Current Topics in Informatics (2 lectures on Model Checking)

(bachelor course KMI/AKTI)

Petr Jančar

December 8, 2022

(*Draft of the notes for the two lectures.*)

We started with recalling the undecidability of general verification problems (for software and hardware systems); in particular we recalled the halting problem and Rice's theorem. Such problems are decidable for finite-state systems but even in this case there is the so-called state-explosion problem already for simple parallel system (consisting of a small number of communicating components), which can easily lead to a large computational complexity of the verified problems.

For concreteness we then recalled Peterson's algorithm for solving the critical section problem, and illustrated that already simple problems in distributed computing can lead to solutions with subtle mistakes. Nevertheless, here it is possible (and feasible) to perform automated analysis that leads either to verifying the desired properties (of the suggested solution-protocol) or to presenting some counterexample(s).

To give a concrete idea of a possible formalization, we first recalled a variant of systems related to the notions of *Kripke structures* and/or *labelled transition systems*. By a *labelled system*, or just a *system* for short, we will here mean a structure

$$\mathcal{L} = (S, \textsc{ap}, \ell, \rightarrow)$$

where

- S is the set of (global) *states* (typically finite, though they might be also infinite);
- $\textsc{ap}$ is a finite set of *atomic propositions* (for expressing some relevant facts about states);
- ℓ is the *labelling function* $\ell : S \to 2^{\textsc{ap}}$ (attaching valid propositions to each state);
- and $\rightarrow \subseteq S \times S$ is the *one-step relation*, for which we rather write $s \rightarrow s'$ instead of $(s, s') \in \rightarrow$ (meaning that the system can go from the state s to the state s' by performing an "instruction").

We want to design a language, typically a set of formulas of a logic, expressing (some relevant) properties, so that for a system $\mathcal{L} = (S, \textsc{ap}, \ell, \rightarrow)$, a state $s \in S$, and a formula φ we give a precise meaning to the expression

$$\mathcal{L}, s \models \varphi, \text{ or just } s \models \varphi \text{ when } \mathcal{L} \text{ is clear from context,}$$

which should capture that the state s has the property expressed by the formula φ.

The expressed properties are typically a sort of *safety properties* (something bad never happens; e.g., two processes are never in their critical section simultaneously), a sort of *liveness properties* (something good happens eventually; e.g., after a process asks to enter the critical section, it will eventually enter), or a (maybe complicated) combination of such properties.

One logic (set of formulas with their semantics) that plays an important role in verification is the so-called *μ-calculus*; we now describe one of its versions. Here the formulas are mainly thought as denoting the sets of states: $[[\varphi]] \subseteq S$, hence $s \models \varphi$ iff $s \in [[\varphi]]$.

It is technically useful to have also formulas with free variables $X, Y, Z, X_1, Y_1, \dots$, from a set $\textsc{Var}$, that are evaluated to sets of states. Formulas, denoted by φ, φ', φ_1, $\dots$, are defined by the following inductive definition, where $\textsc{p} \in \textsc{ap}$ and $X \in \textsc{Var}$:

$$\varphi ::= \textsc{p} \mid X \mid \neg\varphi' \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \langle - \rangle\varphi' \mid [-]\varphi' \mid \mu X.\varphi' \mid \nu X.\varphi'$$

where μ stands for "the least fixpoint" and in $\mu X.\varphi'$ it binds the free occurrences of X in φ', while ν stands for "the greatest fixpoint" and in $\nu X.\varphi'$ it also binds the free occurrences of X in φ'. In fact, there is also a syntactic monotonicity constraint: in formulas $\mu X.\varphi'$ and $\nu X.\varphi'$ we require that X is in the scope of an even number of negations.

Given $\mathcal{L} = (S, \textsc{ap}, \ell, \rightarrow)$, and a *valuation* $v : \textsc{Var} \rightarrow 2^S$, we define the denotation

$$[[\varphi]]_{\mathcal{L},v} \subseteq S$$

by the following structural induction:

- $[[\textsc{p}]]_{\mathcal{L},v} = \{s \in S \mid \textsc{p} \in \ell(s)\}$;
- $[[X]]_{\mathcal{L},v} = v(X)$;
- $[[\neg\varphi]]_{\mathcal{L},v} = S \smallsetminus [[\varphi]]_{\mathcal{L},v}$;
- $[[\varphi_1 \wedge \varphi_2]]_{\mathcal{L},v} = [[\varphi_1]]_{\mathcal{L},v} \cap [[\varphi_2]]_{\mathcal{L},v}$;
- $[[\varphi_1 \vee \varphi_2]]_{\mathcal{L},v} = [[\varphi_1]]_{\mathcal{L},v} \cup [[\varphi_2]]_{\mathcal{L},v}$;
- $[[\,\langle - \rangle\varphi\,]]_{\mathcal{L},v} = \{s \in S \mid \exists s' \in S : (s \rightarrow s' \text{ and } s' \in [[\varphi]]_{\mathcal{L},v})\}$;
- $[[\,[-]\varphi\,]]_{\mathcal{L},v} = \{s \in S \mid \forall s' \in S : (s \rightarrow s' \Rightarrow s' \in [[\varphi]]_{\mathcal{L},v})\}$;
- $[[\mu X.\varphi]]_{\mathcal{L},v}$ is the least fixpoint of the functional $\tau : 2^S \rightarrow 2^S$ where for $A \subseteq S$ we have $\tau(A) = [[\varphi]]_{\mathcal{L},v[X \leftarrow A]}$;
- $[[\nu X.\varphi]]_{\mathcal{L},v}$ is the greatest fixpoint of the above functional τ.

We add that by the valuation $v[X \leftarrow A]$ we denote the valuation $v' : \textsc{Var} \rightarrow 2^S$ such that $v'(X) = A$ and $v'(Y) = v(Y)$ for all $Y \neq X$.

Due to the above monotonicity constraint, the functional $\tau : 2^S \rightarrow 2^S$ is monotonic in the sense that $A \subseteq B$ implies $\tau(A) \subseteq \tau(B)$. Such a functional has the least fixpoint $\textsc{lfp}_\tau$ and the greatest fixpoint $\textsc{gfp}_\tau$ (w.r.t. set inclusion) by the well-known results (Tarski). (We thus have $\textsc{lfp}_\tau = \tau(\textsc{lfp}_\tau)$, $\textsc{gfp}_\tau = \tau(\textsc{gfp}_\tau)$, and for every $C \subseteq S$ where $C = \tau(C)$ we have $\textsc{lfp}_\tau \subseteq C \subseteq \textsc{gfp}_\tau$.) Concretely, we have

$$\text{LFP}_\tau = \bigcap\{X \subseteq S \mid X \supseteq \tau(X)\} \text{ and } \text{GFP}_\tau = \bigcup\{X \subseteq S \mid X \subseteq \tau(X)\}.$$

If S is finite, then LFP_τ is the stable set in the sequence $\emptyset \subseteq \tau(\emptyset) \subseteq \tau(\tau(\emptyset)) \subseteq \cdots$, and GFP_τ is the stable set in the sequence $S \supseteq \tau(S) \supseteq \tau(\tau(S)) \supseteq \cdots$.

Example of φ expressing that there is a computation going via a state where $crit_1 \wedge crit_2$:

$$\mu X.\, ((crit_1 \wedge crit_2) \vee \langle - \rangle X).$$

Hence $[[\varphi]]$ contains all states from which there is a computation visiting a state in which $(crit_1 \wedge crit_2)$ is true (processes 1 and 2 are in the critical section at the same time). The negation $\neg \mu X.\, ((crit_1 \wedge crit_2) \vee \langle - \rangle X)$ is equivalent to the formula $\nu X.\, ((\neg crit_1 \vee \neg crit_2) \wedge [-]X)$.

Generally we have that $\neg \mu X.\varphi(X)$ is equivalent to $\nu X.\neg\varphi(\neg X/X)$.

(To a monotonic functional $\tau : 2^S \to 2^S$ we can attach the functional $\tau' : 2^S \to 2^S$ where $\tau'(A) = \overline{\tau(\overline{A})}$; by $\overline{B}$ we denote $S \smallsetminus B$. We note that $A \subseteq B$ entails $\overline{A} \supseteq \overline{B}$, hence $\tau(\overline{A}) \supseteq \tau(\overline{B})$, and thus $\overline{\tau(\overline{A})} \subseteq \overline{\tau(\overline{B})}$; τ' is thus monotonic as well. We observe that $\tau'(A) = A$ iff $\tau(\overline{A}) = \overline{A}$; hence $\overline{\text{LFP}_\tau} = \text{GFP}_{\tau'}$.)

We considered the computational problem MCMC (model checking μ-calculus):

Instance: $\mathcal{L} = (S, \text{AP}, \ell, \to)$, and a closed formula φ (of the above μ-calculus).

Output: the set $[[\varphi]]_{\mathcal{L}}$.

We have sketched the idea of a polynomial reduction of MCMC to the problem of *parity games*, denoted PG. We describe one possible version of such games.

An *instance of* PG is a directed graph $G = (V, E)$, also called a *game arena*, or just a *game*, where the set V of vertices is a finite subset of $\mathbb{N} = \{0, 1, 2, \dots\}$, and $E \subseteq V \times V$ satisfies $E(v) \neq \emptyset$ for all $v \in V$. (We use the notation $E(v) = \{v' \mid (v, v') \in E\}$.) For technical convenience we also consider the *empty game* where $V = \emptyset$.

A *play from* $v \in V$ is a finite, or infinite, path starting in v, i.e., a sequence $\pi = v_0, v_1, v_2, \dots, v_k$ ($k \in \mathbb{N}$ where $\mathbb{N} = \{0, 1, 2, \dots\}$), or $\pi = v_0, v_1, v_2, \dots$, where $v_0 = v$ and $v_i \in E(v_{i-1})$ for all $i \in [1, k]$, or for all $i \geq 1$. (By $[i, j]$ we denote the set $\{i, i{+}1, i{+}2, \dots, j\}$.)

We imagine two players, an *even player* P_0 (who chooses next moves in even vertices), and an *odd player* P_1 (who chooses in odd vertices). A (partial) P_0*-strategy from* v is a nonempty (finite or infinite) prefix-closed set S of plays from v (if $\pi \in$ S and π' is a nonempty prefix of π, then $\pi' \in$ S) such that for each finite play $v_0, v_1, \dots, v_\ell$ in S the following holds:

- if v_ℓ is even, then there is at most one play $v_0, v_1, \dots, v_\ell, v$ in S (for $v \in E(v_\ell)$), and
- if v_ℓ is odd, then either there is no play $v_0, v_1, \dots, v_\ell, v$ in S or S contains the plays $v_0, v_1, \dots, v_\ell, v$ for all $v \in E(v_\ell)$.

A play $\pi \in$ S that is not a proper prefix of another $\pi' \in$ S (hence π is maximal in S) is called a *branch of* S (it can be finite or infinite). A (partial) P_1*-strategy from* v is defined symmetrically: if v_ℓ is odd, then there is at most one $v_0, v_1, \dots, v_\ell, v$ in S, and if v_ℓ is even, then either there is no play $v_0, v_1, \dots, v_\ell, v$ in S or S contains $v_0, v_1, \dots, v_\ell, v$ for all $v \in E(v_\ell)$. A (P_0- or P_1-) *strategy* S from v is *complete* if each branch of S is infinite.

An *infinite play* π (from some v) is a P_0*-win* if the **least** vertex occurring in π infinitely often is even; if this vertex is odd, then π is a P_1*-win.* A *complete* P_0*-strategy* S *from* v is *winning* if each branch of S is a P_0-win. A *complete* P_1*-strategy* S *from* v is *winning* if each branch of S is a P_1-win. For $i \in \{0,1\}$ we put

$$\textsc{Win}_i(G) = \{v \in V \mid \text{there is a winning } P_i\text{-strategy from } v\}.$$

It is standard to derive that $\textsc{Win}_0(G) \cap \textsc{Win}_1(G) = \emptyset$ and $\textsc{Win}_0(G) \cup \textsc{Win}_1(G) = V$.

We finish a description of the problem PG:

Instance: $G = (V, E)$;

Output: $\textsc{Win}_0(G)$ (and $\textsc{Win}_1(G) = V \smallsetminus \textsc{Win}_0(G)$).

We recall that it is natural to require that an algorithm solving PG returns not only $\textsc{Win}_0(G)$ and $\textsc{Win}_1(G)$ but also the respective winning P_0- and P_1-strategies (for all vertices). It is well known that these strategies can be given compactly since they can be chosen as history-free (a.k.a. positional, or memoryless), where the continuation of $v_0, v_1, \dots, v_\ell$ is determined solely by v_ℓ.

We have shown a proof of the fact that the optimal strategies can be chosen among the positional strategies.

The credit task for students:

Based on this fact, show that the following decision problem PG_{dec}

Instance: $G = (V, E)$ and $v \in V$;

Question: is v in $\textsc{Win}_0(G)$?

is in NP $\cap$ co-NP.

Remark. The possible polynomiality of the problem (i.e., its membership in PTIME) is a well-known open problem, now for more than three decades ...