Branching Time Regular Temporal Logic for Model Checking with Linear Time Complexity

Kiyoharu Hamaguchi, Hiromi Hiraishi and Shuzo Yajima
Department of Information Science, Faculty of Engineering
Kyoto University, Kyoto, 606, Japan.

Abstract Firstly in this paper, we propose a branching time logic BRTL (Branching time Regular Temporal Logic) which has automata connectives as temporal operators. BRTL is more expressive than CTL proposed by Clarke et.al. and it is modest in terms of model checking, i.e. it has a model checking algorithm which runs in time proportional both to the size of a given Kripke structure and to the length of a given formula, as shown in the paper.

Secondly, in order to improve the succinctness of the temporal logic formulas, we introduce the mechanism of *substitutions to Boolean variables* and *references to the Boolean variables*. We propose EBRTL(Extended BRTL), i.e. BRTL with the substitution mechanism, and show examples of descriptions of some temporal properties. We develop its model checking algorithm whose time complexity is linear both in the size of a given Kripke structure and in the length of a given formula and exponential in the number of the Boolean variables used in the formula.

1 Introduction

In order to verify correctness of finite state systems, researchers developed various kinds of propositional temporal logics, applied model checking methods on Kripke structures and succeeded in verifying finite state systems of medium size[1, 2, 3].

CTL proposed in [1] has a model checking algorithm which runs in time $O(\text{Size}(S)\times \text{Len}(f))$ ($\text{Size}(S)$ and $\text{Len}(f)$ are the size of the given Kripke structure and the length of the given CTL formula respectively.) The expressive power of CTL is, however, restricted, because only $\mathcal{U}$(until) and X(next) with path quantifiers are allowed. ECTL[4] is an extension of CTL, which have *automata connectives*[5] as temporal operators and can express an arbitrary ω regular set. The model checking algorithm shown in [4] is exponential in the size of the number of the states of automata connectives, because Muller automata are used as their bases.

One of the two concerns of this paper is to develop a temporal logic system which is more tractable, in terms of model checking, than ECTL and more expressive than CTL. We propose a branching time logic BRTL (Branching Time Regular Temporal Logic) and show a model checking algorithm which runs in time $O(\text{Size}(S)\times \text{Len}(f))$, i.e. linear in the number of the states of automata connectives used in f.

The automata connectives of BRTL are based on restricted deterministic ω finite automata of Büchi type. Although this means that BRTL cannot express all ω regular sets, it can express the properties of repetition of some events as shown by examples.

The other concern is to improve the succinctness of the temporal logic formulas by introducing the mechanism of *substitutions to Boolean variables* and *references to the Boolean variables.* We propose EBRTL(Extended BRTL), i.e. BRTL with the above mechanism, and show examples of descriptions of some temporal properties. We develop its model checking algorithm whose time complexity is $O(\mathrm{Size}(S) \times \mathrm{Len}(f) \times 2^{|V|})$ ($|V|$ is the number of the different Boolean variables used in f).

In Section 2, BRTL is defined and its model checking algorithm is shown. In Section 3, BRTL is extended by introducing the mechanism stated above.

2 Branching Time Regular Temporal Logic

The truth values of propositional logic are represented by T(true) and F(false). X^{ω} and $X^{\dagger}$ represent sets of infinite and finite sequences of elements of a set X respectively. $|X|$ is the number of the elements of X. V_T and V_F are *tautology* and *invalid* formulas respectively.

2.1 Syntax and Semantics

Def. 2.1 *A logic-type deterministic ω finite automaton (ldo-fa)* $A = (Q, \Sigma, P, Br, \delta, q_0, F)$ is defined as follows.

Q and $\Sigma = \{T, F\}^n$ are sets of finite number of states and input symbols respectively. $P = \{p_1, \cdots, p_n\}$ is a set of atomic propositions. F is a set of accepting states and the elements of $Q - F$ are called rejecting states. q_0 is the initial state.

Let BF be a set of all propositional formulas constructed from the elements of P. $Br : Q \times Q \to BF$ is a partial function which satisfies the following two conditions.

We define $BF_q \stackrel{\mathrm{def}}{=} \{f | \exists q'.Br(q, q') = f\}$, for each $q \in Q$.

For any $q \in Q$, (1) $f_1 \wedge f_2 \equiv V_F$ for any$f_1, f_2 \in BF_q$ and (2) $\bigvee_{f \in BF_q} f \equiv V_T$.

We define Edges(A) $\stackrel{\mathrm{def}}{=} \{(q, q') | \exists f.Br(q, q') = f\}$.

$\delta : Q \times \Sigma \to Q$ is a state transition function such that, for $q, q' \in Q$ and $v \in \Sigma$, $\delta(q, v) = q' \Leftrightarrow f = Br(q, q')$ is defined and $f(v) = T$.

For $x \in \Sigma^{\omega}$, we define $Inf(x)$ as the set of the states through which A goes infinitely often. Then the set of words accepted by A is $\{x | Inf(x) \cap F$ is not empty $\}$ and is described by $|A|$. □

Although complementation of an ω finite automaton is not easy in general, even if it is deterministic, an automaton that satisfies the following Cond. 1 can be complemented easily, (i.e. by exchanging the accepting states and the rejecting states.)

Cond. 1 Let A be a ldo-fa $(Q, \Sigma, P, Br, \delta, q_0, F)$. There is no input sequence which transits A from a rejecting state $q_r \in (Q - F)$ to the state q_r via some accepting state $q_a \in F$. □

A logic-type deterministic ω finite automaton A which satisfies Cond. 1 is called *ldo-fa-1.*

It is easy to prove the following two lemmas.

Lem. 2.2 *For an ldo-fa-1 $A = (Q, \Sigma, P, Br, \delta, q_0, F)$, an ldo-fa-1 $\overline{A}$ which accepts $\Sigma^\omega - |A|$ is obtained by exchanging accepting states and rejecting states of A.*

Lem. 2.3 *For ldo-fa-1's $A_i = (Q_i, \Sigma, P, Br_i, \delta_i, q_{i0}, F_i)$ $(i = 1, 2)$, an ldo-fa-1 $A_1|A_2$ which accepts $|A_1| \cup |A_2|$ is obtained as follows:*

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) | q_1 \in Q_1 \ , \ q_2 \in Q_2\}$
- *$Br : Q \times Q \to BF$ is defined by $Br((q_1, q_2), (q_1', q_2')) = Br_1(q_1, q_1') \wedge Br_1(q_2, q_2')$, where $q_i, q_i' \in Q_i$ $(i = 1, 2)$ and both of $Br_1(q_1, q_1')$ and $Br_1(q_2, q_2')$ are defined.*
- $q_0 = (q_{10}, q_{20})$.
- $F = \{(q_1, q_2) | q_1 \in F_1 \text{ or } q_2 \in F_2\}$.

□

The lemmas mean that the languages defined by ldo-fa-1's are closed under Boolean operations over sets.

Let AP be a set of atomic propositions. $S = \langle \Sigma, I, R, \Sigma_0 \rangle$ is called a Kripke structure, where Σ is a set of states, $I : \Sigma \to 2^{AP}$ is an assignment function, $R \subseteq \Sigma \times \Sigma$ is a *total* relation, $\Sigma_0 \subseteq \Sigma$ is a set of initial states. The size of a given Kripke structure, denoted by Size(S), is defined as $|\Sigma| + |R|$.

Def. 2.4 Syntax and Semantics

The syntax of BRTL(Branching Time Regular Temporal Logic) is as follows:

BRTL formulas: The set of all BRTL formulas are described by BF.

- p $\in AP$ is an BRTL formula.
- If f and g are BRTL formulas, then $\neg f$ and $f \vee g$ are BRTL formulas.
- If A is an automaton connective , then $\exists A$ is a BRTL formula.

Automata connectives

If A' is an ldo-fa-1, then $A'[AP/B]$ and $\neg A'[AP/B]$ are automata connectives, where $A'[AP/B]$ is obtained by replacing each atomic proposition $p_i \in AP$ $(1 \leq i \leq n)$ with $f_i \in B \subseteq BF$ corresponding to p_i simultaneously. The numbers of the elements in AP and B have to be equal.

The semantics of BRTL is defined on a Kripke structure $S = \langle \Sigma, I, R, \Sigma_0 \rangle$. $S, s \models f$ means that the BRTL formula f holds at the state s on S. In the following, $p \in AP$, f and g are BRTL formulas and A is an automaton connective.

- $S, s \models p$ iff $I(s) \ni p$
- $S, s \models f \vee g$ iff $S, s \models f$ or $S, s \models g$
- $S, s \models \neg f$ iff $S, s \not\models f$
- $S, s \models \exists A$ iff there exists an infinite sequence $\sigma = s_0 s_1 s_2 \cdots$ starting from s on S and a run (a sequence of states) $q_0 q_1 q_2 \cdots$ in Q such that, $S, s_i \models BR(q_i, q_{i+1})$ holds and all the states which appear infinitely in the run are in F.
- $S, s \models \exists \neg A$ iff $S, s \models \exists \overline{A}$

If $\forall s \in \Sigma_0 . S, s \models f$, then we describe $S \models f$. □

The Boolean operators $\wedge$, $\equiv$ and $\Rightarrow$ are also used. Besides we define $\forall A \overset{\text{def}}{=} \neg\exists\neg A$ and $\forall\neg A \overset{\text{def}}{=} \neg\exists A$.

For BRTL formulas $f = \exists A$ or $\exists\neg A$, where $A = A'[AP/B]$ for some ldo-fa-1 A' and BRTL formulas B, the length of f, denoted as Len(f) is defined as follows:

$\text{Len}(A) \overset{\text{def}}{=} |Q| + |\text{Edges}(A')| + \sum_{f \in Br'(Q\times Q)} \text{Len}(f) + \sum_{g\in B} \text{Len}(g)$

$Br'(Q \times Q)$ represents the set of propositional formulas labeled to transitions of A'. Intuitively the length means the length of the description of the ldo-fa-1 A' and the set of B.

2.2 Comparison with CTL

Def. 2.5 The truth value of a given CTL formula is defined on each state on a given Kripke structure.

While BRTL uses path quantifiers $\exists$ and automata connectives, $\forall Xf$, $\exists Xf$, $\forall[f_1 \mathcal{U} f_2]$ and $\exists[f_1 \mathcal{U} f_2]$ are allowed to be used[1].

The semantics of these formulas are as follows:

- $S, s \models \forall Xf$ ($\exists Xf$) iff $S, s_1 \models f$ for all (some) s_1 such that $(s, s_1) \in R$
- $S, s \models \forall[f_1 \mathcal{U} f_2]$ ($\exists[f_1 \mathcal{U} f_2]$) iff $S, s_i \models f_1$ ($i = 0, 1, 2, \cdots, n$) and $S, s_{n+1} \models f_2$ for all (some) infinite sequences $s = s_0 s_1 s_2 \cdots$ starting from s and an integer $n \geq 0$.

$\square$

Since the automata connectives are not easy to write, we introduce informally a description language like programming languages. For example, the automata connectives of Fig. 1 and Fig. 2 are described by the descriptions (1) and (2) in the following.

- *state:* [a] and [r] represent an accepting state and a rejecting state of an automaton connective respectively. [ac] and [re] are special states. [ac] ([re]) represents an accepting (rejecting) state from which the automaton never transits to the other states for any input.
- *branch:* **if** S_1 **else if** $S_2 \cdots$ **else** S_n **endif** represents a branch from a state. S_i has a higher *precedence* than S_{i+1}.
- *loop:* **loop** S **endloop** represents a loop from a state to the loop.
- *label and goto:* 'L:' is a label to be put on a state and **goto** L means the transition to the state.

If no transition is possible from a state, it is assumed to transit to a rejecting state. [x] [x] is interpreted as [x] V_T [x] (x = a, r, ac or re). The first state is assumed to be the initial state of the automaton.

With the above notations, we can define the temporal operators used in CTL.

Let $Next(f)$ and $Until(f_1, f_2)$ be automata connectives such that

$Next(f) \overset{\text{def}}{=}$ ([r][r]f[ac]) (1)

$Until(f, g) \overset{\text{def}}{=}$ ([r]**if loop** $f_1 \wedge \neg f_2$ **endloop**
else if $\neg f \wedge \neg f_2$ [re] **else if** f_2 [ac] **endif**) (2)

Then,

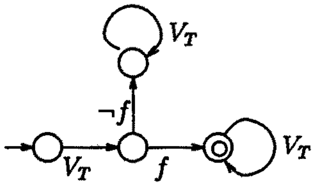

Figure 1: $Next(f)$

$\forall[f_1 \mathcal{U} f_2] \overset{\text{def}}{=} \forall Until(f_1, f_2)$ $\exists[f_1 \mathcal{U} f_2] \overset{\text{def}}{=} \exists Until(f_1, f_2)$
$\forall X f \overset{\text{def}}{=} \forall Next(f)$ $\exists X f \overset{\text{def}}{=} \exists Next(f)$

Temporal properties containing the concept of repetition can also be expressed. For example, an BRTL formula $f = \forall A_R$, where A_R is shown in Fig. 3, means "for a given $m \geq 2$, p is true on every state s_i $(i = km, k = 0, 1, \cdots)$".

It can be shown that this property cannot be expressed by any CTL formula by extending the proof by Wolper in [5] to Kripke structures.

For a Kripke structure $S = \langle \Sigma, I, R, \Sigma_0 \rangle$ and a state $s_0 \in \Sigma$,

$$Re_i(S, s_0) \overset{\text{def}}{=} \{s_i | (s_j, s_{j+1}) \in R, j = 0, 1, 2, \cdots i-1\}$$

Lem. 2.6 *Let p be an atomic proposition. C_i is defined as a set of pairs of a Kripke structure $S = \langle \Sigma, I, R, \Sigma_0 \rangle$ and a state s, i.e. (S, s), satisfying the following conditions.*

1. $\forall s' \in Re_j(s).S, s' \models p \quad (j = 0, 1, \cdots, i)$,
2. $\forall s'' \in Re_{i+1}(s).S, s'' \models \neg p$,
3. $\forall s''' \in Re_j(s).S, s''' \models p \quad (j = i+1, i+2, \cdots)\}$

If CTL formula f has at most n 'X's , then, for any $(S_i, s_i) \in C_i$, $(S_{i'}, s_{i'}) \in C_{i'}$ $(i, i' > n)$,

$S_i, s_i \models f \Leftrightarrow S_{i'}, s_{i'} \models f$ (*)

In other words, for any $i > n$, the truth value of f is same for $(S_i, s_i) \in C_i$. □

Theo. 2.7 *Given an integer $m \geq 2$, $D \overset{\text{def}}{=} \{(S, s) | \forall s' \in Re_{km}.\ S, s' \models p,\ (k = 0, 1, \cdots)\}$, where S is a Kripke structure and s is a state on S.*

There exists no CTL formula f which satisfies $(S, s) \in D \Leftrightarrow S, s \models f$

(Proof) Let us assume that a CTL formula f satisfies the above condition and f has l 'X's.

Let $km - 1 \geq l$. From Lemma 2.6, $S, s \models f \Leftrightarrow S', s' \models f$ holds, for any $(S, s) \in C_{km}$ and any $(S', s') \in C_{km-1}$. Besides $(S, s) \in D$ and $(S', s') \notin D$. Thus there exists no CTL formula that can express the property. □

ECTL[4] has as automata connectives deterministic ω finite automata of Muller type, the class of which is equivalent to ω regular sets, as automata connectives. This means that there are some properties which cannot be expressed in BRTL, but can be expressed in ECTL.

From the above arguments, the relation among CTL, BRTL and ECTL in terms of expressive power is as follows:

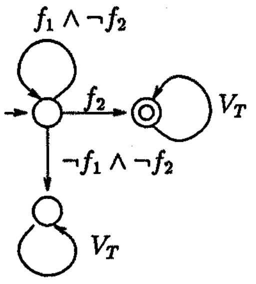

Figure 2: $Until(f_1, f_2)$

$$CTL < BRTL < ECTL$$

2.3 A Model Checking Algorithm

A model checking algorithm is shown in this section. The algorithm is constructed, based on the next lemma.

Lem. 2.8 *The necessary and sufficient condition for $S, s \models \exists A$ is that there exists a finite sequence $(s_0, q_0)(s_1, q_1) \cdots (s_k, q_k)(s_{k+1}, q_{k+1}) \cdots (s_n, q_n)$ satisfying the following conditions:*

1. $(s_i, s_{i+1}) \in R$ $(i = 0, 1, \cdots n-1)$

2. $S, s_i \models Br(q_i, q_{i+1})$ $(i = 0, 1, \cdots n-1)$

3. $s_k = s_n$ *and* $q_k = q_n$

4. $q_j \in F$ $(j = k, k+1, \cdots n)$

Algorithm 2.9 • Input: A Kripke structure S and an BRTL formula f

- Output: $S \models f$?
- Method:

1. The automata connectives of the form $\neg A$ occurred in f are transformed to $\overline{A}$ by exchanging the accepting states and rejecting states.
2. By the following (a)-(c), every subformula g of f is labeled to every state $s \in \Sigma$ if and only if $S, s \models g_i$, in a bottom up manner.

   The output is 'yes' if and only if f is labeled to every $s_0 \in \Sigma_0$.

   (a) When g is an atomic proposition, its truth value on each $s \in \Sigma$ is given.

   (b) When g is $h_1 \vee h_2$ or $\neg h$ for some BRTL formulas h_1, h_2 or h, the truth value of g on each state can be obtained by the truth values of h_1, h_2 or h which are already labeled.

   (c) When g is $\exists A$ for some automata connective A, the following (i)-(iv) are performed.

      i. A graph $G = (V, E)$ is constructed, where
      $V = \{(s, q) | s \in \Sigma, q \in Q\}$ and
      $E = \{((s, q), (s', q')) | (s, s') \in R$ and $S, s \models Br(q, q')\}$.

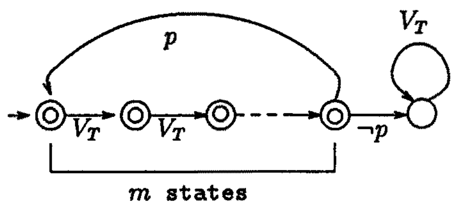

Figure 3: An automata connective representing repetition

ii. The set of vertices of all the strongly connected components on the subgraph G' of G whose vertices are in $V' = \{(s, q_F) | q_F \in F\}$ is obtained. It is described as V_C

iii. The set of the vertices in G which can reach some vertex in V_C is obtained. It is described as V_R

iv. $\exists A$ is labeled to every $s \in \Sigma$ such that $(s, q_0) \in V_R$ (q_0 is the initial state of A).

□

Prop. 2.10 *Algorithm 2.9 runs in time $O(Size(S) \times Len(f))$.*
(Sketch of Proof) This is proved by the following facts:

- In 1, the transformation from $\neg A$ contained in f to $\overline{A}$ can be performed in time $O(\text{Len}(f))$.
- In 2.(b), if the truth values of h_1, h_2 or h are determined on each state $s \in \Sigma$, then the truth values of $h_1 \vee h_2$ and $\neg h$ can be determined in time $O(|\Sigma|)$.
- In 2.(c).i, the determination of $S, s \models Br(q, q')$ for all $s \in \Sigma$ and all $q, q' \in Q$ can be performed in time $\text{O}(\text{Size}(S) \times (\sum_{f \in Br'(Q \times Q)} \text{Len}(f) + \sum_{g \in B} \text{Len}(g)))$. Then the construction of the graph G costs $\text{O}(\text{Size}(S) \times (|Q| + |\text{Edges}(a)|))$ in time, thus in total, $\text{O}(\text{Size}(S) \times (\text{Len}(A)))$.
- In 2.(c).ii, iii and iv, the construction of strongly connected components and the calculation of the vertices reachable to the components can be performed in linear to the size of the graph. (Here the size is the sum of the numbers of edges and vertices.)

□

3 Temporal Logic with Substitutions and References

3.1 Succinctness of Descriptions

Some temporal properties expressed in the linear time temporal logic with □ (always) and ○ (next) as temporal operators cannot be described easily in CTL or BRTL. For example, an assertion for a sequence detector such that the value of a signal line z is 1 iff it recognizes the sequence 110 on a signal line x, is described in the linear time temporal logic as follows.

$$\Box((x \wedge \bigcirc(x \wedge \bigcirc\neg x)) \equiv \bigcirc\bigcirc z)$$

Using BRTL, this property is expressed as

$$\forall\Box\forall A,$$

where $\Box f$ is equivalent to $\neg Until(V_T, \neg f)$ and $A \overset{\text{def}}{=}$

```
[r]  if $x_1$ [r] {
         if $x_1$ [r] {
             if $\neg x_1 \equiv z$ [ac]
             else if $x_1 \wedge \neg z$ [ac]
             else [re]
         }
         else L: [r]  {
             if $\neg z$ [ac]
             else [re]
         }
     else [r] $V_T$ goto L
```

By allowing substitution to Boolean variables and references to them, the property can be expressed succinctly as

$$\forall\Box\forall(\ [\mathtt{r}]\ V_T, v_1 := x\,[\mathtt{r}]\,V_T, v_2 := x\ \ [\mathtt{r}]\ \ (v_1 \wedge v_2 \wedge \neg x) \equiv z\ [\mathtt{ac}]\ )$$

Here v_1 and v_2 are Boolean variables and $v_1 := x$ means that the value of x is substituted to v_1.

Def. 3.1 Syntax of EBRTL(Extended BRTL)

Let $V = \{v_1, v_2, \cdots, v_m\}$ be a set of Boolean variables. AP of BRTL contains V. EBRTL formulas are defined similarly to BRTL formulas.

Automata connectives are similar to those of BRTL except that

- Formulas labeled to the transitions are composed of EBRTL formulas.
- Substitutions $v_i := f$ (f is an EBRTL formula) can be labeled to the transitions. For a transition, substituting to a variable v_i is allowed only once.

More precisely, the function Br is redefined as follows:

$Br : Q \times Q \rightarrow BF \times SUB$ is a partial function which satisfies the following three conditions, where SUB is a class of sets of substitutions.

We define $BF_q \overset{\text{def}}{=} \{f | \exists q'.Br(q, q') = (f, sub), sub \in SUB\}$ for each $q \in Q$.

For any $q \in Q$, (1) $f_1 \wedge f_2 \equiv V_F$ for any$f_1, f_2 \in BF_q$, (2) $\bigvee_{f \in BF_q} f \equiv V_T$ and (3) each element $sub \in SUB$ has at most one substitution of the form $v_i := f$ for each $v_i \in V$. (sub can be empty.) $\Box$

Before defining the semantics, we show a sequence along which $\exists(\ [\mathtt{r}]\ v_1 := x\ [\mathtt{r}]\ v_2 := x\ [\mathtt{r}]\ (v_1 \wedge v_2 \wedge \neg x) \equiv z\ [\mathtt{ac}]\ )$ holds in Fig. 4. The tuple labeled to each state stores the values of v_1 and v_2. We have to give initial values of the Boolean variables. In the example, $v_1 = F$ and $v_2 = F$ are the initial values.

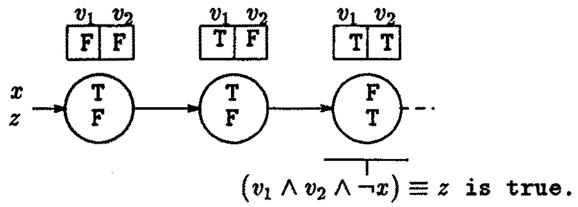

Figure 4: Intuitive Semantics of an EBRTL formula

Note that the change of the values of the Boolean variables caused by substitutions at a state occurs at its next state. This prevents oscillations of the values at each state.

Def. 3.2 Semantics

The semantics of EBRTL is defined for a Kripke structure S. When an EBRTL formula f holds at a state s on S with initial values of Boolean variables represented by a vector notation $\vec{v} = (v_1, v_2, \cdots, v_m)$ ($v_j = F$ or T), we denote as $S, s, \vec{v} \models f$. $\vec{v}[j]$ represents the j-th component of the vector $\vec{v}$.

While the truth value of each atomic propositions except Boolean variables are determined for $(S, s, \vec{v})$ by $I(s)$, that of the Boolean variable v_j $j = 1, 2, \cdots, m$ is the value of the j-th component of $\vec{v_j}$.

Here we define the semantics for $S, s, \vec{v} \models \exists A$ (A is an automaton connective).

$S, s, \vec{v} \models \exists A$ iff there exists an infinite sequence $\tau = t_0 t_1 t_2 \cdots$ that satisfies the following conditions.

Here $t_i = (s_i, \vec{v_i}, q_i) \in \Sigma \times \{T, F\}^m \times Q$.

- $(s_i, s_{i+1}) \in R$
- Let $(f, sub) = Br(q_i, q_{i+1})$. If $(\vec{v_i}[j] := g) \in sub$, then $\vec{v_{i+1}}[j] = \mathrm{T}$ iff $S, s_i, \vec{v_i} \models g$. Otherwise $\vec{v_{i+1}}[j] = \vec{v_i}[j]$.
- Let $(f, sub) = Br(q_i, q_{i+1})$. $S, s_i, \vec{v_i} \models f$.

□

3.2 A Model Checking Algorithm for EBRTL

The outline of a model checking algorithm of EBRTL is shown.

Algorithm 3.3

- Input: A Kripke structure S, an BRTL formula f and initial values of the Boolean variables $\vec{v}$
- Output: $S, s, \vec{v} \models f$ for all the initial states of S ?
- Method: We modify the model checking algorithm shown in Algorithm 2.9 as follows:
  - In constructing the graph G, tuples of the form $(s, q, \vec{v})$, where $s \in \Sigma$, $q \in Q$ and $\vec{v}$ is the vectors of the truth values for $v_j \in V$, are used instead of (s, q).

Prop. 3.4 *The model checking algorithm for EBRTL runs in time* $O(Size(S) \times Len(f) \times 2^{|V|})$.

(Sketch of Proof) This is proved by the fact that the number of the tuples of the form $(s, q, \vec{v})$ is Size(S)$\times$Len(f) $\times$ $2^{|V|}$. □

4 Conclusion

We proposed a temporal logic BRTL which is more expressive than CTL and is more tractable than ECTL in terms of model checking. Furthermore, we provided an extension of BRTL which has the mechanism of substitutions to Boolean variables and references to them.

The linear time complexity of the model checking algorithms of CTL or BRTL is obtained by excluding nondeterminism from their descriptions. CTL* or ETL can express, in a sense, the concept of nondeterminism by allowing formulas of the form $\forall(f \vee g)$ (f and g are the formulas without the path quantifiers $\forall$ and $\exists$) and $A \vee B$ (A and B are automata connectives based on nondeterministic finite automata) respectively.

The mechanism of substitutions and references to Boolean variables also introduces nondeterminism to BRTL, in a sense. In the example of the sequence detector, the branches of the automaton connective of BRTL formula are "folded" by using substitutions. This means that the substitution mechanism is another way to introduce nondeterminism.

Future problems are as follows:

- Determination of the expressive power of EBRTL.
- Determination of the complexity of model checking problem for EBRTL.

References

[1] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications: A Practical Approach. In *10th ACM Symposium on Principles of Programming Languages*, pages 117–126, January 1983.

[2] M. C. Browne, E. M. Clarke, D. L. Dill, and B. Mishra. Automatic Verification of Sequential Circuits Using Temporal Logic. *IEEE Transactions on Computers*, C-35(12):1035–1044, December 1986.

[3] H. Hiraishi. Design Verification of Sequential Machines Based on a Model Checking Algorithm of ϵ-free Regular Temporal Logic. In *Computer Hardware Description Languages and their applications*, pages 249–263, June 1989.

[4] E. M. Clarke and O. Grümberg and R. P. Kurshan. A Synthesis of Two Approached for Verifying Finite State Concurrent Systems. Technical report, Carnegie Mellon University, 1987. manuscript.

[5] P. Wolper. Temporal Logic Can Be More Expressive. In *Proceedings of 22nd Annual Symposium on Foundations of Computer Science*, pages 340–348, 1981.