mathrm{U} \beta_2$ iff $\exists k \geq 0$ $[(M, \pi^k \models^{\hat{d}}_{\mathrm{CTLS}^*} \beta_2)$ and $\forall j$ $(0 \leq j < k$ implies $M, \pi^j \models^{\hat{d}}_{\mathrm{CTLS}^*} \beta_1)]$ iff $\exists k \geq 0$ $[(N, \pi^k \models_{\mathrm{CTL}^*} f([\hat{d}]\beta_2))$ and $\forall j$ $(0 \leq j < k$ implies $N, \pi^j \models_{\mathrm{CTL}^*} f([\hat{d}]\beta_1))]$ (by induction hypothesis) iff $N, \pi \models_{\mathrm{CTL}^*} f([\hat{d}]\beta_1) \mathrm{U} f([\hat{d}]\beta_2)$ iff $N, \pi \models_{\mathrm{CTL}^*} f([\hat{d}](\beta_1 \mathrm{U} \beta_2))$ (by the definition of f). ∎

Lemma 12 *Let f be the mapping defined in Definition 10. For any Kripke structure $N = \langle S, S_0, R, L \rangle$ for* CTL* *and any satisfaction relation $\models_{\mathrm{CTL}^*}$ on N, and any state or path s in N, we can construct a Kripke structure $M = \langle S, S_0, R, \{L^{\hat{d}}\}_{\hat{d} \in \mathrm{SE}} \rangle$ for* CTLS**, sequence-indexed satisfaction relations $\models^{\hat{d}}_{\mathrm{CTLS}^*}$ on M such that for any state or path formula α in $\mathcal{L}^S$,*

$$N, s \models_{\mathrm{CTL}^*} f([\hat{d}]\alpha) \text{ iff } M, s \models^{\hat{d}}_{\mathrm{CTLS}^*} \alpha.$$

Proof. Similar to the proof of Lemma 11. ∎

Theorem 13 (Embedding) *Let f be the mapping defined in Definition 10. For any formula α, α is valid in* CTLS* *iff $f(\alpha)$ is valid in* CTL*.

Proof. By Lemmas 11 and 12. ∎

Theorem 14 (Decidability) *The model checking, validity and satisfiability problems for* CTLS* *are decidable.*

Proof. By the mapping f defined in Definition 10, a formula α of CTLS* can finitely be transformed into the corresponding formula $f(\alpha)$ of CTL*. By Lemmas 11 and 12 and Theorem 13, the model checking, validity and satisfiability problems for CTLS* can be transformed into those of CTL*. Since the model checking, validity and satisfiability problems for CTL* are decidable, the problems for CTLS* are also decidable. ∎

Since the mapping f is a polynomial-time translation, the complexity results for CTLS* are the same as those for CTL*, i.e., the validity, satisfiability and model-checking problems for CTLS* are 2EXPTIME-complete, deterministic 2EXPTIME-complete and PSPACE-complete, respectively.

5 Technical Remarks

In the following paragraph, we remark that the applicability of CTLS* can be extended by adding an operator $-$ (converse), which can satisfy the following axiom schemes:

1. $[--b]\alpha \leftrightarrow [b]\alpha$,
2. $[-(b\ ;\ c)]\alpha \leftrightarrow [-c\ ;\ -b]\alpha$.

The resulting extended logic is called here CTLS$^*_-$. The converse operator $-$ can be used to change the order of sequences in the sequence modal operator; in other words, the priority of information can be changed by $-$. The satisfaction relations of CTLS$^*_-$ are obtained from Definition 4 by adding the following conditions:

1. for any atomic sequence e, $M, s \models^{\hat{d}} [-e]\alpha_1$ iff $M, s \models^{\hat{d}\ ;\ -e} \alpha_1$,
2. $M, s \models^{\hat{d}} [--b]\alpha_1$ iff $M, s \models^{\hat{d}} [b]\alpha_1$,
3. $M, s \models^{\hat{d}\ ;\ --b} \alpha_1$ iff $M, s \models^{\hat{d}\ ;\ b} \alpha_1$,
4. $M, s \models^{\hat{d}} [-(b\ ;\ c)]\alpha_1$ iff $M, s \models^{\hat{d}} [-c\ ;\ -b]\alpha_1$,
5. $M, s \models^{\hat{d}\ ;\ -(b\ ;\ c)} \alpha_1$ iff $M, s \models^{\hat{d}\ ;\ -c\ ;\ -b} \alpha_1$,
6. for any atomic sequence e, $M, \pi \models^{\hat{d}} [-e]\beta_1$ iff $M, \pi \models^{\hat{d}\ ;\ -e} \beta_1$,
7. $M, \pi \models^{\hat{d}} [--b]\beta_1$ iff $M, \pi \models^{\hat{d}} [b]\beta_1$,
8. $M, \pi \models^{\hat{d}\ ;\ --b} \beta_1$ iff $M, \pi \models^{\hat{d}\ ;\ b} \beta_1$,
9. $M, \pi \models^{\hat{d}} [-(b\ ;\ c)]\beta_1$ iff $M, \pi \models^{\hat{d}} [-c\ ;\ -b]\beta_1$,
10. $M, \pi \models^{\hat{d}\ ;\ -(b\ ;\ c)} \beta_1$ iff $M, \pi \models^{\hat{d}\ ;\ -c\ ;\ -b} \beta_1$.

In a similar manner as in the previous sections (with some appropriate modifications), we can obtain the embedding theorem of CTLS$^*_-$ into CTL* and the decidability theorem for the validity, satisfiability, and model-checking problems of CTLS$^*_-$.

We also remark that the proposed framework for the sequence modal operator $[b]$ where b includes $\{\emptyset, ;, -\}$ can also be adapted to CTL [2] and LTL [12]. Namely, we can similarly introduce two extended logics CTLS$_-$ and LTLS$_-$ with the addition of $[b]$ to CTL and LTL, respectively, and we can show the corresponding embedding (into CTL and LTL, respectively) and decidability (w.r.t. validity, satisfiability and model-checking) theorems for CTLS$_-$ and LTLS$_-$. The logics CTLS$_-$ and LTLS$_-$ may be useful for possible applications using *model checking* technologies [3], since CTL and LTL are known to be useful for verifying and specifying concurrent systems by model checking. By the corresponding embedding theorems, CTLS$_-$ and LTLS$_-$ have the same technical situations as CTL and LTL, respectively. For example, CTLS$_-$ has some feasible model checking algorithms, which are deterministic PTIME-complete (see [4] for CTL), but CTLS$_-$ cannot express some important temporal properties such as strong fairness. Similar to LTL, the logic LTLS$_-$ can express almost all the important temporal properties, but LTLS$_-$ has no feasible model checking algorithms, since the model checking problem of LTL is known to be PSPACE-complete [13]. Since CTL and LTL have been rivaled each other [14], the proposed logics CTLS$_-$ and LTLS$_-$ may also be rivaled each other. Although as mentioned in Section 1, the logic CTL* is known to be a result of cooperating and integrating CTL and LTL, the logic CTLS$^*_-$ is regarded as a result of cooperating and integrating CTLS$_-$ and LTLS$_-$. For the purpose of conceptual modeling, the proposed expressive full-computation logics CTLS* and CTLS$^*_-$ are more useful than CTLS$_-$ and LTLS$_-$.

6 Conclusion

This paper presented a new method for conceptual modeling of biological taxonomy for ISA relations, temporal properties, and hierarchical tree structures. For logical reasoning of the conceptual modeling, an extended full computation-tree logic CTLS* with the sequence modal operator $[b]$ was introduced. This logic could be used to appropriately represent hierarchical tree structures, which are useful in formalizing ontologies. The embedding theorem of CTLS* into CTL* was proved. The validity, satisfiability, and model-checking problems of CTLS* were shown to be decidable. The embedding and decidability results allow us to use the existing CTL*-based algorithms to test the satisfiability. Thus it was shown that CTLS* can be used as an executable logic to represent hierarchical tree structures.

Acknowledgment

K. Kaneiwa has been partially supported by the Japanese Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B) 20700147. N. Kamide was supported by the Alexander von Humboldt Foundation and the Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Young Scientists (B) 20700015.

References

1. http://en.wikipedia.org/wiki/Linnaean_taxonomy
2. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic, LNCS 131, 52–71 (1981)
3. Clarke, E.M., Grumberg, O. and Peled, D.A.: Model checking, The MIT Press, 1999.
4. Emerson, E.A. and Clarke, E.M.: Using branching time temporal logic to synthesize synchronization skeletons, Science of Computer Programming 2, pp. 241–266, 1982.
5. Emerson, E.A., Halpern, J.Y.: "Sometimes" and "not never" revisited: on branching versus linear time temporal logic, Journal of the ACM 33 (1), 151–178 (1986)
6. Emerson, E.A, Sistla, P.: Deciding full branching time logic, Information and Control 61, 175–201 (1984)
7. Kamide, N.: Linear and affine logics with temporal, spatial and epistemic operators, Theoretical Computer Science 353 (1-3), pp. 165–207, 2006.
8. Kamide, N. and Kaneiwa, K.: Extended full computation-tree logic with sequence modal operator: Representing hierarchical tree structures, Proceedings of the 22nd Australian Joint Conference on Artificial Intelligence (AI 2009), LNCS 5866, pp. 485–494, 2009.
9. Kaneiwa, K.: Order-sorted logic programming with predicate hierarchy, Artificial Intelligence, 158 (2), pp. 155–188, 2004.
10. Kaneiwa, K. and Mizoguchi, R.: Distributed reasoning with ontologies and rules in order-sorted logic programming, Journal of Web Semantics, 7 (3), pp. 252–270, 2009.

11. Kaneiwa, K. and Satoh, K.: On the complexities of consistency checking for restricted UML class diagrams, Theoretical Computer Science 411(2), pp. 301–323, 2010.
12. Pnueli, A.: The temporal logic of programs. Proceedings of the 18th IEEE Symposium on Foundations of Computer Science, pp. 46–57, (1977)
13. Sistla, A.P. and Clarke, E.M.: The complexity of propositional linear temporal logic, Journal of the ACM 32 (3), pp. 733–749, 1985.
14. Vardi, M.Y.: Branching vs. linear time: final showdown, Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 200), LNCS 2031, pp. 1–22, 2001.
15. Wansing, H.: The logic of information structures. LNAI 681, 163 pages (1993)