Layered automata

A canonical model for ω-regular languages

Presented by Anusha Gupta · ENS Paris-Saclay

Based on joint work of Antonio Casares, Christof Löding and Igor Walukiewicz

Everybody loves automata

Let’s generalise the story from finite words to infinite words.

Finite automata

- Finite automata describe exactly the regular languages.
- Every regular language has a **minimal canonical DFA**, unique up to isomorphism.
- the minimal DFA can be computed in **polynomial** time.
- This canonical model supports **automata learning**.

Parity automata

- An ω-word is an infinite sequence of letters from a finite alphabet.
- Parity automata attach priorities to transitions or states.
- A run is accepting if the **minimum** priority seen infinitely often is even.
- Deterministic parity automata recognise all ω-regular languages.

The deterministic picture breaks

- ω-regular language **need not** have a unique minimal deterministic automaton.

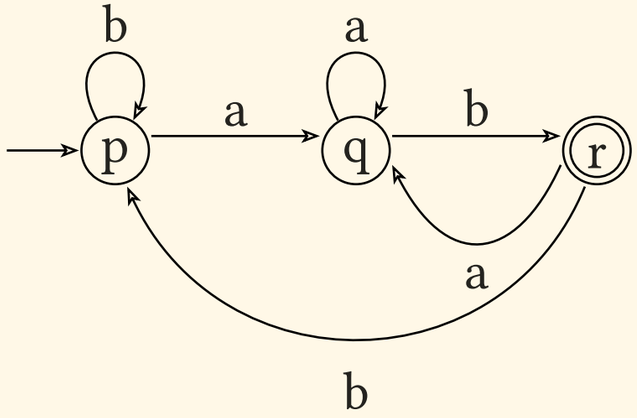
- For deterministic ω-automata expressive enough to recognise all ω-regular languages, minimisation : **NP-hard.**

- **Determinisation** and **Boolean operations** can be costly.

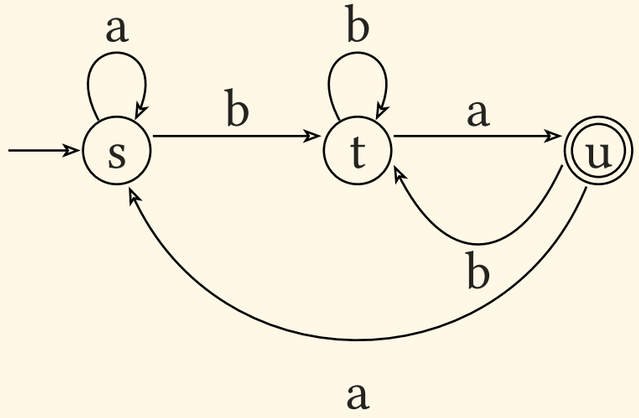
Minimal Büchi automata need not be unique

$$L = \{w \in \{a,b\}^\omega \mid a \text{ and } b \text{ both occur infinitely often}\}$$

Automaton A



Automaton B



- Both are minimal deterministic Büchi automata with three states.
- They are not isomorphic as labelled automata.

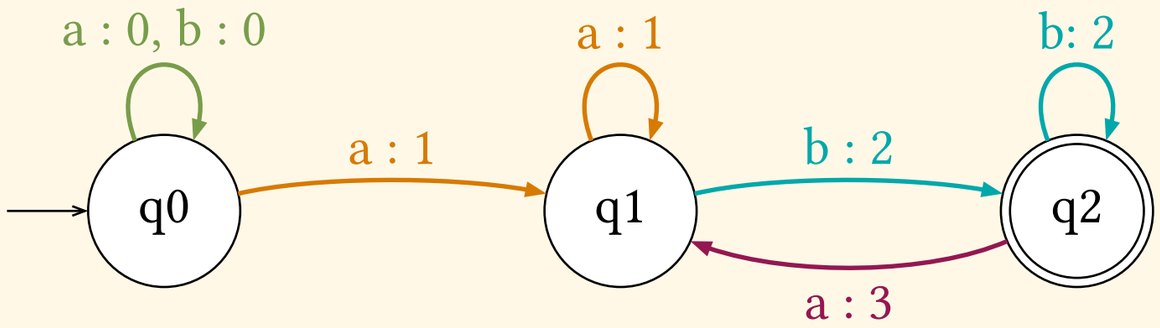
Resolve non-determinism on the fly

An automaton is history-deterministic if its non-determinism can be resolved on-the-fly.

- Think of a letter game: Adam produces the input letters.
- Eve chooses a transition after seeing the prefix so far.
- Eve must use a strategy that does not depend on the future suffix.
- The resulting run must accept exactly when the input word belongs to the language.

Example

$$L = \{w \in \{a, b\}^\omega \mid \text{at least one } a \text{ and infinitely many } b'\text{s occur}\}$$



The game

Adam chooses the next input letter.

Eve chooses a transition after seeing that letter.

Winning condition of Eve: $w \in L \Rightarrow$ the resulting run is accepted by the parity automata.

Eve's strategy

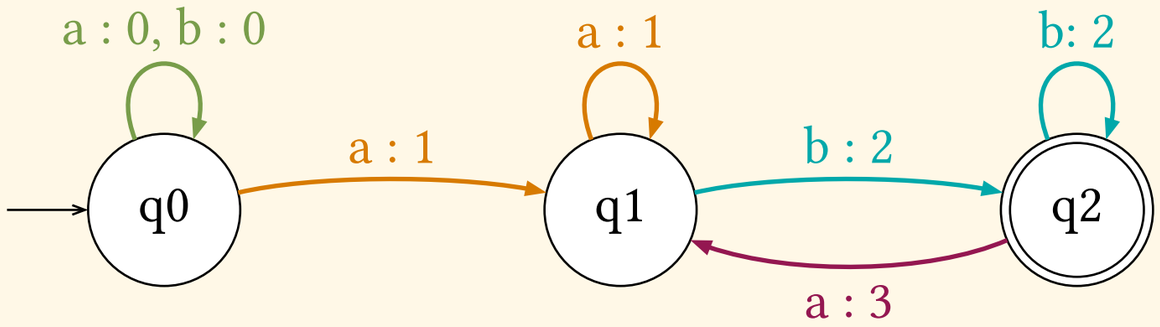
- Stay in q_0 while reading b.
- On the first a, move to q_1.
- Thereafter follow the transitions.

Example play

$$b, a, a, b, a, b, b, \ldots$$

$$q_0, q_1, q_1, q_2, q_1, q_2, q_2, \ldots$$

q_2 is visited infinitely often: Eve wins.



Minimal history-deterministic coBüchi automata

But history-deterministic coBüchi automata cover only a fragment of the ω-regular languages.

- Abu Radi and Kupferman showed that history-deterministic coBüchi automata **admit minimal canonical representations**.
- *WOOOOW!!*
- The canonical automaton can be constructed in **polynomial time**.
- *WOOOWWOOW!!*
- This is a striking positive result for a useful class of automata over infinite words.

History-deterministic coBüchi is not enough

A coBüchi automaton accepts a run if it visits rejecting states only **finitely many times**.

In other words, it can express properties where, eventually, the run stays in accepting states.

Consider the language

$$L = \{w \in \{a, b\}^\omega \mid a \text{ occurs infinitely often}\}$$

$ababab\ldots$ **Accepted** $bbbbbb\ldots$ **Rejected**

This language is ω-regular, but **cannot be recognised by any coBüchi automaton**.

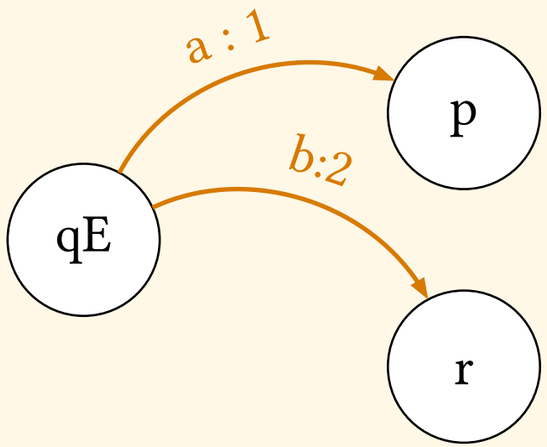
We need a more expressive model to cover all ω-regular languages.

Use alternating parity automata!

A more expressive model, with choices made explicit.

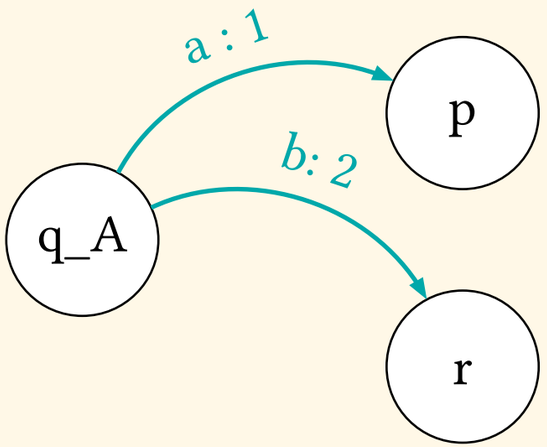
Who chooses the next state?

Existential state Eve chooses



Eve picks one successor. She chooses the branch that helps her win.

Universal state Adam chooses



Adam picks a successor. Eve must win whichever branch he chooses.

Fix an input word: the automaton becomes a parity game.

$w \in L(A)$ iff Eve has a winning strategy.

Why do we require history determinism?

- Eve resolves existential choices; Adam resolves universal choices.
- Both players must make their choices using only the history seen so far.
- No player needs to know the future input to choose a transition.
- This makes the automaton good for games: composing it with a game preserves the winner.

A canonical model for all ω-regular languages

The key idea is to organise deterministic behaviour into layers.

- We want a representation that is expressive enough for every ω-regular language.
- We also want the structural and algorithmic advantages of history-deterministic automata.
- Layered automata provide a formalism for a well-behaved subclass of alternating parity automata.

Layered automata

A sequence of deterministic transition systems connected by morphisms.

The syntax of a layered automaton

$$T_1 \overset{\mu_1}{\leftarrow} T_2 \overset{\mu_2}{\leftarrow} \ldots \overset{\mu_{d-1}}{\leftarrow} T_d$$

- A finite sequence of deterministic transition systems $T_1, \ldots, T_d$.
- For each $x < d$, a morphism $\mu_x : T_{x+1} \to T_x$.
- T_1 is complete, has a distinguished initial state, and all its states are reachable.
- The higher layers may have partial transition functions.

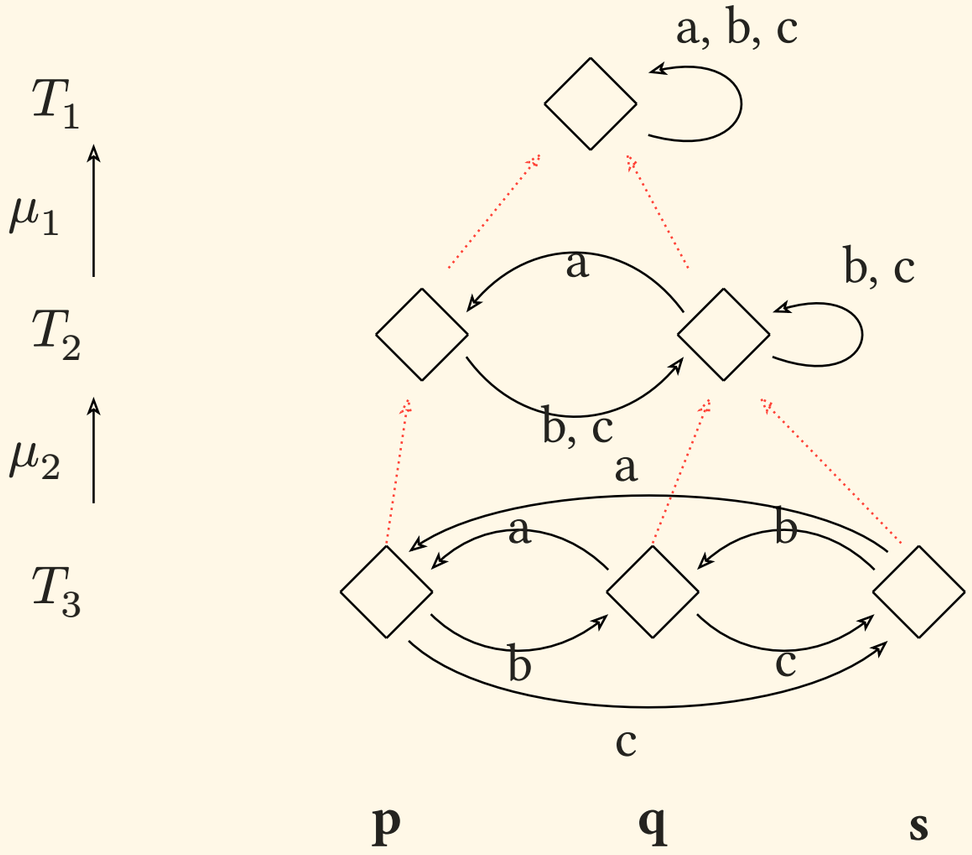
What does a morphism preserve?

$$\mu : V \to V'$$

$$p \xrightarrow{a} q \text{ in } T \text{ implies } \mu(p) \xrightarrow{a} \mu(q) \text{ in } T'$$

- A morphism maps states of one layer to states of the layer below.
- It preserves every transition that is defined.
- Thus the layers are compatible: the image of a transition is the corresponding transition downstairs.

A forest of partial deterministic systems



T_1 is complete. Higher layers may be partial deterministic transition systems.

Every state in T_{x+1} has a unique parent in T_x, specified by μ_x.

The parent maps organise the states into a forest. A leaf is a state with no children.

The leaves become the states of the associated alternating parity automaton.

A missing transition is meaningful

- The first layer is complete, so every input letter has a transition at the root level.
- A transition may be absent in a higher layer.
- When this happens, the semantics can fall back to a shallower layer.
- The deepest layer with a defined transition determines the priority used on that step.

From layers to a parity game

The leaves become the states of an alternating parity automaton.

The leaves are the states

The layered presentation determines an alternating parity automaton.

- Take the leaves of the layered forest as the states of the associated alternating parity automaton.
- Each leaf carries its chain of ancestors.
- To process a letter, inspect the transitions available along that ancestor chain.

Choose the deepest defined transition

$$x = \max \left\{ i : \delta_{i\left(|\mu|_{i(q)}, a\right)} \neq \bot \right\}$$

- Start from a leaf q and read a letter a.
- Look at the ancestor of q in each layer.
- Find the largest index x at which the corresponding a-transition is defined.

The layer index becomes the priority

The hierarchy supplies the parity information.

$$\text{priority}(q, a) = x$$

- The selected layer determines the priority of the transition.
- After the transition, the next state may be any suitable descendant leaf.
- Odd priorities give Eve an existential choice.
- Even priorities give Adam a universal choice.

Why the root layer matters

- T_1 is complete, so the deepest-defined-transition search always has a fallback.
- The root layer must represent the residual behaviour of the language.
- The upper layers refine that behaviour while preserving compatibility through the morphisms.

Consistency and applications

The structural condition behind the good behaviour.

The condition used in the paper

This qualification is important: the main guarantees below are for consistent layered automata.

- Consider two states whose ancestors in the first layer are the same.
- Consistency rules out a word that is strongly accepted from one state and strongly rejected from the other.
- It prevents conflicting strong behaviours inside one root class.
- The paper shows that consistency is equivalent to uniform semantic determinism for the associated alternating automaton.

Consistent layered automata are history-deterministic

Proposition 3.19

- The associated alternating parity automaton admits valid online resolvers for both players.
- The choices can be made from the history seen so far.
- This is the property that makes the model useful in games, verification and synthesis.

The 0–1 probabilistic property

Proposition 3.20

$$\Pr(\rho \text{ is accepting}) \in \{0, 1\}$$

- Fix an input word and consider a random run.
- If the word belongs to the language, the run is accepting with probability 1.
- If the word does not belong to the language, the run is accepting with probability 0.
- This property is useful when composing automata with Markov decision processes.

The consistency condition is tractable

Propositions 3.26–3.28

- Consistency can be checked in polynomial time.
- Emptiness of consistent layered automata is decidable in polynomial time.
- Language inclusion for consistent layered automata is decidable in polynomial time.

Canonicity

Every ω-regular language has a canonical minimal layered automaton.

Canonical minimal layered automata

Theorem 1.1

- Every ω-regular language L has a consistent layered automaton A_L.
- Every equivalent consistent layered automaton contains a subautomaton admitting a surjective morphism onto A_L.
- A_L can be computed in polynomial time from any consistent layered automaton recognising L.

What singles out the canonical representative?

Theorem 4.1

- Normality: there are no evidently useless transitions or unnecessarily high priorities.
- Centralisation: states that can simulate one another within a layer are organised in the same strongly connected component.
- Safe minimality: states are merged when their relevant language behaviour agrees.
- Together, these properties characterise the minimal consistent layered automaton up to isomorphism.

A potentially much smaller representation

— The canonical layered automaton is never larger than the smallest equivalent deterministic automaton in the paper's comparison.

— It can be exponentially smaller than a minimal deterministic parity automaton.

— This does not contradict the hardness of deterministic minimisation: layered automata use an alternating semantics.

Congruences over tuples of finite words

Theorem 6.14

- The finite-word analogy returns: the canonical object can be characterised by a congruence.
- Here the congruence is defined over tuples of finite words.
- It is determined by the language itself, rather than by a chosen automaton presentation.
- The resulting construction characterises the minimal layered automaton.

What remains open?

Conjecture 7.1 and the paper's concluding discussion.

— The paper conjectures that the canonical layered automaton is minimal among all history-deterministic alternating parity automata.

— Other directions include efficient Boolean operations and direct translations from LTL.

— The authors also point to constructions from COCOA representations and learning ω-regular languages.

Conclusions

Layered automata combine expressive power with canonical, tractable structure.