Note sulle Asserzioni alla Hoare

(per il Corso di Metodi Formali per l'Informatica)

Elena Zucca

5 giugno 2003

Introduzione

Una *specifica* è una descrizione di *cosa* deve fare un sistema software. Una particolare implementazione descriverà il *come*. Una specifica rappresenta quindi un contratto ideale tra sviluppatore ed utente.
Una specifica informale è ovviamente spesso ambigua ed imprecisa. Una specifica formale è data invece attraverso delle formule logiche (o *asserzioni*) che hanno una semantica precisa.
Alcuni stili di specifica:

- algebrico (tipi di dato in contesto funzionale),
- asserzioni alla Hoare (paradigma imperativo e object-oriented),
- logica temporale (paradigma concorrente).

L'uso di asserzioni è relativo alla *correttezza* dei programmi. Si noti che "correttezza" è un concetto relativo. Non ha senso ad esempio chiedersi se il comando
`x = y + 1`
è corretto; lo è rispetto alla specifica "rendere `x`, `y` di valore diverso", mentre non lo è rispetto alla specifica "rendere `x` negativo".
Quindi la nozione di correttezza si applica a coppie ⟨specifica, codice⟩. Di conseguenza, per poter definire formalmente cosa significa che un programma è corretto rispetto ad una specifica, entrambi devono essere valutati nello stesso dominio semantico.
Semplicemente *scrivere* la specifica è già un passo in avanti verso l'assicurare la correttezza perchè:

- chiarisce la differenza tra ciò che si vuol fare e come,
- guida la documentazione del software (formalizzazione dei commenti),
- fornisce una base per testing/debugging.

In queste note descriveremo un tipo particolare di asserzioni, le *asserzioni alla Hoare*, che permettono di esprimere formalmente la correttezza di algoritmi descritti tramite un linguaggio imperativo. Un'asserzione alla Hoare è una tripla $\{P\}\ C\ \{R\}$ dove C è un comando (quindi, semanticamente, una trasformazione di stato) e P, R sono predicati[1] sullo stato manipolato da C, detti rispettivamente *precondizione* e *postcondizione*.
Distingueremo asserzioni di *correttezza parziale*, scritte nella forma detta sopra, e asserzioni di *correttezza totale*, scritte $\{P\}\ C\ \{\Downarrow R\}$. La differenza è che per parlare di correttezza totale è necessario che il comando termini, mentre ciò non è richiesto per la correttezza parziale. Ovviamente la differenza è significativa solo nel caso di comandi che possono non terminare, come avviene nel caso di cicli (while, repeat, etc.), di funzioni ricorsive o di errori a run-time. Nel semplice linguaggio che considereremo nel seguito l'unica fonte di non terminazione sarà data dal comando while.
La correttezza parziale potrebbe apparire poco utile in pratica (di qualunque algoritmo che risolva un problema sembra un requisito minimo richiedere che termini!), ma viene studiata perchè spesso risulta conveniente provare separatamente la correttezza parziale e la terminazione.
Ad esempio, la seguente asserzione (in questo caso possiamo indifferentemente usare la correttezza parziale o totale)
$\{\mathtt{x} \geq 9\}$ `x=x+5` $\{\Downarrow \mathtt{x} \geq 13\}$
corrisponde a richiedere che, partendo da uno stato in cui vale $\mathtt{x} \geq 9$, l'esecuzione del comando `x=x+5` termini in uno stato in cui vale $\mathtt{x} \geq 1$. Questa asserzione è intuitivamente valida.

[1] Nell'approccio *estensionale*, vedi dopo.

1 Linguaggio di riferimento

Utilizzeremo, per semplicità espositiva, un linguaggio analogo a quello su cui è stata data la versione originale delle asserzioni alla Hoare, cioè un linguaggio molto semplice MINILW che corrisponde ad un sottoinsieme del linguaggio LW visto nel corso dove si considerano solo comandi ed espressioni, le espressioni non hanno side-effects, l'ambiente è fissato (cioè non vi è un costrutto di blocco) e non si considera l'input/output.
La sintassi di MINILWè la seguente:

$$\begin{array}{lll} \textit{Com} & ::= & \texttt{skip} \mid \textit{Id} = \textit{Exp} \mid \textit{Com}\texttt{;}\,\textit{Com} \mid \texttt{while (}\textit{BExp}\texttt{) \{ }\textit{Com}\texttt{ \}} \mid \\ & & \texttt{if (}\textit{BExp}\texttt{)\{}\textit{Com}\texttt{\} else \{}\textit{Com}\texttt{\}} \\ \textit{Exp} & ::= & \textit{Id} \mid \textit{Exp} + \textit{Exp} \mid \textit{Exp} * \textit{Exp} \mid \textit{Num} \mid \ldots \\ \textit{BExp} & ::= & \texttt{true} \mid \texttt{false} \mid \ldots \end{array}$$

Si noti che, grazie alle assunzioni dette sopra, la semantica formale di MINILW risulta molto più semplice di quella di LW (si provi a darla tutta per esercizio); in particolare, dato che si considera un ambiente fissato (cioè un insieme fissato di identificatori di variabile), le nozioni di ambiente e stato possono "collassare", quindi le funzioni semantiche per comandi ed espressioni hanno il seguente tipo:

$[\![-]\!]_{Com}$: *Com* $\rightarrow$ *States* $\rightarrow$ *States*
$[\![-]\!]_{Exp}$: *Exp* $\rightarrow$ *States* $\rightarrow$ *Value*
$[\![-]\!]_{BExp}$: *BExp* $\rightarrow$ *States* $\rightarrow \{tt, ff\}$

Si noti che è possibile trascurare l'input/output perché, intuitivamente, possiamo sempre trasformare un programma in uno equivalente in cui l'algoritmo vero e proprio sia preceduto da tutte le letture e seguito da tutte le scritture. Ciò corrisponde a dire che il problema (noto nella letteratura originale come "problema della programmazione") di trovare un programma (comando) che, dato un input che soddisfa un requisito iniziale, produca un output che verifichi un requisito finale, formalmente

> dati *P*, *R* predicati (rispettivamente su *Input* ed *Output*), trovare *C* tale che:
> $\forall i \in \textit{Input}.P(i) \Rightarrow R([\![C]\!]\, i)$

si può ridurre, assumendo lo stato coincidente con la memoria, al seguente:

> dati *P*, *R* predicati su *States*, trovare *C* tale che:
> $\forall s \in \textit{States}.P(m) \Rightarrow R([\![C]\!]\, s)$.

2 Asserzioni alla Hoare di correttezza parziale

Specificheremo proprietà dei programmi in MINILW attraverso delle triple dette *asserzioni alla Hoare*, della forma $\{P\}\ C\ \{R\}$, dove *P* e *R* sono dei predicati sullo stato, detti rispettivamente *precondizione* e *postcondizione*, e *C* è un comando. Intuitivamente, il significato di tali asserzioni è il seguente:

> se *P* vale nello stato iniziale (cioè prima di eseguire *C*) e l'esecuzione di *C* termina (producendo quindi uno stato finale), allora *R* deve valere su tale stato.

Si noti che perchè l'asserzione sia valida *non* si richiede che l'esecuzione del comando termini; si parla in questo caso di asserzioni di *correttezza parziale*.

Esempio Un esempio di asserzione alla Hoare di correttezza parziale è il seguente.

$$\{\texttt{x} = x_0 \wedge \texttt{y} = y_0\}\ \texttt{while (x} \neq \texttt{0) \{ x=x-1; y=y+1 \}}\ \{\texttt{x} = 0 \wedge \texttt{y} = x_0 + y_0\}$$

Qui x_0, y_0 denotano due numeri interi arbitrari. Questa asserzione è intuitivamente valida. Infatti, se $x_0 \geq 0$ è facile vedere che l'esecuzione del comando termina ed alla fine xha sicuramente assunto il valore 0 e yha incrementato il suo valore iniziale di una quantità uguale al valore iniziale di x; se invece $x_0 < 0$ l'esecuzione del comando non termina e quindi, in base alla definizione data sopra di correttezza parziale, l'asserzione risulta banalmente valida.

Vedremo tra poco come definire e provare formalmente la validità di un'asserzione. Prima però sono necessarie alcune precisazioni.

Approccio intensionale ed estensionale In generale, si parla di descrizione *estensionale* di una funzione intendendo il grafico, cioè l'insieme delle coppie messe in corrispondenza (che può essere infinito). Una descrizione *intensionale* è invece una descrizione finita data utilizzando un qualche formalismo sintattico. Ovviamente diverse descrizioni intensionali possono corrispondere alla stessa descrizione estensionale.
Nel caso delle asserzioni alla Hoare, per non essere vincolati ad un particolare linguaggio, adotteremo l'approccio estensionale, cioè considereremo triple $\{P\}\ C\ \{R\}$ dove P e R non sono formule (oggetti sintattici) ma oggetti semantici, cioè predicati sullo stato (funzioni totali da stati in valori booleani). In altre parole non ci preoccuperemo di fissare una particolare rappresentazione sintattica per le asserzioni sullo stato; negli esempi utilizzeremo il metalinguaggio matematico e logico usuale.
Ad un'espressione booleana B del linguaggio MINILW corrisponderà ovviamente un predicato $[\![B]\!]$ sullo stato (la sua semantica). Assumiamo infatti l'ipotesi semplificativa che la valutazione delle espressioni di MINILWnon possa dar luogo a non terminazione.

Variabili di programma e variabili logiche Generalizzando quanto detto sopra per le espressioni booleane, ad un'espressione E del linguaggio MINILW corrisponderà una funzione $[\![E]\!]$ da stati a valori (la sua semantica). Tali funzioni potranno essere utilizzate per descrivere predicati sullo stato. Ad esempio, ad un identificatore di variabile `x` è associata la funzione $[\![\texttt{x}]\!]$ che per ogni stato s restituisce $s(\texttt{x})$. L'esempio sopra dovrebbe quindi essere più correttamente scritto nel modo seguente
$\{ [\![\texttt{x}]\!] = x_0 \wedge [\![\texttt{y}]\!] = y_0 \}$ `while (x` $\neq$ `0) { x=x-1; y=y+1 }` $\{ [\![\texttt{x}]\!] = 0 \wedge [\![\texttt{y}]\!] = x_0 + y_0 \}$
Tuttavia, quando non ci sia ambiguità ometteremo le parentesi semantiche per alleggerire la notazione. Bisogna però sempre tenere presente la differenza tra le *variabili di programma* (come `x`, `y` nell'esempio) che possono essere utilizzate nella descrizione di un predicato sullo stato, e denotano una funzione da stati in valori, e le *variabili logiche* che possono essere utilizzate nel metalinguaggio, come x_0, y_0 nell'esempio per indicare due arbitrari valori interi. Per sottolineare tale differenza utilizzeremo in queste note il font `typewriter` per gli elementi del linguaggio MINILW, mentre utilizzeremo il font *italico* per il metalinguaggio.

Operatori su predicati Data un'asserzione P su *States*, indicheremo con $\{P\}$ l'insieme degli stati che soddisfano P e talvolta potremo confondere per comodità le due nozioni.
Diremo che un'asserzione P è *più forte* di un'asserzione Q se se l'insieme degli stati che verificano P è contenuto nell'insieme degli stati che verificano Q.
Indicheremo con *false* e *true* le asserzioni rispettivamente sempre falsa e sempre vera, che risultano quindi essere la più forte e la più debole:

$\{false\} = \emptyset$,
$\{true\} = States$.

Inoltre, se P, Q sono asserzioni sullo stato, allora $\neg P$, $P \vee Q$, $P \wedge Q$, $P \Rightarrow Q$ sono le asserzioni sullo stato definite da:

$(\neg P)(s) = \neg P(s)$,
$(P \vee Q)(s) = P(s) \vee Q(s)$,
$(P \wedge Q)(s) = P(s) \wedge Q(s)$,
$(P \Rightarrow Q)(s) = P(s) \Rightarrow Q(s)$.

In altri termini questi operatori sono semplicemente gli usuali operatori booleani "trasportati" a livello dei predicati sullo stato.
Vale il fatto seguente:

$P \Rightarrow Q$ sse $\{P\} \subseteq \{Q\}$.

Infine, se P è un predicato sullo stato e E, X sono rispettivamente un'espressione ed un identificatore di MINILW, indicheremo con $P[[\![E]\!]/X]$ il predicato sullo stato definito nel modo seguente:
$P[[\![E]\!]/X](s) = P(s[[\![E]\!]\, s/X]$
dove a destra $-[-/-]$ indica l'usuale operazione di aggiornamento su funzioni parziali.[2]

Validità Diremo che un'asserzione alla Hoare di correttezza parziale $\{P\}\ C\ \{R\}$ è *valida*, e scriveremo $\models \{P\}\ C\ \{R\}$, se e solo se:

per ogni $s \in States$, se $P(s) = tt$ e $[\![C]\!]\, s = s'$, allora $R(s') = tt$.

Si noti che se $[\![C]\!]\, s$ è indefinito, cioè il comando C non termina a partire dallo stato iniziale s, l'implicazione risulta vera (poichè la premessa è sempre falsa). Ciò corrisponde al fatto che stiamo considerando la correttezza parziale. Quindi la definizione potrebbe equivalentemente essere data nel modo seguente:

per ogni $s \in States$, se $P(s) = tt$, allora $[\![C]\!]\, s$ è indefinito oppure $R([\![C]\!]\, s) = tt$.

Se un'asserzione $\{P\}\ C\ \{R\}$ è valida, diremo che P è una *precondizione* per C rispetto a R (e viceversa che R è una *postcondizione* per C rispetto a P).

[2]Data una funzione $f\colon A \to B$, due elementi $a \in A$, $b \in B$, $f[b/a]$ indica la funzione da A in B definita da: $f[b/a](a) = b$, $f[b/a](a') = f(a')$ per $a' \neq a$.

Esempio È possibile provare la validità di un'asserzione alla Hoare direttamente dalla definizione. Si consideri ad esempio la seguente asserzione:

$\{\mathtt{x} > 0 \vee \mathtt{y} > 0\}\ \mathtt{if\ (x>0)\{z=x\}\ else\ \{z=y\}}\ \{\mathtt{z} > 0\}$

Per provare che è valida, consideriamo uno stato s su cui vale la precondizione e l'esecuzione del comando termina, cioè

$s(\mathtt{x}) > 0 \vee s(\mathtt{y}) > 0$,
$[\![\mathtt{if\ (x>0)\{z=x\}\ else\ \{z=y\}}]\!]\, s = s'$.

Dobbiamo far vedere che vale la postcondizione sullo stato finale, cioè $s'(\mathtt{z}) > 0$. Infatti, nel caso $s(\mathtt{x}) > 0$ sia vera, $s' = [\![\mathtt{if\ (x>0)\{z=x\}\ else\ \{z=y\}}]\!]\, s = [\![\mathtt{z=x}]\!]\, s = s[s(\mathtt{x})/\mathtt{z}]$, quindi $s'(\mathtt{z}) = s(\mathtt{x}) > 0$; nel caso $s(\mathtt{x}) > 0$ sia falsa, $s(\mathtt{y}) > 0$ e $s' = [\![\mathtt{if\ (x>0)\{z=x\}\ else\ \{z=y\}}]\!]\, s = [\![\mathtt{z=y}]\!]\, s = s[s(\mathtt{y})/\mathtt{z}]$, quindi $s'(\mathtt{z}) = s(\mathtt{y}) > 0$.

Weakest (liberal) precondition Tra tutte le precondizioni P che rendono valida un'asserzione alla Hoare $\{P\}\ C\ \{R\}$, è di particolare interesse *la più debole*, cioè quella implicata da tutte le altre. Infatti questa precondizione può essere vista come un insieme minimo di requisiti da richiedere per ottenere che R valga se il comando termina. Formalmente, definiremo, per ogni C comando e R postcondizione, la *weakest (liberal) precondition*[3] (abbreviato wlp) di C rispetto ad R, denotata $wlp(C,R)$, nel modo seguente:

$wlp(C,R)(s) = tt$ sse $[\![C]\!]\, s = s' \Rightarrow R(s')$.

Si noti che, se $[\![C]\!]\, s$ è indefinito, cioè il comando C non termina a partire dallo stato iniziale s, l'implicazione risulta vera (poichè la premessa è sempre falsa) quindi $wlp(C,R)(s)$ è vera. Ciò corrisponde al fatto che stiamo considerando correttezza parziale. Quindi la definizione potrebbe equivalentemente essere data nel modo seguente:

$wlp(C,R)(s) = tt$ sse $[\![C]\!]\, s$ indefinito oppure $R([\![C]\!]\, s)$.

La definizione può anche essere data in maniera equivalente in termini di insiemi di stati:

$\{wlp(C,R)\} = \{s \mid [\![C]\!]\, s = s' \Rightarrow R(s')\} = \{s \mid [\![C]\!]\, s \text{ indefinito oppure } R([\![C]\!]\, s)\}$.

Ossia, la weakest (liberal) precondition di C rispetto a R è data da tutti gli stati iniziali per cui eseguendo C si ottiene o non terminazione o uno stato finale su cui vale R.
Proviamo ora che $wlp(C,R)$ è effettivamente la precondizione più debole per C rispetto a R, ossia che: è davvero una precondizione per C rispetto a R; ogni altra precondizione per C rispetto a R la implica. Formalmente:

1. $\models \{wlp(C,R)\}\ C\ \{R\}$
2. se $\models \{P\}\ C\ \{R\}$, allora $P \Rightarrow wlp(C,R)$.

Prova di (1) Dobbiamo provare che per ogni $s \in States$, se $wlp(C,R)(s) = tt$ e $[\![C]\!]\, s = s'$, allora $R(s') = tt$. Ma questo segue direttamente dalla definizione di $wlp(C,R)$.

Prova di (2) Sia $\models \{P\}\ C\ \{R\}$. Dobbiamo provare che, per ogni $s \in States$, se $P(s) = tt$ allora $wlp(C,R)(s) = tt$. Ma dalla validità di $\{P\}\ C\ \{R\}$ sappiamo che $[\![C]\!](s) = s' \Rightarrow R(s')$, quindi $wlp(C,R)(s) = tt$.

Esempio Ricaviamo, a titolo di esempio, la wlp del comando $C = \mathtt{if\ (x>0)\{z=x\}\ else\ \{z=y\}}$ rispetto alla postcondizione $\mathtt{z} > 0$. Dalla definizione, $wlp(C, \mathtt{z} > 0)(s) = tt$ sse $[\![C]\!]\, s = s' \Rightarrow \mathtt{z}(s') > 0$. Poichè $[\![C]\!]\, s = [\![\mathtt{z=x}]\!](s) = s[s(\mathtt{x})/\mathtt{z}]$ se $s(\mathtt{x}) > 0$, e $[\![C]\!]\, s = [\![\mathtt{z=y}]\!](s) = s[s(\mathtt{y})/\mathtt{z}]$ se $s(\mathtt{x}) \leq 0$, si ha che $wlp(C, \mathtt{x} > 0)(s) = tt$ sse $s(\mathtt{x}) > 0$ oppure $s(\mathtt{y}) > 0$.

Diamo ora una caratterizzazione esplicita della weakest (liberal) precondition per i comandi di MINILW (omettiamo la caratterizzazione per il comando while).

Skip Per definizione di wlp $wlp(\mathtt{skip}, R)(s) = tt$ sse $[\![\mathtt{skip}]\!]\, s = s' \Rightarrow R(s')$, quindi per definizione di $[\![\mathtt{skip}]\!]$ sse $R(s)$. Quindi $wlp(\mathtt{skip}, R) = R$.

Assegnazione Per definizione di wlp $wlp(X = E, R)(s) = tt$ sse $[\![X = E]\!]\, s = s' \Rightarrow R(s')$, quindi per definizione di $[\![X = E]\!]$ sse $R(s[[\![E]\!]\, s/X])$. Quindi $wlp(X = E, R) = R[[\![E]\!]\, /X]$.

Sequenza Per definizione di wlp $wlp(C_1\,;\,C_2, R)(s) = tt$ sse $[\![C_1\,;\,C_2]\!]\, s = s' \Rightarrow R(s')$, quindi per definizione di $[\![C_1\,;\,C_2]\!]$ sse $[\![C_2]\!]([\![C_1]\!]\, s) = s' \Rightarrow R(s')$ sse $[\![C1]\!]\, s = s'' \Rightarrow wlp(C_2,R)(s'')$ sse $wlp(C_1, wlp(C_2,R))(s)$. Quindi $wlp(C_1\,;\,C_2, R) = wlp(C_1, wlp(C_2,R))$.

If Per definizione di wlp $wlp(\mathtt{if\ (}B\mathtt{)\ \{}C_1\mathtt{\}\ else\ \{}C_2\mathtt{\}}, R)(s) = tt$ sse $[\![\mathtt{if\ (}B\mathtt{)\ \{}C_1\mathtt{\}\ else\ \{}C_2\mathtt{\}}]\!]\, s = s' \Rightarrow R(s')$, quindi per definizione di $[\![\mathtt{if\ (}B\mathtt{)\ \{}C_1\mathtt{\}\ else\ \{}C_2\mathtt{\}}]\!]$ sse $[\![B]\!]\, s = tt$ e $[\![C_1]\!]\, s = s' \Rightarrow R(s')$ oppure $[\![B]\!]\, s = ff$ e $[\![C_2]\!]\, s = s' \Rightarrow R(s')$. Quindi $wlp(\mathtt{if\ (}B\mathtt{)\ \{}C_1\mathtt{\}\ else\ \{}C_2\mathtt{\}}, R) = ([\![B]\!] \wedge wlp(C_1,R)) \vee (\neg\, [\![B]\!] \wedge wlp(C_1,R))$.

Un modo alternativo di provare che un'asserzione alla Hoare $\{P\}\ C\ \{R\}$ è valida è quindi di provare che $P \Rightarrow wlp(C,R)$.

[3] L'appellativo "liberal" distingue questa nozione da quella analoga nel caso di correttezza totale.

(Skip) $$\frac{}{\{R\}\ \texttt{skip}\ \{R\}}$$

(Assign) $$\frac{}{\{R[[\![E]\!]/X]\}\ X = E\ \{R\}}$$

(Seq) $$\frac{\{P\}\ C_1\ \{Q\} \quad \{Q\}\ C_2\ \{R\}}{\{P\}\ C_1 ; C_2\ \{R\}}$$

(If) $$\frac{\{P \wedge [\![B]\!]\}\ C_1\ \{R\} \quad \{P \wedge \neg [\![B]\!]\}\ C_2\ \{R\}}{\{P\}\ \texttt{if (}B\texttt{)}\{C_1\}\ \texttt{else}\ \{C_2\}\ \{R\}}$$

(While) $$\frac{\{INV \wedge [\![B]\!]\}\ C\ \{INV\}}{\{INV\}\ \texttt{while (}B\texttt{) \{ }C\texttt{ \}}\ \{INV \wedge \neg [\![B]\!]\}}$$

(Cons) $$\frac{\{P'\}\ C\ \{R'\} \qquad \begin{array}{l} P \Rightarrow P' \\ R' \Rightarrow R \end{array}}{\{P\}\ C\ \{R\}}$$

Figura 1: Sistema di prova di Hoare per la correttezza parziale

---

Sistema di prova Definiamo ora un *sistema di prova* che ci fornirà un metodo molto più pratico per provare la validità di un'asserzione alla Hoare.
Nel contesto dei metodi di specifica formale, fissato un certo insieme di asserzioni e definitane formalmente la validità, si intende in genere per *sistema di prova* una definizione induttiva, data tramite metaregole, di un sottoinsieme delle asserzioni.
Il sistema di prova di Hoare per la correttezza parziale è dato in Fig.1.
Il sistema di prova è formato da sei regole di inferenza (o metaregole), una per ogni tipo di comando di MINILW più la regola (Cons) che permette di rafforzare la precondizione ed indebolire la postcondizione in un'asserzione alla Hoare.
Il termine metaregole sta a significare che ognuna di esse definisce uno schema di regola, nel senso che le metavariabili (ad esempio C, P, R, E, ...) che vi appaiono possono essere istanziate con arbitrari comandi, predicati, espressioni, etc. Per ogni istanziazione si ottiene una *regola* (nel senso dei sistemi induttivi), cioè una coppia $\langle$ insieme di premesse, conseguenza$\rangle$ in cui premesse e conseguenza sono asserzioni alla Hoare. Il sistema di prova di Hoare definisce quindi induttivamente un sottoinsieme $\mathcal{H}$ delle asserzioni alla Hoare; scriveremo $\vdash \{P\}\ C\ \{R\}$ per indicare che l'asserzione $\{P\}\ C\ \{R\}$ appartiene all'insieme $\mathcal{H}$, ossia è deducibile (ha un albero di prova). Infatti, dato un sistema di prova, l'uso del sistema per provare la validità di un'asserzione consiste nel costruire un albero (etichettato con asserzioni), detto albero di prova, che abbia l'asserzione da provare come radice e dove, per ogni nodo, l'etichetta del nodo e le etichette dei figli corrispondano rispettivamente alla conseguenza ed alle premesse di un'istanziazione di una metaregola. In particolare quindi le foglie devono corrispondere ad istanziazioni di assiomi (regole senza premesse).
Si noti che il sistema *non* fornisce un algoritmo per controllare se un'asserzione è valida o no. Infatti, anzitutto per poter istanziare la regola (Cons) occorre essere in grado di stabilire se valgono le due implicazioni nelle side conditions. Inoltre, per poter istanziare la regola (Seq) in modo da provare la conseguenza è necessario trovare un'asserzione intermedia Q tale che le premesse siano verificate. Analogamente, per poter istanziare la regola (While) in modo da provare la conseguenza è necessario trovare un'asserzione INV tale che la premessa sia verificata. Un'asserzione tale che la premessa $\{INV \wedge [\![B]\!]\}\ C\ \{INV\}$ sia verificata si dice *invariante* del ciclo while; infatti, assumendo che essa valga prima di eseguire il ciclo while, dalla premessa si ha che essa vale ancora ogni volta che si esegue il body del ciclo while, e quindi è una asserzione la cui verità non *varia* qualunque sia il numero di iterazioni eseguite.
Proviamo ora che il sistema di Hoare permette di dedurre tutte e sole le asserzioni valide. Ciò è espresso formalmente dal seguente teorema.

Teorema 2.1 Il sistema di prova di Hoare per la correttezza parziale dato in Fig.1 è sound e completo rispetto alla validità, cioè

$$\models \{P\}\ C\ \{R\} \text{ sse } \vdash \{P\}\ C\ \{R\}.$$

Dato un sistema di prova, dire che il sistema è *sound* significa che non è possibile dedurre cose sbagliate, cioè non è possibile provare asserzioni non valide. Questa è una proprietà "indispensabile" che ci aspettiamo valere per ogni sistema di prova. Invece, dire che il sistema è *completo* significa che è possibile dedurre *tutte* le cose giuste, cioè è possibile provare tutte le asserzioni valide. Questa è una proprietà che non sempre vale per un sistema di prova, perchè in genere il sistema potrebbe non essere sufficientemente potente per riuscire a provare tutte le asserzioni valide. Nel nostro caso, il risultato di completezza vale nell'approccio estensionale, cioè assumendo che la precondizione e la postcondizione siano predicati arbitrari sullo stato. In un approccio intensionale dove la precondizione e la postcondizione sono espresse in un linguaggio di asserzioni fissato, il fatto che la completezza valga o meno dipende da quanto è potente tale linguaggio.

Prova di soundness Dobbiamo provare che $\vdash \{P\}\ C\ \{R\} \Rightarrow \models \{P\}\ C\ \{R\}$. La prova è per induzione sulla definizione di $\mathcal{H}$, cioè dobbiamo provare che ogni (istanziazione di) metaregola è sound nel senso che se le premesse sono valide allora la conseguenza è valida. Nella terminologia dei sistemi induttivi questo corrisponde a provare che l'insieme delle asserzioni valide è chiuso rispetto alla definizione induttiva di $\mathcal{H}$.
Per ogni metaregola relativa ad un tipo di comando daremo una prova intuitiva informale, una prova formale diretta (cioè dalla definizione di validità) che corrisponde ad una formalizzazione della prova intuitiva, ed una prova formale alternativa che utilizza la caratterizzazione della wlp per quel comando (non daremo quindi questa prova formale alternativa per il comando while).

Skip **Prova informale** Se R vale prima di eseguire `skip`, dato che il comando non fa nulla è chiaro che R vale anche dopo.

Prova diretta Dobbiamo provare che per ogni $s \in States$, se $P(s) = tt$, allora $[\![\texttt{skip}]\!]\, s = s' \Rightarrow R(s') = tt$. Poichè $[\![\texttt{skip}]\!]\, s = s$ la tesi vale banalmente.

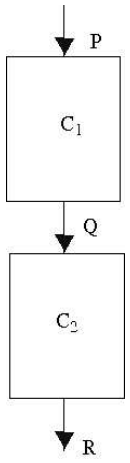
Prova con wlp Dobbiamo provare che $R \Rightarrow R$. Poichè $wlp(\texttt{skip}, R) = R$ la tesi vale banalmente.

Assegnazione **Prova informale** Se $R[[\![E]\!]/X]$ vale prima di eseguire $X = E$, dato che il comando aggiorna lo stato associando $[\![E]\!]$ ad X è chiaro che vale R dopo.

Prova diretta Dobbiamo provare che per ogni $s \in States$, se $P(s) = tt$, allora $[\![X = E]\!]\, s = s' \Rightarrow R(s') = tt$. Poichè $[\![X = E]\!]\, s = s[[\![E]\!]/X]$, dobbiamo provare che $R(s[[\![E]\!]/X]) = tt$, e la tesi vale per definizione dell'operatore di aggiornamento.

Prova con wlp Dobbiamo provare che $R[[\![E]\!]/X] \Rightarrow wlp(X = E, R)$. Poichè $wlp(X = E, R) = R[[\![E]\!]/X]$ la tesi vale banalmente.

Sequenza **Prova informale** Dobbiamo provare che se vale P prima di eseguire $C_1\,;\,C_2$ e il comando termina, allora vale R. Se il comando termina deve ovviamente terminare anche C_1, e dalla prima premessa sappiamo allora che vale Q, quindi vale Q prima di eseguire C_2. Anche C_2 deve ovviamente terminare, quindi dalla seconda premessa sappiamo che vale R.

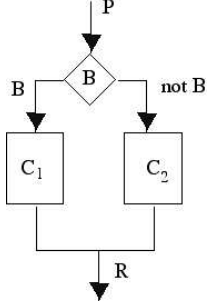


Prova diretta Dobbiamo provare che per ogni $s \in States$, se $P(s) = tt$, allora $[\![C_1\,;\,C_2]\!]\, s = s' \Rightarrow R(s') = tt$, cioè $[\![C_2]\!]([\![C_1]\!]\, s) = s' \Rightarrow R(s') = tt$. Se $[\![C_2]\!]([\![C_1]\!]\, s) = s'$, deve esistere uno stato s'' tale che $[\![C_1]\!]\, s = s''$ e $[\![C_2]\!]\, s'' = s'$. Allora dal fatto che la prima premessa è valida si ha che $Q(s'') = tt$, e da questo più la validità della seconda premessa si ha che $R(s') = tt$.

Prova con wlp Dobbiamo provare che $P \Rightarrow wlp(C_1, wlp(C_2, R))$. Dalla prima premessa sappiamo che $P \Rightarrow wlp(C_1, Q)$; dalla seconda premessa sappiamo che $Q \Rightarrow wlp(C_2, R)$ e quindi, dato che l'operatore $wlp(C, -)$ è monotono[4], che $wlp(C_1, Q) \Rightarrow wlp(C_1, wlp(C_2, R))$. Dai due fatti segue la tesi per transitività dell'implicazione.

If **Prova informale** Dobbiamo provare che se vale P prima di eseguire `if (`B`)` $\{C_1\}$ `else` $\{C_2\}$ e il comando termina, allora vale R. Eseguire il comando significa anzitutto valutare l'espressione B; poichè abbiamo assunto che la valutazione delle espressioni non possa dar luogo a non terminazione, otterremo *tt* o *ff*. Nel primo caso si esegue C_1, l'esecuzione termina per l'ipotesi quindi dalla prima premessa sappiamo che vale R dopo. Analogamente nel secondo caso.

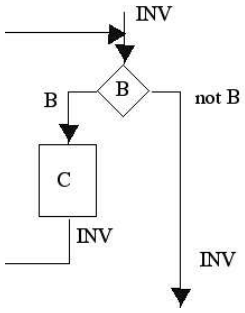
[4]Cioè se $P \Rightarrow Q$, allora per ogni comando C si ha che $wlp(C, P) \Rightarrow wlp(C, Q)$; lo si provi per esercizio.



Prova diretta Dobbiamo provare che per ogni $s \in States$, se $P(s) = tt$, allora $[\![\texttt{if (}B\texttt{) \{}C_1\texttt{\} else \{}C_2\texttt{\}}]\!]\, s = s' \Rightarrow R(s') = tt$. Sia $[\![B]\!]\, s = tt$. Allora $[\![\texttt{if (}B\texttt{) \{}C_1\texttt{\} else \{}C_2\texttt{\}}]\!]\, s = [\![C_1]\!]\, s$, e la tesi segue dal fatto che la prima premessa è valida. Analogamente se $[\![B]\!]\, s = ff$.

Prova con wlp Dobbiamo provare che $P \Rightarrow ([\![B]\!] \wedge wlp(C_1, R)) \vee (\neg\, [\![B]\!] \wedge wlp(C_1, R))$. Dalla prima premessa sappiamo che $P \wedge [\![B]\!] \Rightarrow wlp(C_1, R)$; dalla seconda premessa sappiamo che $P \wedge \neg\, [\![B]\!] \Rightarrow wlp(C_2, R)$. Dai due fatti segue facilmente la tesi.

While **Prova informale** Dobbiamo provare che se vale *INV* prima di eseguire `while (B) { C }` e il comando termina, allora vale $INV \wedge \neg\, [\![B]\!]$. Se il comando termina vuol dire che si è eseguito un numero finito di volte il body, sia $n \geq 0$ questo numero. È semplice allora provare per induzione aritmetica su n che ogni volta all'inizio dell'esecuzione del body vale *INV*: infatti questo vale la prima volta ed ogni volta continua a valere in base alla premessa. All'ultimo passo quindi vale ancora *INV*, vale $\neg B$ perchè altrimenti si effettuerebbe ancora un passo e quindi non si fa nulla e vale *INV* alla fine.



Prova diretta Dobbiamo provare che per ogni $s \in States$, se $INV(s) = tt$, allora $[\![\texttt{while (}B\texttt{) \{}C\texttt{\}}]\!]\, s = s' \Rightarrow (INV \wedge \neg\, [\![B]\!])(s') = tt$. Proviamo questa tesi per induzione sulla definizione della funzione $[\![\texttt{while (}B\texttt{) \{}C\texttt{\}}]\!]$.

- Sia $[\![B]\!]\, s = ff$, quindi $[\![\texttt{while (}B\texttt{) \{}C\texttt{\}}]\!]\, s = s$. Allora la tesi vale banalmente.
- Sia $[\![B]\!]\, s = tt$, quindi $[\![\texttt{while (}B\texttt{) \{}C\texttt{\}}]\!]\, s = s' = [\![\texttt{while (}B\texttt{) \{}C\texttt{\}}]\!]([\![C]\!]\, s)$; allora esiste s'' tale che $s'' = [\![C]\!]\, s$, quindi dalla premessa sappiamo che vale $INV(s'')$ e per ipotesi induttiva, dato che $[\![\texttt{while (}B\texttt{) \{}C\texttt{\}}]\!]\, s'' = s'$, si ha la tesi.

Cons Dobbiamo provare che se $\{P'\}\ C\ \{R'\}$ è valida, allora lo è anche $\{P\}\ C\ \{R\}$ se $P \Rightarrow P'$ e $R' \Rightarrow R$; questo segue banalmente dalla definizione di validità.

Prova di completezza Dobbiamo provare che $\models \{P\}\ C\ \{R\} \Rightarrow \vdash \{P\}\ C\ \{R\}$. A tale scopo, proveremo che

(*) dato un comando C ed una postcondizione R si ha $\vdash \{wlp(C, R)\}\ C\ \{R\}$.

Infatti a questo punto $\vdash \{P\}\ C\ \{R\}$ si può provare istanziando la regola (Cons), infatti le condizioni a lato sono verificate perchè se $\models \{P\}\ C\ \{R\}$ sappiamo che $P \Rightarrow wlp(C, R)$.
La prova di (*) è per induzione sulla struttura dei comandi di MINILW, cioè dobbiamo provare che l'implicazione sopra vale per un comando C (con R arbitraria) se vale per i sottocomandi che lo compongono.

Skip Sappiamo che $wlp(\texttt{skip}, R) = R$, e $\vdash \{R\}$ `skip` $\{R\}$ si ottiene istanziando la regola (Skip).

Assegnazione Sappiamo che $wlp(X = E, R) = R[[\![E]\!]/X]$, e $\vdash \{R[[\![E]\!]/X]\}$ `skip` $\{R\}$ si ottiene istanziando la regola (Assign).

Sequenza Sappiamo che $wlp(C_1\texttt{;}\, C_2, R) = wlp(C_1, wlp(C_2, R))$, e $\vdash \{wlp(C_1, wlp(C_2, R))\}\ C_1\texttt{;}\, C_2\ \{R\}$ si ottiene istanziando la regola (Seq) nel modo seguente:

$$\frac{\{wlp(C_1, wlp(C_2, R))\}\ C_1\ \{wlp(C_2, R)\} \quad \{wlp(C_2, R)\}\ C_2\ \{R\}}{\{wlp(C_1, wlp(C_2, R))\}\ C_1\texttt{;}\, C_2\ \{R\}}\ (Seq)$$

e osservando che le premesse sono deducibili per ipotesi induttiva.

(Skip+Cons) $\dfrac{}{\{P\}\ \texttt{skip}\ \{R\}} \quad P \Rightarrow R$

(Assign+Cons) $\dfrac{}{\{P\}\ X = E\ \{R\}} \quad P \Rightarrow R[[\![E]\!]/X]$

(While+Cons) $\dfrac{\{INV \wedge [\![B]\!]\}\ C\ \{INV\}}{\{P\}\ \texttt{while (}B\texttt{) \{ }C\texttt{ \}}\ \{R\}} \quad \begin{array}{l} P \Rightarrow INV \\ INV \wedge \neg [\![B]\!] \Rightarrow R \end{array}$

Figura 2: Regole derivate

If Sia $P_{if} = wlp(\texttt{if (}B\texttt{)}\ \{C_1\}\ \texttt{else}\ \{C_2\}, R) = ([\![B]\!] \wedge wlp(C_1, R)) \vee (\neg [\![B]\!] \wedge wlp(C_1, R))$. Allora $\vdash \{P_{if}\}\ \texttt{if (}B\texttt{)}\ \{C_1\}\ \texttt{else}\ \{C_2\}\ \{R\}$ si ottiene con il seguente albero di prova:

$$\dfrac{\dfrac{\{wlp(C_1,R)\}\ C_1\ \{R\}}{\{P_{if} \wedge [\![B]\!]\}\ C_1\ \{R\}}\textit{(Cons)} \quad \dfrac{\{wlp(C_2,R)\}\ C_2\ \{R\}}{\{P_{if} \wedge \neg [\![B]\!]\}\ C_2\ \{R\}}\textit{(Cons)}}{\{P_{if}\}\ \texttt{if (}B\texttt{)}\ \{C_1\}\ \texttt{else}\ \{C_2\}\ \{R\}}\textit{(If)}$$

osservando che le premesse sono deducibili per ipotesi induttiva.

While Sia $INV = wlp(\texttt{while (}B\texttt{) \{ }C\texttt{ \}}, R)$. Allora $\vdash \{INV\}\ \texttt{while (}B\texttt{) \{ }C\texttt{ \}}\ \{R\}$ si ottiene con il seguente albero di prova:

$$\dfrac{\dfrac{\dfrac{\{wlp(C,INV)\}\ C\ \{INV\}}{\{[\![B]\!] \wedge INV\}\ C\ \{INV\}}\textit{(Cons)}}{\{INV\}\ \texttt{while (}B\texttt{) \{ }C\texttt{ \}}\ \{\neg [\![B]\!] \wedge INV\}}\textit{(While)}}{\{INV\}\ \texttt{while (}B\texttt{) \{ }C\texttt{ \}}\ \{R\}}\textit{(Cons)}$$

osservando che la premessa $\{wlp(C,INV)\}\ C\ \{INV\}$ è deducibile per ipotesi induttiva e che le istanziazioni della regola (Cons) sono lecite perchè valgono le due seguenti implicazioni.

1. $\neg [\![B]\!] \wedge INV \Rightarrow R$,
2. $[\![B]\!] \wedge INV \Rightarrow wlp(C, INV)$.

Prova di (1) Sia s tale che $[\![B]\!]\, s = ff$ e $INV(s) = tt$. Dobbiamo far vedere che $R(s) = tt$. Poichè $[\![B]\!]\, s = ff$, $[\![\texttt{while (}B\texttt{) \{ }C\texttt{ \}}]\!]\, s = s$, e dalla definizione di INV si ha la tesi.

Prova di (2) Sia s tale che $[\![B]\!]\, s = tt$ e $INV(s) = tt$. Dobbiamo far vedere che $wlp(C, INV)(s) = tt$, cioè che se $[\![C]\!]\, s = s'$ allora $INV(s') = tt$, cioè che se $[\![\texttt{while (}B\texttt{) \{ }C\texttt{ \}}]\!]\, s' = s''$ allora $R(s'') = tt$. Sia effettivamente $[\![C]\!]\, s = s'$ e $[\![\texttt{while (}B\texttt{) \{ }C\texttt{ \}}]\!]\, s' = s''$; poichè $[\![B]\!]\, s = tt$, $[\![\texttt{while (}B\texttt{) \{ }C\texttt{ \}}]\!]\, s = [\![\texttt{while (}B\texttt{) \{ }C\texttt{ \}}]\!]\, s' = s''$, e poichè sappiamo che $INV(s) = tt$ si ha la tesi.

Regole derivate Nel seguito useremo per comodità le regole derivate in Fig.2 ottenute combinando una regola relativa ad un tipo di comando con la regola (Cons).

Esempio Mostriamo ora un esempio di prova di validità di un'asserzione di correttezza parziale utilizzando il sistema di Hoare. Consideriamo l'asserzione

$$\{\mathtt{x} = x_0 \wedge \mathtt{y} = y_0\}\ C\ \{\mathtt{x} = 0 \wedge \mathtt{y} = x_0 + y_0\}$$

dove C indica il comando $\texttt{while (x} \neq \texttt{0) \{ x=x-1; y=y+1 \}}$.
Un albero di prova per quest'asserzione è il seguente, dove come invariante del ciclo abbiamo scelto $INV = \mathtt{x} + \mathtt{y} = x_0 + y_0$.

$$\dfrac{\dfrac{\dfrac{}{\{INV \wedge \mathtt{x} \neq 0\}\ \texttt{x=x-1}\ \{INV[\mathtt{y}+1/\mathtt{y}]\}}\textit{(Assign+Cons)} \quad \dfrac{}{\{INV[\mathtt{y}+1/\mathtt{y}]\}\ \texttt{y=y+1}\ \{INV\}}\textit{(Assign)}}{\{INV \wedge \mathtt{x} \neq 0\}\ \texttt{x=x-1; y=y+1}\ \{INV\}}\textit{(Seq)}}{\{\mathtt{x} = x_0 \wedge \mathtt{y} = y_0\}\ C\ \{\mathtt{x} = 0 \wedge \mathtt{y} = x_0 + y_0\}}\textit{(While+Cons)}$$

L'istanziazione della regola (While+Cons) è corretta perchè le side conditions sono banalmente verificate:

- $\mathtt{x} = x_0 \wedge \mathtt{y} = y_0 \Rightarrow INV$,
- $(INV \wedge \mathtt{x} = 0) \Rightarrow (\mathtt{x} = 0 \wedge \mathtt{y} = x_0 + y_0)$.

Inoltre, l'istanziazione della regola (Assign+Cons) è corretta perchè $INV \wedge \mathtt{x} \neq 0$ implica $INV[\mathtt{y}+1/\mathtt{y}][\mathtt{x}-1/\mathtt{x}] = INV$.

Esempio Mostriamo un altro esempio che illustra come possiamo esprimere e provare la non terminazione di un comando. Consideriamo l'asserzione

$$\{\mathtt{x} < 0\}\ C\ \{\mathit{false}\}$$

dove C indica il comando `while (x` $\neq$ `0) { x=x-1; y=y+1 }`.
Si noti che, in base alla definizione, richiedere che questa asserzione sia valida corrisponde a richiedere che il comando C non termini a partire da stati che verificano la precondizione.
Un albero di prova per quest'asserzione è il seguente, dove come invariante del ciclo abbiamo scelto $\mathtt{x} < 0$.

$$\dfrac{\dfrac{\dfrac{}{\{\mathtt{x} < 0\}\ \mathtt{x=x-1}\ \{\mathtt{x} < 0[\mathtt{y}+1/\mathtt{y}]\}}\textit{(Assign+Cons)} \quad \dfrac{}{\{\mathtt{x} < 0[\mathtt{y}+1/\mathtt{y}]\}\ \mathtt{y=y+1}\ \{\mathtt{x} < 0\}}\textit{(Assign)}}{\{\mathtt{x} < 0\}\ \mathtt{x=x-1;y=y+1}\ \{\mathtt{x} < 0\}}\textit{(Seq)}}{\{\mathtt{x} < 0\}\ C\ \{\mathit{false}\}}\textit{(While+Cons)}$$

L'istanziazione della regola (While+Cons) è corretta perchè le side conditions sono banalmente verificate:

- $\mathtt{x} < 0 \Rightarrow \mathtt{x} < 0$,
- $(\mathtt{x} < 0 \wedge \mathtt{x} = 0) \Rightarrow \mathit{false}$.

Inoltre, l'istanziazione della regola (Assign+Cons) è corretta perchè $\mathtt{x} < 0$ implica $\mathtt{x} < 0[\mathtt{y}+1/\mathtt{y}][\mathtt{x}-1/\mathtt{x}] = \mathtt{x} - 1 < 0$.

3 Asserzioni alla Hoare di correttezza totale

Le asserzioni alla Hoare *di correttezza totale* sono triple della forma $\{P\}\ C\ \{\Downarrow R\}$, dove su P, R e C valgono le stesse assunzioni e la stessa terminologia che nel caso delle asserzioni di correttezza parziale.
Intuitivamente, il significato di tali asserzioni è il seguente:

> se P vale nello stato iniziale (cioè prima di eseguire C), allora l'esecuzione di C deve terminare producendo uno stato finale su cui vale R.

Si noti che, in questo caso, perchè l'asserzione sia valida si richiede che l'esecuzione del comando termini.

Esempio Un esempio di asserzione alla Hoare di correttezza totale è il seguente.

$$\{\mathtt{x} = x_0 \wedge \mathtt{y} = y_0 \wedge x_0 \geq 0\}\ \mathtt{while\ (x} \neq \mathtt{0)\ \{\ x=x-1;y=y+1\ \}}\ \{\Downarrow \mathtt{x} = 0 \wedge \mathtt{y} = x_0 + y_0\}$$

Questa asserzione è intuitivamente valida. Infatti, se $x_0 \geq 0$ è facile vedere che l'esecuzione del comando termina ed alla fine $\mathtt{x}$ ha sicuramente assunto il valore 0 e $\mathtt{y}$ ha incrementato il suo valore iniziale di una quantità uguale al valore iniziale di $\mathtt{x}$.

Validità Diremo che un'asserzione alla Hoare di correttezza totale $\{P\}\ C\ \{R\}$ è *valida*, e scriveremo $\models \{P\}\ C\ \{\Downarrow R\}$, se e solo se:

> per ogni $s \in \mathit{States}$, se $P(s) = \mathit{tt}$, allora $R([\![C]\!]\, s) = \mathit{tt}$.

Si noti che se $[\![C]\!]\, s$ è indefinito, cioè il comando C non termina a partire dallo stato iniziale s, l'implicazione risulta falsa. Ciò corrisponde al fatto che stiamo considerando la correttezza totale.

Weakest precondition Analogamente al caso parziale, tra tutte le precondizioni P che rendono valida un'asserzione alla Hoare di correttezza totale $\{P\}\ C\ \{\Downarrow R\}$, è di particolare interesse *la più debole*, cioè quella implicata da tutte le altre. Questa precondizione può essere vista come un insieme minimo di requisiti da richiedere per ottenere che il comando termini in uno stato in cui vale R. Formalmente, definiremo, per ogni C comando e R postcondizione, la *weakest precondition* (abbreviato wp) di C rispetto ad R, denotata $wp(C,R)$, nel modo seguente:

$$wp(C,R)(s) = \mathit{tt} \text{ sse } R([\![C]\!]\, s).$$

Si noti che, se $[\![C]\!]\, s$ è indefinito, cioè il comando C non termina a partire dallo stato iniziale s, la wp risulta falsa.
La definizione può anche essere data in maniera equivalente in termini di insiemi di stati:

$$\{wp(C,R)\} = \{s \mid R([\![C]\!]\, s)\}.$$

Ossia, la wp di C rispetto a R è data da tutti gli stati iniziali per cui eseguendo C si ottiene uno stato finale su cui vale R.
Si può facilmente provare, analogamente al caso parziale, che $R([\![C]\!]\, s)$ è effettivamente la precondizione più debole per C rispetto a R. Inoltre, si vede facilmente che la caratterizzazione esplicita della wlp data precedentemente per i comandi di MINILW coincide con quella della wp.

(Skip) $$\frac{}{\{R\}\ \texttt{skip}\ \{\Downarrow R\}}$$

(Assign) $$\frac{}{\{R[[\![E]\!]/X]\}\ X = E\ \{\Downarrow R\}}$$

(Seq) $$\frac{\{P\}\ C_1\ \{\Downarrow Q\} \quad \{Q\}\ C_2\ \{\Downarrow R\}}{\{P\}\ C_1 ; C_2\ \{\Downarrow R\}}$$

(If) $$\frac{\{P \wedge [\![B]\!]\}\ C_1\ \{\Downarrow R\} \quad \{P \wedge \neg [\![B]\!]\}\ C_2\ \{\Downarrow R\}}{\{P\}\ \texttt{if (}B\texttt{)}\ \{C_1\}\ \texttt{else}\ \{C_2\}\ \{\Downarrow R\}}$$

(While) $$\frac{\{INV \wedge [\![B]\!]\}\ C\ \{\Downarrow INV\}}{\{INV\}\ \texttt{while (}B\texttt{)}\ \{\ C\ \}\ \{\Downarrow INV \wedge \neg [\![B]\!]\}}$$

$\exists t$: *States* $\to P$ funzione di terminazione per `while (`B`)` $\{\ C\ \}$ e *INV*

(Cons) $$\frac{\{P'\}\ C\ \{\Downarrow R'\}}{\{P\}\ C\ \{\Downarrow R\}} \quad \begin{array}{l} P \Rightarrow P' \\ R' \Rightarrow R \end{array}$$

Figura 3: Sistema di prova di Hoare per la correttezza totale

---

Sistema di prova Il sistema di prova nel caso totale differisce da quello nel caso parziale solo per la regola relativa al ciclo while. In questo caso infatti occorre anche garantire la terminazione del comando a partire dalla precondizione data.
Prima di dare la nuova regola, premettiamo alcune definizioni.
Un *ordine parziale* su un insieme P è una relazione binaria su P tale che, per ogni $x, y, z \in P$:

- $x \leq x$ ($\leq$ è *riflessiva*),
- se $x \leq y$ e $y \leq x$ allora $x = y$ ($\leq$ è *antisimmetrica*),
- se $x \leq y$ e $y \leq z$ allora $x \leq z$ ($\leq$ è *transitiva*).

Se $\leq$ è un ordine parziale su P diremo anche che P è un *insieme parzialmente ordinato* (rispetto a $\leq$). Dato $\leq$, possiamo definire la relazione binaria $<$ su P di *ordine parziale stretto* come segue:

$x < y$ sse $x \leq y$ e $x \neq y$.

È facile vedere che è anche sempre possibile definire un ordine parziale a partire da un ordine parziale stretto; quindi è indifferente dare l'uno o l'altro.
Se P è un insieme parzialmente ordinato rispetto a $\leq$, allora ogni suo sottoinsieme P' è ancora un insieme parzialmente ordinato rispetto alla restrizione di $\leq$ a P'.
Una *catena (strettamente) discendente infinita* in un insieme P parzialmente ordinato è un sottoinsieme C di P tale che, per ogni $x \in C$, esiste $y \in C$ tale che $y < x$.
Un insieme P parzialmente ordinato è *noetheriano* (*ben fondato*) se non contiene catene discendenti infinite.
Sia `while (`B`)` $\{\ C\ \}$ un comando while e *INV* un'asserzione. Una *funzione di terminazione* per `while (`B`)` $\{\ C\ \}$ e *INV* è una funzione t: *States* $\to P$ con P insieme parzialmente ordinato che soddisfa le seguenti due condizioni:

1. $t(\{INV \wedge [\![B]\!]\}) \subseteq N$ con N sottoinsieme noetheriano di P;
2. per ogni s in $\{INV \wedge [\![B]\!]\}$, $t([\![C]\!]\, s) < t(s)$.

Il sistema di prova di Hoare per la correttezza totale è dato in Fig.3. Le regole sono analoghe a quelle date per la correttezza parziale, tranne la regola del while.

Teorema 3.1 Il sistema di prova di Hoare per la correttezza totale dato in Fig.3) è sound e completo rispetto alla validità, cioè

$\models \{P\}\ C\ \{\Downarrow R\}$ sse $\vdash \{P\}\ C\ \{\Downarrow R\}$.

Prova di soundness Analogamente a quanto visto per il caso parziale, dobbiamo provare che ogni (istanziazione di) metaregola è sound nel senso che se le premesse sono valide allora la conseguenza è valida. Diamo solo la prova per la regola (While); per le altre metaregole la prova è analoga a quella data nel caso parziale.

While Il fatto che se vale *INV* prima di eseguire `while (B) { C }` ed il comando termina, allora vale $INV \wedge \neg [\![B]\!]$ si può provare in modo analogo a quanto visto per il caso parziale. Dobbiamo quindi provare che se vale *INV* prima di eseguire `while (B) { C }`, allora il comando termina. Supponiamo per assurdo che il comando non termini. Consideriamo allora la seguente successione di elementi di P (l'insieme parzialmente ordinato codominio della funzione di terminazione *t*):

- $p_0 = t(s)$,
- $p_1 = t([\![C]\!]\, s)$,
- $p_2 = t([\![C]\!]([\![C]\!]\, s))$,
- $\ldots$,
- $p_i = t([\![C]\!]^i(s))$,
- $\ldots$

Si vede facilmente che, per ogni $i \geq 0$, $[\![C]\!]^i(s) \in \{INV \wedge [\![B]\!]\}$. Infatti, per ogni $i \geq 0$, $[\![C]\!]^i(s) \in \{[\![B]\!]\}$ perchè in caso contrario l'esecuzione del comando while terminerebbe, contro l'ipotesi. A questo punto, si prova per induzione aritmetica che, per ogni $i \geq 0$, $[\![C]\!]^i(s) \in \{INV\}$, dato che vale $INV(s)$ per ipotesi ed è valida la premessa della regola (While).

Allora, in base alla prima condizione che una funzione di terminazione deve soddisfare, si ha che, per ogni $i \geq 0$, $p_i \in N$ con N insieme noetheriano. Inoltre, in base alla seconda condizione che una funzione di terminazione deve soddisfare, si ha che, per ogni $i \geq 0$, $p_{i+1} < p_i$. Abbiamo quindi costruito una catena discendente infinita di elementi di N, il che è impossibile.

Prova di completezza Analogamente a quanto visto per il caso parziale, è sufficiente provare che

(*) dato un comando *C* ed una postcondizione *R* si ha $\vdash \{wp(C,R)\}\ C\ \{\Downarrow R\}$.

La prova di (*) è per induzione sulla struttura dei comandi di MINILW. Diamo solo la prova per la regola (While); per le altre metaregole la prova è analoga a quella data nel caso parziale.

While Sia $INV = wp(\texttt{while (}B\texttt{) \{ }C\texttt{ \}}, R)$. Allora $\vdash \{INV\}$ `while (`B`) { `C` }` $\{\Downarrow R\}$ si ottiene con un albero di prova analogo a quello visto per il caso parziale; dobbiamo però in questo caso dare anche una funzione di terminazione che ci consenta di istanziare la regola (While). Poichè $\{INV\}$ `while (`B`) { `C` }` $\{\Downarrow R\}$ è per definizione valida, sappiamo che per ogni stato s per cui vale $wp(C,R)(s)$, il comando termina dopo un certo numero di iterazioni, sia N_s, con $N_s \geq 0$. Definiamo allora *t* come segue:

t: *States* $\rightarrow \mathbb{N}$
$t(s) = N_s$, se $s \in \{INV \wedge [\![B]\!]\}$,
$t(s)$ = un valore arbitrario, ad esempio 0, altrimenti.

Allora *t* verifica le due condizioni richieste per essere una funzione di terminazione. Infatti la prima vale banalmente poichè $\mathbb{N}$ è noetheriano. La seconda vale poichè, per ogni s in $\{INV \wedge [\![B]\!]\}$, poichè per definizione $[\![\texttt{while (}B\texttt{) \{ }C\texttt{ \}}]\!]\, s = [\![\texttt{while (}B\texttt{) \{ }C\texttt{ \}}]\!]([\![C]\!]\, s)$, deve essere $N_s = N_{[\![C]\!]\, s} + 1$.

Esempio Mostriamo ora un esempio di prova di validità di un'asserzione di correttezza totale utilizzando il sistema di Hoare. Consideriamo l'asserzione

$$\{\mathtt{x} = x_0 \wedge \mathtt{y} = y_0 \wedge x_0 \geq 0\}\ C\ \{\Downarrow \mathtt{x} = 0 \wedge \mathtt{y} = x_0 + y_0\}$$

dove *C* indica il comando `while (x` $\neq$ `0) { x=x-1; y=y+1 }`.
Un albero di prova per quest'asserzione è il seguente, dove come invariante del ciclo abbiamo scelto

$$INV = \mathtt{x} + \mathtt{y} = x_0 + y_0 \wedge \mathtt{x} \geq 0$$

e come funzione di terminazione t: *States* $\rightarrow \mathbb{Z}$ definita da $t(s) = s(\mathtt{x})$.

$$\dfrac{\dfrac{\dfrac{}{\{INV \wedge \mathtt{x} \neq 0\}\ \mathtt{x=x-1}\ \{\Downarrow INV[\mathtt{y}+1/\mathtt{y}]\}}\,(Assign+Cons) \quad \dfrac{}{\{INV[\mathtt{y}+1/\mathtt{y}]\}\ \mathtt{y=y+1}\ \{\Downarrow INV\}}\,(Assign)}{\{INV \wedge \mathtt{x} \neq 0\}\ \mathtt{x=x-1;y=y+1}\ \{\Downarrow INV\}}\,(Seq)}{\{\mathtt{x} = x_0 \wedge \mathtt{y} = y_0 \wedge x_0 \geq 0\}\ C\ \{\Downarrow \mathtt{x} = 0 \wedge \mathtt{y} = x_0 + y_0\}}\,(While+Cons)$$

L'istanziazione della regola (While+Cons) è corretta perchè le side conditions sono banalmente verificate:

- $\mathtt{x} = x_0 \wedge \mathtt{y} = y_0 \wedge x_0 \geq 0 \Rightarrow \mathit{INV}$,
- $(\mathit{INV} \wedge \mathtt{x} = 0) \Rightarrow (\mathtt{x} = 0 \wedge \mathtt{y} = x_0 + y_0)$;
- t è una funzione di terminazione per C e INV perchè è facile vedere che:
  - $t(\{\mathit{INV} \wedge \mathtt{x} \neq 0\}) \subseteq \mathbb{N}$,
  - per ogni s in $\{\mathit{INV} \wedge \mathtt{x} \neq 0\}$, $t([\![\mathtt{x=x-1;\ y=y+1}]\!]\, s) < t(s)$.

Inoltre, l'istanziazione della regola (Assign+Cons) è corretta perchè $\mathit{INV} \wedge \mathtt{x} \neq 0$ implica $\mathit{INV}[\mathtt{y}+1/\mathtt{y}][\mathtt{x}-1/\mathtt{x}] = \mathtt{x}+\mathtt{y} = x_0 + y_0 \wedge \mathtt{x} - 1 \geq 0$.

4 Esercizi

Notazioni Utilizziamo `if (B) {C}` come abbreviazione per `if (B) {C} else {skip}`. *Pari*(x), con x identificatore, indica il predicato sugli stati vero sse il valore di x è pari. Assumiamo di avere in MINILW il tipo degli array di interi (indiciati da 1 ad una costante positiva data, con le operazioni e notazioni usuali), ed il tipo delle stringhe (su un certo alfabeto) con le operazioni: `Empty`, `Hd`, `Tl` e `Cons` con l'usuale significato. Si noti che in presenza di operazioni parziali (come `Hd` e `Tl`) ad un'espressione booleana B di MINILW corrisponde un predicato sullo stato $[\![B]\!]$ che vale vero se se l'espressione è vera, falso se l'espressione è falsa *oppure non definita*.

1. Sia $C =$ `while (x ≠ 0) { x=x-1; y=y+1 }`, x_0, y_0 due costanti intere. Si provi che:
   (a) $\models \{\mathtt{x} = x_0 \wedge \mathtt{y} = y_0\}\ C\ \{\mathtt{x} = 0 \wedge \mathtt{y} = y_0 + x_0\}$ (soluzione data precedentemente),
   (b) $\models \{\mathtt{x} = 0 \wedge \mathtt{y} = y_0\}\ C\ \{\Downarrow \mathtt{y} = y_0\}$,
   (c) $\models \{\mathtt{x} < 0\}\ C\ \{\mathit{false}\}$ (soluzione data precedentemente),
   (d) $\models \{\mathtt{x} = x_0 \wedge \mathtt{y} = y_0 \wedge x_0 \geq 0\}\ C\ \{\Downarrow \mathtt{x} = 0 \wedge \mathtt{y} = y_0 + x_0\}$ (soluzione data precedentemente).
2. Si provi, direttamente dalla definizione, utilizzando la wp e utilizzando il sistema di Hoare che è valida $\{\mathtt{x} > 0 \vee \mathtt{y} > 0\}$ `if (x>0){z=x} else {z=y}` $\{\Downarrow \mathtt{z} > 0\}$.
3. Si trovi C che renda valide entrambe le asserzioni

   $\{\mathtt{x} < 7\}\ C\ \{\Downarrow \mathtt{y} = 0\}$,
   $\{\mathtt{x} \geq 7\}\ C\ \{\mathit{false}\}$.
4. Si trovi P che renda valida $\{\mathtt{x} = x_0 \wedge \mathtt{y} = y_0 \wedge P\}$ `while (y ≠ x) { x=x+1; y=y-1 }` $\{\Downarrow \mathtt{x} = \mathtt{y}\}$.
5. Si verifichi la validità della seguente asserzione, organizzando la prova in modo modulare.

   $\{\mathit{true}\}$

   ```
   if (A<B) {T=A;A=B;B=T}
   if (B<C) {T=B;B=C;C=T;if (A<B){T=A;A=B;B=T}}
   ```

   $\{\mathtt{A} \geq \mathtt{B} \geq \mathtt{C}\}$
6. Si trovi un comando C che renda valide entrambe le asserzioni

   $\{\mathit{Pari}(\mathtt{x}) \wedge \mathtt{x} \geq 0\}\ C\ \{\Downarrow \mathtt{y} = 0\}$,
   $\{\neg \mathit{Pari}(\mathtt{x}) \vee \mathtt{x} < 0\}\ C\ \{\mathit{false}\}$.
7. Si trovi la condizione P necessaria e sufficiente perchè valga

   $\{P \wedge \mathtt{x} = x_0 \wedge \mathtt{y} = y_0\}$ `s=0;while (x!=y){x=x-1;s=s+1}` $\{\Downarrow \mathtt{s} = x_0 - y_0\}$.
8. Scrivere un algoritmo iterativo che calcoli l'n-simo termine della successione di Fibonacci (n costante positiva) e provarne formalmente la correttezza (la successione di Fibonacci è definita induttivamente da $f_0 = 1$, $f_1 = 1$, $f_{i+2} = f_i + f_{i+1}$ per $i \geq 0$).
9. Si completi il seguente algoritmo iterativo scegliendo B in modo che esso risulti corretto rispetto alla specifica indicata (`A` è un array di dimensione n, con n costante positiva). La postcondizione può essere espressa informalmente dicendo che tutti gli elementi dispari devono precedere tutti quelli pari.

   $\{\mathtt{first} = 1 \wedge \mathtt{last} = n\}$

```
while (B) {
  z=A[last];
  if (Pari(z)){last=last-1} else {A[last]=A[first];A[first]=z;first=first+1}
}
```

$\{\Downarrow \forall i, j \in [1..n].\mathit{Pari}(\texttt{A[i]}) \wedge \neg \mathit{Pari}(\texttt{A[j]}) \Rightarrow j < i\}$

10. Si verifichi la validità della seguente asserzione:

    $\{\texttt{x} = \texttt{y}\}$ `if (x>=0)`$\{$`if (y>=0)`$\{$`x=x+2`$\}\}$ `else`$\{$`if (y<0)`$\{$`y=y-2`$\}\}$ $\{\Downarrow \texttt{x} = \texttt{y} + 2\}$.

    Si valuti inoltre, detto *C* il comando sopra, $wp(C, \texttt{x} = \texttt{y} + 2)$.

11. Si consideri il seguente algoritmo iterativo che ordina in modo crescente gli elementi di un array `A` di dimensione n (n costante positiva fissata).

    $\{\texttt{i} = 1\}$

    ```
    while (i<n) {
      if (A[i]>A[i+1]) {swap(A,i);i=1}
      else {i=i+1}
    }
    ```

    $\{\Downarrow \forall i \in [1..n-1].\texttt{A}[i] \leq \texttt{A}[i+1]\}$

    Si è assunto che `swap(A,i)` sia un comando che scambia tra loro l'elemento di posto `i` ed il successivo. Si provi la correttezza totale dell'algoritmo rispetto alla specifica indicata, formalizzando le asserzioni su `swap(A,i)` necessarie nel corso della prova.

12. Il seguente algoritmo fornisce, data una stringa $\bar{w}$, la stringa rovesciata $\bar{w}^R$.

    $\{\texttt{rev} = \texttt{Empty} \wedge \texttt{w} = \bar{w}\}$

    `while (w!=Empty) {rev=Cons(Hd(w),rev);w=Tl(w)}`

    $\{\Downarrow \texttt{rev} = \bar{w}^R\}$

    Si provi formalmente la correttezza totale dell'algoritmo rispetto alla precondizione e postcondizione indicate.

13. Il seguente algoritmo controlla se una stringa $\bar{w}$ è *palindroma* (cioè $\bar{w} = \bar{w}^R$).

    $\{\texttt{pal} \wedge \texttt{w} = \bar{w}\}$

    `while (w!=Empty && pal) {pal=Hd(w)==Last(w);w=Tl(w);w=CancelLast(w)}`

    $\{\Downarrow \texttt{pal} = (\bar{w} = \bar{w}^R)\}$

    Si è assunto di avere a disposizione due funzioni ausiliarie (parziali) `Last` e `CancelLast` che data una stringa restituiscono rispettivamente l'ultimo elemento e la stringa privata dell'ultimo elemento.

    L'algoritmo dato *non* è corretto rispetto alla specifica indicata. Si motivi tale fatto dando un controesempio. Si modifichi l'algoritmo in modo da renderlo corretto e si provi la correttezza della versione modificata.

14. Si consideri il seguente algoritmo iterativo (x_0, y_0 costanti intere).

    $\{\texttt{diff} = 0 \wedge \texttt{x} = x_0 \wedge \texttt{y} = y_0\}$

    ```
    while (x!=y) {
      if (x>y){x=x-1} else {x=x+1};
      diff=diff+1
    }
    ```

    $\{\Downarrow \texttt{diff} = \mid x_0 - y_0 \mid\}$

    Se ne provi formalmente la correttezza totale rispetto alla specifica indicata utilizzando il sistema di Hoare.

15. Si provi formalmente che vale (con `x`, `y` variabili intere e k costante intera positiva)

    $\{\mid \texttt{x} - \texttt{y} \mid = k\}$

    ```
    if (x>y){x=x-1};
    if (y>x){y=y-1}
    ```

$\{\Downarrow | \texttt{x} - \texttt{y} | = k - 1\}$

16. Si consideri il seguente comando *C*, dove `x`, `y` sono stringhe.

    $C =$ `while (x!=Empty) { Push(Hd(x),y);Pop(x) }`

    Si provi che, assumendo che i comandi `Push(Hd(x),y)` e `Pop(x)` siano corretti rispetto ad opportune asserzioni (specificare quali), vale la seguente asserzione:

    $\{\texttt{x} = \bar{x} \wedge \texttt{y} = \texttt{Empty}\}\ C\ \{\Downarrow \texttt{y} = \bar{x}^R\}$

    dove $\bar{x}$ è una costante di tipo stringa.

17. Si trovi $wp($`if (x>=0){if (y<0){z=x}} else {if (y>=0){z = x}}`$, \texttt{z} = \texttt{y})$.