Temporal reasoning, Branching-time formalism, Linear-time formalism, Computation tree logic, Linear-time temporal logic, Model checking.

Abstract: A new temporal logic called linear-time computation tree logic (LCTL) is obtained from computation tree logic (CTL) by adding some modified versions of the temporal operators of linear-time temporal logic (LTL). A theorem for embedding LCTL into CTL is proved. The model-checking, validity and satisfiability problems of LCTL are shown to be deterministic PTIME-complete, EXPTIME-complete and deterministic EXPTIME-complete, respectively.

1 INTRODUCTION

It is known that computation tree logic (CTL) (Clarke and Emerson, 1981) and linear-time temporal logic (LTL) (Pnueli, 1977) are the most useful temporal logics for verifying concurrent systems by model checking (Clarke et al., 1999). CTL has some feasible model checking algorithms, which are deterministic PTIME-complete (Emerson and Clarke, 1982), but CTL cannot express some important temporal properties such as strong fairness. LTL can express almost all important temporal properties, but LTL has no feasible model-checking algorithms. The model-checking problem of LTL is indeed PSPACE-complete (Sistla and Clarke, 1985). Although CTL and LTL have been rivaled each other (Vardi, 2001), cooperating CTL and LTL is considered to be a good choice to obtain a more useful model checking tool. Full computation-tree logic (CTL∗) (Emerson and Sistla, 1984; Emerson and Halpern, 1986) is known to be a result of cooperating CTL and LTL. However, the model-checking problem of CTL∗ is PSPACE-complete. This paper tries to obtain a cooperative and feasible approach to the traditional issue of “branching-time versus linear-time”. The proposed logic in this paper includes CTL and subsumes some versions of the linear-time temporal operators of LTL (i.e., cooperative). The proposed logic also has the same complexity result as CTL model-checking (i.e., feasible).

The results of this paper are then summarized as follows. A new computation tree logic called linear-time computation tree logic (LCTL) is obtained from CTL by adding some bounded versions of the linear-time temporal operators of LTL. A theorem for embedding LCTL into CTL is proved. The model-checking, validity and satisfiability problems of LCTL are shown to be deterministic PTIME-complete, EXPTIME-complete and deterministic EXPTIME-complete, respectively. The embedding and decidability results indicate that we can reuse the existing CTL-based algorithms for model-checking, validity and satisfiability. This fact is regarded as an advantage of LCTL. The proposed bounded linear-time temporal operators, which are regarded as finite approximations of the usual linear-time temporal operators, have the central role for obtaining the complexity results. Although the standard LTL operators have an infinite (unbounded) time domain, i.e., the setω of natural numbers, the proposed bounded operators have a bounded time domain which is restricted by a fixed positive integer l, i.e., the setωl := {x ∈ ω | x ≤ l}. Despite this restriction, the proposed bounded operators can derive almost all the typical LTL axioms including the time induction axiom.

1By “feasible”, we mean “computable in practice”. There is a widespread opinion that PTIME computability is the correct mathematical model of feasible computation.
2 LINEAR-TIME COMPUTATION TREE LOGIC

Formulas of LCTL are constructed from countably many atomic formulas, → (implication) ∧ (conjunction), ∨ (disjunction), ¬ (negation), X (next), G (globally), F (eventually), U (until), X_L (linear next), G_L (linear globally), F_L (linear eventually), A (all computation paths) and E (some computation path) where X_L, G_L and F_L are based on a bounded time domain. The symbols X, G, F, U, X_L, G_L and F_L are called temporal operators, and the symbols A and E are called path quantifiers. The symbol ATOM is used to denote the set of atomic formulas. An expression A ≡ B is used to denote the syntactical identity between A and B.

Definition 2.1 Formulas α are defined by the following grammar, assuming p ∈ ATOM:

\[ α ::= p \mid α\neg α \mid α ∧ α \mid α ∨ α \mid ¬ α \mid X α \mid G α \mid F α \mid A X α \mid E X α \mid AG α \mid EG α \mid AF α \mid EF α \mid A(α U α) \mid E(α U α) \]

Note that pairs of symbols like AG and EU are indivisible, and that the symbols X, G, F and U cannot occur without being preceded by an A or an E. Similarly, every A or E must have one of X, G, F and U to accompany it. Some operators are redundant as those in CTL, because some operators can be obtained by the other operators (e.g., AG α := ¬EF¬α).

The symbol ω is used to represent the set of natural numbers. Lower-case letters i, j, k, m and n are sometimes used to denote any natural numbers. An expression X^n_α for any m ∈ ω is defined inductively by X^0_α := α and X^{n+1}_α := X_X^n_α. The symbols ≤ and ≥ are used to represent a linear order on ω. The symbol ω₀ is used to represent the set \{i ∈ ω | i ≤ l\}. In the following discussion, the number l is fixed as a certain positive integer.

Definition 2.2 A structure \( ⟨S, S_0, R, \{L^m\}_{m∈ω}⟩ \) is called a time-indexed Kripke structure if:

1. S is the set of states,
2. S₀ is a set of initial states and S₀ ⊆ S,
3. R is a binary relation on S which satisfies the condition: \( ∀S ∈ S \exists S' ∈ S | (S, S') ∈ R \),
4. \( L^m (m ∈ ω) \) are functions from S to the power set of a nonempty subset AT of ATOM.

A path in a time-indexed Kripke structure is an infinite sequence of states, \( π = s_0, s_1, s_2, \ldots \) such that \( ∀i \geq 0 | (s_i, s_{i+1}) ∈ R \).

The logic LCTL is then defined as a time-indexed Kripke structure with satisfaction relations \( |=^m (m ∈ ω) \).

Definition 2.3 Let AT be a nonempty subset of ATOM. Satisfaction relations \( |=^m (m ∈ ω) \) on a time-indexed Kripke structure \( M = ⟨S, S₀, R, \{L^m\}_{m∈ω}⟩ \) are defined inductively as follows (s represents a state in S):

1. for any \( p ∈ AT, M, s |=^m p \) iff \( p ∈ L^m (s) \),
2. \( M, s |=^m α_1 → α_2 \) iff \( M, s |=^m α_1 \) implies \( M, s |=^m α_2 \),
3. \( M, s |=^m α_1 ∧ α_2 \) iff \( M, s |=^m α_1 \) and \( M, s |=^m α_2 \),
4. \( M, s |=^m α_1 ∨ α_2 \) iff \( M, s |=^m α_1 \) or \( M, s |=^m α_2 \),
5. \( M, s |=^m ¬α_1 \) iff \( not[M, s |=^m α_1] \),
6. for any \( m ≤ l − 1, M, s |=^m X_L α \) iff \( M, s +^l |=^m α \),
7. for any \( m ≥ l, M, s |=^m X_L α \) iff \( M, s +^l |=^m α \),
8. for any \( n ∈ ω, M, s |=^n α \) iff \( M, s +^l |=^m α \) for all \( n ∈ ω \),
9. \( M, s |=^m G_L α \) iff \( M, s |=^m α \) for all \( n ∈ ω \),
10. \( M, s |=^m F_L α \) iff \( M, s |=^m α \) for some \( n ∈ ω \),
11. \( M, s |=^m A X α \) iff \( ∀s₁ ∈ S | (s, s₁) ∈ R \) implies \( M, s₁ |=^m α \),
12. \( M, s |=^m E X α \) iff \( ∃s₁ ∈ S | (s, s₁) ∈ R \) and \( M, s₁ |=^m α \),
13. \( M, s |=^m AG α \) iff for all paths \( π = s_0, s_1, s_2, \ldots \) where \( s ∈ S₀ \) and all states \( s_i \) along \( π \), we have \( M, s_i |=^m α \),
14. \( M, s |=^m EG α \) iff there is a path \( π = s_0, s_1, s_2, \ldots \) where \( s ∈ S₀ \) and for all states \( s_i \) along \( π \), we have \( M, s_i |=^m α \),
15. \( M, s |=^m AF α \) iff for all paths \( π = s_0, s_1, s_2, \ldots \) where \( s ∈ S₀ \), there is a state \( s_i \) along \( π \) such that \( M, s_i |=^m α \),
16. \( M, s |=^m EF α \) iff there is a path \( π = s_0, s_1, s_2, \ldots \) where \( s ∈ S₀ \) and for some state \( s_i \) along \( π \), we have \( M, s_i |=^m α \),
17. \( M, s |=^m A(α₁ U α₂) \) iff for all paths \( π = s_0, s_1, s₂, \ldots \) where \( s ∈ S₀ \), there is a state \( s_j \) along \( π \) such that \([M, s_j |=^m α₂] \) and \( ∀j (0 ≤ j < k \) implies \( M, s_j |=^m α₁)] \),
18. \( M, s |=^m E(α₁ U α₂) \) iff there is a path \( π = s_0, s_1, s₂, \ldots \) where \( s ∈ S₀ \) and for some state \( s_k \) along \( π \), we have \([M, s_k |=^m α₂] \) and \( ∀j (0 ≤ j < k \) implies \( M, s_j |=^m α₁)] \).

We can naturally consider the unbounded version LCTL ω which is obtained from LCTL by deleting the conditions 7 and 8 and replacing the conditions 6, 9 and 10 by the standard conditions:

6'. \( M, s |=^m X L α \) iff \( M, s |=^m +^l α \),
9'. \( M, s |=^m G L α \) iff \( M, s |=^m +^l α \) for all \( n ∈ ω \),
10'. \( M, s |=^m F L α \) iff \( M, s |=^m +^n α \) for some \( n ∈ ω \).

523
However, the decidability of validity, satisfiability, and model-checking problems for LCTL cannot be shown using the proposed embedding-based method. The logic LCTLω is embeddable into the infinitary version CTLω which is obtained from CTL by adding the infinitary conjunction and disjunction connectives ∨ and ∧. But, logics with ∨ and ∧ are known to be undecidable, and hence such an embedding result cannot imply the decidability.

**Definition 2.4** A formula α is valid (satisfiable) in LCTL if and only if M,s ⊨ α holds for any (some) time-indexed Kripke structure M = ⟨S, S₀, R, {L^n}(m ∈ ω)⟩, any (some) s ∈ S, and any (some) satisfaction relations |=^m (m ∈ ω) on M.

**Definition 2.5** Let M be a time-indexed Kripke structure S, S₀, R, {L^n}(m ∈ ω) for LCTL, and |=^m (m ∈ ω) be satisfaction relations on M. Then, the model checking problem of LCTL is defined by: for any formula α, find the set {s ∈ S | M,s ⊨ α}.

Let C be a finite set of formulas. Then, expressions ∧ and ∨ represent the conjunction and disjunction of all elements of C, respectively. An expression α → β is used to represent (α → β ∧ β → α).

**Proposition 2.6** The following formulas are valid in LCTLω for any formulas α and β.

1. X_L(α ∨ β) ↔ X_Lα ∨ X_Lβ where ∈ {→, ∧, ∨}.
2. X_L(¬α) ↔ ¬(X_Lα).
3. G_Lα → α.
4. G_Lα → X_Lα.
5. G_Lα → X_LG_Lα.
7. α ∧ G_L(α → X_Lα) → G_Lα (time induction).
8. for any n ∈ ω, X_L[n]α ↔ X_Lα.
9. G_Lα ↔ {X_Lα | n ∈ ω}.
10. F_Lα ↔ ∨ {X_Lα | n ∈ ω}.

A satisfaction relation |= on a Kripke structure M = ⟨S, S₀, R, L⟩ for CTL is defined by the same conditions 1–5 and 9–16 as in Definition 2.3 by deleting the superscript “m”. The validity, satisfiability and model-checking problems for CTL are defined similarly as those for LCTL.

Remark that |=^ω of LCTL includes |= of CTL, and hence LCTL is an extension of CTL.

### 3 Embedding and Complexity

**Definition 3.1** Let AT be a non-empty subset of ATOM, and AT^m (m ∈ ω) be the sets {p^m | p ∈ AT^m} of atomic formulas where p^0 := p (i.e., AT^0 := AT). The language L^L (the set of formulas) of LCTL is defined using AT, X_L, G_L, F_L, →, ∧, ∨, X, F, G, U, A and E. The language L of CTL is obtained from L^L by adding ∪ m∈ω AT^m and deleting {X_L, G_L, F_L}.

A mapping f from L^L to L is defined inductively by:

1. for any p ∈ AT, f(X_L^n p) := p^m ∈ AT^m, esp., f(p := p).
2. f(X_L^n(α ∨ β)) := f(X_L^nα) ∨ f(X_L^nβ) where ∈ {∧, ∨, →}.
3. f(X_L^nα) := f(X_L^nα) where ∈ {¬A, X, E, A, ¬X, E, ¬A}.
4. f(X_L^n(α U β)) := f(X_L^nα) U f(X_L^nβ) where ∈ {A, E}.
5. f(X_L^nG_Lα) := ∨ {f(X_L^nα) | n ∈ ω}.
6. f(X_L^nF_Lα) := ∨ {f(X_L^nα) | n ∈ ω}.

**Lemma 3.2** Let f be the mapping defined in Definition 3.1. For any time-indexed Kripke structure M := ⟨S, S₀, R, {L^n}(m ∈ ω)⟩ for LCTL, and any satisfaction relations |=^m (m ∈ ω) on M, we can construct a Kripke structure N := ⟨S, S₀, R, L⟩ for CTL and a satisfaction relation |= on N such that for any formula α in L^L and any state s in S,

M,s |=^m α iff N,s |= f(X_L^nα).

**Proof.** Let AT be a nonempty subset of ATOM, and AT^m be the sets {p^m | p ∈ AT} of atomic formulas. Suppose that M is a time-indexed Kripke structure ⟨S, S₀, R, {L^n}(m ∈ ω)⟩ such that

L^n (m ∈ ω) are functions from S to the power set of AT.

Suppose that N is a Kripke structure ⟨S, S₀, R, L⟩ such that
Suppose moreover that for any \( s \in S \) and any \( p \in \text{AT} \), \( p \in L^m(s) \) iff \( p^m \in L(s) \).

The lemma is then proved by induction on the complexity of \( \alpha \).

- **Base step:**
  - Case \( \alpha \equiv \rho \in \text{AT} \): We obtain: \( M, s \models \rho \) iff \( \rho \in L(s) \) iff \( N, s \models \rho^m \) iff \( N, s \models f(X_{\rho}^m) \) (by the definition of \( f \)).

- **Induction step:**
  - Case \( \alpha \equiv \beta \land \gamma \): We obtain: \( M, s \models \beta \land \gamma \) iff \( M, s \models \beta \) and \( M, s \models \gamma \) iff \( N, s \models f(X_{\beta}^m) \) and \( N, s \models f(X_{\gamma}^m) \) (by induction hypothesis) iff \( N, s \models f(X_{\beta \land \gamma}^m) \) (by the definition of \( f \)).

- Case \( \alpha \equiv \beta \lor \gamma \): Similar to Case \( \alpha \equiv \beta \land \gamma \).

- Case \( \alpha \equiv \neg \beta \): We obtain: \( M, s \models \neg \beta \) iff \( f(M, s \models \beta) \) iff \( f(N, s \models f(X_{\beta}^m) \) (by induction hypothesis) iff \( N, s \models \neg f(X_{\beta}^m) \) iff \( N, s \models f(X_{\neg \beta}^m) \) (by the definition of \( f \)).

- Case \( \alpha \equiv X_{\beta} \): Subcase \( m \leq l - 1 \): We obtain: \( M, s \models X_{\beta} \) iff \( M, s \models X_{\beta}^{m+1} \) iff \( N, s \models f(X_{\beta}^{m+1}) \) (by induction hypothesis).

  Subcase \( m \geq l \): We obtain: \( M, s \models X_{\beta} \) iff \( M, s \models X_{\beta}^{m+1} \) iff \( N, s \models f(X_{\beta}^{m+1}) \) (by induction hypothesis).

- Case \( \alpha \equiv G_{\beta} \): We obtain: \( M, s \models G_{\beta} \) iff \( M, s \models G_{\beta}^{m} \) for any \( n \in \omega \) iff \( N, s \models f(X_{\beta}^{m}) \) for any \( n \in \omega \) (by induction hypothesis) iff \( N, s \models \bigwedge_{n \in \omega} f(X_{\beta}^{m+n}) \) (by the definition of \( f \)).

- Case \( \alpha \equiv F_{\beta} \): Similar to Case \( \alpha \equiv G_{\beta} \).

- Case \( \alpha \equiv AX_{\beta} \): We obtain: \( M, s \models AX_{\beta} \) iff \( \forall \beta, (s, 1) \in R \) implies \( M, s_{1} \models \beta \) iff \( \forall s_{1} \in S \) \( (s, s_{1}) \in R \) implies \( N, s_{1} \models f(X_{\beta}^{m}) \) (by induction hypothesis) iff \( N, s \models AX_{\beta} \) iff \( N, s \models f(X_{\beta}^{m} AX_{\beta}) \) (by the definition of \( f \)).

- Case \( \alpha \equiv EX_{\beta} \): Similar to Case \( \alpha \equiv AX_{\beta} \).

- Case \( \alpha \equiv AG_{\beta} \): We obtain: \( M, s \models AG_{\beta} \) iff for all states \( s \equiv s_{0} \), and all states \( s_{i} \) along \( \pi \), we have \( M, s_{i} \models \beta \).

- Case \( \alpha \equiv AFG_{\beta} \): We obtain: \( M, s \models AFG_{\beta} \) iff for all states \( s \equiv s_{0} \), and all states \( s_{i} \) along \( \pi \), we have \( N, s_{i} \models f(X_{\beta}^{m}) \) (by induction hypothesis) iff \( N, s \models A f(X_{\beta}^{m}) \) (by the definition of \( f \)).
the model-checking, validity and satisfiability problems for LCTL are deterministic PTIME-complete, EXPTIME-complete and deterministic EXPTIME-complete, respectively.

4 CONCLUDING REMARKS

In this paper, a new logic, linear-time computation tree logic (LCTL), was introduced by “cooperating” CTL and LTL, and the deterministic PTIME-completeness (i.e., the existence of “feasible” algorithms) of the LCTL model-checking problem was shown. It was thus shown that there is a cooperative and feasible approach to the traditional issue of “branching-time versus linear-time”.

In the following, we give some remarks on the idea of bounding time and on the concept of combining logics.

To restrict the time domain of the LTL operators is not a new idea. Such an idea was discussed in (Biere et al., 2003; Cerrito et al., 1999; Cerrito and Mayer, 1998; Hodkinson et al., 2000). For example, by using and introducing a bounded time domain and the notion of bounded validity in a semantics, bounded tableaux calculi (with temporal constraints) for propositional and first-order LTLs were introduced by Cerrito, Mayer and Prand (Cerrito et al., 1999; Cerrito and Mayer, 1998). It is also known that to restrict the time domain is a technique to obtain a decidable or efficient fragment of first-order LTL (Hodkinson et al., 2000). Restricting the time domain implies not only some purely theoretical merits discussed above, but also some practical merits for describing temporal databases and planning specifications (Cerrito et al., 1999; Cerrito and Mayer, 1998), and for implementing an efficient model checking algorithm called bounded model checking (Biere et al., 2003). Such practical merits are due to the fact that there are problems in computer science and artificial intelligence where only a finite fragment of the time sequence is of interest (Cerrito et al., 1999).

As mentioned in (Sernadas and Sernadas, 2003), there are some general theories for various combined modal logics (Sernadas and Sernadas, 2003), including the theories of fusion, product and fibring. Various combined modal logics have been studied based on these theories. The proposed logic LCTL may be categorized by a fusion of CTL and a bounded-time version of LTL.

ACKNOWLEDGEMENTS

This research was supported by the Alexander von Humboldt Foundation and by the Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Young Scientists (B) 20700015.

REFERENCES


