This advance publication article will be replaced by the finalized version after proofreading.
A Subclass of Mu-Calculus with the Freeze Quantifier Equivalent to Büchi Register Automata

Yoshiaki TAKATA†, Member, Akira ONISHI††, Ryoma SENDA†††, Nonmembers, and Hiroyuki SEKI†††, Fellow

SUMMARY Register automaton (RA) is an extension of finite automaton for dealing with data values in an infinite domain. In the previous work, we proposed disjunctive \( \mu \)-calculus (\( \mu_1 \)-calculus), which is a subclass of modal \( \mu \)-calculus with the freeze quantifier, and showed that it has the same expressive power as RA. However, \( \mu_1 \)-calculus is defined as a logic on finite words, whereas temporal specifications in model checking are usually given in terms of infinite words. In this paper, we re-define the syntax and semantics of \( \mu_1 \)-calculus to be suitable for infinite words and prove that the obtained temporal logic, called \( \mu_1^{\omega} \)-calculus, has the same expressive power as Büchi RA. We have proved the correctness of the equivalent transformation between \( \mu_1^{\omega} \)-calculus and Büchi RA with a proof assistant software Coq.

In this paper, we re-define the syntax and semantics of \( \mu_1^{\omega} \)-calculus to be suitable for infinite words, and we prove that the obtained temporal logic, called \( \mu_1^{\omega} \)-calculus, has the same expressive power as Büchi RA. We have proved the correctness of the equivalent transformation between \( \mu_1^{\omega} \)-calculus and Büchi RA with a proof assistant software Coq.

The main difference between \( \mu_1 \)-calculus and \( \mu_1^{\omega} \)-calculus is that the former requires only the least solution of a system of equations while the latter has to consider both the least and greatest solutions of systems of equations. For example, a proposition “every (resp. some) odd position in a given infinite word satisfies a given atomic proposition \( p \)” which can be represented by Büchi RA, is represented in \( \mu_1^{\omega} \)-calculus as the greatest (resp. least) solution of a recursive equation \( V = p \wedge XXV \) (resp. \( V = p \vee XXV \)). To deal with both the least and greatest solutions, \( \mu_1 \)-calculus in [7] classifies each variable into two types, \( \mu \) and \( \nu \). A \( \mu \)-type (resp. \( \nu \)-type) variable represents its least (resp. greatest) solution. Moreover, \( \mu_1 \)-calculus has a syntactic restriction that a \( \mu \)-type variable and a \( \nu \)-type one cannot mutually depend on each other. Similar restriction is imposed in usual modal \( \mu \)-calculus [8] that has \( \mu \) and \( \nu \) constructs instead of systems of equations; any two constructs cannot be mutually a subformula of each other. We do not follow these restrictions in \( \mu_1^{\omega} \)-calculus, because they prevent simple correspondence between states of a Büchi RA and variables in \( \mu_1^{\omega} \)-calculus equations. Focusing on the fact that we only need the same expressive power as Büchi RA, we define the semantics carefully and keep the syntax and semantics of \( \mu_1^{\omega} \)-calculus simple without introducing restrictions on the dependence between variables.

1. Introduction

Register automaton (RA) [1] is an extension of finite automaton (FA) that has registers for dealing with data values. RA has been used as a formal model in various areas such as verification of systems [2]. Model checking is a technique to verify whether every run of a model \( M \) of a system satisfies a given specification \( \varphi \). For a finite-state model of a system and a linear temporal logic (LTL) specification, the model checking can be reduced to the emptiness problem of a Büchi automaton [3]. LTL model checking has also been shown to be decidable for pushdown systems (PDS) [4] and register pushdown systems (RPDS) [5]. These model-checking methods rely on the fact that every LTL formula can be transformed into an equivalent Büchi automaton. To extend these model-checking methods to models with registers, we need an appropriate temporal logic to give a specification, which can be transformed into an equivalent Büchi RA.

In the previous work, we proposed disjunctive \( \mu \)-calculus (\( \mu_1 \)-calculus) [6], which is a subclass of modal \( \mu \)-calculus with the freeze quantifier (\( \mu_1 \)-calculus) [7] and is shown to have the same expressive power as RA. However, \( \mu_1 \)-calculus is defined as a logic on finite words, whereas temporal specifications in model checking are usually given in terms of infinite words.

2. Preliminaries

Let \( \mathbb{N} \) be the set of natural numbers not including zero. Let \( [n] = \{1, \ldots, n\} \) for \( n \in \mathbb{N} \). \( \mathcal{P}(X) \) denotes the power set of a set \( X \). \( X^\omega \) is the set of infinite sequences over \( X \).

Let \( A \) be a finite set of atomic propositions. Let \( \Sigma = \mathcal{P}(A) \) and we call it an alphabet. We assume a countable set \( \mathcal{D} \) of data values. A sequence \( w \in (\Sigma \times \mathcal{D})^\omega \) is called an infinite data word. For \( w = (b_1,d_1)(b_2,d_2) \ldots \in (\Sigma \times \mathcal{D})^\omega \), we let \( w_i = (b_i,d_i) \) and \( w[i] = (b_i,d_i)(b_{i+1},d_{i+1}) \ldots \) for \( i \in \mathbb{N} \). We also let \( f((b,d)) = b \) and \( s((b,d)) = d \) for \( (b,d) \in (\Sigma \times \mathcal{D}) \). Let \( x, y \in \mathcal{D} \) be a data value designated as the initial value of registers.

An assignment of data values to \( k \) registers is a \( k \)-tuple \( \theta \in D^k \). The value of \( r \)-th register in an assignment \( \theta \) is
denoted by $\theta(r)$. For $R \subseteq [k]$ and $d \in \mathbb{D}$, $\theta[R \leftarrow d]$ is the assignment obtained from $\theta$ by updating the value of the $r$-th register for every $r \in R$ to $d$. Namely, $\theta[R \leftarrow d](r) = d$ for $r \in R$ and $\theta[R \leftarrow d](r) = \theta(r)$ for $r \notin R$. Let $\bot^k$ be the assignment that assigns $\bot$ to all the $k$ registers.

3. Disjunctive $\mu$-calculus with the freeze quantifier ($\mu^{\bot}_{\mathbf{a}}$-calculus)

In the proposed logic named $\mu^{\bot}_{\mathbf{a}}$-calculus, each specification is given as a system of equations and each equation is given by a formula in the disjunctive LTL with the freeze quantifier ($\mu^{\bot}_{\mathbf{a}}$)-calculus defined as follows.

Definition 1: $\mu^{\bot}_{\mathbf{a}}$ formulas over a set $\mathcal{V}$ of variables, a set $\mathcal{A}$ of atomic propositions, and $k$ registers are defined by the following $\psi$, where $V \in \mathcal{V}$, $p \in \mathcal{A}$, $R \subseteq [k]$ and $d \in \mathbb{D}$.

- $\psi ::= V \mid \psi \land \psi \mid \downarrow_p X \psi \land \phi \mid tt$, $\psi ::= tt \mid \mathbf{e}f \mid p \mid \neg p \mid \uparrow_r \mid \neg \uparrow_r \mid \phi \land \phi$.

Basic $\mu^{\bot}_{\mathbf{a}}$ formulas are formulas defined by the above $\phi$. Let $\Phi_k$ denote the basic set of $\mu^{\bot}_{\mathbf{a}}$ formulas.

Temporal operator $X$ (next) has the same meaning as in the usual LTL. The freeze operator $\downarrow_r$ represents the storing of an input data value into registers specified by $R$. The look-up operator $\uparrow_r$ represents the proposition that the stored value of the $r$-th register equals the input data value. Let $\downarrow \psi$ be an abbreviation of $\downarrow_r X \psi \land tt$. We consider $\phi \in \Phi_k$ is also a $\mu^{\bot}_{\mathbf{a}}$ formula since $\phi$ is equivalent to $tt \land \phi$.

A basic $\mu^{\bot}_{\mathbf{a}}$ formula describes a proposition on a single position in an infinite data word. The semantics for a basic $\mu^{\bot}_{\mathbf{a}}$ formula $\phi \in \Phi_k$ is provided by the relation $w, i, \theta \models \phi$, defined as follows.

Definition 2: For an infinite data word $w$, a position $i \in \mathbb{N}$, and an assignment $\theta$, $w, i, \theta \models p := p \in \text{fst}(w_i)$.

$\downarrow \psi \models p \models \text{snd}(w_i)$.

$\uparrow_r \models \theta(r) = \text{snd}(w_i)$.

$\downarrow \psi \models \text{tt}$ always holds.

where $p \in \mathcal{A}$ and $r \in [k]$. The semantics for negation and conjunction are the same as usual.

Definition 3: A system of equations (or a system) $\sigma$ over a set $\mathcal{V} = \{V_1, \ldots, V_t\}$ of $t$ variables is a mapping from $\mathcal{V}$ to $\mu^{\bot}_{\mathbf{a}}$ formulas over $\mathcal{V}$. A system $\sigma$ intuitively represents a set of equations $V_1 = \sigma(V_1), \ldots, V_t = \sigma(V_t)$. For each system $\sigma$, one variable is designated as its main variable. Moreover, one or more variables are designated as $\omega$-variables, and the set $\mathcal{Var}_\sigma (\subseteq \mathcal{Var})$ of $\omega$-variables is provided. We assume that $\mathcal{Var}_\sigma$ contains an $\omega$-variable $V_{\text{tt}}$, such that $\sigma(V_{\text{tt}}) = \text{tt}$. Moreover, we assume that for every $V \in \mathcal{Var}_\sigma$ and $V' \in \mathcal{Var}$, $\sigma(V) = \sigma(V')$ implies $V = V'$.

Intuitively, an $\mu^{\bot}_{\mathbf{a}}$ formula containing a variable $V$ equals the $\mu^{\bot}_{\mathbf{a}}$ formula obtained by replacing $V$ with $\sigma(V)$. For a recursively defined variable $V$, if $V \in \mathcal{Var}_\omega$, then we consider $\sigma$ intuitively equals the formula obtained by replacing $V$ with $\sigma(V)$ infinitely many times. If $V \notin \mathcal{Var}_\omega$, then $\sigma$ intuitively equals the formula obtained by replacing $V$ with $\sigma(V)$ arbitrary finite times. For example, a formula $\Phi_\omega$ using the global operator $G$ in $\mu^{\bot}_{\mathbf{a}}$, which means “every position of a given infinite word satisfies $\phi$, can be simulated by an equation $V = \phi \land XV$ where $V \in \mathcal{Var}_\omega$. On the other hand, a formula $\phi \land \psi$ using the until operator $U$ in $\mu^{\bot}_{\mathbf{a}}$, which means “some position of a given infinite word satisfies $\phi$ and every position before that position satisfies $\phi$,” can be simulated by an equation $V = \psi \lor (\phi \land XV)$ where $V \in \mathcal{Var}_\omega$.

Formally, we introduce a 5-tuple $(i, \theta(j, \theta', x))$ where $i, j \in \mathbb{N}$ are positions in an infinite word, $\theta, \theta'$ are assignments, and $x$ is an $\omega$-variable, and let the “value” of each variable $V \in \mathcal{Var}$ be a set of these 5-tuples. Intuitively, the value of a variable $V$ contains $(i, \theta(j, \theta', x))$ if and only if $(w, i, \theta)$ satisfies $V$ under the assumption that $(w, j, \theta')$ satisfies $x$, where $w$ is a given infinite word. The sentence “$(w, i, \theta)$ satisfies $V$” intuitively means that $\sigma(V)$ holds at position $i$ in word $w$ when the contents of registers equal $\theta$. If we were considering only the least solution of a system of equations, then it was sufficient to let the value of a variable be a set of a pair $(i, \theta)$ as the same as $\mu^{\bot}_{\mathbf{a}}$-calculus in [6]. Instead of considering an infinite substitution of $\sigma(V)$ for an $\omega$-variable $V$, we consider an infinite sequence $i_1, i_2, \ldots$ of positions each of which satisfies $V$. To handle such a sequence, we use 5-tuple $(i, \theta(j, \theta', V))$ instead of pair $(i, \theta)$, where $(j, \theta')$ represents “the next position” of $(i, \theta)$ satisfying $V$ (and the contents of registers at that position). The tuple asserts that whether $(w, i, \theta)$ satisfies $V$ depends on whether $(w, j, \theta')$ satisfies $V$. The latter may also depend on whether some $(w, j_1, \theta_1)$ satisfies $V$. We consider $(w, i, \theta)$ satisfies $V$ if such an infinite chain of dependence exists.

An environment is the mapping from a variable to the above-mentioned set of 5-tuples.

Definition 4: An environment for a given infinite data word $w \in (\mathbb{D} \times \mathbb{D})^{\omega}$ is a mapping of type $\mathcal{V} \rightarrow \mathcal{P}(\mathbb{N} \times \mathbb{D} \times \mathbb{D} \times \mathcal{Var}_\omega)$. Let $\mathcal{Env}_w$ be the set of environments for $w$.

Definition 5: The following relation gives the semantics for each variable in $\mu^{\bot}_{\mathbf{a}}$ formulas over a given environment. For an infinite data word $w$, a position $i \in \mathbb{N}$, an assignment $\theta$, an environment $u \in \mathcal{Env}_w$, and a variable $V \in \mathcal{Var}$, $w, i, \theta \models_u V := \exists j. \theta', x : (i, \theta(j, \theta', x)) \in u(V)$, $i < j$ and $w, j, \theta' \models_u x$.

A system $\sigma$ of equations specifies an environment under which all equations $V = \sigma(V)$ hold. Such an environment can be seen as a solution of the system of equations. We define an augmented version of the semantic relation $\models_u$ as follows, and let a solution of $\sigma$ be an environment $u$ satisfying $w, i, \theta(j, \theta', x) \models_u V \iff w, i, \theta, j, \theta', x \models_u \sigma(V)$.

Definition 6: For an infinite data word $w$, positions $i, j \in \mathbb{N}$, and assignments $\theta, \theta'$.
such that \( i \leq j \), assignments \( \theta, \theta' \), \( x \in \text{Var}_{\omega} \), and an environment \( u \in \text{Env}_{\omega} \),

\[
\begin{align*}
& w, i, \theta, j, \theta', x \models_{u} V \iff (i, \theta; j, \theta', x) \in u(V), \\
& w, i, \theta, j, \theta', x \models_{u} \psi_{1} \lor \psi_{2} \iff (w, i, \theta, j, \theta', x \models_{u} \psi_{1}) \\
& \quad \text{or} (w, i, \theta, j, \theta', x \models_{u} \psi_{2}), \\
& w, i, \theta, j, \theta', x \models_{u} 1_{R}^{X} \wedge \phi \iff i < j \text{ and } w, i, \theta \models \phi \\
& \quad \text{and} (w, i + 1, \theta \models \neg \text{snd}(w_{i})), j, \theta' \models \psi_{u}), \\
& w, i, \theta, j, \theta', x \models_{u} \top \iff \theta \models \top' \quad \text{and } x = V_{\text{tt}}.
\end{align*}
\]

The (least) solution of \( \sigma \) can be represented as the least fixed point of the variable updating function \( F_{\tau,w} \) in Definition 7. We define a partial order \( \preceq \) over \( \text{Env}_{\omega} \) as:

\[
u_{1} \preceq \nu_{2} \iff u_{1}(V) \preceq u_{2}(V) \quad \text{for every } V \in \text{Var}.
\]

Let \( \emptyset \) denote the environment that maps every \( V \in \text{Var} \) to \( \emptyset \). The least upper bound \( u_{1} \cup u_{2} \) of \( u_{1}, u_{2} \in \text{Env}_{\omega} \) can be represented as \( (u_{1} \cup u_{2})(V) = u_{1}(V) \cup u_{2}(V) \) for every \( V \in \text{Var} \). Then, \( \text{Env}_{\omega,\emptyset} \) is a complete partial order (CPO)\(^{9}\) [9, Sect. 5.2] with the least element \( \emptyset \).

**Definition 7:** For a system \( \sigma \) of equations over \( \text{Var} \) and an infinite data word \( w \in (\Sigma \times D)\omega \), we define a variable updating function \( F_{\sigma_{w}} : \text{Env}_{\omega} \rightarrow \text{Env}_{\omega} \) as:

\[
F_{\sigma_{w}}(u)(V) :=
\begin{cases}
(i, \theta; j, \theta', x) \models_{u} \sigma(V) & \text{if } V \in \text{Var}_{\omega}, \\
(i, \theta; j, \theta', V) \models_{i} \sigma_{x} & \text{if } V \in \text{Var},
\end{cases}
\]

for every \( u \in \text{Env}_{\omega} \) and \( V \in \text{Var} \).

**Definition 8:** For a variable updating function \( F \), a fixed point of \( F \) is an environment \( u \) that satisfies \( F(u) = u \). A fixed point \( u \) of \( F \) is called the least fixed point of \( F \), denoted by \( \text{lfp}(F) \), if \( u \preceq u' \) for any fixed point \( u' \) of \( F \).

We can prove that \( F_{\sigma_{w}} \) is continuous,\(^{10} \) which implies that at least fixed point \( \text{lfp}(F_{\sigma_{w}}) = \bigcup_{n=0}^{\infty} F_{\sigma_{w}}^{n}(\emptyset) \).

**Definition 9:** For an infinite data word \( w \) and a system \( \sigma \) with its main variable \( V_{1} \), we say \( w \) satisfies \( \sigma \), denoted by \( w \models \sigma \), if \( w, 1, \bot \vdash \models_{\text{lfp}(F_{\sigma_{w}})} V_{1} \), where \( \models_{\text{lfp}(F_{\sigma_{w}})} \) is the relation defined in Definition 5.

**Example 1:** Consider a system \( \sigma_{1} \) of equations:

\[
\sigma_{1} = \{ V_{\text{tt}} = \top, V_{1} = \top, V_{2} = V_{2} \lor (XV_{2} \wedge (\neg V_{1} \land p_{1})) \}, \quad V_{3} = \downarrow_{R}XV_{2}
\]

over \( \text{Var} = \{ V_{\text{tt}}, V_{1}, V_{2}, V_{3} \} \) with \( \text{Var}_{\omega} = \{ V_{\text{tt}} \} \). The main variable is \( V_{2} \). This system of equations is intuitively equivalent to \( \downarrow_{R}X((\neg V_{1} \land p_{1}) \lor \top) \) where \( U \) is the until operator in \( \Sigma \times D \).

LTL, and an infinite data word \((b_{1},d_{1})(b_{2},d_{2}) \ldots \in (\Sigma \times D)^{\omega} \) satisfies \( \sigma_{1} \) if and only if \( d_{1} = d_{0} \) for \( \exists \theta > 1 \) and \( p_{1} \in b_{1} \) and \( d_{1} \neq d_{1} \) for \( \forall \theta \in \{ 2, \ldots, n-1 \} \). On the other hand, the same system of equations \( \sigma_{2} = \sigma_{1} \) with \( \text{Var}_{\omega} = \{ V_{\text{tt}}, V_{2} \} \) intuitively equivalent to \( \downarrow_{R}X((\neg V_{1} \land p_{1}) \lor \top) \) where \( W \) is the weak until operator.\(^{11} \)

Every infinite data word that satisfies \( \sigma_{1} \) also satisfies \( \sigma_{2} \).

Moreover, an infinite data word \((b_{1},d_{1})(b_{2},d_{2}) \ldots \) such that \( p_{1} \in b_{1} \) and \( d_{1} \neq d_{1} \) for \( \forall \theta \geq 2 \) also satisfies \( \sigma_{2} \).

Definition 10: A Büchi register automaton with \( k \) registers \((k\text{-BRA})\) over \( \Sigma \) and \( D \) is a quadruple \( \mathcal{A} = (Q,q_{0},\delta,F) \), where \( Q \) is a finite set of states, \( q_{0} \in Q \) is the initial state, \( \delta \subseteq Q \times (F_{k} \times \{ \epsilon \}) \times \mathcal{P}([k]) \times Q \) is a set of transition rules, and \( F \subseteq Q \) is a set of accepting states.

A transition rule \((q,\phi,R,\rho) \in \delta \) is written as \((q,\phi,R) \rightarrow \rho \) for readability. A transition rule whose second element \( \epsilon \) is \( \epsilon \) is called an \( \epsilon \)-rule. For each \( \epsilon \)-rule \((q,\epsilon,R) \rightarrow \epsilon \) \( \delta \), \( R \) must be \( \emptyset \).

For example, a transition rule whose second element is \( p_{1} \land \top \) can be applied to the current position \( i \) in an input data word \( w \) if \( p_{1} \in \text{fst}(w_{i}) \) and \( \text{snd}(w_{i}) \) equals the contents of the second register. An \( \epsilon \)-rule represents a transition without consuming the input. Note that we can eliminate \( \epsilon \)-rules in the same way as usual non-deterministic finite automata.

The third element \( \delta \subseteq \{ k \} \) of a transition rule designates registers to be updated. \( R \) can be empty, which represents no register is updated.

For a \( k \text{-BRA} \( \mathcal{A} = (Q,q_{0},\delta,F) \), every triple \((q,\theta,\psi) \in \) \( (\phi \models \psi) \) or \((\psi \models \phi) \).
$Q \times \Delta^k \times (\Sigma \times D)^{\omega}$ is called an instantaneous description (ID) of $\mathcal{A}$, where $q$, $\theta$, and $w\prime$ represent the current state, the current assignment, and the remaining suffix of the input data word, respectively. We define the transition relation $r_{\mathcal{A}}$ over IDs by the following inference rules:

\[
\frac{(q, \phi, R) \rightarrow q' \in \delta \cdot w, i, \theta \models \phi}{(q, \theta, w^{[i]}); r_{\mathcal{A}} (q', \theta, R \models \text{send}(w_i))}, w^{[i+1]})
\]
\[
\frac{(q, \xi, \theta) \rightarrow q' \in \delta \cdot i \in \mathbb{N}}{(q, \theta, w^{[i]}); r_{\mathcal{A}} (q', \theta, w^{[i]}))}
\]

We say that $\mathcal{A}$ accepts $w$ if there exist $q_f \in F$, $i_1, i_2, \ldots \in \Delta^k$ such that $i_1 < i_2 < \ldots$ and $(q_0, t, \omega) \vdash_{\mathcal{A}} (q_f, t, w^{[i_1]}); r_{\mathcal{A}} (q_f, t, w^{[i_2]}); r_{\mathcal{A}} \ldots$. The language of $\mathcal{A}$ is defined as $L(\mathcal{A}) = \{ w \mid \mathcal{A} \text{ accepts } w \}$.

5. Transformation between $\mu^1_{\text{doo}}$-calculus and Büchi RA

Proofs of theorems are omitted due to space limitation and included in a longer version of this paper. Proof scripts for the Coq proof assistant are also available.\[\cite{http://arxiv.org/abs/2406.11351}\]\[\cite{https://github.com/ytakata69/proof-mucal-bra}\]

Theorem 1: Let $\sigma$ be a system of equations over $\text{Var}$, whose main variable is $V_i$, in the normol form described in Lemma 1. The following Büchi RA $\mathcal{A} = (Q, \sigma(V_i), \delta, F)$ is equivalent to $\sigma$; i.e., $L(\mathcal{A}) = \{ w \mid w \models \sigma \}$.

$Q = \{ \sigma(V_i) \mid V_i \in \text{Var} \}$,
$\delta = \{ (tt, tt, \theta) \rightarrow tt \}$
$\cup \{ (V_i \lor V_j, e, \theta) \rightarrow \sigma(V_j) \mid V_j \lor V_i \in Q \}$
$\cup \{ (V_i \lor V_j, e, \theta) \rightarrow \sigma(V_j) \mid V_i \lor V_j \in Q \}$
$\cup \{ (\downarrow X V_i \land \phi, \theta) \rightarrow \sigma(V_i) \mid \downarrow X V_i \land \phi \in Q \}$,
$F = \{ \sigma(V_{\omega}) \mid V_{\omega} \in \text{Var}_{\omega} \}$.

Theorem 1 is a direct consequence of the following Lemmas 2 and 3:

Lemma 2: $w, i, \theta, j, \theta', x \equiv_{\text{typ}(\text{For}, i)} V$ iff $\sigma(V_i) \theta, w^{[i]}); r_{\mathcal{A}} (\sigma(x), \theta', w^{[i]}))$ and $\sigma(x) \in F$.

Lemma 3: $w, i, \theta \equiv_{\text{typ}(\text{For}, i)} V$ iff $\sigma(V_i) \theta, w^{[i]}); r_{\mathcal{A}} (q_f, \theta, w^{[i]}); r_{\mathcal{A}} (q_r, \theta, w^{[i]}); r_{\mathcal{A}} \ldots$ for some $q_f \in F$ and $i_1, i_2, \ldots \text{ such that } i < i_1 < i_2 < \ldots$.

Example 2: A Büchi RA equivalent to $\sigma_1$ in Example 1 is shown in Figure 1. By additionally designating the state $V_1 \lor V_2^\prime (\equiv \sigma_2(V_2))$ as an accepting state in Figure 1, we obtain a Büchi RA equivalent to $\sigma_2$ in Example 1.

Theorem 2: Let $\mathcal{A} = (Q, q_0, \delta, F)$ be a Büchi RA without $e$-rules, where every $q \in Q$ has at least one state transition from it. We define the set $\text{Var}$ of variables, the set $\text{Var}_{\omega}$ of $\omega$-variables, and the system $\sigma$ of equations over $\text{Var}$, whose main variable is $V_{\omega}$, as follows. Then, $\sigma$ is equivalent to $\mathcal{A}$.

\[\begin{align*}
&\begin{array}{c}
\downarrow XV_2 (1) \quad \downarrow XV_2 \quad \neg t_1 \land \neg p_1, \emptyset \\
(V_1 \lor V_2^\prime) \quad e^{-} \\
Xtt \land t_1 \quad \\
\end{array} \\
\begin{array}{c}
\downarrow t_1, \emptyset \\
\end{array}
\end{align*}\]

Fig 1 Büchi RA equivalent to $\{ V_{tt} = tt, V_1 = t_1, V_2 = V_1 \lor (XV_2 \land \neg t_1 \land \neg p_1) \}$ when $\text{Var}_{\omega} = \{ V_{tt} \}$.

$Var = \{ V_q \mid q \in Q \} \cup \{ V_r \mid r \in \delta \} \cup \{ V_{tt} \}$,
$Var_{\omega} = \{ V_q \mid q \in F \} \cup \{ V_{tt} \}$,
$\sigma = \{ V_q = \bigvee_{r \in \delta_{q_0}} V_r \mid q \in Q \}$
$\cup \{ V_r = \downarrow_{R} XV_i' \land \phi \mid r \in \delta, r = (q, \phi, R) \rightarrow q' \}$
$\cup \{ V_{tt} = tt \}$,
where $\delta_{q_0} = \{ r \in \delta \mid r = (q, \phi, R) \rightarrow q' \}$ for $\exists \phi, R, q'$.\]

Proof sketch. Construct BRA $\mathcal{A}' = (Q', q_0', \delta', F')$ by applying Theorem 1 to $\sigma$ obtained from a given A as above. It is sufficient to show $L(\mathcal{A}) = L(\mathcal{A}')$.

6. Conclusion

In this paper, we proposed $\mu^1_{\text{doo}}$-calculus, which is a subclass of $\mu$-calculus with the freeze quantifier and has the same expressive power as Büchi RA. We showed a transformation from a system of equations in $\mu^1_{\text{doo}}$-calculus to an equivalent Büchi RA as well as its reverse. We proved the correctness of the transformations. We have also provided formal proofs of those theorems using the Coq proof assistant.

Future work includes developing a formal verification method based on RA and the proposed calculus.

References