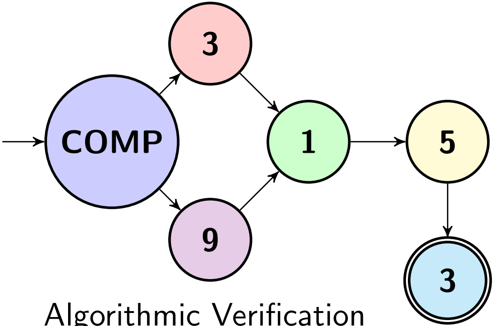


Algorithmic Verification

Safety and Liveness, Fairness

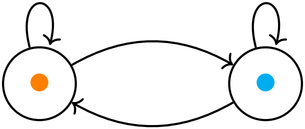
Dr. Liam O'Connor
CSE, UNSW (for now)
Term 1 2020

Behaviours

Recall

The infinite traces of a Kripke structure are called *behaviours*. So they are infinite sequences of state labels $\subseteq (2^{\mathcal{P}})^{\omega}$.

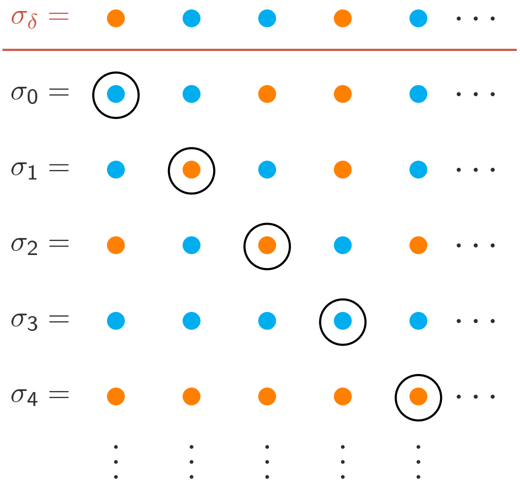
How many behaviours for these automata?



Cantor's Uncountability Argument

Result

It is impossible in general to enumerate the space of all behaviours.



Proof

Suppose there $\exists$ a sequence $\sigma_0\sigma_1\sigma_2\ldots$ that enumerates all behaviours. Then we can construct a devilish sequence σ_δ that differs from any σ_i at the ith position, and thus is not in our sequence. **Contradiction!**

Metric for Behaviours

We define the *distance* $d(\sigma, \rho) \in \mathbb{R}_{\geq 0}$ between two behaviours σ and ρ as follows:

$$d(\sigma, \rho) = 2^{-\sup\{\ i \in \mathbb{N} \mid \sigma|_i = \rho|_i\}}$$

(we say that $2^{-\infty} = 0$) Intuitively, we consider two behaviours to be close if there is a long prefix for which they agree.

Observations

- $d(x, y) = 0 \Leftrightarrow x = y$
- $d(x, y) = d(y, x)$
- $d(x, z) \leq d(x, y) + d(y, z)$

This forms a *metric space* and thus a *topology* on behaviours.

Topology

Definition

A set S of subsets of U is called a *topology* if it contains $\emptyset$ and U, and is closed under union and finite intersection. Elements of S are called *open* and complements of open sets are called *closed*.

Example (Sierpiński Space)

Let $U = \{0, 1\}$ and $S = \{\emptyset, \{1\}, U\}$.

Questions

- What are the closed sets of the Sierpiński space?
- Can a set be *clopen* i.e. both open and closed?

Topology for Metric Spaces

Our metric space can be viewed as a topology by defining our open sets as (unions of) *open balls*:

$$B(\sigma, r) = \{ \ \rho \mid d(\sigma, \rho) < r \ \}$$

This is analogous to open and closed ranges of numbers.

Why do we care?

Viewing behaviours as part of a metric space gives us notions of limits, convergence, density and many other mathematical tools.

Limits and Boundaries

Consider a sequence of behaviours $\sigma_0\sigma_1\sigma_2\ldots$. The behaviour σ_ω is called a *limit* of this sequence if the sequence *converges* to σ_ω, i.e. for any positive ε:

$$\exists n.\ \forall i \geq n.\ d(\sigma_i, \sigma_\omega) < \varepsilon$$

The *limit-closure* or *closure* of a set A, written $\overline{A}$, is the set of all the limits of sequences in A.

Question

Is $A \subseteq \overline{A}$?

A set A is called *limit-closed* if $\overline{A} = A$. It is easy (but not relevant) to prove that limit-closed sets and closed sets are the same.
A set A is called *dense* if $\overline{A} = (2^{\mathcal{P}})^\omega$ i.e. the closure is the space of all behaviours.

Properties

Recall

A linear temporal property is a set of behaviours.

1. A *safety* property states that something **bad** does not happen. For example:

   I will never run out of money.

   These are properties that may be violated by a finite prefix of a behaviour.

2. A *liveness* property states that something **good** will happen. For example:

   If I start drinking now, eventually I will be smashed.

   These are properties that can always be satisfied eventually.

Properties Examples

Try to express these in LTL. Are they safety or liveness?

- *When I come home, there must be beer in the fridge* – **Safety**
- *When I come home, I'll drop on the couch and drink a beer* – **Liveness**
- *I'll be home later* – **Liveness**
- *The program never allocates more than 100MB of memory* — **Safety**
- *The program will allocate at least 100MB of memory* – **Liveness**
- *No two processes are simultaneously in their critical section* — **Safety**
- *If a process wishes to enter its critical section, it will eventually be allowed to do so* – **Liveness**

Safety Properties are Limit Closed

Let P be a safety property.

- Assume that there exists a sequence of behaviours $\sigma_0\sigma_1\sigma_2\ldots$ such that every $\sigma_i \in P$ but their limit $\sigma_\omega \notin P$.
- For σ_ω to violate the safety property P, there must be a specific state in σ_ω where shit hit the fan. That is, there must be a specific k such that any behaviour with the prefix $\sigma_\omega|_k$ is not in P.
- For σ_ω to be the limit of our sequence, however, that means there is a particular point in our sequence i after which all σ_j for $j \geq i$ agree with σ_ω for the first $k+1$ states. According to the above point, however, those σ_j cannot be in P.

Contradiction.

Liveness Properties are Dense

Let P be a liveness property. We want to show that $\overline{P}$ contains all behaviours, that is, that any behaviour σ is the limit of some sequence of behaviours in P.

- If $\sigma \in P$, then just pick the sequence $\sigma\sigma\sigma\ldots$ which trivially converges to σ.
- If $\sigma \notin P$:
  - It must not "do the right thing eventually", i.e. no finite prefix of σ ever fulfills the promise of the liveness property.
  - However, every finite prefix $\sigma|_i$ of σ could be extended differently with some ρ_i such that $\sigma|_i\rho_i$ is in P again.
  - Then, $\lim_{i\to\infty}(\sigma|_i\rho_i) = \sigma$ and thus σ is the limit of a sequence in P.

The Big Result

Alpern and Schneider's Theorem

Every property is the intersection of a safety and a liveness property

$$P = \underbrace{\overline{P}}_{\text{closed}} \cap \underbrace{(2^{\mathcal{P}})^{\omega} \setminus (\overline{P} \setminus P)}_{\text{dense}}$$

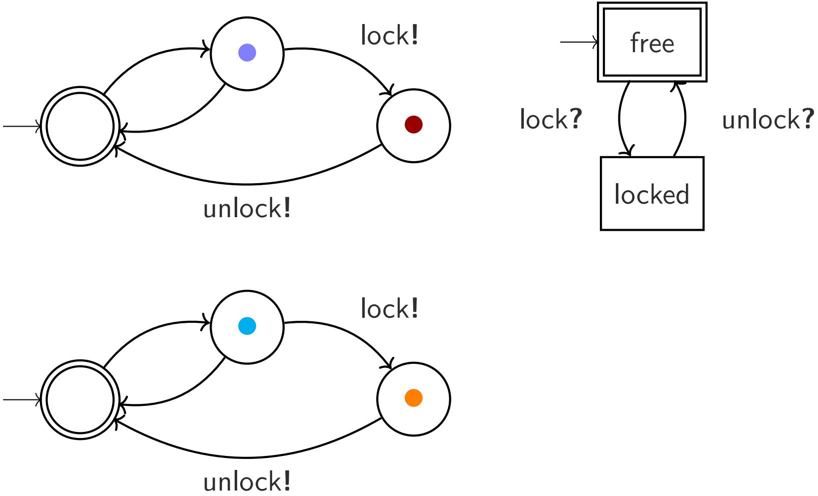
Why are these two components closed and dense? Also, let's do the set theory reasoning to show this equality holds.
If there's time: Let's also prove that every property is the intersection of two liveness properties.

Decomposing Safety and Liveness

Let's break these up into their safety and liveness components.

- The program will allocate exactly 100MB of memory.
- If given an invalid input, the program will return the value -1.
- The program will sort the input list.

Critical Sections



Does the product satisfy **G**(● ⇒ **F** ●) (*eventual entry*)?

Fairness

Definition

Fairness is a scheduling constraint that ensures that if a process is ready to move, it will eventually be allowed to move.

Two types of fairness:

- **Weak Fairness** — If a process is continuously ready, it will eventually be scheduled:

$$\mathbf{G}(\mathbf{G}\ \text{Ready} \Rightarrow \mathbf{F}\ \text{Scheduled})$$

- **Strong Fairness** — If a process is ready infinitely often, it will eventually be scheduled.

$$\mathbf{G}(\mathbf{GF}\ \text{Ready} \Rightarrow \mathbf{F}\ \text{Scheduled})$$

Bibliography

- Baier/Katoen: Principles of Model Checking, Section 3.3 (parts), 3.4 (parts), 3.5
- Bowen Alpern and Fred B. Schneider: *Defining Liveness*, Information Processing Letters 21(4):181-185, October 1985.