Monadic Second Order Logic and Automata on Infinite Words: Büchi's Theorem

R. Dustin Wehr

December 18, 2007

Büchi's theorem establishes the equivalence of the satisfiability relation for monadic second-order logic, and the acceptance relation for Büchi automata. The development of the theories of monadic second-order logic and Büchi automata follows Thomas's survey[6] closely, in that all of the concepts and results found in this report are also in [6]. However, because the scope of Thomas's survey is much greater, he develops the theories in a more general (and more complicated) way than is necessary to understand Büchi's theorem, and he only sketches the proof of Büchi's theorem, which is given in detail here.

Two theories concerned with infinite words

For both of the theories considered in this report, the relations of interest depend on infinite words. We will not consider finite words, although those theories can be developed in a similar way to what follows[1]. An *alphabet* A is a finite set of symbols a, a_i. A word over A is a sequence w such that $w(i) \in A$ for all $i \in \mathbb{N}$. A *language* is just a set of words.

An automata *recognizes* a language if it *accepts* all the words of the language and no others. A sentence (closed formula) in a given logic *defines* a language if it *is satisfied* by all the words of the language and no others. The purpose of the next two sections is to give precise definitions of relations for *accepts* and *is satisfied*.

[1] In [6], Thomas develops in parallel the theories for finite and infinite words, trees and graphs

Monadic Second Order Logic

Monadic second-order logic with successor (MSOL[S]) is a small fragment of second-order logic, and an extension of *first-order logic with successor* (FOL[S]). We will define the first-order part first. Let x, y, x_i, y_i denote first-order variables that range over $\mathbb{N}$. The well-formed formulas of FOL[S] are defined by:

$$\begin{aligned}\text{FOL}[\mathsf{S}]\text{ formulas } \phi, \psi \quad := \quad & \mathsf{S}(x, y) \mid x = y \mid Q_a(x) \\ & \mid \neg\psi \mid \psi \vee \phi \mid \phi \wedge \psi \mid \psi \Rightarrow \phi \\ & \mid \forall x.\psi \mid \exists x.\psi\end{aligned}$$

The satisfiability relation $w, p_1, \ldots, p_n \models \psi$ for FOL[S] is defined by:

$w, p_1, \ldots, p_n \models \mathsf{S}(x_i, x_j)$	if $p_i = p_j + 1$
$w, p_1, \ldots, p_n \models x_i = x_j$	if $p_i = p_j$
$w, p_1, \ldots, p_n \models Q_a(x_i)$	if $w(p_i) = a$
$w, p_1, \ldots, p_n \models \forall x_{n+1}.\, \psi$	if $w, p_1, \ldots, p_{n+1} \models \psi$ for all $p_{n+1} \in \mathbb{N}$
$w, p_1, \ldots, p_n \models \exists x_{n+1}.\, \psi$	if $w, p_1, \ldots, p_{n+1} \models \psi$ for some $p_{n+1} \in \mathbb{N}$
$w, p_1, \ldots, p_n \models \psi \wedge \phi$	if $w, p_1, \ldots, p_n \models \psi$ and $w, p_1, \ldots, p_n \models \phi$
$w, p_1, \ldots, p_n \models \psi \vee \phi$	if $w, p_1, \ldots, p_n \models \psi$ or $w, p_1, \ldots, p_n \models \phi$
$w, p_1, \ldots, p_n \models \psi \Rightarrow \phi$	if $w, p_1, \ldots, p_n \models \phi$ whenever $w, p_1, \ldots, p_n \models \psi$
$w, p_1, \ldots, p_n \models \neg\psi$	if it is not the case that $w, p_1, \ldots, p_n \models \psi$

To get MSOL[S], we add quantification over second-order variables X, Y, X_i, Y_i, and atomic formulas $X(x)$ for testing set membership. Second-order variables are also called *monadic predicates*, and they range over infinite subsets of $\mathbb{N}$.

$$\begin{aligned}\text{MSOL}[\mathsf{S}]\text{ formulas } \phi, \psi \quad := \quad & \mathsf{S}(x, y) \mid x = y \mid Q_a(x) \mid X(x) \\ & \mid \neg\psi \mid \psi \vee \phi \mid \phi \wedge \psi \mid \psi \Rightarrow \phi \\ & \mid \forall x.\psi \mid \exists x.\psi \mid \forall X.\psi \mid \exists X.\psi\end{aligned}$$

Note that if we use $<$ instead of S in FOL then we get a strictly more expressive logic.[2][3] The same cannot be said about MSOL, since we can simulate $<$ using second-order quantification and S (see below).

The satisfaction relation $w, P_1, \ldots, P_m, p_1, \ldots, p_n \models \psi$ for MSOL[S] has all the cases of the relation for FOL[S], converted in the expected way. We add the new

[2] A logic is said to be more expressive than another if the languages defined by the sentences of the first are a subset of the languages defined by the sentences of the second.

[3] In fact, first-order logic with $<$ defines the same languages as linear temporal logic.

cases:

$$
\begin{array}{ll}
w, P_1, \ldots, P_m, p_1, \ldots, p_n \models X_i(x_j) & \text{if } p_j \in P_i \\
w, P_1, \ldots, P_m, p_1, \ldots, p_n \models \forall X_{m+1}.\, \psi & \text{if } w, P_1, \ldots, P_{m+1}, p_1, \ldots, p_n \models \psi \\
 & \text{for all } P_{m+1} \subseteq \mathbb{N} \\
w, P_1, \ldots, P_m, p_1, \ldots, p_n \models \exists X_{m+1}.\, \psi & \text{if } w, P_1, \ldots, P_{m+1}, p_1, \ldots, p_n \models \psi \\
 & \text{for some } P_{m+1} \subseteq \mathbb{N}
\end{array}
$$

Büchi Automata

A Büchi automaton is a (nondeterministic) finite automaton that reads infinite words and uses the *Büchi acceptance condition* (defined below). A Büchi automaton $\mathcal{A}$ is a tuple $\langle Q, A, q_0, \Delta, F\rangle$, where Q is the finite set of states, A is the alphabet, $q_0 \in Q$ is the start state, $\Delta \subseteq Q \times A \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accept states. A *run* of $\mathcal{A}$ on w is a sequence of states ρ such that $\rho(0) = q_0$, and $(\rho(i), w(i), \rho(i+1)) \in \Delta$ for all $i \geq 0$.

The *Büchi acceptance condition* for a run is that at least one accept state occurs infinitely often in the run. There are several other simple acceptance conditions that are known to define automata that recognize exactly the same languages as Büchi automata. For example, a *Muller automata* is defined with a set of accepting *sets of states*, rather than a single set of accepting states. The acceptance condition for a run is that one of the accepting sets is exactly the set of states that occur infinitely often in the run. The definitions are also *inequivalent* in some interesting ways; for example, there is an effective procedure that transforms any Büchi automaton into an equivalent *deterministic* Muller automaton[4], though not every Büchi automaton can be translated into a deterministic Büchi automaton.

We will use the following two lemmas in the proof of Büchi's theorem.

Lemma. *The languages recognized by Büchi automata are closed under complement.*

I won't prove this, but one way to do it is to prove that Büchi automata recognize the same languages as deterministic Muller automata, and then prove that Muller automata are closed under complementation. The latter construction is trivial; change the set S of accepting sets to $\mathcal{P}(Q)/S$, where $\mathcal{P}(Q)$ is the powerset of Q.

Lemma. *The languages recognized by Büchi automata are closed under intersection[5].*

Proof. Let $\mathcal{A}_Q = \langle A, Q, \Delta_Q, q_0, F_Q\rangle$ and $\mathcal{A}_R = \langle A, R, \Delta_R, r_0, F_R\rangle$ be two Büchi automata. We define a third automata $\mathcal{A}$ that recognizes the intersection of the

languages of $\mathcal{A}_Q$ and $\mathcal{A}_R$. The strategy used is to enter an accept state of $\mathcal{A}$ only after entering an accept state of $\mathcal{A}_Q$ and *then* entering an accept state of $\mathcal{A}_R$. So a state of $\mathcal{A}$ has one of three types:

1. A state in which we are waiting to enter an accept state of $\mathcal{A}_Q$
2. A state in which we are waiting to enter an accept state of $\mathcal{A}_R$.
3. An accept state

We introduce a flag that takes values $\{\mathsf{waitQ}, \mathsf{waitR}, \mathsf{accept}\}$ in order to label each state with its type. $\mathcal{A}$ is given by

$$\langle A, Q \times R \times \{\mathsf{waitQ}, \mathsf{waitR}, \mathsf{accept}\}, \Delta, \langle q_0, r_0, \mathsf{waitQ}\rangle, F_Q \times F_R \times \{\mathsf{accept}\}\rangle$$

Let $f, f' \in \{\mathsf{waitQ}, \mathsf{waitR}, \mathsf{accept}\}$.
$(\langle q, r, f\rangle >, a, \langle q', r', f'\rangle) \in \Delta$ iff $(q, a, q') \in \Delta_Q$ and $(r, a, r') \in \Delta_R$ and one of the following conditions is satisfied:

- $f = \mathsf{waitQ}$, $q' \in F_Q$, and $f' = \mathsf{waitR}$.
- $f = \mathsf{waitR}$, $r' \in F_R$, and $f' = \mathsf{accept}$.
- $f = \mathsf{accept}$ and $f' = \mathsf{waitQ}$.
- $f = f'$.

□

Büchi's Theorem

Theorem (Büchi's theorem (1962)[1])**.** *A language is recognizable by a Büchi automaton if and only if it is definable in MSOL[*S*].*

Proof. First we prove that for every Büchi automata $\mathcal{A}$ there exists a sentence ψ such that $\mathcal{A}$ accepts w iff $w \models \psi$. The following formula schema does that by encoding a run of $\mathcal{A}$ on w in $\|Q\|$ second-order variables, where Q is the set of states of $\mathcal{A}$. In particular, if ρ is a run of $\mathcal{A}$ on w then $x \in X_i$ iff $\rho(x) = q_i$.
For readability, define $\mathsf{First}(x) = \forall y.\, (x = y) \vee (x < y)$.

$$\begin{aligned} w \models \exists X_1 \ldots \exists X_{\|Q\|}. \quad & \forall x.\, \mathsf{First}(x) \Leftrightarrow X_1(x) \\ \wedge \quad & \Big(\bigwedge_{i \neq j} \forall x.\, \neg(X_i(x) \wedge X_j(x))\Big) \\ \wedge \quad & \forall x.\, \forall y.\, \Big(\mathsf{S}(x,y) \Rightarrow \bigvee_{(q_i,a,q_j)\in\Delta} X_i(x) \wedge Q_a(x) \wedge X_j(y)\Big) \\ \wedge \quad & \forall x.\, \exists y.\, (x < y) \wedge \Big(\bigvee_{q_j \in F} \bigvee_{(q_i,a,q_j)\in\Delta} X_i(y) \wedge Q_a(y)\Big) \end{aligned}$$

The first three lines of the conjunction formula establish that $X_1, \ldots, X_k$ represents a valid run of $\mathcal{A}$ on w. The last line says that the run satisfies the Büchi acceptance condition.

To simplify the proof for the other direction, we will work with a smaller, equivalent logic that has only second-order variables. First-order variables are simulated with singleton sets.

$$\begin{array}{rcl} \text{MSOL}_0[\mathsf{S}] \text{ formulas } \phi, \psi & := & \mathsf{S}(X,Y) \mid X \subseteq Y \mid X \subseteq Q_a \\ & & \mid \neg\psi \mid \psi \wedge \phi \\ & & \mid \exists X.\psi \end{array}$$

I will show how to translate between $\text{MSOL}_0[\mathsf{S}]$ and $\text{MSOL}[\mathsf{S}]$. We can then conclude that $\text{MSOL}_0[\mathsf{S}]$ defines the same languages as $\text{MSOL}[\mathsf{S}]$.

First we do the translation from $\text{MSOL}[\mathsf{S}]$ to $\text{MSOL}_0[\mathsf{S}]$. The usual conditions on variable binding must hold, but here we just informally assume that we have an injective function that assigns to each first-order variable x a second-order variable X_x. We first need to translate satisfiability statements in $\text{MSOL}[\mathsf{S}]$:

$$w, P_1, \ldots, P_m, p_1, \ldots, p_n \models \psi$$

to satisfiability statements in $\text{MSOL}_0[\mathsf{S}]$:

$$w, P_1, \ldots, P_m, \{p_1\}, \ldots, \{p_n\} \models [X_{m+1}/x_1] \cdots [X_{m+n}/x_n]\psi$$

So now the remainder of the algorithm only needs to work on formulas with no free first-order variables. For readability, let Sing be a predicate for testing whether a set is a singleton, and $\mathsf{AboveClosed}$ for whether a set contains all the numbers greater than an element of the set.

$$\begin{array}{lcl} \mathsf{Sing}(X) & = & \forall Y.\, Y \subseteq X \Rightarrow X \subseteq Y \\ \mathsf{AboveClosed}(X) & = & \forall x.\, \forall y.\, X(x) \wedge \mathsf{S}(x,y) \Rightarrow X(y) \end{array}$$

$$\begin{array}{rcl} (x < y) & \rightarrow & \neg(x = y) \wedge \Big(\forall X.\, [X(x) \wedge \mathsf{AboveClosed}(X)] \Rightarrow X(y)\Big) \\ \mathsf{S}(x,y) & \rightarrow & \mathsf{S}(X_x, X_y) \\ (x = y) & \rightarrow & X_x \subseteq X_y \wedge X_y \subseteq X_x \\ Q_a(y) & \rightarrow & \mathsf{Sing}(X_y) \wedge X_y \subseteq Q_a \\ X(y) & \rightarrow & \mathsf{Sing}(X_y) \wedge X_y \subseteq X \\ \psi \rightarrow \phi & \rightarrow & \neg\psi \vee \phi \\ \psi \vee \phi & \rightarrow & \neg(\neg\psi \wedge \neg\phi) \\ \forall x.\psi & \rightarrow & \neg\exists x.\, \neg\psi \\ \forall X.\psi & \rightarrow & \neg\exists X.\, \neg\psi \\ \exists x.\psi & \rightarrow & \exists X_x.\, \mathsf{Sing}(X_x) \wedge \psi \end{array}$$

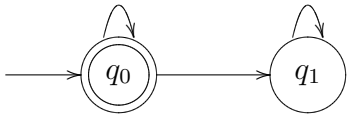
The other direction is simpler:

$$\begin{array}{rcl} \mathsf{S}(X,Y) & \rightarrow & \mathsf{Sing}(X) \wedge \mathsf{Sing}(Y) \wedge \exists x.\, \exists y.\, X(x) \wedge Y(y) \wedge \mathsf{S}(x,y). \\ X \subseteq Y & \rightarrow & \forall x. X(x) \Rightarrow Y(x) \\ X \subseteq Q_a & \rightarrow & \forall x. X(x) \Rightarrow Q_a(x) \end{array}$$

The second part of the theorem is proved by induction on the structure of formulas, so we need to take open formulas into account. For that purpose, we define a function encode that takes n subsets of $\mathbb{N}$ and a word over A to a word over $A \times \{0,1\}^n$. If $w' = \mathsf{encode}(w, P_1, \ldots, P_n)$ then $w'(i) = (w(i), b_1, \ldots, b_n)$ where $b_j = 1$ iff $i \in P_j$ and $b_j = 0$ otherwise. Thus, membership of i in the P_j is encoded in the i-th letter of the word.

We can now state the proposition: For every formula ψ with free variables $X_1, \ldots, X_n$ there exists a Büchi automaton $\mathcal{A}$ such that $w, P_1, \ldots, P_n \models \psi$ iff $\mathcal{A}$ accepts $\mathsf{encode}(w, P_1, \ldots, P_n)$.

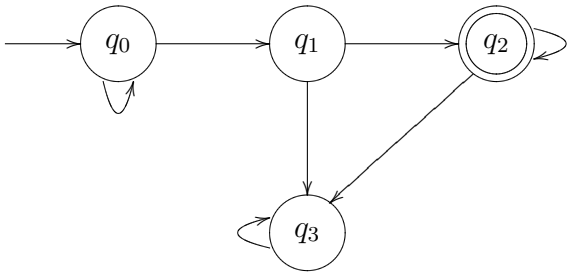
There are three base cases, for the three types of atomic formulas.
Suppose $\psi = X_i \subseteq X_j$. We construct an automaton with this shape:



At step k, you check whether k is a counterexample to $X_i \subseteq X_j$, i.e. you stay in q_0 upon reading $(a, b_1, \ldots, b_n)$ iff $b_i = b_j$ or $b_i = 0$, $b_j = 1$. Otherwise you transition to q_1. Once in q_1, you stay there, since a counterexample has been found. $(q_0, (a, b_1, \ldots, b_n), q_0) \in \Delta$ iff $b_i = b_j$ or $b_i = 0$, $b_j = 1$.
$(q_0, (a, b_1, \ldots, b_n), q_1) \in \Delta$ otherwise, i.e. iff $b_i = 1$, $b_j = 0$.
$(q_1, (a, b_1, \ldots, b_n), q_1) \in \Delta$ for all $a, b_1, \ldots, b_n$.

Suppose $\psi = X_i \subseteq Q_a$, i.e. if $k \in X_i$ then the k-th letter of the word is a. The shape of the automaton is the same as that used for $X_i \subseteq X_j$. You stay in q_0 upon reading $(a', b_1, \ldots, b_n)$ iff $b_i = 1$ implies $a' = a$. Otherwise you transition to q_1 and stay there.
$(q_0, (a', b_1, \ldots, b_n), q_0) \in \Delta$ iff $b_i = 1$ implies $a' = a$.
$(q_0, (a', b_1, \ldots, b_n), q_1) \in \Delta$ otherwise, i.e. iff $b_i = 1$ and $a \neq a'$.
$(q_1, (a', b_1, \ldots, b_n), q_1) \in \Delta$ for all $a', b_1, \ldots, b_n$.

Suppose $\psi = \mathsf{S}(X_i, X_j)$. We construct an automaton with this shape:



Recall that if X_i occurs in $\mathsf{S}(X_i, X_j)$ then $X_i = \{k\}$ for some $k \in \mathbb{N}$. You stay in q_0 until step k, and then in the next step, from q_1, you verify that $X_j = \{k+1\}$. Then from q_2, you forever check that nothing else is in X_i or X_j.
$(q_0, (a, b_1, \ldots, b_n), q_1) \in \Delta$ iff $b_i = 1$ and $b_j = 0$.
$(q_0, (a, b_1, \ldots, b_n), q_0) \in \Delta$ iff $b_i = b_j$ or $b_i = 0, b_j = 1$.
$(q_1, (a, b_1, \ldots, b_n), q_2) \in \Delta$ iff $b_i = 0$ and $b_j = 1$.
$(q_1, (a, b_1, \ldots, b_n), q_3) \in \Delta$ iff $b_i = b_j$ or $b_i = 1, b_j = 0$.
$(q_2, (a, b_1, \ldots, b_n), q_2) \in \Delta$ iff $b_i = b_j = 0$.
$(q_2, (a, b_1, \ldots, b_n), q_3) \in \Delta$ iff $b_i = 1$ or $b_j = 1$.
$(q_2, (a, b_1, \ldots, b_n), q_3) \in \Delta$ for all $a, b_1, \ldots, b_n$.

Suppose $\psi = \psi_1 \wedge \psi_2$. By the induction hypothesis there are Büchi automata that recognize the languages defined by ψ_1 and ψ_2. Since the language defined by $\psi_1 \wedge \psi_2$ is just the intersection of the languages defined by ψ_1 and ψ_2, this case amounts to proving that the set of languages recognizable by Büchi automata is closed under intersection.

Suppose $\psi = \neg\psi'$. By the induction hypothesis there is a Büchi automaton hat recognizes the language defined by ψ'. Since the language defined by $\neg\psi'$ is just the complement of the language defined by ψ', this case amounts to proving that the set of languages recognizable by Büchi automata is closed under complementation.

Suppose $\psi = \exists X_n.\, \psi'$, and ψ' may have free variables $X_1, \ldots, X_n$. By the induction hypothesis there is an $\mathcal{A}' = \langle Q, A, q_0, \Delta', F\rangle$ that accepts $\mathsf{encode}(w, P_1, \ldots, P_n)$ iff $w, P_1, \ldots, P_n \models \psi'$. We use $\mathcal{A}'$ to construct an $\mathcal{A}$ that accepts $\mathsf{encode}(w, P_1, \ldots, P_{n-1})$ iff $w, P_1, \ldots, P_{n-1} \models \exists X_n.\, \psi'$. Equivalently, $\mathcal{A}$ accepts $\mathsf{encode}(w, P_1, \ldots, P_{n-1})$ iff $w, P_1, \ldots, P_n \models \psi'$ for *some* P_n. It suffices to use $\mathcal{A} = \langle Q, A, q_0, \Delta, F\rangle$, where $(q_i, (a, b_1, \ldots, b_{n-1}), q_j) \in \Delta$ iff $(q_i, (a, b_1, \ldots, b_{n-1}, 1), q_j) \in \Delta'$ or $(q_i, (a, b_1, \ldots, b_{n-1}, 0), q_j) \in \Delta'$.

□

Other definability results

It would be a shame to develop enough of monadic second order logic to understand Büchi's theorem, and not even mention some other interesting results. These results are for logics over finite words, and so are not related to Büchi automata in any obvious way. They are here because the definitions in this report can be used to help explain them.

Every instance of the formula schema given in the first part of the proof of Büchi's theorem is a formula of *existential monadic second order logic with successor* (EMSOL[S]), which is the fragment of monadic second-order logic with the formulas $\exists X_1 \ldots \exists X_k . \psi$, where ψ is a formula of FOL[S]. But we showed that *any* MSOL[S] formula has an equivalent Büchi automaton, and thus any MSOL[S] formula has an equivalent EMSOL[S] formula. There is a result very similar to Büchi's theorem that establishes an equivalence between monadic second-order logic and finite automata on *finite* words, and the proof similarly leads us to the conclusion that for finite words also, every MSOL[S] formula has an equivalent EMSOL[S] formula.

Now, consider EMSOL[S] but where quantification over second-order predicates of higher arity is allowed. Fagin proved that the resulting logic, called (general) existential second-order logic, defines exactly the languages in the complexity class NP[3] [4].

General existential second-order logic is much more expressive than EMSOL[S]. Consider the following smaller alteration of EMSOL[S]: the successor predicate is dropped, and quantification over second-order predicates is restricted to binary relations on $\{1, \ldots, n\}^2$ (where n is the length of the word) that are *matchings*. R is a matching on $\{1, \ldots, n\}^2$ if the following conditions hold:

1. Each $i \in \{1, \ldots, n\}$ occurs in at most one pair in R
2. If $(i, j) \in R$ then $i < j$.
3. If (i, j) and (k, l) are in R and $i < k < j$ then $i < k < l < j$.

Lautemann, Schwentick and Thérien proved that the resulting logic defines the context-free languages [2].

[4] Thomas suggests *Finite Model Theory* by Ebbinghaus and Flum for details on this result and other results of *descriptive complexity theory* (including logics that characterize P, PSPACE, NLOGSPACE, etc.)

References

[1] J.R. Büchi. On a decision method in restricted second-order arithmetic. In *1960 Int. Congr. for Logic, Methodology and Philosophy of Science*, pages 1–11, Stanford, 1962. SIAM-AMS, Stanford university press.

[2] D. Therien C. Lautemann, Th. Schwentick. Logics for context-free languages. In J. Tiuryn L. Pacholski, editor, *Computer Science Logic*, volume 933 of *Lecture Notes in Computer Science*, pages 205–216. 1995.

[3] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R.M. Karp, editor, *Complexity of computation*, volume 7, pages 43–73. SIAM-AMS, 1974.

[4] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.

[5] P. Panangaden. Shown in comp 525 lecture. October 2007.

[6] W. Thomas. Languages, automata, and logic. Technical Report 9607, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel, Germany, 1996.