July, 2019

Probabilistic Non-Interference

Amir Abas Kabiri Zamani 97131100

Overview

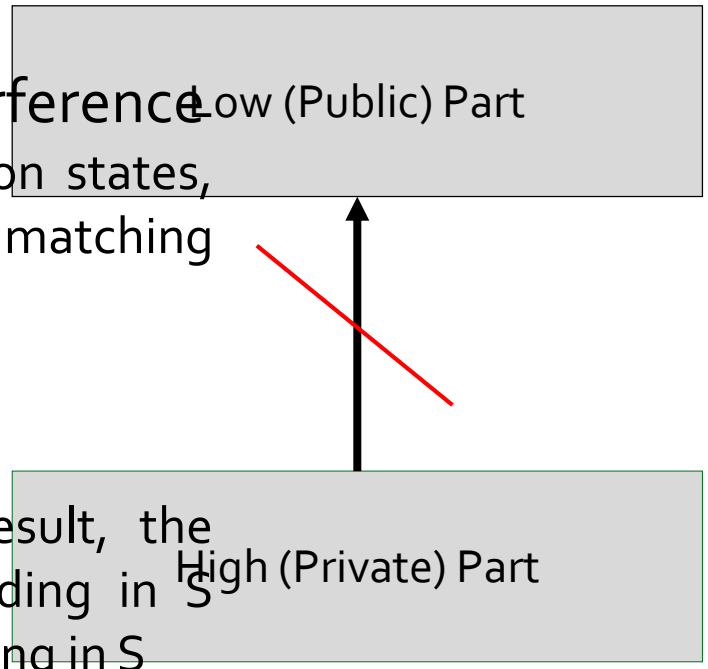
1. Introduction
2. The programming language
   - Syntax
   - Semantic
3. Non-Interference
4. Syntactic Criteria
5. References

What we are going to discuss?

- Present an Isabelle formalization of probabilistic noninterference for a multi-threaded language with uniform scheduling
- Resumption-based notion of probabilistic noninterference

Setting the Scene

- Resumption-based (or bisimulation-based) Non-Interference
  - Requires that each execution chunk is compatible with on states, and that this property is also resumed in the matching continuations.



- Trace-based Non-Interference
  - For each possible indistinguishability class S of the result, the (cumulated) measure of all traces starting in s and ending in S equates with the measure of all traces starting in t and ending in S

Language

The syntax and semantic of language under inspection

2

Syntax

- The type com, of commands, ranged over by c,d, is defined as follows
- atom, of atoms, ranged over by a
  - assignments such x=x+y
- A command is called finished if it is either Done or, inductively, a Par or ParT-composition of finished commands:

data com = Atm **atom** | Done | Seq **com com** | While **test com** | Par (**com list**) | ParT (**com list**)

finished Done;

$(\bigwedge_{i=0}^{n-1} \text{finished } c_i) \Longrightarrow \text{finished } (\text{Par } [c_0, \ldots, c_{n-1}]);$

$(\bigwedge_{i=0}^{n-1} \text{finished } c_i) \Longrightarrow \text{finished } (\text{ParT } [c_0, \ldots, c_{n-1}]).$

List of Commands

- **test**, of tests
  - booleans such x+y>2
- **choice**, of (probabilistic) choices, ranged over by *ch:*
  - choices are flexible enough to cover the standard "if" conditions, as well as stateless probabilistic choice

Syntax

- Par$[c_0, \ldots, c_{n-1}]$ and ParT$[c_0, \ldots, c_{n-1}]$ are two variants of parallel composition of the thread pool $[c_0, \ldots, c_{n-1}]$, written in concrete syntax as $c_0 || \ldots || c_{n-1}$ and $c_0 ||_T \ldots ||_T c_{n-1}$,respectively. They differ in that the latter is termination sensitive, removing finished threads from the thread pool

Semantic

- The semantics of the language indicates the immediate steps available to a command in a given state, where the steps are assigned weights that sum up to 1. It is parameterized by the following data:
  - a type of (memory) states, **state**, ranged over by s; t
  - an execution function for the atoms, aexec : **atom** → **state** → **state**
  - an evaluation function for the tests, tval : **test** → **state** → **bool**
  - an evaluation function for the choices, cval : **choice** → **state** → [0;1]

Semantic

- Fo
  -
  -
  -
  -
- Th gned
  we the
  fur
  -
  -
  -

c	wt $c\ s\ i$	cont $c\ s\ i$	eff $c\ s\ i$
atm	1	Done	aexec $c\ s$
Done	1	Done	s
Seq $c_1\ c_2$	wt $c_1\ s\ i$	c_2, if finished c_1 Seq (cont $c_1\ i$) c_2, otherwise	eff $c_1\ s\ i$
Ch $ch\ c_1\ c_2$	cval $ch\ s$, if $i = 0$ $1 -$ cval $ch\ s$, if $i = 1$	c_1, if $i = 0$ c_2, if $i = 1$	s
While $tst\ d$	1	Seq d (While $tst\ d$), if tval $tst\ s$ Done, otherwise	s
Par $[c_0, \ldots, c_{n-1}]$	$\frac{1}{n} *$ wt $c_k\ s\ j$	Par $[c_0, \ldots,$ cont $c_k\ s\ j, \ldots, c_{n-1}]$	eff $c_k\ s\ j$
ParT $[c_0, \ldots, c_{n-1}]$	$\frac{1}{m} *$ wt $c_k\ s\ j$, if $\neg$ finished c_k 0, otherwise	ParT $[c_0, \ldots,$ cont $c_k\ s\ j, \ldots, c_{n-1}]$	eff $c_k\ s\ j$

Semantic

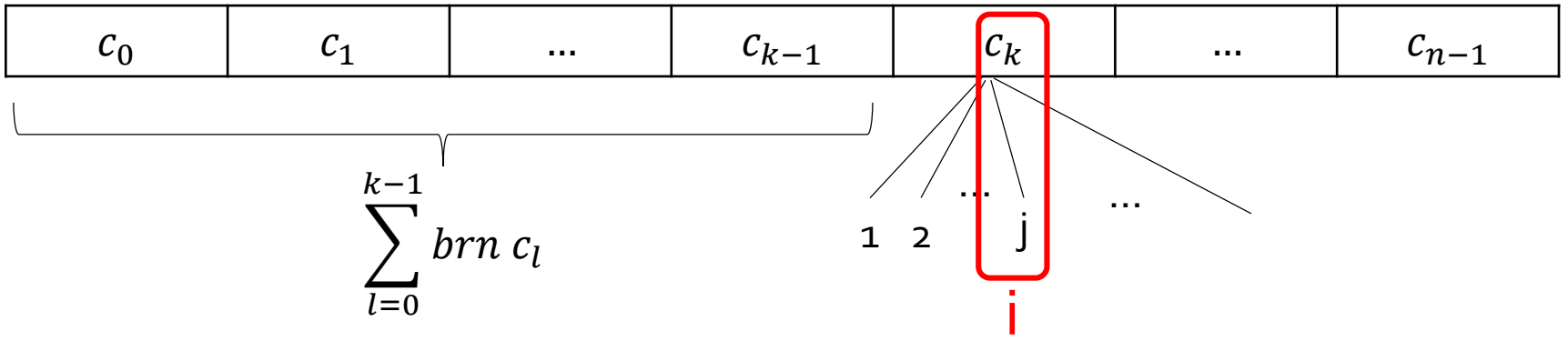
- The semantics of commands:
  - The semantics of atm and Done are straightforward: there is one single available step (hence brn is 1) with weight 1, transiting to the terminating continuation Done. For Done, there is no effect on the state (i.e., the state s remains unchanged), and for atm, the effect is given by aexec.
  - A *Seq c1 c2* command obtains its branching, weight and effect from c1, and its continuation is the continuation of c1 if unfinished and is c2 otherwise.
  - *While tst d* performs an unfolding step to the continuation *Seq d* if the test is *True* and to *Done* otherwise, in both cases with weight 1 and no effect.
  - A choice command Ch *ch* c1 c2 is assumed to perform an effectless branching according to ch. It has 2 branches, labeled 0 and 1: the left one with weight cval ch s and continuation c1, and the right one with complementary weight 1-cval ch s and continuation c2.

Semantic

- If c has the form Par $[c_0, \dots, c_{n-1}]$, then it consists of n threads, $c_0, \dots, c_{n-1}$, running in parallel under a uniform scheduler assigning them equal probabilities. The *branching* of c is thus the sum the *branchings* of c_l for $l \in \{0, \dots, n-1\}$.
- A branch label i of c determines uniquely the numbers k and j so that i corresponds to the j's branch of ck, via the equation $i = \left(\sum_{l=0}^{k-1} brn\ c_l\right) + j$. The weight of i in c is:

  [the probability of picking thread ck (out of n possibilities)] X [the weight of j in ck]

  $= \left(\frac{1}{n}\right) \times wt\ c_k\ s\ j$



Semantic

- The *i-continuation* from c is obtained by replacing, in the thread pool, ck with its *j-continuation* cont ck s j. The *i-effect* of c is the *j-effect* of ck, eff ck s j.
- ParT behaves like Par, except that it is termination-sensitive, in that finished threads are not taken into consideration (being assigned weight 0), and the choice is made among the m unfinished threads

Non-Interference

Inspecting NI w.r.t Probability

3

Indistinguishability

- We fix a relation $\sim$ on states, called *indistinguishability*, where $s \sim t$ is meant to say "s and t are indistinguishable by the attacker."

$$s \sim t \equiv \forall x \in \mathbf{var}.\ \mathsf{sec}\ x = \mathsf{lo} \Longrightarrow s\ x = t\ x.$$

- *Noninterference* of a program states that its execution is compatible with the *indistinguishability* relation: given two *indistinguishable* states s and t, (partially) executing the program once starting from s and once starting from t yield *indistinguishable* states.

Resumption-Based Noninterference

- We define following notions coinductively as greatest fixed points as the weakest predicates satisfying certain equations:
  - Self isomorphism (Siso): if started in *indistinguishable* states, executions take the same branches with the same probabilities
    - Given any two initial memory states that are indistinguishable by the attacker, the executions of c proceed identically
  - Discreetness (discr): during the computation, the states stay indistinguishable from the initial state.
    - c may never change the low part of the memory during its execution

$$\begin{aligned}\text{siso}\ c \equiv\ & (\forall s\ t\ i.\ s \sim t\ \wedge\ i < \text{brn}\ c \Longrightarrow \text{cont}\ c\ s\ i = \text{cont}\ c\ t\ i\ \wedge\ \text{eff}\ c\ s\ i \sim \text{eff}\ c\ t\ i) \wedge \\ & (\forall s\ i.\ i < \text{brn}\ c \Longrightarrow \text{siso}\ (\text{cont}\ c\ s\ i)) \\ \text{discr}\ c \equiv\ & \forall s\ i.\ i < \text{brn}\ c \Longrightarrow s \sim \text{eff}\ c\ s\ i\ \wedge\ \text{discr}\ (\text{cont}\ c\ s\ i)\end{aligned}$$

Probabilistic Bisimulation

- To introduce probabilistic bisimulation, we need a few preparations. Given $I \subseteq C$ for $\{0, \ldots, brn\ c - 1\}$, we write $Wt\ c\ s\ I$ for the cumulated weights from (c; s) of the labels in I, namely $\sum_{i \in I} wt\ c\ s\ i$. Given sets A and P, we say P is a *partion* of A, written, part A P, if P consists of mutually disjoint sets whose union is A.

Probabilistic Bisimulation

- The following predicate $match_c^C$ shows, for a relation on commands Θ and two commands c and d, how the steps taken by c and d are matched unambiguously and weight-exhaustively, so that their effects are indistinguishable and their continuations are in Θ:

$$
\begin{aligned}
&\mathsf{match}_c^C\ \theta\ c\ d \equiv \\
&\forall s\ t.\ s \sim t \Longrightarrow \exists P\ Q\ F. \\
&\mathsf{part}\ \{0,\ldots,\mathsf{brn}\ c-1\}\ P \wedge \mathsf{part}\ \{0,\ldots,\mathsf{brn}\ d-1\}\ Q \wedge [F : P \to Q\ \text{bijection}] \wedge \\
&(\forall I \in P.\ \mathsf{Wt}\ c\ s\ I = \mathsf{Wt}\ d\ t\ (F\ I) \wedge \\
&\qquad (\forall i \in I.\ \forall j \in F\ I.\ \mathsf{eff}\ c\ s\ i \sim \mathsf{eff}\ d\ t\ j \wedge \theta\ (\mathsf{cont}\ c\ s\ i)\ (\mathsf{cont}\ d\ t\ j)))
\end{aligned}
$$

Probabilistic Bisimulation

- Thus, $match_c^c \Theta\ c\ d$ states that there exist partitions P and Q of the branches of c and d and a bijective correspondence $F : P \longrightarrow Q$ so that, for any corresponding sets of branches I and F I:
  - The cumulated weights are the same.
  - For any pair (i ; j) of branches in these sets, the effects are indistinguishable and the continuations are in Θ.

Probabilistic Bisimulation

- Strong bisimilarity, $\approx_s$, is defined as follows:
  - $\forall c, d. c \approx_s d \iff match_c^c(\approx_s) c\ d.$
- A good during-execution noninterference candidate should be compositional with the language constructs and weaker than both siso and discr. It turns out that $\approx_s$ has many of these characteristics, in particular, it will be shown to commute with all the constructs except for While and ParT.

Probabilistic Bisimulation

- To compensate for the lack of ParT-compositionality of $\approx_s$, we introduce a weaker relation, $\approx_{01}$ that we call 01-bisimilarity because it requires a step to be matched by either no step (a stutter move) or one step:

$$
\begin{aligned}
&\mathsf{match}^{\mathsf{C}}_{01\mathsf{C}}\ \theta\ c\ d \equiv \\
&\forall s\ t.\ s \sim t \Longrightarrow \exists P\ Q\ I_0\ F. \\
&\mathsf{part}\ \{0,\ldots,\mathsf{brn}\ c-1\}\ P \wedge \mathsf{part}\ \{0,\ldots,\mathsf{brn}\ d-1\}\ Q \wedge I_0 \in P \wedge [F : P \to Q\ \text{bijection}] \\
&(\forall I \in P - \{I_0\}.\ \frac{\mathsf{Wt}\ c\ s\ I}{1-\mathsf{Wt}\ c\ s\ I_0} = \frac{\mathsf{Wt}\ d\ t\ (F\ I)}{1-\mathsf{Wt}\ d\ t\ (F\ I_0)} \wedge \\
&\qquad (\forall i \in I.\ \forall j \in F\ I.\ \mathsf{eff}\ c\ s\ i \sim \mathsf{eff}\ d\ t\ j \wedge \theta\ (\mathsf{cont}\ c\ s\ i)\ (\mathsf{cont}\ d\ t\ j))) \wedge \\
&(\forall i \in I_0.\ s \sim \mathsf{eff}\ c\ s\ i \wedge \theta\ (\mathsf{cont}\ c\ s\ i)\ d) \wedge \\
&(\forall j \in F\ I_0.\ t \sim \mathsf{eff}\ d\ t\ j \wedge \theta\ c\ (\mathsf{cont}\ d\ t\ j))
\end{aligned}
$$

Probabilistic Bisimulation

- o1-bisimilarity, $\approx_{01}$, is now defined analogously to $\approx_s$, as the largest relation satisfying $\forall c, d. c \approx_{01} d \iff match^{c}_{01c}(\approx_{01}) c\ d$. The notion of security associated to a bisimilarity is its diagonal version, which we call *self bisimilarity*. Thus, c is called *self strongly-bisimilar* if $c \approx_s c$ and self o1-bisimilar if $c \approx_{01} c$.

Compositionality

- Here we establish the compositionality properties of the two resumption-based notions w.r.t. the language constructs and discuss their relative strengths and weaknesses.

Compositionality

- First, we need atomic properties of preservation and compatibility adapted to the abstract notions of state and *indistinguishability* relation. An atom atm is called $\sim -preserving$, written pres atm, if $\forall s.$ aexec $atm\ s \sim s$; it is called $\sim -compatible$, written cpt atm, if $\forall st.\ s \sim t \Longrightarrow$ aexec $atm\ s \sim$ aexec $atm\ t$. A test tst is called $\sim -compatible$, written cpt $tst, if\ \forall s\ t.\ s \sim t \Longrightarrow$ tval $tst\ t$. A choice ch is called $\sim -compatible$, written cpt ch, $if\ \forall s\ t.\ s \sim t \Longrightarrow$ cval $ch\ s = $ cval $ch\ t$.

Compositionality

- In the setting of Example 2, for atoms, $\sim -preservation$ means no assignment to low variables and $\sim -compatible$ means no direct leaks. Moreover, for tests, $\sim -compatiblity$ means no dependence on high variables. A stateless choice is always compatible and an "if" choice is compatible if it is so as a test.

Final Table

c	discr c	siso c	$c \approx_s c$	$c \approx_{01} c$
atm	pres atm	cpt atm	cpt atm	cpt atm
Seq c_1 c_2	discr c_1 discr c_2	siso c_1 siso c_2	siso c_1 $c_2 \approx_s c_2$ $c_1 \approx_s c_1$ discr c_2	siso c_1 $c_2 \approx_{01} c_2$ $c_1 \approx_{01} c_1$ discr c_2
Ch ch c_1 c_2	discr c_1 discr c_2	cpt ch siso c_1 siso c_2	cpt ch $c_1 \approx_s c_1$ $c_2 \approx_s c_2$	cpt ch $c_1 \approx_{01} c_1$ $c_2 \approx_{01} c_2$
While tst d	discr d	cpt tst siso d	False	False
Par $[c_0, \ldots, c_{n-1}]$	discr c_l $0 \leq l < n$	siso c_l $0 \leq l < n$	$c_l \approx_s c_l$ $0 \leq l < n$	False
ParT $[c_0, \ldots, c_{n-1}]$	discr c_l $0 \leq l < n$	False	False	$c_l \approx_s c_l$ $0 \leq l < n$

References

- Popescu, Andrei, Johannes Hölzl, and Tobias Nipkow. "Formalizing probabilistic noninterference." International Conference on Certified Programs and Proofs. Springer, Cham, 2013.
- Popescu, Andrei, Johannes Hölzl, and Tobias Nipkow. "Proving concurrent noninterference." International Conference on Certified Programs and Proofs. Springer, Berlin, Heidelberg, 2012.
- Sabelfeld, Andrei, and David Sands. "Probabilistic noninterference for multi-threaded programs." Proceedings 13th IEEE Computer Security Foundations Workshop. CSFW-13. IEEE, 2000.

July, 2019

Thank You

Amir Abass Kabiri Zamani

kabirizamani@aut.ac.ir