Behavioural Equivalences on Finite-State Systems are PTIME-hard

Petr Jančar   Zdeněk Sawa

Department of Computer Science, FEI
Technical University of Ostrava
17. listopadu 15, Ostrava-Poruba 708 33
Czech Republic

26 November 2005

Presented Result

The presented result will be published in journal Computing and Informatics (Volume 24, Number 5, 2005)

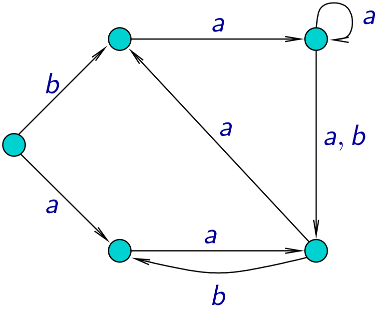
Introduction

- Importance of verification
- Two main types of problems:
  - model checking
  - equivalence checking
- Computational complexity

Labelled Transition Systems

Labelled transition system is a tuple $\mathcal{T} = (S, \mathcal{A}, \longrightarrow)$ where:

- S – set of **states** (can be infinite),
- $\mathcal{A}$ – finite set of **actions**,
- $\longrightarrow \subseteq S \times \mathcal{A} \times S$ – **transition relation**
  (we usually write $s \stackrel{a}{\longrightarrow} s'$ instead of $(s, a, s') \in \longrightarrow$)



Equivalence checking

The **equivalence checking** problems are problems of the following form:

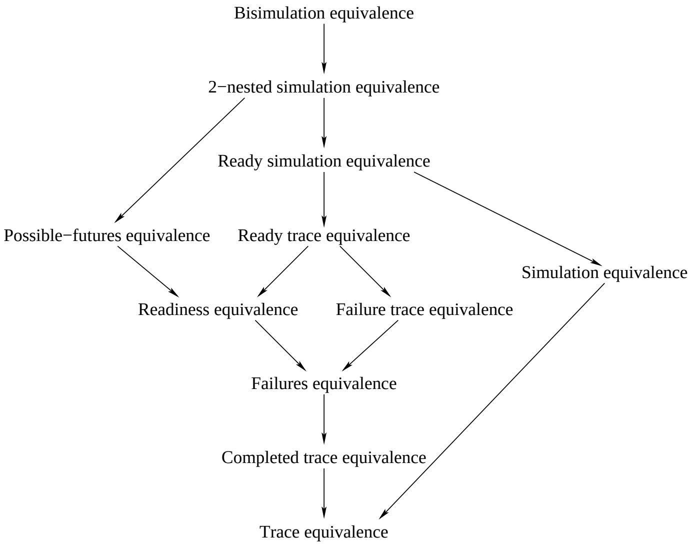
Problem

Instance: Two labelled transition systems (their descriptions).

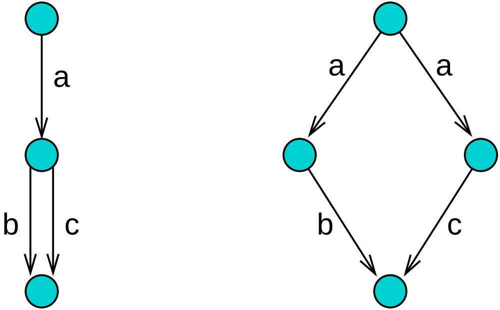
Question: Is the behaviour of these two system equivalent with respect to some given notion of equivalence?

Remark: We often consider two states of one labelled transition system instead of two systems (we can take their disjoint union).

Linear time / branching time spectrum



Difference between Equivalences



Both systems can perform sequences of actions *ab* and *ac* (and nothing else), but their behaviour is quite different.

Trace Preorder

$\mathcal{T} = (S, \mathcal{A}, \longrightarrow)$ – labelled transition system

$Tr(s)$ – the set of all sequences of actions that can be performed starting in a state s

Definition

Relation $\sqsubseteq_{tr}$ such that $s \sqsubseteq_{tr} s'$ iff $Tr(s) \subseteq Tr(s')$, is called **trace preorder**.

Definition

Relation $\equiv_{tr}$ such that $s \equiv_{tr} s'$ iff $s \sqsubseteq_{tr} s'$ and $s' \sqsubseteq_{tr} s$, is called **trace equivalence**.

Bisimulation Equivalence

Binary relation $\mathcal{R}$ on the set of states of a labelled transition system is called a **bisimulation** relation iff for every pair of states $(s, t) \in \mathcal{R}$ we have:

- If $s \xrightarrow{a} s'$ for some action a and a state s', then $t \xrightarrow{a} t'$ for some state t' such that $(s', t') \in \mathcal{R}$.
- If $t \xrightarrow{a} t'$ for some action a and a state t', then $s \xrightarrow{a} s'$ for some state s' such that $(s', t') \in \mathcal{R}$.

Definition

Relation $\sim$ such that $s \sim t$ iff there is some bisimulation $\mathcal{R}$ such that $(s, t) \in \mathcal{R}$ is called a **bisimulation equivalence** or **bisimilarity**.

Theorem

The following problem is PTIME-hard for any binary relation $\mathcal{R}$ between bisimulation equivalence and trace preorder (i.e., such that $\sim \subseteq \mathcal{R} \subseteq \sqsubseteq_{tr}$):

Instance: An acyclic finite-state system and two of its states s, s'.

Question: Is $(s, s') \in \mathcal{R}$?

PTIME-hardness

Definition

An algorithm in in LOGSPACE iff it uses at most $O(\log n)$ bits of memory for an input instance of size n.

Definition

A problem P is PTIME-hard iff for every $P' \in \text{PTIME}$ there is a LOGSPACE reduction from P' to P.

Remark: PTIME-hard problems are hardly parallelizable.
If we show that a problem P is PTIME-hard, it means that there is no efficient parallel algorithm solving P unless $\text{NC} = \text{PTIME}$.

Definition

A parallel algorithm is **efficient** iff for an instance of size n it works in time $O(\log^k n)$ on $O(n^{k'})$ processors where k, k' are (small) constants.

Class of efficient parallel algorithms is denoted NC.
It is not difficult to show that

$$\mathrm{NC} \subseteq \mathrm{PTIME}$$

but it is generally conjectured that

$$\mathrm{NC} \subsetneq \mathrm{PTIME}$$

and in particular that for any PTIME-hard problem P

$$P \notin \mathrm{NC}$$

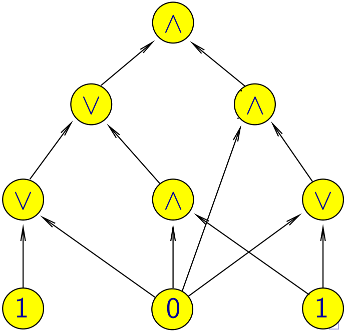
Proof Idea

We show a logspace reduction from the following PTIME-complete problem:

Problem MCVP:

Instance: Monotone boolean circuit and its input.

Question: Is on the output the value 1 ?

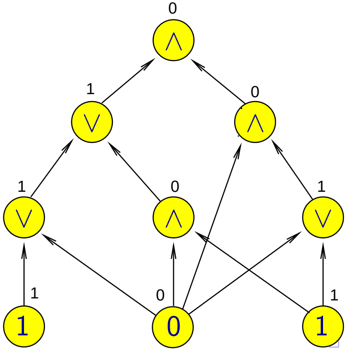


We show a logspace reduction from the following PTIME-complete problem:

Problem MCVP:

Instance: Monotone boolean circuit and its input.

Question: Is on the output the value 1 ?



Proof Idea

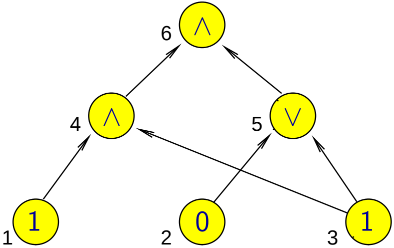
The algorithm constructs for a given boolean circuit a labelled transition system with two distinguished states s, s' such that:

on the output of the circuit is $1 \quad \Rightarrow \quad s \sim s' \quad \Rightarrow \quad (s, s') \in \mathcal{R}$

on the output of the circuit is $0 \quad \Rightarrow \quad s \not\sqsubseteq_{tr} s' \quad \Rightarrow \quad (s, s') \notin \mathcal{R}$

(Remark: $\sim \subseteq \mathcal{R} \subseteq \sqsubseteq_{tr}$)

An Example of the Construction



For the given circuit the algorithm constructs the following labelled transition system . . .

An Example of the Construction