COMP4161 T3/2019
Advanced Topics in Software Verification

Assignment 2

This assignment starts on **Friday 18th of October** and is due on **Friday 1st of November 6PM**. We will accept Isabelle theory (.thy) files.

The assignment is take-home. This does NOT mean you can work in groups. Each submission is personal. For more information, see the plagiarism policy: https://student.unsw.edu.au/plagiarism

Submit using `give` on a CSE machine:

```
give cs4161 a2 files ...
```

For example:

```
give cs4161 a2 a2.thy
```

Hint: there are hints at the end of this document.

1 A theory of network connectivity (25 marks)

1.1 Background

In this section, we will consider a simple model of connectivity between devices in a hierarchical network.

We will use natural numbers to stand for device identifiers, and will describe network topologies as a set of *connectivity conditions*. The conditions are of two kinds:

1. *Joinable* n m means that the devices m and n are connected, in the sense that they can send messages to each other via some (unspecified) route.

2. *Above* n m, pronounced "m is above n", means that device m is an uplink for n. In other words, m can forward messages to other devices in the network on behalf of n.

From an initial set A of conditions describing the network topology, we can construct a set *connection* A of all connections between devices that can be established in the network. *connection* A is defined inductively by the following rules:

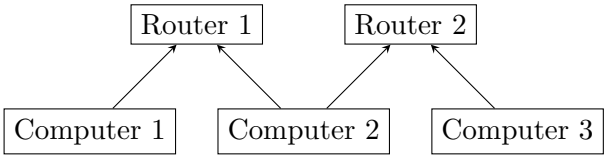

Figure 1: An example topology.

1. I can forward messages on behalf of myself:

$$\frac{}{\mathit{Above}\ a\ a \in \mathit{connection}\ A}$$

2. The connections in the initial topology are available:

$$\frac{\varphi \in A}{\varphi \in \mathit{connection}\ A}$$

3. If you and I are connected, then I and you are connected:

$$\frac{\mathit{Joinable}\ a\ b \in \mathit{connection}\ A}{\mathit{Joinable}\ b\ a \in \mathit{connection}\ A}$$

4. My uplink's uplink is my (virtual) uplink:

$$\frac{\mathit{Above}\ a\ b \in \mathit{connection}\ A \qquad \mathit{Above}\ b\ c \in \mathit{connection}\ A}{\mathit{Above}\ a\ c \in \mathit{connection}\ A}$$

5. We are connected if we share a common uplink:

$$\frac{\mathit{Above}\ a\ b \in \mathit{connection}\ A \qquad \mathit{Above}\ c\ b \in \mathit{connection}\ A}{\mathit{Joinable}\ a\ c \in \mathit{connection}\ A}$$

6. If my uplink is connected to someone, then so am I:

$$\frac{\mathit{Above}\ a\ b \in \mathit{connection}\ A \qquad \mathit{Joinable}\ b\ c \in \mathit{connection}\ A}{\mathit{Joinable}\ a\ c \in \mathit{connection}\ A}$$

As an example, consider the topology illustrated in Figure 1, where arrows represent the *Above* relationship. Here Computers 1 and 2 are joinable because of their common uplink (Router 1), and so are Computers 2 and 3. Computers 1 and 3 are *not* joinable because there is no common uplink to forward messages between the routers.

1.2 Questions

(a) (3 marks) Implement the topology described in Figure 1 as a set of *Above* judgements, and prove that computers 1 and 2 are joinable.

Prove that the connectivity relation satisfies these properties:

(b) (2 marks) $\varphi \in \mathit{connection}\ A \Longrightarrow \varphi \in \mathit{connection}\ (A \cup B)$

(c) (3 marks) $\mathit{connection}\ (\mathit{connection}\ A) = \mathit{connection}\ A$

(d) (2 marks) $\varphi \in \mathit{connection}\ (A \cup B) \Longrightarrow \exists\, C\ D.\ C \subseteq \mathit{connection}\ A \wedge D \subseteq \mathit{connection}\ B \wedge \varphi \in \mathit{connection}\ (C \cup D)$

(e) (2 marks) $\varphi \in \mathit{connection}\ \emptyset \Longrightarrow \mathit{is\text{-}refl}\ \varphi$

(f) (2 marks) $\mathit{is\text{-}refl}\ \varphi \Longrightarrow \varphi \in \mathit{connection}\ A$

(g) (2 marks) $\varphi \in \mathit{connection}\ A \Longrightarrow \varphi \in \mathit{connection}\ (A - \{\varphi \mid \mathit{is\text{-}refl}\ \varphi\})$

(h) (3 marks) $[\![\mathit{Above}\ a\ b \in \mathit{connection}\ A;\ \bigwedge a\ b.\ \mathit{Above}\ a\ b \notin A]\!] \Longrightarrow a = b$

(i) (3 marks) $[\![\varphi \in \mathit{connection}\ (C \cup D);\ C \subseteq \mathit{connection}\ A;\ D \subseteq \mathit{connection}\ B]\!] \Longrightarrow \varphi \in \mathit{connection}\ (A \cup B)$

(j) (3 marks) $\mathit{connection}\ A = \mathit{connection}\ B \Longrightarrow \mathit{connection}\ (A \cup C) = \mathit{connection}\ (B \cup C)$

2 A theory of communicating devices (15 marks)

2.1 Background

We will now extend our (static) model of a network to include the dynamic behaviour of processes communicating through the network. For simplicity we do not model data: communication is just a synchronous handshake with no information being exchanged. The state of the network is described by this datatype:

datatype process =
Cond condition
| Par process process
| Input nat process
| Output nat process
| Nil

A network may: contain connectivity assumptions (*Cond* φ); consist of multiple processes running in parallel, possibly on different devices (*Par* P Q); contain a process that does nothing (*Nil*); have device m ready to receive a message and then continue as P (*In* m P); or have a device ready to send a message (*Output* m P).

The behaviour of the network is captured with a small-step transition relation *semantics* A P α P', meaning that in a network with topology A, the process P can perform the action α and transition to the state P'. The actions can be input, output or internal (*LTau*):

datatype action = LInput nat | Output nat | LTau

The *frame* of a process collects all its top-level conditions, which taken together describe the current network topology. Conditions underneath *In* and *Out* prefixes represent ways the topology may change in the future.

An output or input prefix may send or receive a message, consuming the prefix:

$$\frac{}{\mathit{semantics}\ A\ (\mathit{Input}\ n\ P)\ (\mathit{LInput}\ n)\ P}$$

$$\frac{}{\mathit{semantics}\ A\ (\mathit{Output}\ n\ P)\ (\mathit{LOutput}\ n)\ P}$$

If a process P can perform an action, it may also perform that action with Q sitting inertly in parallel. The derivation of the transition from P may use the connectivity information present in *frame* Q:

$$\frac{\mathit{semantics}\ (A \cup \mathit{frame}\ Q)\ P\ \alpha\ P'}{\mathit{semantics}\ A\ (\mathit{Par}\ P\ Q)\ \alpha\ (\mathit{Par}\ P'\ Q)}$$

If two parallel processes P and Q can perform an input and output, respectively, and if the devices are joinable, then P and Q may synchronise to perform a communication action (*LTau*):

$$\frac{\begin{array}{c}\mathit{semantics}\ (A \cup \mathit{frame}\ Q)\ P\ (\mathit{LOutput}\ n)\ P' \\ \mathit{semantics}\ (A \cup \mathit{frame}\ P)\ Q\ (\mathit{LInput}\ m)\ Q' \\ \mathit{Joinable}\ n\ m \in \mathit{connection}\ (A \cup \mathit{frame}\ P \cup \mathit{frame}\ Q)\end{array}}{\mathit{semantics}\ A\ (\mathit{Par}\ P\ Q)\ \mathit{LTau}\ (\mathit{Par}\ P'\ Q')}$$

Symmetric versions of the rules for parallel and communication are elided.

2.2 Questions

Prove that the semantics satisfies the following:

(a) (5 marks) $\mathit{semantics}\ A\ P\ \alpha\ Q \Longrightarrow \mathit{semantics}\ (A \cup B)\ P\ \alpha\ Q$

(b) (5 marks) $\mathit{semantics}\ A\ P\ \alpha\ Q \Longrightarrow \exists\, \beta\ Q'.\ \mathit{semantics}\ \emptyset\ P\ \beta\ Q'$

(c) (5 marks) $[\![\mathit{semantics}\ A\ P\ \alpha\ Q;\ \mathit{connection}\ A = \mathit{connection}\ B]\!] \Longrightarrow \mathit{semantics}\ B\ P\ \alpha\ Q$

3 A theory of behavioural equivalence (60 marks)

3.1 Background

It is often useful to be able to say when two distinct processes have the same behaviour. For the purposes of this exercise, we will use *trace equivalence* for this purpose. Trace equivalence may be intuitively characterised as follows. Imagine you are presented with a log (aka trace) of all the events that transpired when a process ran to completion. You know that the log describes an execution of either P or Q, but you haven't been told which. By inspecting the log, can you figure out which process the log originates from? If the answer to this question is always no for every log, we say that P and Q are trace equivalent.

Let's formalise this intuitive understanding. A process is *stuck* if it has run to completion; that is, if it has no outgoing transitions:

$$\mathit{stuck}\ P \equiv \forall\, \alpha\ P'.\ \mathit{semantics}\ \emptyset\ P\ \alpha\ P' \longrightarrow \mathit{False}$$

The relation *list-trans* lifts *semantics* from single actions to sequences of actions. A *trace* of a process is a sequence of actions it can take that lead to a stuck state. The set of all such traces is given by

$$\mathit{traces\text{-}of}\ P = \{\mathit{tr} \mid \exists\, Q.\ \mathit{list\text{-}trans}\ \emptyset\ P\ \mathit{tr}\ Q \wedge \mathit{stuck}\ Q\}$$

Finally, two processes are trace equivalent if they have the same sets of traces—or in other words, if for every trace we cannot tell which process it originated from:

$$\mathit{trace\text{-}eq}\ P\ Q = (\mathit{traces\text{-}of}\ P = \mathit{traces\text{-}of}\ Q)$$

3.2 Questions

(a) (3 marks) Show that trace equivalence is indeed an equivalence relation:

$\mathit{trace\text{-}eq}\ P\ P$

$\mathit{trace\text{-}eq}\ P\ Q \Longrightarrow \mathit{trace\text{-}eq}\ Q\ P$

$[\![\mathit{trace\text{-}eq}\ P\ Q;\ \mathit{trace\text{-}eq}\ Q\ R]\!] \Longrightarrow \mathit{trace\text{-}eq}\ P\ R$

(b) (4 marks) $[\![stuck\ P;\ stuck\ Q]\!] \Longrightarrow trace\text{-}eq\ P\ Q$

(c) (8 marks) $stuck\ R \Longrightarrow traces\text{-}of\ P \subseteq traces\text{-}of\ (Par\ P\ R)$

(d) (4 marks) The previous question used set inclusion instead of trace equivalence because inclusion does not hold in the other direction: it is possible to introduce more traces by adding a stuck parallel component. Give an example of concrete P and R such that the following proposition is *false*:

$stuck\ R \Longrightarrow traces\text{-}of\ (Par\ P\ R) \subseteq traces\text{-}of\ P$

(e) (7 marks) Use your counterexample from the previous question to formally prove

$(\bigwedge P\ R.\ stuck\ R \Longrightarrow traces\text{-}of\ (Par\ P\ R) \subseteq traces\text{-}of\ P) \Longrightarrow False$

Fill in the gaps to complete a proof that parallel composition is associative up to trace equivalence:

(f) (6 marks) $semantics\ A\ (Par\ P\ (Par\ Q\ S))\ \alpha\ R \Longrightarrow \exists\, P'\ Q'\ S'.\ R = Par\ P'\ (Par\ Q'\ S') \wedge semantics\ A\ (Par\ (Par\ P\ Q)\ S)\ \alpha\ (Par\ (Par\ P'\ Q')\ S')$

(g) (2 marks) $list\text{-}trans\ A\ (Par\ (Par\ P\ Q)\ S)\ tr\ (Par\ (Par\ P'\ Q')\ S') = list\text{-}trans\ A\ (Par\ P\ (Par\ Q\ S))\ tr\ (Par\ P'\ (Par\ Q'\ S'))$

(h) (4 marks) $stuck\ (Par\ P\ Q) = (stuck\ P \wedge stuck\ Q)$

(i) (4 marks) $trace\text{-}eq\ (Par\ (Par\ P\ Q)\ S)\ (Par\ P\ (Par\ Q\ S))$

Finally, prove that *Nil* is the unit of parallel composition, and that parallel composition commutes:

(j) (9 marks) $trace\text{-}eq\ P\ (Par\ P\ Nil)$

(k) (9 marks) $trace\text{-}eq\ (Par\ P\ Q)\ (Par\ Q\ P)$

4 Hints

- You are allowed to use sledgehammer as much as you like.
- You are allowed—encouraged, in fact—to use solutions to questions as lemmas in the answers to other questions.
- Most proofs will require induction of one kind or the other. The relevant induction rules are `connection.induct`, `semantics.induct` and `list.induct`.

- The assumptions of theorems stated in the `assumes-show` format are accessible via the fact named `assms`. For example, you can do `simp add: assms` or `rule assms(1)`. The assumptions can be added directly to the goal state by beginning your proof with the command `using assms`; that is usually what you want to do before starting an induction.

- The equivalent of `spec` for the meta-logic universal quantifier is called `meta_spec`.

- `semantics.simps` is sometimes useful for doing case analysis on the derivation of a transition, but is loop-prone. It is often more useful to do this case analysis using specialised elimination rules like `par` and `nil`. Another option is to specialise `semantics.simps` to the pattern you want to decompose before adding it to the simpset. For example, if your assumptions mention a transition from a process *Par P Q*, adding the following will decompose it into assumptions about transitions from *P* and *Q*.

```
simp add: semantics.simps[where ?a2.0="Par P Q" for P
                    Q,simplified]
```

- Some inductive proofs will require you to strengthen the induction hypothesis in order to close the proof. Think about which variables should be `arbitrary`.

- For the later exercises, your life will be much easier if you decompose your proofs into auxiliary lemmas. For example, proofs of trace equivalence naturally decompose into separate proofs about *stuck* and *list-trans*. For proofs about *list-trans*, a good strategy is to first prove a similar lemma for single-step transitions (*semantics*), then lift them to *list-trans* by induction on the action sequence.