1. Introduction

This is a condensed extract from section 6.10 (Proof rules) of the text. Hopefully, it will serve as a convenient reference while doing assertional proofs. It also introduces some terminology (in boxes) used in homeworks.

2. Hoare-triples

Hoare-triples express properties of program statements when they execute without interference from the environment. A Hoare-triple has the form {P} S {Q}, where P and Q are predicates and S is a program statement. P and Q are referred to as the **precondition** and the **postcondition**, respectively, of the Hoare-triple.

- For S that is *non-blocking* and not preceded by an input assumption/condition:
  {P}S{Q} means that the execution of S starting from *any* state satisfying P always terminates (i.e., no infinite loop, no fault) in a state that satisfies Q, assuming that S's environment does not affect intermediate states of S's execution.
- For S that is *blocking* with guard B and action C (e.g., "await (B) C" or "oc{B} C"):
  {P}S{Q} means {P and B}C{Q}.
- For S that is preceded by input assumption/condition B:
  {P}S{Q} means {P and B}S{Q}.

Here are some examples of Hoare-triples. Next to each we indicate whether or not it is valid.

- {true} if x ≠ y then x ← y+1 {(x = y+1) or (x = y)} (valid)
- {x = n} for (i in 0..10) do x ← x+i {x = n+55} (valid)
- {x = 3} x ← y+1 {x = 4} (invalid; e.g., if y = 1 holds at start)

We say "S unconditionally establishes Q from P" to mean that {P} S {Q} holds.
We say "S unconditionally establishes Q" to mean that {true} S {Q} holds.
We say "S unconditionally preserves P" to mean that {P} S {P} holds.

3. Proof rules for safety assertions

Invariance induction rule

Inv P holds for program `M` if the following hold:

- for the initial atomic step e of `M`: `{true}` e `{`P`}`
- for every non-initial atomic step e of `M`: `{`P`}` e `{`P`}`

We say "P satisfies the invariance induction rule" to mean it satifies the above conditions.

Invariance induction rule

Inv P holds for program `M` if the following hold for some predicate R:

- *Inv* `R`
- for the initial atomic step e of `M`: `{true}` e `{`$R \Rightarrow P$`}`
- for every non-initial atomic step e of `M`: `{`P `and` R`}` e `{`$R \Rightarrow P$`}`

We say "P satisfies the invariance induction rule assuming *Inv* R" to mean it satifies the above conditions.

Unless rule

P *unless* Q holds for program `M` if the following hold:

- for every non-initial atomic step `e` of `M`: `{`P `and not` Q`}` `e` `{`P `or` Q`}`

We say "P `and` Q follows from the unless rule" to mean it satifies the above conditions.

Closure rules

Inv P holds if P holds.

Inv P holds if the following hold:
Inv Q
- *Inv* $(Q \Rightarrow P)$

P *unless* Q holds if *Inv* $(P \ \Rightarrow \ Q)$ holds.

P *unless* Q holds if the following hold:

- R *unless* S
- *Inv* $(P \ \Rightarrow \ R)$
- *Inv* $(S \ \Rightarrow \ Q)$

We say an assertion holds via closure of assertions $Q_1, \cdots, Q_n$" to mean that the former follows by applying closure rules to the latter.

4. Proof rules for progress assertions

For an atomic step `e`, let the predicate `e.enabled` mean that a thread is at `e` and `e` is unblocked (if it has a guard). Formally,

$$\texttt{e.enabled} = \begin{cases} \texttt{thread at e} & \text{if e is nonblocking} \\ \texttt{(thread at e) and B} & \text{if e has guard B (e.g., oc\{B\}S)} \end{cases}$$

Weak-fair rule

P *leads-to* Q holds for program `M` if the following hold, where `e` is an atomic step of `M` subject to weak fairness:

- (P and not Q) $\Rightarrow$ e.enabled
- {P and not Q} e {Q}
- for every non-initial atomic step `f` of `M`: {P and not Q} f {P or Q}

We say "P *leads-to* Q via weak-fair rule" to mean that P and Q satifies the above conditions.

Strong-fair rule

P *leads-to* Q holds for program `M` if the following hold, where `e` is an atomic step of `M` subject to strong fairness:

- (P and not Q and not e.enabled) *leads-to* (Q or e.enabled)
- {P and not Q} e {Q}
- for every non-initial atomic step `f` of `M`: {P and not Q} f {P or Q}

We say "P *leads-to* Q via strong-fair rule" to mean that P and Q satifies the above conditions.

Closure rules

- P *leads-to* (Q_1 or Q_2) holds if the following hold:
  - P *leads-to* P_1 or Q_2
  - P_1 *leads-to* Q_1

- P *leads-to* Q holds if the following hold for some predicate R:
  - *Inv* R
  - (P and R) *leads-to* ($R \Rightarrow Q$)

- (P_1 and P_2) *leads-to* Q_2 holds if the following hold for some predicate Q_1:
  - P_1 *leads-to* Q_1
  - P_2 *unless* Q_2
  - *Inv* ($Q_1 \Rightarrow$ (not P_2))

- P *leads-to* Q holds if, for some function F on a lower-bounded partial order $(Z, \prec)$, the following hold:
  - P *leads-to* (Q or forsome(x in $Z : F(x)$))
  - forall(x in Z:
    $F(x)$ *leads-to* (Q or forsome(w in $Z : w \prec x$ and $F(w)$)))

  [This is just induction over a well-founded order.]

We say P *leads-to* Q via closure of assertions $L_1, \cdots, L_n$" to mean that the former follows by applying closure rules to the latter.