ω-Automata and corresponding Logics

Swen Jacobs
Seminar: Logical Aspects of XML
Saarland University
Programming Systems Lab

Abstract

We present to the reader the correspondence between Büchi automata on infinite words and S1S, the monadic second-order logic on infinite strings. To reach this goal, we show that Büchi automata are closed under boolean operations such as union, intersection and complementation. Special emphasis will be on the complementation proof, which is the most difficult step in this work. Also, the according steps for the correspondence between Muller automata on infinite trees and S2S are mentioned, but not carried out.

1 Introduction

In this work, we first want to provide the reader with essential knowledge about ω-automata, which are essentially finite automata over input words or trees. The main difference to normal and more or less widely known finite word and tree automata is that in case of ω-automata the input word or tree is infinite. To clarify things, we will only use the term ω-automata if we mean both types. If we specifically talk about automata on infinite strings, we call them ω-word-automata, if we talk about automata on infinite trees, we use the term ω-tree-automata.

We view these automata with respect to the logics they decide, which will turn out to be the monadic second order logic (MSO) with one or two successors (S1S or S2S), respectively. We give a short introduction to these logics. We will also present the theorems leading to this conclusion. Having seen the correspondence between finite automata and WS1S as well as between finite tree automata and WS2S in [Kerber03], the proofs are trivial, except for one thing: In order to do these proofs, we need to show the closure of ω-automata under boolean operations. For the infinite word case, we will show closure of Büchi-word-automata, which are a special case of ω-word-automata. For the infinite tree case, we will only give the proof ideas.

2 ω-word-automata

When we discuss ω-word-automata, our basis clearly are finite automata. However, since our input words are no longer finite, we need a different acceptance condition. While the acceptance condition for automata over finite strings is rather canonical, there are several ways to define acceptance of a run which includes infinitely many visits to finitely many states. One could either define an acceptance condition on the states which are or are not visited on the run (which is called *Staiger-Wagner acceptance*, introduced in [SW74]), or one could restrict the acceptance condition to view only those states which are visited infinitely often in a run. In the following, we will use the latter approach and introduce two acceptance conditions working in this way.

Definition 1 (Finite ω-word-automaton). A *finite ω-word-automaton* is a tuple $\mathcal{A} = (Q, A, q_0, \Delta, Acc)$, where Q is a finite set of states, A an input alphabet, q_0 an initial state, $\Delta \subseteq Q \times A \times Q$ a transition relation, and Acc an acceptance component. A run of automaton $\mathcal{A}$ on a given input ω-word $\alpha = \alpha(0)\alpha(1)...$ with $\alpha(i) \in A$ is a sequence $\rho = \rho(0)\rho(1)... \in Q^\omega$ such that $\rho(0) = q_0$ and $(\rho(i), \alpha(i), \rho(i+1)) \in \Delta$ for $i \geq 0$. If the automaton is *deterministic*, the transition relation is replaced by a function $\delta : Q \times A \rightarrow Q$, and a run then has to satisfy $\rho(i+1) = \delta(\rho(i), \alpha(i))$ for $i \geq 0$.

To obtain a functional automaton, we still need to specify the acceptance component. Let $\exists^\omega$ be the quantifier for "there exist infinitely many" and consider the set $In(\rho) = \{q \in Q | \exists^\omega\ i : \rho(i) = q\}$, the *infinity set* of run ρ.

Now we can define two important acceptance conditions, which are requirements on the infinity set of a run ρ:

- Büchi condition[Büchi62]: $In(\rho) \cap F \neq \emptyset$ for a set $F \subseteq Q$ of *final states*. This condition requires that at least one of the final states occurs infinitely often in the run ρ.

- Muller condition[Muller63]: $In(\rho) \in \mathcal{F}$, for $\mathcal{F} \subseteq 2^Q$, a set of accepting subsets of Q. This condition requires that exactly the infinity set of the run is specified as an accepting set.

There are other acceptance conditions for ω-automata, like the *Rabin, Rabin chain* or *Streett* condition. However, because we don't need them for our proofs we omit their definitions.

In accordance to the acceptance condition, in the following we will speak of a *Büchi-word-automaton* if we mean an ω-word-automaton with Büchi acceptance condition (and similarly for the Muller condition). Even though it is straightforward, let us define acceptance of a word by a Büchi- or Muller-word-automaton:

Definition 2 (Acceptance of Büchi-word-automata). An ω-word α is accepted by a Büchi-word-automaton A, if there exists a run ρ of A on α such that $In(\rho) \cap F \neq \emptyset$.

Definition 3 (Acceptance of Muller-word-automata). An ω-word α is accepted by a Muller-word-automaton A, if there exists a run ρ of A on α such that $In(\rho) \in \mathcal{F}$.

After defining different acceptance conditions for the same kind of automaton, one question arises immediately: Which one is more powerful?

Proposition 1. *Nondeterministic Büchi- and Muller-word-automata recognize the same class of ω-languages.*

Proof. It is straightforward that the more general Muller condition can simulate the Büchi condition: for a given set of final states F, let the set of accepting sets $\mathcal{F}$ contain every subset S of Q such that at least one element $s \in S$ is from F. The other direction, simulating Muller by Büchi, is not that easy and only works when allowing nondeterminism: To simulate a Muller-word-automaton, the Büchi-word-automaton at first needs to nondeterministically chose the subset $S \in \mathcal{F}$ which will be equal to the infinity set of the run. Furthermore, it needs to nondeterministically chose the point of time in the run from which on only states from S occur[1]. Finally, it has to be tested that every state from S is indeed visited infinitely often, which is done by a subset construction. As one can easily see, this simulation is very expensive, while the other direction works on the same state set with only different acceptance condition. □

Let's get back to the statement that we need nondeterminism when simulating Muller by Büchi:

Proposition 2. *Deterministic Büchi-word-automata recognize less languages than nondeterministic Büchi-word-automata.*

Proof. It is easy to show that there are ω-languages which cannot be recognized by deterministic Büchi-word-automata. One example is the language over $A = \{a, b\}$ with only finitely many a's in every ω-word. It is clearly recognized by Muller-word-automata (and thus also by nondeterministic Büchi-word-automata), but not by deterministic Büchi-word-automata: Suppose there would be a Büchi-word-automaton accepting the language. Then it would accept the ω-word b^ω, i.e. on the corresponding run there are infinitely many accepting states. We chose one of them, say it appears after reading b^{i_0}. Then the automaton must also accept $b^{i_0}ab^\omega$, i.e. we find another accepting state at $b^{i_0}ab^{i_1}$. In this way we can construct a word $b^{i_0}ab^{i_1}ab^{i_2}a...$ with infinitely many a's which is nonetheless accepted by the automaton, thus obtaining a contradiction. □

[1]One can easily confirm that such a point must exist: If for some state $q \notin In(\rho)$ there would not be a point after which it does not occur anymore, then it would need to occur infinitely often, which is a contradiction.

3 Completeness of ω-word-automata

In order to show the correspondence between ω-word-automata and S1S, we need to show the closure of the automata under the following operations:

1. union
2. intersection
3. projection
4. cylindrification
5. complement

We prove closure for Büchi-word-automata, as far as possible:

1. Union:
Given two automata $\mathcal{A}_1 = (Q_1, A_1, q_1, \Delta_1, F_1)$ and $\mathcal{A}_2 = (Q_2, A_2, q_2, \Delta_2, F_2)$ with respective languages $L(\mathcal{A}_1) = \mathcal{L}_1$ and $L(\mathcal{A}_2) = \mathcal{L}_2$, the automaton for $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$ is constructed by building the union of all respective sets, i.e.

$$\mathcal{A} = (Q_1 \cup Q_2, A_1 \cup A_2, q_0, \Delta_1 \cup \Delta_2 \cup \{(q_0, \epsilon, q_1), (q_0, \epsilon, q_2)\}, F_1 \cup F_2),$$

and q_0 is a new initial state with ϵ-transitions to the old initial states q_1 and q_2.

2. Intersection:
Given two automata as above, the automaton for the language $\mathcal{L} = \mathcal{L}_1 \cap \mathcal{L}_2$ is constructed with the help of a double product automaton plus an indice. That means, we first construct the standard product automaton (as for finite automata) for the given two automata. We copy the resulting automaton and mark the tuple-states from the original product automaton with a 1, and those of the copy with a 2. Thus, in the resulting automaton, we have tuple states of the form (q_a, q_b, i), where $q_a \in Q_1$, $q_b \in Q_2$ and $i \in \{1, 2\}$. The construction works as follows: we start in $(q_1, q_2, 1)$, i.e. in the initial state of the original product automaton. When we reach a final state from $\mathcal{A}_1$, i.e. a tuple state where the first element is from F_1, with the next transition we jump into the copy of the product automaton (and into the according state of the computation, of course). Here we continue the computation until we reach a final state from $\mathcal{A}_2$, and then jump back into the original product automaton. As accepting states we only need the accepting states of $\mathcal{A}_1$ in the original product automaton, i.e. all states $(q_a, q_b, 1)$, where $q_a \in F_1$ and $q_b \in Q_2$. This construction works because if we visit any of these final states infinitely often, we must also visit some accepting state from $\mathcal{A}_2$ in the copy of the product automaton infinitely often (otherwise we would not be allowed to return to the original).

3. Projection: Given an automaton over a tuple alphabet $A = A_1 \times A_2$, the automaton for the first projection is constructed easily: We may keep the states, initial state and accepting states of the original automaton, our new alphabet is of course only A_1. For every transition $(q, (a_1, a_2), q') \in \Delta$ in the original automaton, there is a transition (q, a_1, q') in the new automaton.

4. Cylindrification: Cylindrification is just as easy: we keep the states as above, and for any transition $(q, (a_1), q') \in \Delta$ of the original automaton, we have all possible transitions $(q, (a_1, a_2), q')$ with any a_2 from the new alphabet.

5. Complement: We have already seen that complementation is the most difficult standard operation for finite automata, word and tree alike. The infinite word case is no exception. In fact, complementation of ω-word-automata is such a challenging task, that we will do it in a section of its own.

4 Determinisation and Complementation

Complementation of ω-word-automata is a big problem. The reason for this is that the known complementation algorithms need deterministic automata, but we have already seen that deterministic Büchi-word-automata are not as expressive as nondeterministic ones. That means, for a given nondeterministic Büchi-word-automaton, there may not even be a deterministic automaton, and thus no way to build the complementary automaton in a relatively easy way. Nonetheless, complementation of Büchi-word-automata is possible:

Theorem 1 (McNaughton's Theorem [McNaughton66]). *A Büchi-word-automaton can be transformed effectively into an equivalent deterministic Muller-word-automaton.*

This theorem is the key to our complementation problem, because a deterministic Muller-word-automaton can be easily complemented: we only need to switch accepting and non-accepting sets of states. After that complementation, we may go back to a Büchi-word-automaton, as we have already seen (losing the determinism, however).

Complementation of Büchi automata has been proved in several different ways: Büchi himself already proved that "his" automata were closed under complement in [Büchi62]. McNaughton's theorem of course has been proved by McNaughton, and his proof was then sharpened by the widely-known proof of Safra in [Safra88] which uses "Safra trees" to construct macro-states similar to the subset construction for finite automata. Even after Safra, there have been new proofs for complementation of Büchi automata: Wolfgang Thomas developed a new proof in [Thomas2], which involves transforming the nondeterministic Büchi automaton into a *weak alternating automaton*, and goes on to use the determinism of weak infinite games to complete the proof.

In the following, we will give an entire look on Safra's proof. To this end, we will first explain Safra's construction itself, then give an example of how it is applied to a nondeterministic Büchi automaton, go on to show why Safra's construction is correct and finally prove McNaughton's theorem.

4.1 Safra's construction[Safra88]

Before we explain Safra's construction, let us revisit the subset construction known from finite automata and show why it is not sufficient in the infinite word case:

Suppose we have a Büchi automaton with three states q_0, q_1 and q_2 such that a transition from q_0 to q_1 is always possible (plus transitions to stay in q_0), but from q_1 all transitions directly go on to q_2. Furthermore, there is no transition leaving q_2 and q_1 is the only accepting state. Clearly, no ω-word can be accepted by such an automaton, because the final state can only be visited once.

Now take a look at the subset construction: from the macrostate $\{q_0\}$, we move on to $\{q_0, q_1\}$, and from there to $\{q_0, q_1, q_2\}$, where we stay forever. Now every macrostate which includes a final state is a final state in the subset construction, so instead of accepting no ω-words, our subset construction would accept all ω-words.

The reason for this lies in the way the subset construction is supposed to work: essentially, for every finite word, it computes all possible states the automaton can reach with this word. If this set of reachable states includes at least one final state of a finite automaton, we know that there is an accepting run, i.e. the automaton accepts the given word. This is not enough for ω-word-automata, however: we not only need to reach a final state once, but infinitely often. Thus, an according construction for ω-word-automata must not only check if a final state is reachable, but it must find out if there is a loop such that a final state can be visited infinitely often. As we will see, this is exactly what Safra's construction does:

Essentially, Safra's construction *is* a subset construction, with one new concept: whenever the computed macrostate includes final states, an own thread of macrostates is split off, represented by a child node in the Safra tree. Those Safra trees will be the states of the resulting deterministic automaton.

Definition 4 (Safra Tree). A *Safra tree* over a finite non-empty set of states Q with $|Q| = n$ is a finite, ordered tree. The nodes of the tree have names from $\{1, ..., 2n\}$ and labels from $2^Q \setminus \{\emptyset\}$. Furthermore, every node may be marked or unmarked. The label of every node in the tree is a proper superset of the union of labels of its children. Labels of nodes with the same parent are pairwise disjoint.

As this definition only captures Safra trees, but not their development, we still need to define the algorithm which will be used for determinization of Büchi-word-automata. Let us first discuss it informally:

The root node of the given Safra tree will always contain the momentary reachable states, just like the original subset construction. Now, if during the computation this set of states contains any final states, a child node is introduced, containing exactly those final states. After that, the subset construction is applied both to the parent and the child node separately. If in the following we get to another macrostate where the root contains final states, a new child is split off, containing those states. The same is done if any of the child states should contain final states. However, we need to ensure that our tree remains within the boundaries of the Safra tree definition, i.e. the label of a parent node must always be a proper superset of the union of labels of its children, and the labels of those children have to be disjoint. The second property is ensured by a horizontal merge, i.e. if some state is included in more than one brother-node, it is deleted from all nodes except the oldest. This may result in empty labels for some nodes, in which case the whole node is deleted. The first property is in some sense the center of Safra's construction: If the union of the labels of child nodes equals the label of the parent node, all child nodes are deleted and the parent node is marked[2]. With this, we come to the end of the construction: the acceptance condition. Every Safra tree we created represents one state in a new (deterministic) Muller-word-automaton. The set of accepting sets $\mathcal{F}$ contains a set of Safra trees if there is a node with the same name in every tree and this node is marked in at least one of those trees.

Now, let us revisit the construction and formulate an algorithm which develops Safra's construction:

Algorithm 1 (Safra's Construction). Given a Büchi-word-automaton $\mathcal{A} = (Q, A, q_0, \Delta, F)$, we build a deterministic Muller automaton in the following way:

1. Start with the Safra tree consisting of only one node named "1" and labeled with the set $\{q_0\}$, consisting only of the initial state.

   From here on, repeat until no more trees are added:

2. All nodes are unmarked.

3. If any of the nodes of the current tree contains final states, a new youngest child node is added with a free name[3] and labeled with this set of final states.

4. Apply the subset construction on the label of every node of the tree

5. For all brother nodes: if some brother nodes contain the same state, remove it from all but the oldest brother

6. If there are nodes labeled with the empty set, remove them

7. If there are nodes where the union of child-labels equals the parent label, mark the parent node and remove all descendants

   end repeat.

Let us take a look at the marked states: *what does it mean if some state is marked?* Suppose from some set of states Q_1 the automaton reaches Q_2 after

[2]As one can easily confirm, there cannot be states in labels of child nodes which are not in the parent node

[3]usually this would be the lowest number which is not currently used in the tree

reading input u_1, with a set of final states $F_2 \subseteq Q_2$. Suppose further that after reading input v_1, the automaton goes on and reaches the same set of states from both F_2 and Q_2, thus the node with Q_2 in the Safra tree would be marked. As one can easily see, this mark means that for any state q_2 in the marked node, there is some state $q_1 \in Q_1$ such that the automaton can reach q_2 from q_1 when reading u_1v_1 and pass through F_1 in the process. The same argument holds if during input v_1, more macrostates are split off.

If we continue this scheme, we can see how acceptance of the old automaton relates to acceptance of the constructed one: If we can find an ω-word $\omega_1\omega_2$... such that after each input ω_i some marked macrostate is reached, then both of our automata are accepting. For the constructed automaton this is clear, since the run on this ω-word passes through the same node, which in the process is marked not only once, but infinitely often. For the original Büchi-word-automaton acceptance is easily poved: for each sequence ω_1, $\omega_1\omega_2$, ... pick some finite run of the automaton which passes through F at least i times (as seen above). These finite runs then form an infinite tree which is finitely branching. By applying König's Lemma, we yield an infinte run with infinitely many visits to F.

Thus, we can use Safra trees with marked nodes to detect successful runs of the Büchi-word-automaton. In the following section we want to use this fact to prove McNaughton's theorem.

4.2 Proof of McNaughton's Theorem

Given a Büchi-word-automaton $\mathcal{A} = (Q, A, q_0, \Delta, F)$, the desired deterministic Muller-word-automaton $\mathcal{B}$ is constructed as follows: Use Safra's construction (Algorithm 1) to define state set and transitions of $\mathcal{B}$. Initial state of $\mathcal{B}$ is always q_0, the alphabet is of course the same as for $\mathcal{A}$. We define acceptance of $\mathcal{B}$ as follows: a set S of Safra trees, i.e. states of $\mathcal{B}$, is in $\mathcal{F}$ if some node k is in every tree in S, and k is marked at least once.

What we need to show is that $L(\mathcal{A}) = L(\mathcal{B})$:

- $L(\mathcal{A}) \supseteq L(\mathcal{B})$

  If α is an ω-word such that $\alpha \in L(\mathcal{B})$, then there is a run of $\mathcal{B}$ on α, such that some node k is contained in all Safra trees from the infinity set, and marked in at least one of those trees. By the argument from the last section, α can be split into infinitely many input words such that after each input word, a marked node is reached. Thus, there is a run of $\mathcal{A}$ on α with infinitely many visits to F. Hence $\mathcal{A}$ accepts α.

- $L(\mathcal{A}) \subseteq L(\mathcal{B})$

  Suppose $\mathcal{A}$ accepts α, say by a run ρ which passes through $q \in F$ infinitely often. Consider the run of Safra trees of $\mathcal{B}$ on α. The root macrostate of each Safra tree in this run must be nonempty, since it contains at least $\rho(i)$. If the root is marked infinitely often, $\mathcal{B}$ accepts and we are done. Otherwise, after the last time the root has been marked, a final state

$q \in F$ will be reached at some later point of the run ρ (since there are infinitely many final states in ρ), and thus be put into a son macrostate of the root. From this point on, the states from the run ρ will appear in the label of this son, or get associated to some older brother of this node by a horizontal merge operation (step 5 in the algorithm). However, such a merge can only happen a finite number of times – from then on, the states of ρ will be associated to some fixed son of the root. This son cannot be deleted anymore because we know that the root has already been marked for the last time. Now we continue as above: if this son of the root is marked infinitely often, we are done. Otherwise, proceed as above, continuing the computation with some son of this son. Because the depth of a Safra tree is bounded by $|Q| - 1$, at some time a node must be found which is marked infinitely often. Thus $\mathcal{B}$ accepts and we are done. $\square$

5 Correspondence between ω-Word-Automata and S1S

Now that we have poved closure of Büchi-word-automata, together with what we have already seen in [Kerber03], it is fairly easy to show that ω-word-automata correspond to S1S, i.e. are a possibility to decide S1S. The proof works exactly the same way, except that we may have infinite sets in our S1S-formulas, which corresponds to the fact that we have automata over infinite words.

However, with these results at hand, another result becomes apparent:

Proposition 3. *Over ω-words, any S1S-formula is equivalent to a WS1S-formula.*

The proof idea is to express every S1S-formula in such a way that infinite sets are declared indirectly, i.e. by specific properties which only an infinite set may have. When we return to automata, this means that we do not need ω-word-automata to decide S1S: instead of testing an input word with an ω-automaton, we can test each prefix of the word with a finite automaton. Only if infinitely many prefixes are accepted, the ω-word is accepted.

6 ω-Tree-Automata and S2S

For the case of automata on infinite trees, there are similar results available as for the infinite word case. Resembling what we have already shown, one can extend the proof from [Kerber03] to the infinite tree case, once closure of ω-tree-automata under boolean operations has been proved. This is done fairly easy, again with the exception of the complement. Complementation of ω-tree-automata is an even bigger problem than for ω-word-automata, and to our knowledge there is no constructive proof available. To make the proof understandable, a game-theoretic view of automata may help. Such proofs can be seen e.g. in [Thomas96] or [Finkbeiner02]. However, originally this is already a result of Rabin, as in [Rabin69].

The main results of these proofs are the following:

Theorem 2. *For any S1S-formula ϕ one can construct effectively a ω-tree-automaton $\mathcal{A}$ such that a tree t is accepted by $\mathcal{A}$ iff t satisfies ϕ.*

Theorem 3 (Rabin Tree Theorem). *The theory S2S is decidable.*

In contrast to S1S, there is no result stating that S2S-formulas over ω-trees may be substituted by WS2S-formulas.

7 Conclusion

We have shown that for any S1S-formula, an equivalent ω-word-automaton can be constructed, thus showing decidability of S1S. To this end, we have proved closure of Büchi-word-automata and thus ω-word-automata under boolean operations, especially complementation. This most difficult step was shown resembling the proof of Safra, including Safra's construction for determinization of Büchi-word-automata.

For S2S and ω-tree-automata, there are similar results available, although an entire look at the proofs was out of the scope of this work.

References

[Finkbeiner02] B. Finkbeiner. Lecture: Automata, Games & Verification, WS 2002/03.

[Thomas96] W. Thomas. Languages, Automata, and Logic. Technical Report for Univiversity of Kiel, 1969, pp. 28-61.

[Kerber03] J. Kerber. The Correspondence between Automata and Logics. Seminar: Logic Aspects of XML, 2003.

[Büchi62] J.R. Büchi. On a decision method in restricted second-order arithmetic. Proc. 1960 Int. Congr. For Logic, Methodology and Philosophy of Science. Stanford Universal Press, Stanford 1962, pp.1-11.

[Muller63] D.E. Muller. Infinite sequences and finite machines. Proc. 4th IEEE Symp. On switching Circuit Theory and Logic Design, 1963, pp. 3-16.

[Rabin69] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. Trans. Amer. Math. Soc. 141, 1969, pp. 1-35.

[Rabin72] M.O. Rabin. Automata on infinite objects and Churchs problem. Amer. Math. Soc., Providence, RI, 1972.

[Safra88] S. Safra. On the complexity of ω-automata. Proc. 29th IEEE Symp. on Foundations of Computer Science, 1988, pp. 319-327.

[McNaughton66] R. McNaughton. Testing and generating infinite sequences by a finite automaton. Inform. Contr. 9, 1966, pp.521-530.

[SW74] L. Staiger and K. Wagner. Authomatentheoretische und automatenfreie Charakterisierungen topologischer Klassen regulrer Folgemengen. Elektron. Informationsverarbeitung und Kybernetik, EIK 10, 1974, pp.379-392.

[Thomas2] W. Thomas. Complementation of Bchi Automata revisited. J. Karhumki et al., Eds., Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa, Springer-Verlag, pp. 109-122.