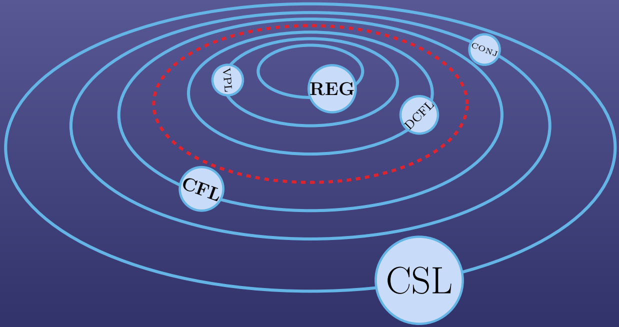
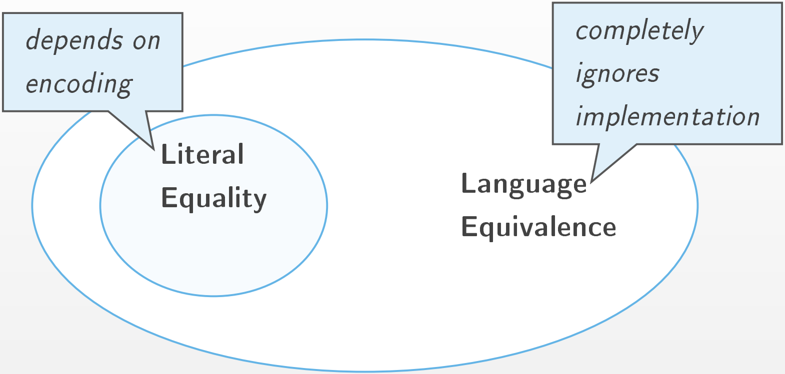
Bauman Moscow State University
Th. Computer Science Dept.

Equivalences of Finite Automata



Antonina Nepeivoda
a_nevod@mail.ru

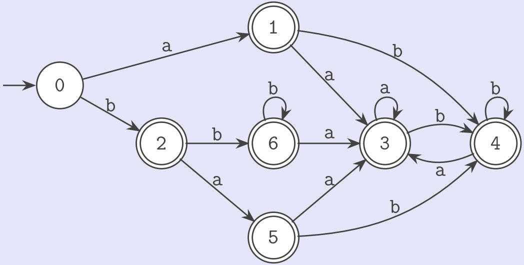
Machines Comparison



Problem: *how to find an equivalence that is sustainable to irrelevant implementation details (such as node naming) but tracks parsing-relevant properties?*

Equality of DFA and NFA

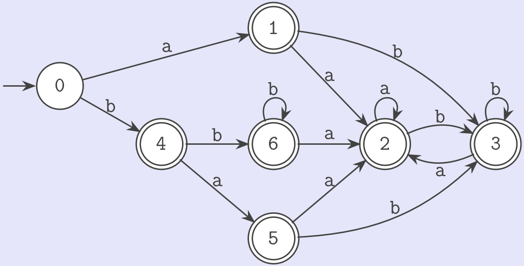
Given a DFA, its states can be *canonically named*: i.e. state number is determined by the string marking the shortest path from the starting state to the state considered. Then the numeration depends only on the chosen linear order on strings.



Above is the state numeration with respect to the military (length- lexicographic) order, given $\mathtt{a} \prec \mathtt{b}$*:*
$\varepsilon \prec \mathtt{a} \prec \mathtt{b} \prec \mathtt{aa} \prec \mathtt{ab} \prec \mathtt{ba} \prec \mathtt{bb}$.

Equality of DFA and NFA

Given a DFA, its states can be *canonically named*: i.e. state number is determined by the string marking the shortest path from the starting state to the state considered. Then the numeration depends only on the chosen linear order on strings.



Another numeration is induced by the "vanilla" lexicographic order, given a $\prec$ b*: now* $\varepsilon \prec$ a $\prec$ aa $\prec$ ab $\prec$ b $\prec$ ba $\prec$ bb.

Equality of DFA and NFA

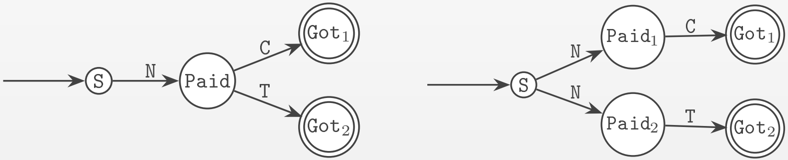
Given a DFA, its states can be *canonically named*: i.e. state number is determined by the string marking the shortest path from the starting state to the state considered. Then the numeration depends only on the chosen linear order on strings.
However, the canonical numeration does not work for NFA, provided that the string sets marking paths to the states can coincide.

Hence, DFA equality up to the state renaming can be thought as the literal coincidence of the canonically ordered DFA; but the canonical order does not work for recognising NFA equality.

Behavioral Equivalence and Language Equivalence

Language equivalence tracks only admissible actions, but not a way the actions are performed. The following example is well-known.

- N is «put a note into the machine»;
- C is «order a coffee»;
- T is «order a tea».



The given two machines have the same language, but are distinct from the point of view of a user:

- the first requires a note, and then asks what drink is ordered;
- the second asks for a choice when taking money, then requires to press the button that prepares it.

Bisimulation of Labelled Transition Systems

Bisimulation is a relation $\sim$ *between states of the systems* $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *satisfying the following property:*

- *If* $q_1 \sim q_2$ $(q_1 \in \mathcal{T}_1,\ q_2 \in \mathcal{T}_2)$, *then for every transition* $q_1 \xrightarrow{\gamma} q_1'$ *in* $\mathcal{T}_1$ *there is a transition* $q_2 \xrightarrow{\gamma} q_2'$ *in* $\mathcal{T}_2$ *such that* $q_1' \sim q_2'$, *and vice versa.*

Starting and final (if any) states must be bisimilar.

Every state machine $\mathscr{A}$ can be represented as a labelled transition system.

- $\mathscr{A}_1$ and $\mathscr{A}_2$ are bisimilar $\Leftrightarrow$ their LTS $\mathcal{T}_1$ and $\mathcal{T}_2$ are bisimilar.

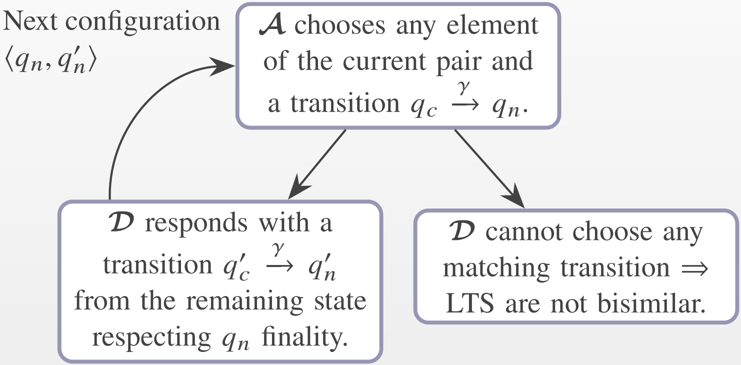
Labelled Transition Systems *versus* NFA

Labelled transitions are not necessarily finite; moreover, LTS contain no final states.

Existence of final states can be modelled via introducing *endmarkers* (usually denoted \$). Then every final state of an NFA has a transition by the endmarker to the unique «bottom» state.

Bisimulation Game

$\mathcal{T}_1$ and $\mathcal{T}_2$ bisimilarity checking technique can be formulated as a two-player game with an initial configuration $\langle q_S, q'_S \rangle$:



- Attacker's winning strategy always leads to the fact that any possible play is finite.

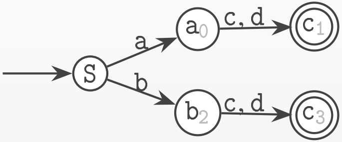
Equivalent Trim DFA are Bisimilar

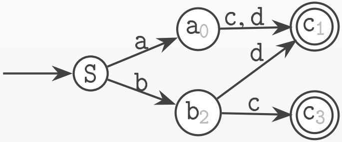
Given two non-bisimilar trim DFA $\mathscr{A}_1$ and $\mathscr{A}_2$, we assume by the contradiction that the player $\boldsymbol{\mathcal{A}}$ has a winning strategy. The strategy is completely determined by a finite input string ω.

- If after reading the string ω one DFA ends up in a final state, while the other ends up in a non-final state, then ω witnesses that their languages do not coincide.
- If after reading the string ω one DFA (say $\mathscr{A}_1$) ends up in a state with an outgoing transition by some $\gamma \in \Sigma$, while the other does not has such a transition, then no string in $\mathscr{L}(\mathscr{A}_1)$ with the prefix $\omega\gamma$ can belong to $\mathscr{L}(\mathscr{A}_2)$. The DFA are trim $\Rightarrow$ at least one string prefixed by $\omega\gamma$ is in $\mathscr{L}(\mathscr{A}_1)$.

Bisimulation and Equality

Bisimilar DFA are not necessarily equal, even if cardinalities of their state sets are also equal.





In this example, the distinction between the states $\mathtt{c}_1$ and $\mathtt{c}_3$ is redundant: they are indistinguishable with respect to the languages that can be recognised from them, namely, $\mathscr{L}(\mathtt{c}_1) = \mathscr{L}(\mathtt{c}_3) = \{\varepsilon\}$. Hence, $\mathtt{c}_1 \sim \mathtt{c}_3$.

We could *merge* the bisimilar states with no impact to the recognised DFA language; conversely, if we know that the DFA states coincide wrt their languages, then we know they are behaviorally equivalent.