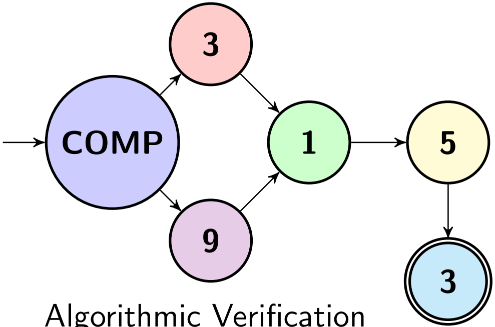


Algorithmic Verification

LTL Model Checking and Büchi Automata

Dr. Liam O'Connor
CSE, UNSW (for now)
Term 1 2020

LTL Model Checking

$$\begin{tikzcd}
\mathcal{M} & \models & \varphi \\
\text{Kripke Structure} \arrow[d] & ??? & \text{LTL Formula} \arrow[d] \\
L(\mathcal{M}_A) & \subseteq & L(\varphi_A) \\
\text{Büchi Automaton} & & \text{Büchi Automaton}
\end{tikzcd}$$

Büchi Automata

Büchi Automata are like finite automata, but their languages are of infinite-length strings, so they work well for behaviours $\in (2^{\mathcal{P}})^{\omega}$.

Büchi Automata

Definition

A (generalized) Büchi automaton is a 5-tuple $(Q, I, \Sigma, \delta, F)$ where

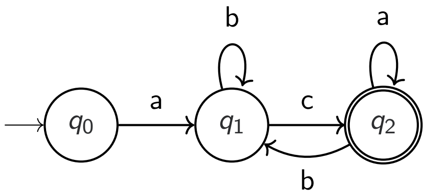
- Q is a set of states.
- $I \subseteq Q$ is a set of initial states.
- Σ is our alphabet of actions.
- $\delta : (Q \times \Sigma) \to 2^Q$ is our transition relation.
- $F \subseteq Q$ is a set of final states.

Language

We consider $\sigma \in L(A)$ for a Büchi automaton A iff it visits a particular final state infinitely often. More formally, define $\mathsf{inf}(\rho) = \{\ q \mid q \text{ appears infinitely often in } \rho\ \}$, then we say

$$\mathsf{trace}(\rho) \in L(A) \Leftrightarrow \mathsf{inf}(\rho) \cap F \neq \emptyset$$

Example



- `acaaaaaaa...` **Accepted**
- `acbcbcbcb...` **Accepted**
- `acbbbbbbb...` **Rejected**

Exercise

Let $\Sigma = \{0, 1\}$. Define Büchi automata for the following languages.

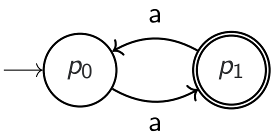
- $L_1 = \{v \in \Sigma^\omega \mid \text{0 occurs in } v \text{ exactly once } \}$
- $L_2 = \{v \in \Sigma^\omega \mid \text{every 0 is followed at least one 1}\}$
- $L_3 = \{v \in \Sigma^\omega \mid v \text{ contains infinitely many 1s}\}$
- $L_4 = (01)^*\Sigma^\omega$

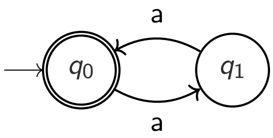
Closure Properties

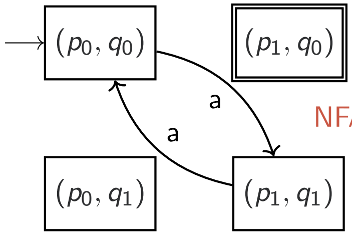
Büchi Automata are closed under:

- Union (same as NFAs)
- Intersection (as we will show)
- Complement (as we will refer to textbooks — it's hard)

Intersection of GBAs







NFA product doesn't work!

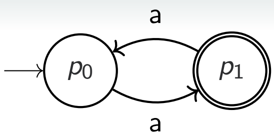
Triple Product

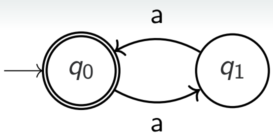
An accepting cycle of a product of Büchi automata $P \times Q$ must cycle through accepting states of both P and Q infinitely often. Arbitrarily, we shall say it must alternate by visiting a final state of Q then P then Q and so on. This doesn't affect expressivity because we are only concerned with infinite strings.

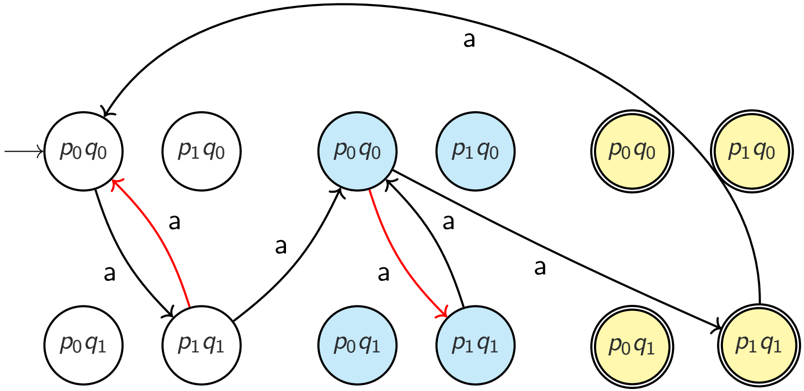
Key idea

Make three copies of the product: $P \times Q \times \{0, 1, 2\}$.

- Copy '0' is marked with initial states $I_P \times I_Q$.
- Copy '2' is entirely marked as final states.
- Transition relation like normal product, but:
  - We move from copy 0 to copy 1 when moving to a state $\in F_Q$.
  - We move from copy 1 to copy 2 when moving to a state $\in F_P$.
  - All transitions from copy 2 move back to copy 0.


a
p_0
p_1
a


a
q_0
q_1
a


a
$p_0\,q_0$
$p_1\,q_0$
$p_0\,q_0$
$p_1\,q_0$
$p_0\,q_0$
$p_1\,q_0$
a
a
a
a
a
a
$p_0\,q_1$
$p_1\,q_1$
$p_0\,q_1$
$p_1\,q_1$
$p_0\,q_1$
$p_1\,q_1$

Büchi Product

Let $A_1 = (Q_1, I_1, \Sigma_1, \delta_1, F_1)$ and $A_2 = (Q_2, I_2, \Sigma_2, \delta_2, F_2)$.

Definition

Define $A_\cap$ with $Q = Q_1 \times Q_2 \times \{0,1,2\}$, and $I = I_1 \times I_2 \times \{0\}$, $\Sigma = \Sigma_1 \cap \Sigma_2$ and $F = Q_1 \times Q_2 \times \{3\}$.
We define δ as follows:

$$\begin{array}{lll}
((q_1, q_2, 0), a, (q_1', q_2', 0)) \in \delta & \text{iff} & (q_i, a, q_i') \in \delta_i \ (i = 1,2) \wedge q_1' \notin F_1 \\
((q_1, q_2, 0), a, (q_1', q_2', 1)) \in \delta & \text{iff} & (q_i, a, q_i') \in \delta_i \ (i = 1,2) \wedge q_1' \in F_1 \\
((q_1, q_2, 1), a, (q_1', q_2', 1)) \in \delta & \text{iff} & (q_i, a, q_i') \in \delta_i \ (i = 1,2) \wedge q_1' \notin F_2 \\
((q_1, q_2, 1), a, (q_1', q_2', 2)) \in \delta & \text{iff} & (q_i, a, q_i') \in \delta_i \ (i = 1,2) \wedge q_1' \in F_2 \\
((q_1, q_2, 2), a, (q_1', q_2', 0)) \in \delta & \text{iff} & (q_i, a, q_i') \in \delta_i \ (i = 1,2)
\end{array}$$

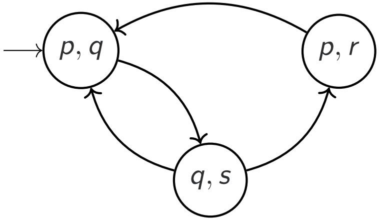
LTL Model Checking

$$\begin{aligned} & L(A_M) \subseteq L(A_\Phi) \\ \equiv\ & L(A_M) \cap L(A_\Phi)^C = \emptyset \\ \equiv\ & L(A_M) \cap L(A_{\neg\Phi}) = \emptyset \\ \equiv\ & L(A_M \times A_{\neg\Phi}) = \emptyset \end{aligned}$$

We still need to know how to:

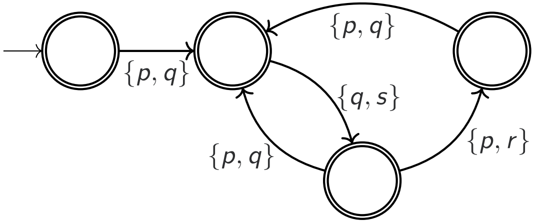
- Determine if $L(A) = \emptyset$ for a Büchi automaton A.
- Convert a Kripke structure M to a Büchi automaton A_M
- Convert a LTL formula Φ to a Büchi automaton A_Φ.

Büchi from Kripke



How to convert

We add a new initial state, move labels on the states to all incoming edges, and make all states final.



Büchi Automata Emptiness

Theorem (Büchi pumping)

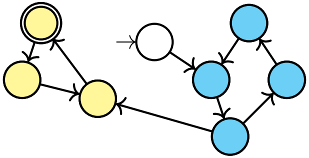
Given a Büchi Automaton $A = (Q, I, \Sigma, \delta, F)$ then $L(A) \neq \emptyset$ iff there exists $v, w \in \Sigma^*$ with lengths $\leq |Q|$ such that $vw^\omega \in L(A)$.

We need to find a final state that is:

- Reachable from an initial state.
- Reachable from itself — a cycle.

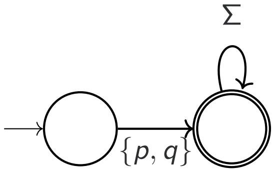
How to detect cycles?

We use **Strongly Connected Components!**. Many algorithms exist (see online).

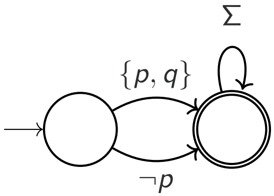


How to convert from LTL formulae to Büchi Automata? For atomic formulae, it's straightforward:

- $p \wedge q$



- $p \Rightarrow q$



We can manually construct them for temporal formulae, but how to do so systematically?

Methods of LTL to Büchi

Many exist. All are complicated.

- Tableau Methods (Kersten, Manna, McGuire, Pnueli or Geth, Peled, Vardi, Wolper)
- Automata Theoretic (Vardi)
- Local and Eventuality Automata (Vardi, Wolper)

Local and Eventuality Automata

1. Reduce number of operators to just **UNTIL** and **X**.
2. Construct a *local automaton* for Φ — describes behaviours that satisfy the safety component of Φ.
3. Construct a *eventuality automaton* for Φ — ensures "termination", the liveness aspect of Φ.
4. Intersect the two automata, then reduce the alphabet to just atomic propositions.

Closure and Maximal Subsets

Closure

The *closure* Cl(Φ) of an LTL formula Φ is the set of all subformulae of Φ and their negation.

What is the closure of ● **U** ● ?

Maximal Subsets

Define Sub(Φ) of an LTL formula Φ as the set of all maximal subsets of Cl(Φ) that are *locally consistent* (not contradictory).

Local Automaton

Definition

The local automaton for A^L_Φ for a formula Φ is defined as $(Q, I, \Sigma, \delta, F)$ where:

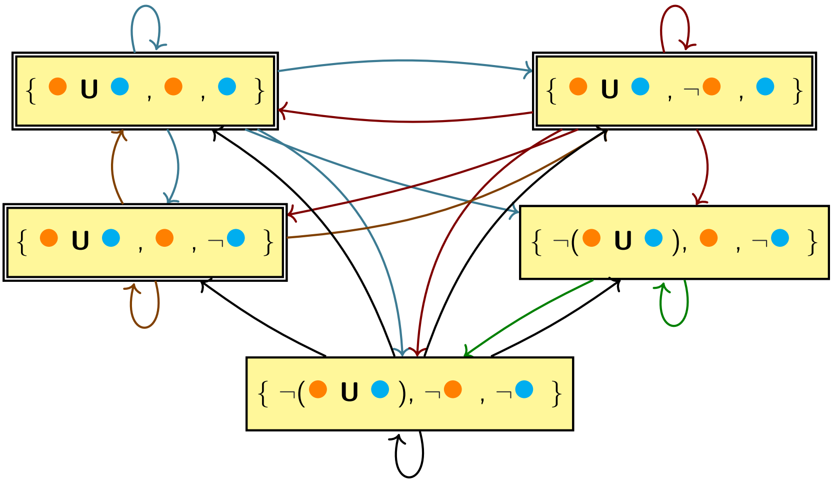
- $Q = \mathrm{Sub}(\Phi)$
- $I = \{\ S \in \mathrm{Sub}(\Phi) \mid \Phi \in S\ \}$
- $\Sigma = 2^{\mathrm{Cl}(\Phi)}$
- $F = I$
- $q \in \delta(p, a)$ if $a = p$ and
  - $\mathbf{X}\varphi \in p$ if $\varphi \in q$
  - $\varphi\ \mathbf{U}\ \psi \in p$ if $\psi \in p$ or $\varphi \in p \wedge (\varphi\ \mathbf{U}\ \psi) \in q$

Whats the local automaton for **X**● ?

Example

Whats the local automaton for ● **U** ● ?

(the edge actions are always just the origin state, so they're omitted)



Eventuality Automaton

The local automaton accepts just the safety part of our formula. So, our example on the previous slide would accept an infinite sequence of ● .

To ensure that the second part of **UNTIL** actually happens, we use an *eventuality automaton*.

Eventuality Automaton

The eventuality automaton A^E_Φ for a formula Φ is defined as $(Q, I, \Sigma, \delta, F)$ where the states Q are all sets of **UNTIL** formulae in $\mathsf{Cl}(\Phi)$, the initial and final state is $\emptyset$, the actions Σ are the same as the local automaton $\mathsf{Sub}(\Phi)$, and δ is defined as follows:

$q \in \delta(p, a)$ iff a is consistent with p and

When $p = \emptyset$: For all $(\varphi \ \mathbf{U}\ \psi) \in a$ one has $(\varphi \ \mathbf{U}\ \psi) \in q$ iff $\psi \notin a$

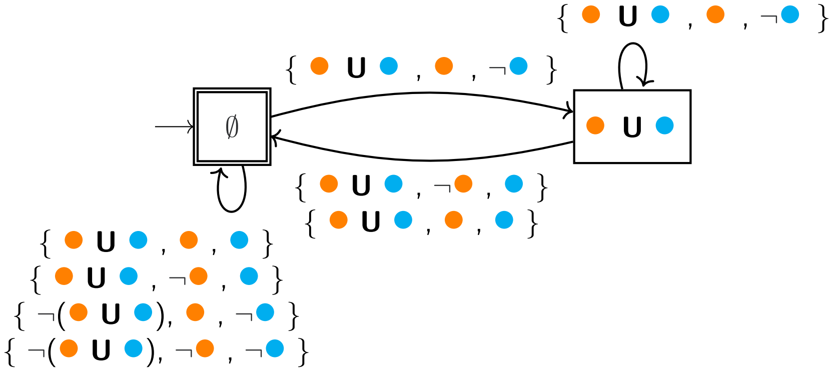
When $p \neq \emptyset$: For all $(\varphi \ \mathbf{U}\ \psi) \in p$ one has $(\varphi \ \mathbf{U}\ \psi) \in q$ iff $\psi \notin a$

Example

The current state of the eventuality automaton reflects the set of **UNTIL** formulae we are waiting on.

Example for ● **U** ● :

No other consistent edges!



Alphabet Reduction

Our model automata A_M has just atomic propositions for actions, but our formula automaton $A^L_\Phi \times A^E_\Phi$ includes temporal propositions in the actions.

Solution

After computing the product of local and eventuality automata, however, we can simply remove all negations and temporal propositions from the actions, leaving only atomic propositions behind.
Then we can compute the final product of our model with our negated formula as normal.

Complexity

- Each node in a local automaton contains each subformula, $|Q|$ exponential in size of formula.
- Eventuality automata has each combination of **UNTIL**s, $|Q|$ exponential in number of **UNTIL**s.
- Then product, reduction to reachable states, alphabet reduction, and final product.
- Then SCCs to find cycles, check emptiness.

Tons of overhead. Other methods are smarter (but even more complicated).

SPIN

Liam: Whirlwind tour of SPIN, preview of next lecture

Bibliography

- Baier/Katoen: Principles of Model Checking, Section 5.2