PROPERTIES PROOF METHOD IN IPCL APPLICATION TO REAL-WORLD SYSTEM CORRECTNESS PROOF

Mykyta Kartavov, Taras Panchenko, Nataliya Polishchuk

***Abstract**: The correctness proof for programs with parallelism, and interleaving concurrency with shared memory in particular, is complicated problem because the state of separate execution thread can be changed even in ready-to-run (waiting) time due to possible inference of one execution path over the other via shared data or messaging mechanics. Classical methods like Floyd-Hoare cannot be applied directly to this case and new non-trivial methods are required to cope with such a complexity. The safety property proof of real-world system using method for software correctness proof in Interleaving Parallel Composition Language (for defined custom class of programs – namely server software for Symmetric Multi-Processing architecture like DB-server or Web-server) is the subject of this article. Operational semantics of the system is defined in terms of state transitions. Program Invariant as well as Pre- and Post- conditions are formulated in accordance with the methodology. Conclusions about adequacy of the Method usage for such a kind of tasks (thanks to flexibility of composition-nominative platform) and its practicality as well as ease of use for real-world systems have been made based on this and other authors' works.*

***Keywords**: software correctness, safety proof, concurrent program, interleaving, invariant, IPCL, composition approach, composition-nominative languages, formal verification.*

***ACM Classification Keywords**: F.3.1 Theory of Computation - LOGICS AND MEANINGS OF PROGRAMS - Specifying and Verifying and Reasoning about Programs, D.2.4 Software - SOFTWARE ENGINEERING - Software/Program Verification.*

Introduction

The problem of software correctness proof is quite relevant nowadays. There are a lot of scientific researches and methods devoted to program verification, but nevertheless the problem stays relevant, because most of methods are either too complicated for practical application or too theorized (which makes them impractical – and here emerges the question of transferring these theoretical results into a more practical field), or simply unable to cope with real tasks. At the same time, contemporary software becomes more and more parallel (the increase in processors' core clock frequency started to slow down, forcing the increase in number of them, therefore stimulating code parallelization), but classical formal methods of verification are not well suited for such kind of tasks, where mutual in influence between parallel processes is present [Panchenko, 2006, Panchenko, 2007, Panchenko, 2008, Panchenko2, 2007]. Special interest is devoted to systems based on shared memory concurrency which are less investigated [Panchenko, 2006]. Those are supercomputers with UMA and NUMA memory architecture, SMP-based computer hardware architectures, operating systems, database management systems (DBMS), centralized databases and data warehouses (for example, in Business Intelligence systems), server-side software in client-server environments, etc.

Regarding the necessity to prove the correctness of programs, it is mostly related to so-called safety-critical systems – systems, whose failure or malfunction may result in death or injury to people health, loss or severe damage to property and/or equipment or environmental harm. According to Trusted Computer System Evaluation Criteria [DD, 1985] (the famous “Orange book”), formal specification and verification of programs that are claimed to be of the highest level of reliability is needed. In a more contemporary of Computer System Evaluation Criteria, which is standard ISO/IEC 15408 [ISO, 2005] (“Common Criteria”), formal verification is demanded for 3 out of 7 levels of reliability – EAL5-EAL7 (Evaluation Assurance Level), which means that requirements have strengthened even more.

The Problem

System Description

In this work we will prove the safety property of partial correctness of an Infosoft e-Detailing 1.0 software system. This software is designed for making (almost) synchronous presentations by one speaker (manager) to a numerous audience (client). The usage of this system basically lies in switching slides on a manager's device which is almost immediately followed by an automatic switching to the same slide on each of the clients' devices. This product is commercial, hence we are not going to include the source code, but we are including the same (slightly simplified) code written in compositional language IPCL [Panchenko, 2004], which is going to be used in proofs. Compositional nominative languages IPCL provide means of working with any kind of parallelism [Panchenko2, 2008] and cover the most common class of parallel programs – MIMD architecture, according to Flynn's taxonomy.

The most important from the correct functioning point of view is to make sure that every client will see the same slide that manager has switched to. Work of the system consists of cycles, namely switching a slide on manager's device and then switching a slide on all of clients' devices. The amount of such cycles is unlimited, the only stopping criteria is that everyone has left their presentation session. Typical cycle would look like this: manager sends to the server, and clients are reading from it, the index of a current slide (currently using HTTP + AJAX + Long Poll technologies) – in such a way the asynchronous slide replication is achieved on all the devices.

The problem statement is to prove the correctness of one typical slide switching cycle. More precisely: if manager has switched to a new slide (this slide has index $slideM$) and notified the server about it (S common variable on the server, which holds the current slide index for every client to read, and at the beginning $S \neq slideM$, in other words manager has switched to a slide that is different from the previous one), then eventually all the clients will have their own slide index (for each client i – $sldieC_i$) equal to the slide index that manager has switched to, i.e. for each client we have:

$$\forall i(valueOf(slideC_i) = valueOf(S) = valueOf(slideM))$$

All presentations by default begin from the first slide, in other words at the beginning we have:

$$valueOf(slideM) = valueOf(slideC_i) = valueOf(S) = 1$$

Stages in Correctness Proof

The sources of manager and client programs in IPCL and verification using method for program's properties correctness proof [Panchenko, 2006, Panchenko, 2008, Panchenko, 2004] are given below, namely:

- The notion of a state is specified;
- Transition system is constructed (model of execution of the program **manager** || **client** $\boldsymbol{^n}$);
- Starting and final states as well as precondition and postcondition are specified;
- Invariant of the software system is introduced, and it is proven that each of macrotransitions keeps it true, and that precondition on starting states implies invariant, and invariant on final states implies postcondition.

System Formal Model

Sources of manager and client programs with labels (in accordance with the method for program's properties correctness proof):

manager ≡ **[M1]** S := slideM **[M2]**

client ≡ **[C1]** newSlide := S;

while **[C2]** (slideC = newSlide) do

    [C3] newSlide := S;

[C4] slideC := newSlide **[C5]**

The whole software system will have the following structure:

$$\textbf{program} = \textbf{manager} \mid\mid \textbf{client}^{\,n}.$$

The power here is understood in a sense [Panchenko, 2006, Panchenko, 2007, Panchenko, 2008, Panchenko2, 2007, Panchenko, 2004], i.e. parallel execution of n instances of a program in an interleaving manner.

States of the System

The state of such program will have the following structure:

$$State = (M, CS, [S \mapsto S_0], [slideM \mapsto SM], CD)$$

where $M \in \{\mathbf{M1},\mathbf{M2}\}$ – manager's labels, $CS = (s_1,\ldots,s_n)$ – clients' labels, where n – number of clients, $\forall i\ (s_i \in \{\mathbf{C1},\mathbf{C2},\mathbf{C3},\mathbf{C4},\mathbf{C5}\})$, $[S \mapsto S_0]$ – global (common) data, $[slideM \mapsto SM]$ – manager's local data, $CD = (d_1,\ldots,d_n)$ – clients' local data, where $\forall i\ (d_i = [newSlide \mapsto NSi, slideC \mapsto SCi])$, $\forall i\ (\{S_0, SM, NS_i, SC_i\} \subset \mathbf{N})$, – slide indices. $States$ is a set of all possible states.

Here we will use standard in composition-nominative approach denomination composition $A \Rightarrow$ [Redko, 1978, Nikitchenko, 1998], which returns the value of variable with name A over the data d:

$$A\Rightarrow (d) = w \Leftrightarrow [A \mapsto w] \in d$$

Also we will denote $s_i(S) = Pr_i(Pr_2(S))$ and $d_i(S) = Pr_i(Pr_5(S))$.

Transition System

The transition system will have following macro-transitions (the scheme of transitions):

$$Transitions = \{\ S_1 \to S_2 \mid S_1, S_2 \in States \wedge (Tr_1(S_1,S_2) \vee Tr_2(S_1,S_2) \vee Tr_3(S_1,S_2) \vee Tr_4(S_1,S_2) \vee Tr_5(S_1,S_2) \vee Tr_6(S_1,S_2))\ \}$$

where each of $Tr_i(S_1,S_2)$ corresponds to some of possible program atomic steps (which will be executed in interleaving manner somehow due to concurrent environment during runtime execution path) and defines the semantics of such a step (i.e. corresponding transition between states), namely:

1) for **manager**'s move from label **M1** to label **M2**:

$Tr_1(S_1,S_2) = (Pr_1(S_1) = \mathbf{M1}) \wedge (Pr_1(S_2) = \mathbf{M2}) \wedge (Pr_2(S_1) = Pr_2(S_2)) \wedge (Pr_3(S_1) = d) \wedge (Pr_3(S_2) = d \nabla [S \mapsto SM]) \wedge (Pr_4(S_1) = Pr_4(S_2) = [slideM \mapsto SM]) \wedge (Pr_5(S_1) = Pr_5(S_2))$

2) for **client**'s move **C1** → **C2** (pre-while-cycle assignment):

$Tr_2(S_1,S_2) = (Pr_1(S_1) = Pr_1(S_2)) \wedge (Pr_3(S_1) = Pr_3(S_2) = [S \mapsto S_0]) \wedge (Pr_4(S_1) = Pr_4(S_2)) \wedge \exists j (s_j(S_1) = \mathbf{C1} \wedge s_j(S_2) = \mathbf{C2} \wedge \forall i\ (i \neq j \rightarrow s_i(S_1) = s_i(S_2) \wedge d_i(S_1) = d_i(S_2)) \wedge (d_j(S_1) = d) \wedge (d_j(S_2) = d \nabla [newSlide \mapsto S_0]))$

3) for **client**'s move **C2** → **C4** ($false$ value of while-cycle condition):

$Tr_3(S_1,S_2) = (Pr_1(S_1) = Pr_1(S_2)) \wedge (Pr_3(S_1) = Pr_3(S_2)) \wedge (Pr_4(S_1) = Pr_4(S_2)) \wedge (Pr_5(S_1) = Pr_5(S_2)) \wedge \exists j\ (s_j(S_1) = \mathbf{C2} \wedge s_j(S_2) = \mathbf{C4} \wedge \forall i\ (i \neq j \rightarrow s_i(S_1) = s_i(S_2)) \wedge slideC{\Rightarrow}(d_j(S_1)) \neq newSlide{\Rightarrow}(d_j(S_1)))$

4) for **client**'s move **C2** → **C3** ($true$ value of while-cycle condition):

$Tr_4(S_1,S_2) = (Pr_1(S_1) = Pr_1(S_2)) \wedge (Pr_3(S_1) = Pr_3(S_2)) \wedge (Pr_4(S_1) = Pr_4(S_2)) \wedge (Pr_5(S_1) = Pr_5(S_2)) \wedge \exists j\ (s_j(S_1) = \mathbf{C2} \wedge s_j(S_2) = \mathbf{C3} \wedge \forall i\ (i \neq j \rightarrow s_i(S_1) = s_i(S_2)) \wedge slideC{\Rightarrow}(d_j(S_1)) = newSlide{\Rightarrow}(d_j(S_1)))$

5) for **client**'s move **C3** → **C2** (while-cycle body assignment statement execution):

$Tr_5(S_1,S_2) = (Pr_1(S_1) = Pr_1(S_2)) \wedge (Pr_3(S_1) = Pr_3(S_2) = [S \mapsto S_0]) \wedge (Pr_4(S_1) = Pr_4(S_2)) \wedge \exists j$ $(s_j(S_1) = \mathbf{C3} \wedge s_j(S_2) = \mathbf{C2} \wedge \forall i\ (i \neq j \to s_i(S_1) = s_i(S_2) \wedge d_i(S_1) = d_i(S_2)) \wedge (d_j(S_1) = d) \wedge (d_j(S_2) = d\ \nabla\ [newSlide \mapsto S_0]))$

6) for **client**'s move **C4** → **C5** (post-while-cycle assignment):

$Tr_6(S_1,S_2) = (Pr_1(S_1) = Pr_1(S_2)) \wedge (Pr_3(S_1) = Pr_3(S_2)) \wedge (Pr_4(S_1) = Pr_4(S_2)) \wedge \exists j\ (s_j(S_1) = \mathbf{C4} \wedge s_j(S_2) = \mathbf{C5} \wedge \forall i\ (i \neq j \to s_i(S_1) = s_i(S_2) \wedge d_i(S_1) = d_i(S_2)) \wedge (d_j(S_1) = d) \wedge (d_j(S_2) = d\ \nabla\ [slideC \mapsto newSlide \Rightarrow (d_j(S_1))]))$

Invariant of the Program

Now let us fix Starting states for the transition system described:

$$StartStates=\{S \in States | Pr_1(S)=\mathbf{M1} \wedge \forall i\ (s_i(S)=\mathbf{C1})\}$$

and Final states for this system are:

$$StopStates=\{S \in States | Pr_1(S)=\mathbf{M2} \wedge \forall i\ (s_i(S)=\mathbf{C5})\}$$

To prove the safety condition we formulate Precondition:

$$PreCond(S) = \forall i\ (slideC \Rightarrow (d_i(S)) = S \Rightarrow (Pr_3(S))) \wedge (slideM \Rightarrow (Pr_4(S)) \neq S \Rightarrow (Pr_3(S)))$$

and Postcondition:

$$PostCond(S) = \forall i\ (slideC \Rightarrow (d_i(S)) = S \Rightarrow (Pr_3(S))) \wedge (slideM \Rightarrow (Pr_4(S)) = S \Rightarrow (Pr_3(S)))$$

The invariant of **program** system:

$Inv(S) = I_1(S) \wedge I_2(S) \wedge I_3(S) \wedge I_4(S) \wedge I_5(S)$, where

$$I_1(S) = (Pr_1(S) = M2 \rightarrow S \Rightarrow (Pr_3(S)) = slideM \Rightarrow (Pr_4(S))),$$

$$I_2(S) = (Pr_1(S) = M1 \rightarrow (S \Rightarrow \big(Pr_3(S)\big) \neq slideM \Rightarrow \big(Pr_4(S)\big) \wedge \forall\, i\ (slideC \Rightarrow \big(d_i(S)\big) = S \Rightarrow \big(Pr_3(S)\big) \wedge s_i(S) \in \{C1, C2, C3\}))),$$

$$I_3(S) = \forall\, i\ (s_i(S) = C4 \rightarrow (slideC \Rightarrow \big(d_i(S)\big) \neq S \Rightarrow \big(Pr_3(S)\big) \wedge newSlide \Rightarrow (d_i(S)) = S \Rightarrow (Pr_3(S)))),$$

$$I_4(S) = \forall\, i\ (s_i(S) = C5 \rightarrow slideC \Rightarrow (d_i(S)) = S \Rightarrow (Pr_3(S))),$$

$$I_5(S) = \forall\, i\ (s_i(S) = C1 \vee slideC \Rightarrow \big(d_i(S)\big) = newSlide \Rightarrow \big(d_i(S)\big) \vee Pr_1(S) = M2 \wedge newSlide \Rightarrow (d_i(S)) = S \Rightarrow (Pr_3(S))).$$

The Proof

To proof the safety condition of the program we need to be sure that Precondition (S) implies Invariant $Inv(S)$ over all $S \in StartStates$, Postcondition $PostCont(S)$ follows from Invariant $Inv(S)$ over all $S \in StopStates$ and also the $Inv(S)$ preserves $true$ value over all possible $Transitions$. In other words, this needs to be proven in accordance with the Method:

$$\forall\, S \in StartStates\ \big(PreCond(S) \rightarrow Inv(S)\big) \wedge$$
$$\forall\, S \in StopStates\ \big(Inv(S) \rightarrow PostCond(S)\big) \wedge$$
$$\forall\, (S_1 \rightarrow S_2) \in Transitions \wedge S_1, S_2 \in States\ (Inv(S_1) \rightarrow Inv(S_2)).$$

To make the proof simple, let's prove this in terms of first order logic, using Gödel's completeness theorem:

1. $PreCond(S) = true \vDash Inv(S) = true, S \in StartStates$

$a.\ S \in StartStates \vDash (I_1(S) \land I_3(S) \land I_4(S) \land I_5(S))$

$b.\ (S \in StartStates \land \forall\, i\ (slideC \Rightarrow (d_i) = S \Rightarrow (Pr_3(S))) \land slideM \Rightarrow (Pr_4(S))$
$\neq S \Rightarrow (Pr_3(S)))$

$\vDash (S \Rightarrow (Pr\_3\ (S)) \neq slideM \Rightarrow (Pr\_4\ (S)) \land \forall\, i\ (slideC \Rightarrow (d\_i\ (S)) = S$
$\Rightarrow (Pr\_3\ (S)) \land s\_i\ (S) \in \{C1, C2, C3\})) \vDash I\_2\ (S)$

$\left(I_1(S) \land I_2(S) \land I_3(S) \land I_4(S) \land I_5(S)\right) \vDash Inv(S)$ ■

2. $Inv(S) = true \vDash PostCond(S) = true, S \in StopStates$

$a.\ \left(S \in StopStates \land I_4(S)\right) \vDash \forall\, i\ (slideC \Rightarrow (d_i) = S \Rightarrow (Pr_3(S)))$

$b.\ \left(S \in StopStates \land I_1(S)\right) \vDash S \Rightarrow \left(Pr_3(S)\right) = slideM \Rightarrow (Pr_4(S))$

$c.\ (\forall\, i\ (slideC \Rightarrow (d_i) = S \Rightarrow (Pr_3(S))) \land S \Rightarrow (Pr_3(S)) = slideM$
$\Rightarrow (Pr_4(S))) \vDash PostCond(S)$ ■

3. $Tr_1(S_1, S_2) = true \land Inv(S_1) = true \vDash Inv(S_2) = true$

$a.\ (Pr_3(S_1) = d \land Pr_3(S_2) = d \nabla [S \mapsto SM] \land Pr_4(S_1) = Pr_4(S_2)$
$= [slideM \mapsto SM]) \vDash S \Rightarrow \left(Pr_3(S_2)\right) = slideM \Rightarrow \left(Pr_4(S_2)\right)$
$\vDash I_1(S_2)$

$b.\ Pr_1(S_2) = M2 \vDash I_2(S_2)$

$c.\ (I_2(S_1) \land Pr_2(S_1) = Pr_2(S_2)) \vDash \forall\, i\ (s_i(S_2) \notin \{C4, C5\}) \vDash (I_3(S_2) \land I_4(S_2))$

$d.\ (Pr_1(S_1) = M1 \land I_5(S_1)) \vDash \forall\, i\ (slideC \Rightarrow (d_i(S_1)) = newSlide$
$\Rightarrow (d_i(S_1)))$

$(\forall\, i\ (slideC \Rightarrow (d_i(S_1)) = newSlide \Rightarrow (d_i(S_1))) \land Pr_5(S_1) = Pr_5(S_2))$
$\vDash \forall\, i\ (slideC \Rightarrow (d_i(S_2)) = newSlide \Rightarrow (d_i(S_2))) \vDash I_5(S_2)$

$\left(I_1(S) \land I_2(S) \land I_3(S) \land I_4(S) \land I_5(S)\right) \vDash Inv(S)$ ■

4. $Tr_2(S_1, S_2) = true \land Inv(S_1) = true \vDash Inv(S_2) = true$

$a.\ (I_1(S_1) \land Pr_1(S_1) = Pr_1(S_2) \land Pr_3(S_1) = Pr_3(S_2) \land Pr_4(S_1) = Pr_4(S_2)) \vDash I_1(S_2)$

$b.\ (Pr_1(S_1) = Pr_1(S_2) \land Pr_3(S_1) = Pr_3(S_2) \land Pr_4(S_1) = Pr_4(S_2) \land d_j(S_1) = d \land d_j(S_2) = d \nabla [newSlide \mapsto S_0] \land \forall i = \overline{1,n} \land i \neq j \left(d_i(S_1) = d_i(S_2)\right) \land s_j(S_1) = C1 \land s_j(S_2) = C2 \land \forall i = \overline{1,n} \land i \neq j\ (s_i(S_1) = s_i(S_2))) \vDash I_2(S_2)$

$c.\ (d_j(S_1) = d \land d_j(S_2) = d \nabla [newSlide \mapsto S_0] \land \forall i = \overline{1,n} \land i \neq j\ (d_i(S_1) = d_i(S_2)) \land Pr_3(S_1) = Pr_3(S_2)) \vDash \forall i\ (slideC \Rightarrow (d_i(S_1)) = slideC \Rightarrow (d_i(S_2)))$

$(I_3(S_1) \land I_4(S_1) \land \forall i\ (slideC \Rightarrow (d_i(S_1)) = slideC \Rightarrow (d_i(S_2))) \land d_j(S_1) = d \land d_j(S_2) = d \nabla [newSlide \mapsto S_0] \land \forall i = \overline{1,n} \land i \neq j\ (d_i(S_1) = d_i(S_2))) \vDash (I_3(S_2) \land I_4(S_2))$

$d.\ \left(I_1(S_2) \land I_2(S_2) \land I_5(S_1) \land d_j(S_1) = d \land d_j(S_2) = d \nabla [newSlide \mapsto S_0] \land \forall i = \overline{1,n} \land i \neq j \left(d_i(S_1) = d_i(S_2)\right) \land Pr_3(S_2) = [S \mapsto S_0]\right) \vDash I_5(S_2)$

$\left(I_1(S) \land I_2(S) \land I_3(S) \land I_4(S) \land I_5(S)\right) \vDash Inv(S)$ ■

5. $Tr_3(S_1, S_2) = true \land Inv(S_1) = true \vDash Inv(S_2) = true$

$a.\ (I_1(S_1) \land I_5(S_1) \land Pr_1(S_1) = Pr_1(S_2) \land Pr_3(S_1) = Pr_3(S_2) \land Pr_4(S_1) = Pr_4(S_2) \land Pr_5(S_1) = Pr_5(S_2)) \vDash (I_1(S_2) \land I_5(S_2))$

$b.\ (slideC \Rightarrow (d_j(S_2)) \neq newSlide \Rightarrow (d_j(S_2)) \land Pr_5(S_1) = Pr_5(S_2) \land I_5(S_1)) \vDash Pr_1(S_1) = M2$

$(Pr_1(S_1) = Pr_1(S_2) \land Pr_1(S_1) = M2) \vDash Pr_1(S_2) = M2 \vDash I_2(S_2)$

$c.\ (Pr_3(S_1) = Pr_3(S_2) \land Pr_5(S_1) = Pr_5\ (S_2) \land \forall i = \overline{1,n} \land i \neq j\ (s_i(S_1) = s_i(S_2))) \vDash (\forall i = \overline{1,n} \land i \neq j\ (s_i(S_2) = C4 \rightarrow (slideC \Rightarrow \left(d_i(S_2)\right) \neq S \Rightarrow (Pr_3(S_2)) \land newSlide \Rightarrow (d_i(S_2)) = S \Rightarrow (Pr_3(S_2)))) \land \forall i = \overline{1,n} \land i \neq j\ (s_i(S_2) = C5 \rightarrow slideC \Rightarrow (d_i(S_2)) = S \Rightarrow (Pr_3(S_2))))$

$$s_j(S_2) = C4 \vDash (s_j(S_2) = C5 \rightarrow slideC \Rightarrow (d_j(S_2)) = S \Rightarrow (Pr_3(S_2)))$$

$$(Pr_3(S_1) = Pr_3(S_2) \wedge Pr_5(S_1) = Pr_5(S_2) \wedge s_j(S_2) = C4 \wedge slideC \Rightarrow (d_j(S_2)) \neq newSlide \Rightarrow (d_j(S_2)) \wedge I_5(S_2)) \vDash (s_j(S_2) = C4 \rightarrow (slideC \Rightarrow (d_j(S_2)) \neq S \Rightarrow (Pr_3(S_2)) \wedge newSlide \Rightarrow (d_j(S_2)) = S \Rightarrow (Pr_3(S_2))))$$

$$(\forall i = \overline{1,n} \wedge i \neq j\ (s_i(S_2) = C4 \rightarrow (slideC \Rightarrow \left(d_i(S_2)\right) \neq S \Rightarrow \left(Pr_3(S_2)\right) \wedge newSlide \Rightarrow (d_i(S_2)) = S \Rightarrow (Pr_3(S_2)))) \wedge \forall i = \overline{1,n} \wedge i \neq j\ (s_i(S_2) = C5 \rightarrow slideC \Rightarrow (d_i(S_2)) = S \Rightarrow (Pr_3(S_2))) \wedge (s_j(S_2) = C4 \rightarrow (slideC \Rightarrow \left(d_j(S_2)\right) \neq S \Rightarrow (Pr_3(S_2)) \wedge newSlide \Rightarrow (d_j(S_2)) = S \Rightarrow (Pr_3(S_2)))) \wedge (s_j(S_2) = C5 \rightarrow slideC \Rightarrow (d_j(S_2)) = S \Rightarrow (Pr_3(S_2)))) \vDash (I_3(S_2) \wedge I_4(S_2))$$

$$\left(I_1(S) \wedge I_2(S) \wedge I_3(S) \wedge I_4(S) \wedge I_5(S)\right) \vDash Inv(S)\ \blacksquare$$

6. $Tr_4(S_1, S_2) = true \wedge Inv(S_1) = true \vDash Inv(S_2) = true$

$a.$ $(I_1(S_1) \wedge I_5(S_1) \wedge Pr_1(S_1) = Pr_1(S_2) \wedge Pr_3(S_1) = Pr_3(S_2) \wedge Pr_4(S_1) = Pr_4(S_2) \wedge Pr_5(S_1) = Pr_5(S_2)) \vDash (I_1(S_2) \wedge I_5(S_2))$

$b.$ $(Pr_1(S_1) = Pr_1(S_2) \wedge Pr_3(S_1) = Pr_3(S_2) \wedge Pr_4(S_1) = Pr_4(S_2) \wedge Pr_5(S_1) = Pr_5(S_2) \wedge s_j(S_1) = C2 \wedge s_j(S_2) = C3 \wedge \forall i = \overline{1,n} \wedge i \neq j\ (s_i(S_1) = s_i(S_2))) \vDash I_2(S_2)$

$c.$ $(Pr_3(S_1) = Pr_3(S_2) \wedge Pr_5(S_1) = Pr_5(S_2) \wedge \forall i = \overline{1,n} \wedge i \neq j\ (s_i(S_1) = s_i(S_2))) \vDash (\forall i = \overline{1,n} \wedge i \neq j\ (s_i(S_2) = C4 \rightarrow (slideC \Rightarrow (d_i(S_2)) \neq S \Rightarrow (Pr_3(S_2)) \wedge newSlide \Rightarrow (d_i(S_2)) = S \Rightarrow (Pr_3(S_2)))) \wedge \forall i = \overline{1,n} \wedge i \neq j\ (s_i(S_2) = C5 \rightarrow slideC \Rightarrow (d_i(S_2)) = S \Rightarrow (Pr_3(S_2))))$

$$\big(Pr_3(S_1) = Pr_3(S_2) \wedge Pr_5(S_1) = Pr_5(S_2) \wedge s_j(S_2) = C3\big) \vDash ((s_j(S_2) = C4 \rightarrow (slideC \Rightarrow (d_j(S_2)) \neq S \Rightarrow (Pr_3(S_2)) \wedge newSlide \Rightarrow (d_j(S_2)) = S \Rightarrow (Pr_3(S_2)))) \wedge (s_j(S_2) = C5 \rightarrow slideC \Rightarrow (d_j(S_2)) = S \Rightarrow (Pr_3(S_2))))$$

$$(\forall\, i = \overline{1,n} \wedge i \neq j\ (s_i(S_2) = C4 \rightarrow (slideC \Rightarrow \big(d_i(S_2)\big) \neq S \Rightarrow (Pr_3(S_2)) \wedge newSlide \Rightarrow (d_i(S_2)) = S \Rightarrow (Pr_3(S_2)))) \wedge \forall\, i = \overline{1,n} \wedge i \neq j\ (s_i(S_2) = C5 \rightarrow slideC \Rightarrow (d_i(S_2)) = S \Rightarrow (Pr_3(S_2))) \wedge (s_j(S_2) = C4 \rightarrow (slideC \Rightarrow (d_j(S_2)) \neq S \Rightarrow (Pr_3(S_2)) \wedge newSlide \Rightarrow (d_j(S_2)) = S \Rightarrow (Pr_3(S_2)))) \wedge (s_j(S_2) = C5 \rightarrow slideC \Rightarrow (d_j(S_2)) = S \Rightarrow (Pr_3(S_2)))) \vDash (I_3(S_2) \wedge I_4(S_2))$$

$$\big(I_1(S) \wedge I_2(S) \wedge I_3(S) \wedge I_4(S) \wedge I_5(S)\big) \vDash Inv(S)\ \blacksquare$$

7. $Tr_5(S_1, S_2) = true \wedge Inv(S_1) = true \vDash Inv(S_2) = true$

$a.$ $(I_1(S_1) \wedge Pr_1(S_1) = Pr_1(S_2) \wedge Pr_3(S_1) = Pr_3(S_2) \wedge Pr_4(S_1) = Pr_4(S_2)) \vDash I_1(S_2)$

$b.$ $(Pr_1(S_1) = Pr_1(S_2) \wedge Pr_3(S_1) = Pr_3(S_2) \wedge Pr_4(S_1) = Pr_4(S_2) \wedge d_j(S_1) = d \wedge d_j(S_2) = d \nabla [newSlide \mapsto S_0] \wedge \forall\, i = \overline{1,n} \wedge i \neq j \big(d_i(S_1) = d_i(S_2)\big) \wedge s_j(S_1) = C3 \wedge s_j(S_2) = C2 \wedge \forall\, i = \overline{1,n} \wedge i \neq j\ (s_i(S_1) = s_i(S_2))) \vDash I_2(S_2)$

$c.$ $(d_j(S_1) = d \wedge d_j(S_2) = d \nabla [newSlide \mapsto S_0] \wedge \forall\, i = \overline{1,n} \wedge i \neq j\ (d_i(S_1) = d_i(S_2)) \wedge Pr_3(S_1) = Pr_3(S_2)) \vDash \forall\, i\ (slideC \Rightarrow (d_i(S_1)) = slideC \Rightarrow (d_i(S_2)))$

$(I_3(S_1) \wedge I_4(S_1) \wedge \forall\, i\ (slideC \Rightarrow (d_i(S_1)) = slideC \Rightarrow (d_i(S_2))) \wedge d_j(S_1) = d \wedge d_j(S_2) = d \nabla [newSlide \mapsto S_0] \wedge \forall\, i = \overline{1,n} \wedge i \neq j\ (d_i(S_1) = d_i(S_2))) \vDash (I_3(S_2) \wedge I_4(S_2))$

$d.$ $(I_1(S_2) \wedge I_2(S_2) \wedge I_5(S_1) \wedge d_j(S_1) = d \wedge d_j(S_2) = d \nabla [newSlide \mapsto S_0] \wedge \forall\, i = \overline{1,n} \wedge i \neq j\ (d_i(S_1) = d_i(S_2)) \wedge Pr_3(S_2) = [S \mapsto S_0]) \vDash I_5(S_2)$

$$\big(I_1(S) \wedge I_2(S) \wedge I_3(S) \wedge I_4(S) \wedge I_5(S)\big) \vDash Inv(S)\ \blacksquare$$

8. $Tr_6(S_1, S_2) = true \wedge Inv(S_1) = true \vDash Inv(S_2) = true$

$$a.\ (I_1(S_1) \wedge Pr_1(S_1) = Pr_1(S_2) \wedge Pr_3(S_1) = Pr_3(S_2) \wedge Pr_4(S_1) = Pr_4(S_2)) \vDash I_1(S_2)$$

$$b.\ (I_5(S_1) \wedge Pr_1(S_1) = Pr_1(S_2) \wedge Pr_3(S_1) = Pr_3(S_2) \wedge d_j(S_1) = d \wedge d_j(S_2) = d \nabla [slideC \mapsto newSlide \Rightarrow d_j(S_2)] \wedge \forall i = \overline{1,n} \wedge i \neq j\ (d_i(S_1) = d_i(S_2))) \vDash I_5(S_2)$$

$$c.\ (Pr_3(S_1) = Pr_3(S_2) \wedge \forall i = \overline{1,n} \wedge i \neq j\ (d_i(S_1) = d_i(S_2)) \wedge \forall i = \overline{1,n} \wedge i \neq j\ (s_i(S_1) = s_i(S_2))) \vDash (\forall i = \overline{1,n} \wedge i \neq j\ (s_i(S_2) = C4 \rightarrow (slideC \Rightarrow (d_i(S_2)) \neq S \Rightarrow (Pr_3(S_2)) \wedge newSlide \Rightarrow (d_i(S_2)) = S \Rightarrow (Pr_3(S_2)))) \wedge \forall i = \overline{1,n} \wedge i \neq j\ (s_i(S_2) = C5 \rightarrow slideC \Rightarrow (d_i(S_2)) = S \Rightarrow (Pr_3(S_2))))$$

$$s_j(S_2) = C5 \vDash (s_j(S_2) = C4 \rightarrow slideC \Rightarrow (d_j(S_2)) \neq S \Rightarrow (Pr_3(S_2)))$$

$$\big(I_3(S_1) \wedge S_j(S_1) = C4\big) \vDash newSlide \Rightarrow (d_j(S_1)) = S \Rightarrow (Pr_3(S_1))$$

$$(d_j(S_2) = d \nabla [slideC \mapsto newSlide \Rightarrow d_j(S_2)] \wedge newSlide \Rightarrow (d_j(S_1)) = S \Rightarrow (Pr_3(S_1))) \vDash (s_j(S_2) = C5 \rightarrow slideC \Rightarrow (d_j(S_2)) = S \Rightarrow (Pr_3(S_2)))$$

$$(\forall i = \overline{1,n} \wedge i \neq j\ (s_i(S_2) = C4 \rightarrow slideC \Rightarrow \big(d_i(S_2)\big) \neq S \Rightarrow (Pr_3(S_2))) \wedge \forall i = \overline{1,n} \wedge i \neq j\ (s_i(S_2) = C5 \rightarrow slideC \Rightarrow (d_i(S_2)) = S \Rightarrow (Pr_3(S_2))) \wedge s_j(S_2) = C4 \rightarrow slideC \Rightarrow (d_j(S_2)) \neq S \Rightarrow (Pr_3(S_2)) \wedge s_j(S_2) = C5 \rightarrow slideC \Rightarrow (d_j(S_2)) = S \Rightarrow (Pr_3(S_2))) \vDash (I_3(S_2) \wedge I_4(S_2))$$

$$d.\ (I_2(S_1) \wedge s_j(S_1) = C4 \wedge Pr_1(S_1) = Pr_1(S_2)) \vDash Pr_1(S_1) = Pr_1(S_2) = M2$$

$$Pr_1(S_2) = M2 \vDash I_2(S_2)$$

$$\big(I_1(S) \wedge I_2(S) \wedge I_3(S) \wedge I_4(S) \wedge I_5(S)\big) \vDash Inv(S)\ \blacksquare$$

Conclusion

Partial correctness of the software system, namely Infosoft e-Detailing 1.0, according to an initial problem statement has been proven using Correctness Proof Methodology [Panchenko, 2006, Panchenko, 2008, Panchenko, 2004] in an IPCL language [Panchenko, 2004]. Considering the difficulties in the process of such proof in parallel environments, we can state:

- Correctness Proof Method in IPCL is well suited for the verification of parallel programs or the software correctness proof in terms of safety properties;
- The Method allows shortening the proof at the expense of choosing an adequate abstraction level [Nikitchenko, 1998] due to universality of a compositional nominative approach [Nikitchenko, 1998, Redko, 1978] and by fixing the appropriate basic function set of semantic algebra.

Taking into account flexibility of the Methodology, existence of Simplified State Model reasoning in some cases [Panchenko2, 2007], and universal nature of the approach [Panchenko2, 2008], it can be recommended for program properties proof (particularly safety property or partial correctness) for wide range of software which is executed in interleaving concurrency environment with shared memory, primarily for server-side software of client-server complexes.

The same conclusion is obtained in [Polishchuk, 2015] also.

Bibliography

[Panchenko, 2006] T. Panchenko, Compositional Methods for Software Systems Specification and Verification [in Ukrainian]. PhD thesis, Taras Shevchenko National University of Kyiv, 2006.

[Panchenko, 2007] T. Panchenko, Parallel Addition to Shared Variable Correctness Proof in IPCL [in Ukrainian], Bulletin of Taras Shevchenko National University of Kyiv. Series: Physical and Mathematical Sciences, no. 4, pp. 187—190, 2007.

[Panchenko, 2008] T. Panchenko, The Method for Program Properties Proof in Compositional Nominative Languages IPCL [in Ukrainian], Problems of Programming, no. 1, pp. 3—16, 2008.

[Panchenko2, 2007] T. Panchenko, Simplified State Model for Properties Proof Method in IPCL Languages and its Usage with Advances [in Ukrainian], in Proceedings of the International Scientific Conference “Theoretical and Applied Aspects of Program Systems Development” (TAAPSD’2007), pp. 319—322, 2007.

[DD, 1985] Department of Defense Standard 5200.28-STD ”Trusted Computer System Evaluation Criteria”. National Security Institute, 1985.

[ISO, 2005] European Union Agency for Network and Information Security, ISO/IEC 15408- 1/2/3 Information technology – Security techniques – Evaluation criteria for IT security, 2005.

[Panchenko, 2004] T. Panchenko, The Methodology for Program Properties Proof in Compositional Languages IPCL [in Ukrainian], in Proceedings of the International Conference “Theoretical and Applied Aspects of Program Systems Development” (TAAPSD’2004), pp. 62—67, 2004.

[Panchenko2, 2008] T. Panchenko, Formalization of Parallelism Forms in IPCL [in Ukrainian], Bulletin of Taras Shevchenko National University of Kyiv. Series: Physical and Mathematical Sciences, no. 3, pp. 152—157, 2008.

[Redko, 1978] V. Redko, Compositions of programs and composition programming [in Russian], Programming, no. 5, pp. 3—24, 1978.

[Nikitchenko, 1998] M. Nikitchenko, Technical Report IT-TR: 1998-020. A Composition Nominativej Approach to Program Semantics. Technical University of Denmark, 1998.

[Polishchuk, 2015] N.V. Polishchuk, M.O. Kartavov and T.V. Panchenko, Safety Property Proof using Correctness Proof Methodology in IPCL, in Proceedings of the 5th International Scientific Conference “Theoretical and Applied Aspects of Cybernetics, Kyiv: Bukrek, pp.37–44, 2015.

Authors' Information

Mykyta Kartavov *– the 4th year student at the Theory and Technology of Programming Department, Faculty of Cybernetics, Taras Shevchenko National University of Kyiv, 64/13, Volodymyrska Street, Kyiv, Ukraine, 01601*

e-mail: howard.lons@gmail.com

Major Fields of Scientific Research: Theory and Technology of Programming, Software Engineering, Software Correctness, Formal Methods

Taras Panchenko *– PhD, Associate Professor at the Theory and Technology of Programming Department, Faculty of Cybernetics, Taras Shevchenko National University of Kyiv, 64/13, Volodymyrska Street, Kyiv, Ukraine, 01601*

e-mail: tp@infosoft.ua

Major Fields of Scientific Research: Theory and Technology of Programming, Software Engineering, Software Correctness, Formal Methods

Nataliya Polishchuk *– the 4th year student at the Theory and Technology of Programming Department, Faculty of Cybernetics, Taras Shevchenko National University of Kyiv, 64/13, Volodymyrska Street, Kyiv, Ukraine, 01601*

e-mail: natischka@yandex.ua

Major Fields of Scientific Research: Theory and Technology of Programming, Software Engineering, Software Correctness, Formal Methods