M1 MPRI : Automates et Applications

Cours 4
Inifite word automata and temporal logics

kn@lri.fr

February 6, 2024

Outline

Let’s come back to words

After considering words, we have added more structure by considering trees.

However, some systems are “infinite” by nature. For instance, a network server that receives connections can be modelled by a system that loops infinitely, receiving requests and sending out responses.
For such systems, we would like to ensure that a property holds infinitely many times, or that a property becomes true as soon as another becomes false.

Infinite words

Definition

Let Σ be an alphabet. An infinite word over Σ is a total function from $\mathbb{N}_1 \to \Sigma$.

Recall : $\mathbb{N}_1$ is the set of strictly positive integers. Previously, a word was a function $\{1, \ldots, n\} \to \Sigma$, for a fixed n.

Set of inifinite words

Definition

We write Σ^ω the set of infinite words Σ. We write Σ^∞ the set of finite or infinite words (therefore, $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$).

Warning, do not mix up:

- a finite language of infinite words
- a finite language of finite words

Examples

Let $\Sigma = \{a, b\}$:

- $\{a, b, ab\}$ finite set of finite words
- $a^*b = \{b, ab, aab, aaab, \dots\}$ is an infinite set of finite words
- $\left\{ \begin{array}{l} \mathbb{N}_1 \to \Sigma \\ n \mapsto a \end{array}, \begin{array}{l} \mathbb{N}_1 \to \Sigma \\ n \mapsto b \end{array} \right\}$ is a finite set of infinite words
- $\left\{ \begin{array}{l} \mathbb{N}_1 \to \Sigma \\ n \mapsto a, n < k \\ n \mapsto b, n \geq k \end{array} \middle| \, k \in \mathbb{N}_1 \right\}$ is an infinite set of infinite words

Side remark

The symbol ω denotes the smallest *infinite ordinal*. Ordinal numbers are a generalization of natural number. If we take a set-theoretic definition of natural numbers:

- $0 \equiv \varnothing$
- $1 \equiv \{0\} \equiv \{\varnothing\}$
- $2 \equiv \{0, 1\} \equiv \{\varnothing, \{\varnothing\}\}$
- $3 \equiv \{0, 1, 2\} \equiv \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}$
- $4 \equiv \{0, 1, 2, 3\} \equiv \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}, \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}\}$
- $\ldots$
- $\omega \equiv \mathbb{N} \equiv \{\varnothing, \ldots\}$: first infinite set
- $\omega + 1 \equiv \{0, 1, \ldots, \omega\}$

Side remark (2)

- $\omega + 2 \equiv \{0, 1, \ldots, \omega, \omega + 1\}$
- $\ldots$
- $\omega + \omega \equiv \omega 2$
- $\ldots$
- ω^2
- $\ldots$
- ω^ω
- $\ldots$
- $\left.\omega^{\omega^{\omega^{\cdots^{\omega}}}}\right\}\omega \text{ times} \equiv \epsilon_0$

Side remark (3)

- one can define arithmetic operations on such objects
- one can go beyond ϵ_0 to reach uncountable ordinals
- one can reason using transfinite induction, which is a recursion principle on ordinals
- when needed, we might use ω, but only as a notation
- a close but distinct concepts is the one of cardinal number (which represents the size of sets and allows one to say that the size of $\mathbb{N}$ is “smaller” than the size of $\mathbb{R}$)
- Georg Cantor (1845-1918) lost his sanity by discovering that

Outline

ω power

Definition (ω power of a language)

Let $L \subseteq \Sigma^*$ be a finite words language. We define L^ω with:

$$L^\omega = \{\sigma = u_1 u_2 \dots u_n \dots \mid \forall i \geq 1, u_i \in L \setminus \{\epsilon\}\}$$

In other words, it's the set of all *infinite* words that can be built by concatenating infinitely many non-empty words from L.

ω-regular languages

The notion of *regular language* can be extended to infinite words:

Definition (Langage ω-régulier)

A language $L \subseteq \Sigma^\omega$ is ω-regular if and only if:

- *$L = A^\omega$ where A is a regular (finite words) language*
- *or $L = A \cdot B$ where A is regular and B is ω-regular*
- *or $L = A \cup B$ where both A and B are ω-regular*

ω-regular languages (2)

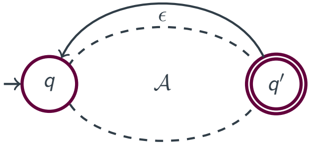
The only way to construct on ω-regular language is to take the ω power of a regular language. Then, one may add to its infinite words a finite prefixes (described by a regular language) or take the union of two ω-regular languages.

Recognizable languages of infinite words

ω-regular languages are build from regular languages by concatenating infinitely any words. Can we use automata to characterize such languages?

Recognizable languages of infinite words

ω-regular languages are build from regular languages by concatenating infinitely any words. Can we use automata to characterize such languages?



If an automaton performes an "infinite loop" through q', it will recognize a word of $L_{\mathcal{A}}^{\omega}$.

Outline

Automaton

Definition (Nonterministic finite automaton)

A non deterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, I, F)$ *where:*

- Q *is a set of states*
- Σ *is an alphabet*
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ *is a transition function*
- $I \subseteq Q$ *is a set of initial states*
- $F \subseteq Q$ *is a set of accepting states*

(nothing new here, that's the definition from the first lecture)

Infinite run

We extend the notion of a *run* to infinite words:

Definition

Let $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ be an automaton and u a word of length $\underline{n} \leq \omega$. We call run of $\mathcal{A}$ for u a function $r : \underline{n} \to Q$ such that:

- $r(0) \in I$
- $\forall i \in \underline{n}, r(i+1) \in \delta(r(i), u(i+1))$

Recall: ordinal numbers (including natural numbers) are both "numbers" and sets of sets. We write $\underline{n}$ to say that we consider n (an number or ω) as the set that contains all its predecessors.

States reached by a run

Definition (States reached by a run)

Let r be a run of $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ for the word u :

if $|u| = \underline{n} < \omega$: the set of reached states of r is $\{r(n)\}$

if $|u| = \omega$: the set of reached sates of r is

$$\{q \mid \#\{i \mid r(i) = q\} \notin \mathbb{N}\}$$

We write $\lim(r)$ the set of reached states of r

In the definition above, $\#S$ denotes the cardinal of S.

For infinite words, the set of reached states is the set set of states that appear infinitely many times in the run.

Büchi's acceptance condition

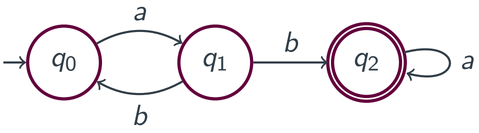
Definition (Accepting run)

Let $\mathcal{A} = (Q, \Sigma, \delta, I, F)$, be an automaton, $u \in \Sigma^{\infty}$ a word and r a run of $\mathcal{A}$ for u. We say that r is accepting if and only if $\lim(r) \cap F \neq \varnothing$.
We write $L_{\mathcal{A}}$ the set of finite words recognized by $\mathcal{A}$ and $L^{\omega}_{\mathcal{A}}$ the set of infinite words recognized by $\mathcal{A}$.

The condition that a run goes “infinitely many times through an accepting state” is called Büchi's acceptance condition. An infinite word langauge is recognized by such an condition is said to be Büchi-recognizable.
Julius Richard Büchi, 1924-1984, Swiss mathematician.

Example

The set L of infinite words starting with an finite sequence of ab followed by an infinite sequence of as is Büchi-recognizable:


$$\begin{tikzcd} \arrow[r] & q_0 \arrow[r, bend left, "a"] & q_1 \arrow[l, bend left, "b"] \arrow[r, "b"] & q_2 \arrow[loop right, "a"] \end{tikzcd}$$

Remark: This automaton is not special in any way. Its the same as the one recognizing $(ab)^+a^*$ for finite words. We just change the acceptance condition to account for infinite words.

Determinization ?

Since we haven't changed our automata (only the notion of accepting run), the notion of deterministic automaton is the same: An automaton is deterministic if and only if:

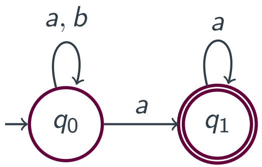
- I is a singleton
- $\forall q \in Q, x \in \Sigma,\ \delta(q, x)$ is a singleton.

Determinization ? (2)

Theorem

Non-deterministic Büchi automata recognize strictly more languages than deterministic Büchi automata.

Proof: Consider the language $(a|b)^* a^\omega$. It is recognized by the automaton:

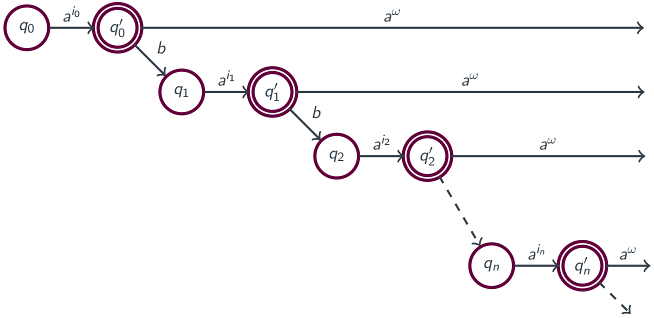

$$\begin{tikzcd} \arrow[r] & q_0 \arrow[loop above, "{a, b}"] \arrow[r, "a"] & q_1 \arrow[loop above, "a"] \end{tikzcd}$$

Determinization ? (3)

Assume there exists a *deterministic* automaton $\mathcal{A} = (Q, \{a, b\}, \delta, q_0, F)$ that recognizes the language. Call $n = |Q|$. Consider the infinite word $m_0 = a^\omega$. This word is in the language. Therefore, there is a run which goes infinitely many time through a state in F. Call q'_0 the first state of F reached by the run, after having read i_0 letters.

Consider now word $m_1 = a^{i_0} b a^\omega$. As the automaton is deterministic the run for m_1 and the one for m_0 are equal up to q'_0. Since m_1 is a word from the language q'_0 has a transition to a state q_1. Then from this state, the run eventually goes infinitely many times through a state $q'_1 \in F$.

Déterminisation ? (4)



The word $a^{i_0}ba^{i_1}ba^{i_2}b\ldots a^{i_{n+1}}ba^\omega$, goes through $n+1$ accepting states (states q'_i), however there are n states in total in the automaton. Therefore some of the q'_is are the same, and there is a cycle going from a state q'_i to a state q'_j. These states have a transition b. Thus, the automaton can recognize words with infinitely many b, which are not words of the language, contradiction.

Decision problem

Theorem

The emptiness of the language of a Büchi automaton is decidable in linear time.

Proof: it is sufficient to find a final state q_f :

- accessible from an initial state q_0
- accessible from itself

If such a state q_f exists, a run that goes infinitely many times through q_f exists, and the language is not empty. State accessibility in a graph.

Decision problems (2)

Theorem

Deciding whether the language of a Büchi automaton is universal is decidable in exponential time.

Note: We can construct the complement of even non-deterministic Büchi automata and test the emptiness. The construction causes an exponential blow up in the number of states. E.g. see:

The complementation problem for Büchi automata with applications to temporal logic, A. Sisla, M. Vardi, P. Wolper, TCS, 1987.

Closure properties

- Büchi automata are closed under union
- Büchi automata are closed under complement
- Büchi automata are closed under intersection

Naïve (and non working) product of automata

Consider the construction used for the finite word case. It does not work:

- $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, I_1, F_1)$.
- $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, I_2, F_2)$.

Assume that $\mathcal{A}_1$ recognizes words $a(ba)^\omega$ and $\mathcal{A}_2$ recognizes words $(ab)^\omega$. A run for $\mathcal{A}_1$ has the form $q_0 q_1 q_2 q_1 q_2 \ldots$ with q_2 accepting. A run for $\mathcal{A}_2$ as the form $p_0 p_1 p_0 p_1 \ldots$, with p_1 accepting. With a naïve product construction, we would have a run:
$(q_0, p_0)(q_1, p_1)(q_2, p_0)(q_1, p_1), \ldots$ with *no accepting states*. However, both languages are the same! (infinite sequences of ab starting with a).

Better product automaton

$\mathcal{A}_{1\wedge 2} = (Q_1 \times Q_2 \times \{1,2\}, \Sigma, \delta, I_1 \times I_2 \times \{1\}, F)$ avec :

- $\delta((q_1, q_2, i), a) = (q_1', q_2', f(q_1, q_2, i)),\ \ q_1' \in \delta_1(q_1, a), q_2' \in \delta_2(q_2, a)$
- $f(q_1, q_2, 1) = 2$ si $q_1 \in F_1$
- $f(q_1, q_2, 2) = 1$ si $q_2 \in F_2$
- $f(q_1, q_2, i) = i$ sinon
- $F = Q_1 \times F_2 \times \{2\}$

Integer 2 means that the last accepting state came from F_1 (therefore we are now looking for a state inf F_2). Likewise, 1 means that the last accepting state was in F_2 (we are looking for a state in F_1). A final state $(q, q_f, 2)$ means that we are in a final state for $\mathcal{A}_2$, but along the current run, we passed through a final state of $\mathcal{A}_1$.

Outline

Transition systems

Consider a transition system (program, a network protocol, a set of processes, . . .) that one can model with an automaton. Finite words allow to describe safety properties. E.g., we can make it so that that accepting states are forbidden states, and ask whether the language of the automaton is empty. If that is not the case, we can find a word, that is a sequence of transitions, that leads from an initial state to a forbidden state (that is a bug).

Transition systems (2)

There are however other interesting properties, beyond safety. For instance, liveness properties:

- Are all processes of a system running at some point?
- Does my server always return to a state where it accepts connexions?
- ...

Infinite words allow to capture such properties.

LTL

Definition (Linear temporal logic)

Let AP be a set of symbols called atomic properties. A formula of the linear temporal logic (LTL) is a finite production of the grammar:

$$\begin{array}{rcll}
\phi & ::= & p & p \in AP \ \textit{(proposition)} \\
 & | & \phi_1 \vee \phi_2 & \textit{(disjonction)} \\
 & | & \neg\phi & \textit{(negation)} \\
 & | & \mathcal{X}\phi & \textit{(next)} \\
 & | & \phi_1 \mathcal{U} \phi_2 & \textit{(until)}
\end{array}$$

LTL (2)

Formula of LTL are interpreted over infinite sequences of sets of atomic propositions. For instance consider $AP = \{\text{wait}, \text{log}, \text{recv}, \text{send}\}$. We can consider the infinite trace:

$$\begin{array}{cccccccc} \text{wait} & \text{wait} & & & \text{wait} & & & \\ & & \text{recv} & \text{send} & & \text{recv} & \text{send} & \dots \\ \text{log} & \text{log} & & & \text{log} & & & \end{array}$$

We want to ask questions such as:

- do I always come back to logging?
- do I always send something after having received?
- if I'm waiting, do I eventually receive something?

Semantics of LTL

Definition

Let w be an infinite sequence of $\mathcal{P}(AP)^\omega$. The truth value of a formula in LTL is given by:

$$\begin{array}{lcl} w \vdash p & \Leftrightarrow & p \in w(0) \\ w \vdash \phi_1 \vee \phi_2 & \Leftrightarrow & w \vdash \phi_1 \text{ or } w \vdash \phi_2 \\ w \vdash \neg\phi & \Leftrightarrow & w \nvdash \phi \\ w \vdash \mathcal{X}\phi & \Leftrightarrow & w^1 \vdash \phi \\ w \vdash \phi_1 \mathcal{U} \phi_2 & \Leftrightarrow & \exists i \geq 0 \text{ such that } w^i \vdash \phi_2 \text{ et } \forall k < i,\ \ w^k \vdash \phi_1 \end{array}$$

Here $w^i = w(i)w(i+1)\ldots$ is the infinite suffix of w starting at index i.

Notations

We can use derived notations:

$$
\begin{array}{lcll}
\phi_1 \wedge \phi_2 & \equiv & \neg(\neg\phi_1 \vee \neg\phi_2) & \text{conjunction} \\
\phi_1 \Rightarrow \phi_2 & \equiv & \phi_1 \vee \neg\phi_2 & \text{implication} \\
\phi_1 \Leftrightarrow \phi_2 & \equiv & (\phi_1 \Rightarrow \phi_2) \wedge (\phi_2 \Rightarrow \phi_1) & \text{equivalence} \\
\top & \equiv & p \vee \neg p & \text{true, for } p \in AP \\
\bot & \equiv & \neg\top & \text{false} \\
\mathcal{F}\phi & \equiv & \top\mathcal{U}\phi & \text{future} \\
\mathcal{G}\phi & \equiv & \neg\mathcal{F}\neg\phi & \text{global}
\end{array}
$$

Exemples

$AP = \{\mathsf{wait}, \mathsf{log}, \mathsf{recv}, \mathsf{send}\}$.

- $\mathcal{G}(\mathsf{recv} \Rightarrow \mathcal{X}\mathsf{send})$

Exemples

$AP = \{\text{wait}, \text{log}, \text{recv}, \text{send}\}$.

- $\mathcal{G}(\text{recv} \Rightarrow \mathcal{X}\text{send})$: "it is always true that I send after having received"
- $\mathcal{F}\neg\text{log}$

Exemples

$AP = \{\text{wait}, \text{log}, \text{recv}, \text{send}\}$.

- $\mathcal{G}(\text{recv} \Rightarrow \mathcal{X}\text{send})$: “it is always true that I send after having received”
- $\mathcal{F}\neg\text{log}$: “at some point I don’t log”
- $\mathcal{G}\neg(\text{send} \wedge \text{recv})$

Exemples

$AP = \{\mathsf{wait}, \mathsf{log}, \mathsf{recv}, \mathsf{send}\}$.

- $\mathcal{G}(\mathsf{recv} \Rightarrow \mathcal{X}\mathsf{send})$: “it is always true that I send after having received”
- $\mathcal{F}\neg\mathsf{log}$: “at some point I don’t log”
- $\mathcal{G}\neg(\mathsf{send} \wedge \mathsf{recv})$: “i never send and receive at the same time”

Let’s revisit some properties

Safety : some formula ϕ never occurs: $\mathcal{G}\neg\phi$

Liveness: some event ϕ_2 always occurs whenever ϕ_1 occurs: $\mathcal{G}(\phi_1 \Rightarrow \mathcal{X}\phi_2)$.

Strong fairness: an event a occurs infinitely many times,and when it does, an event b also occurs

Let's revisit some properties

Safety : some formula ϕ never occurs: $\mathcal{G}\neg\phi$

Liveness: some event ϕ_2 always occurs whenever ϕ_1 occurs: $\mathcal{G}(\phi_1 \Rightarrow \mathcal{X}\phi_2)$.

Strong fairness: an event a occurs infinitely many times,and when it does, an event b also occurs: $\mathcal{G}\mathcal{F}\phi_a \Rightarrow \mathcal{G}\mathcal{F}\phi_b$

How to decide such properties?

LTL to Büchi automata

Definition

For all formula of LTL, one can construct a Büchi automaton that recognizes the same languages as the formula.

Remark : the language of ϕ is the set of infinite words $w \in \mathcal{P}(AP)^\omega$ such that $w \vdash \phi$. Complexity is EXPTIME.