Correttezza di programmi - modelli della concorrenza

Esercizio 1
Dimostrare la correttezza parziale e totale della seguente tripla:

```
{ k > 0 }
x := k;
y := 0;
while x > 0 do
    x := x - 1;
    y := y + n;
endwhile
{ y = k * n }
```

Soluzione
Cerchiamo un invariante per il ciclo:

x	y
k	0
k - 1	n
k -2	2*n
k - 3	3*n

Ipotesi: $i \equiv y = (k-x)n \wedge x \geq 0$.

Per dimostrare che i è un invariante, dobbiamo dimostrare la tripla

$$\{i \wedge x > 0\} \quad x := x-1; y := y+n \quad \{i\}.$$

Utilizziamo la regola dell'assegnamento con la postcondizione che vogliamo trovare. I due assegnamenti sono indipendenti, per cui possiamo eseguire le sostituzioni simultaneamente:

$$\vdash \{y+n = (k-(x-1))n \wedge x-1 \geq 0\} \quad x := x-1; y := y+n \quad \{i\};$$

con qualche passaggio algebrico possiamo riscrivere questa tripla come:

$$\vdash \{y = (k-x)n \wedge x > 0\} \quad x := x-1; y := y+n \quad \{i\}.$$

Poiché $(x > 0 \wedge x \geq 0) \longrightarrow (x > 0)$, possiamo applicare la regola di implicazione e ottenere la tripla:

$$\vdash \{y = (k-x)n \wedge x \geq 0 \wedge x > 0\} \quad x := x-1; y := y+n \quad \{i\}.$$

Questo conclude la dimostrazione che i è un invariante. Per la regola di iterazione della correttezza parziale, chiamato W l'intero ciclo while, valgono le triple:

$$\vdash \{i\} \quad W \quad \{y = (k-x)n \wedge x \geq 0 \wedge x \leq 0\}$$

$$\vdash \{i\} \quad W \quad \{y = kn\}. \tag{1}$$

Osserviamo che la tripla 1 ha la postcondizione che desideriamo ottenere. Per concludere la dimostrazione, dobbiamo ora considerare gli assegnamenti iniziali. Visto che speriamo di poter usare la regola di composizione per unire la tripla che considera gli assegnamenti alla tripla 1, applichiamo la regola dell'assegnamento considerando la precondizione di 1 come postcondizione; dato che i due assegnamenti considerano variabili distinte, possiamo eseguirli simultaneamente:

$$\vdash \{0 = (k-k)n \wedge k \geq 0\} \quad x := k; y := 0 \quad \{y = (k-x)n \wedge x \geq 0\}.$$

Poiché $(k > 0) \longrightarrow k \geq 0$, vale la tripla:

$$\vdash \{k > 0\} \quad x := k; y := 0 \quad \{i\}. \tag{2}$$

che ha la precondizione della tripla che vogliamo dimostrare. Per finire di dimostrare la correttezza parziale basta applicare la regola di composizione considerando le triple 2 e 1.

Per dimostrare la correttezza totale del programma, dobbiamo dimostrare che termina. Il programma termina per ogni esecuzione, se dopo un numero finito di iterazioni $x \leq 0$. Per dimostrarlo, dobbiamo trovare un'espressione E tale che:

1. esista un invariante inv per il ciclo, e $inv \rightarrow E \geq 0$;
2. $\{inv \wedge x > 0 \wedge E = E_0\} \quad x := x - 1; y := y + n \quad \{E < E_0\}$.

Possiamo considerare $E \equiv x$ e $inv = i$. La condizione 1 è immediatamente soddisfa, perché $x \geq 0$ è parte dell'invariante. Per verificare la condizione 2 applichiamo la regola di assegnamento:

$$\vdash \{x - 1 < E_0\} \quad x := x - 1; y := y + n \quad \{x < E_0\}.$$

Poiché $x - 1 < E_0$ è equivalente a scrivere $x \leq E_0$ e $(x = E_0) \rightarrow (x \leq E_0)$, possiamo scrivere:

$$\vdash \{x = E_0\} \quad x := x - 1; y := y + n \quad \{x < E_0\}.$$

Appicando nuovamente la regola di implicazione sulla precondizione, otteniamo la tripla che dimostra la correttezza totale.

Esercizio 2
Scrivere un programma che calcola il foriale di un numero intero positivo e dimostrarne la correttezza parziale e totale.

Soluzione
Proposta di programma che calcola il foriale:

```
x := 0; f := 1;
while x < n do
    x := x+1;
    f := f * x
endwhile
```

Per dimostrare che questo programma (P) è corretto, dobbiamo dimostrare la tripla:

$$\{n \geq 0\} \quad P \quad \{f = n!\}. \tag{3}$$

Come primo passo, cerchiamo un invariante per il ciclo:

x	f
0	1
1	1
2	1*2
3	1*2*3
4	1*2*3*4

Un ipotesi ragionevole per l'invariante sembra essere $i \equiv f = x! \wedge x \leq n$. Per dimostrare che lo è davvero dobbiamo dimostrare la tripla:

$$\{i \wedge x < n\} \quad x := x + 1; f := f * x \quad \{i\}.$$

Siccome entrambi gli assegnamenti coinvolgono la variabile x, ne eseguiamo uno alla volta. Visto che vogliamo fissare la postcondizione, partiamo dal secondo, e otteniamo la tripla:

$$\vdash \{f * x = x! \wedge x \leq n\} \quad f := f * x\{i\}. \tag{4}$$

Eseguiamo anche il primo assegnamento, considerando come postcondizione la precondizione ottenuta dall'assegnamento precedente:

$$\vdash \{f * (x+1) = (x+1)! \wedge (x+1) \leq n\} \quad x := x + 1 \quad \{f * x = x! \wedge x \leq n\}.$$

Questa tripla si può riscrivere come

$$\vdash \{f = x! \wedge x < n\} \quad x := x + 1 \quad \{f * x = x! \wedge x \leq n\}. \tag{5}$$

Applichiamo la regola di composizione alle triple 4 e 5:

$$\vdash \{f = x! \wedge x < n\} \quad x := x + 1; f = f * x \quad \{i\}$$

Poiché $(x < n \wedge x \leq n) \rightarrow (x < n)$, per la regola di implicazione vale la tripla

$$\vdash \{f = x! \wedge x \leq n \wedge x < n\} \quad x := x + 1; f = f * x \quad \{i\},$$

quindi i è davvero un invariante per il ciclo. Per la regola di iterazione per la correttezza parziale, detto W l'intero ciclo while, vale la tripla

$$\vdash \{i\} \quad W \quad \{f = x! \wedge x \leq n \wedge x \geq n\}.$$

$$\vdash \{i\} \quad W \quad \{f = n!\}. \tag{6}$$

Consideriamo ora gli assegnamenti iniziali. Siccome sono indipendenti possiamo considerarli simultaneamente. Applichiamo la regola di assegnamento considerando come postcondizione la precondizione di 6:

$$\{1 = 0! \wedge n \geq 0\} \quad x := 0; f := 1 \quad \{f = x! \wedge x \leq n\} \tag{7}$$

Per concludere, applicando la regola di composizione alle triple 7 e 6 otteniamo la tripla che volevamo dimostrare.

Per dimostrare la correttezza totale, dobbiamo trovare un'espressione E tale che:

1. esista un invariante inv per il ciclo, e $inv \rightarrow E \geq 0$;
2. $\{inv \wedge x < n \wedge E = E_0\} \quad x := x + 1; f := f * x \quad \{E < E_0\}$.

Come espressione consideriamo $E \equiv n - x$. Il punto 1 è verificato prendendo come invariante quello usato per la correttezza parziale. Per verificare il punto 2 utilizziamo la regola di assegnamento. Visto che nella postcondizione f non compare, il secondo assegnamento non ha alcun effetto nel determinare la precondizione, quindi, anche se entrambi gli assegnamenti utilizzano la variabile x, non è necessario fare due passaggi.

$$\vdash \{n - (x + 1) < E_0\} \quad x := x + 1; f := f * x \quad \{n - x < E_0\}.$$

Con ragionamenti affatto analoghi a quelli svolti nell'esercizio precedente, possiamo ricavare la tripla che dimostra la correttezza totale.

Esercizio 3
Dimostrare la correttezza parziale e totale della seguente tripla:

```
{ p >= 0 }
n := 1; s := 0; k := 0;
while k < p do
    s := s + n;
    n := n + 2;
    k := k + 1;
endwhile
{ s = p * p }
```

Soluzione
Come primo passo cerchiamo un invariante per il ciclo while.

s	n	k
0	1	0
1	3	1
4	5	2
9	7	3
16	9	4

Guardando la tabella sembrerebbe che $i \equiv s = k^2 \wedge k \leq p$ sia un invariante per il ciclo. Per dimostrarlo dobbiamo far vedere che vale la tripla

$$\{s = k^2 \wedge k \leq p \wedge k < p\} \quad s := s + n; n := n + 2; k := k + 1 \quad \{s = k^2 \wedge k \leq p\}. \tag{8}$$

Gli ultimi due assegnamenti considerano variabili distinte, quindi possiamo eseguire le sostituzioni simultaneamente; aspettiamo a considerare il primo, visto che n appare anche nel

secondo assegnamento.

$$\vdash \{s = (k+1)^2 \wedge k+1 \leq p\} \quad n := n+2; k := k+1 \quad \{s = k^2 \wedge k \leq p\}$$
$$\vdash \{s = k^2 + 2k + 1 \wedge k < p\} \quad n := n+2; k := k+1 \quad \{s = k^2 \wedge k \leq p\}.$$

Applichiamo il primo assegnamento considerando come postcondizione $s = k^2 + 2k + 1 \wedge k < p$:

$$\vdash \{s + n = k^2 + 2k + 1 \wedge k < p\} \quad s := s + n \quad \{s = k^2 + 2k + 1 \wedge k < p\}.$$

Applicando la regola di composizione derivo la tripla:

$$\vdash \{s + n = k^2 + 2k + 1 \wedge k < p\} \quad s := s + n; n := n+2; k := k+1 \quad \{k = s^2 \wedge k \leq p\}.$$

A questo punto abbiamo applicato tutte le regole, ma non siamo riusciti a dimostrare la tripla a cui eravamo interessati. Osserviamo che saremmo riusciti nella dimostrazione se avessimo potuto affermare che $n = 2k + 1$. Guardando la tabella all'inizio dell'esercizio, notiamo che in effetti, in tutti i casi che abbiamo considerato, vale $n = 2k + 1$. Potremmo sospettare che questa sia una parte dell'invariante. In questo caso il nostro nuovo invariante sarebbe $i' \equiv s = k^2 \wedge k \leq p \wedge n = 2k + 1$. Se i' è davvero un invariante, vale la tripla

$$\vdash \{i' \wedge k < p\} \quad s := s + n; n := n+2; k := k+1 \{i' \wedge k < p\}.$$

Proviamo a dimostrarla con dei passaggi analoghi a quelli svolti in precedenza.

$$\vdash \{s = k^2 + 2k + 1 \wedge k < p \wedge n + 2 = 2(k+1) + 1\} \quad n := n+2; k := k+1 \quad \{s = k^2 \wedge k \leq p \wedge n = 2k+1\};$$
$$\vdash \{s = k^2 + 2k + 1 \wedge k < p \wedge n = 2k+1\} \quad n := n+2; k := k+1 \quad \{s = k^2 \wedge k \leq p \wedge n = 2k+1\};$$
$$\vdash \{s + n = k^2 + 2k + 1 \wedge k < p \wedge n = 2k+1\} \quad s := s + n \quad \{s = k^2 + 2k + 1 \wedge k < p \wedge n = 2k+1\};$$
$$\vdash \{s = k^2 \wedge k < p \wedge n = 2k+1\} \quad s := s + n \quad \{s = k^2 + 2k + 1 \wedge k < p \wedge n = 2k+1\}.$$

Utilizzando la regola di composizione e la regola di implicazione, otteniamo la tripla che dimostra che i' è un invariante. Per la regola di iterazione per la correttezza parziale, chiamato W l'intero ciclo while, vale la tripla:

$$\vdash \{s = k^2 \wedge k \leq p \wedge n = 2k+1\} W \{s = k^2 \wedge k \leq p \wedge n = 2k+1 \wedge k \geq p\}, \tag{9}$$

che equivale alla tripla

$$\vdash \{s = k^2 \wedge k \leq p \wedge n = 2k+1\} W \{s = p^2 \wedge n = 2p+1\}.$$

Poiché $(s = p^2 \wedge n = 2p+1) \rightarrow (s = p^2)$, per la regola d'implicazione vale:

$$\vdash \{s = k^2 \wedge k \leq p \wedge n = 2k+1\} W \{s = p^2 \wedge n = 2p+1\}. \tag{10}$$

Infine, consideriamo gli assegnamenti iniziali. Visto che considerano tutti variabili distinte, possiamo effettuare simultaneamente le sostituzioni da fare per applicare la regola di assegnamento:

$$\vdash \{0 = 0 \wedge 0 \leq p \wedge 1 = 1\} \quad n := 1; s := 0; k := 0 \quad \{s = k^2 \wedge k \leq p \wedge n = 2k+1\},$$
$$\vdash \{p \geq 0\} \quad n := 1; s := 0; k := 0 \quad \{s = k^2 \wedge k \leq p \wedge n = 2k+1\}. \tag{11}$$

Applicando la regola di composizione a 11 e 10, otteniamo la tripla che dimostra la correttezza parziale del programma.

Per dimostrare la correttezza totale, dobbiamo trovare un'espressione E tale che:

1. esista un invariante inv per il ciclo, e $inv \rightarrow E \geq 0$;

2. $\{inv \wedge k < p \wedge E = E_0\} \quad s := s + n; n := n + 2; k := k + 1 \quad \{E < E_0\}$.

Consideriamo $E \equiv p - k$. Il punto 1 è verificato se utilizziamo i' come invariante. Per il punto 2 applichiamo la regola dell'assegnamento. Dal momento che l'unica variabile nella postcondizione è k e che l'assegnamento che riguarda k non inflisce sulle altre variabilii, possiamo considerare assieme tutti e tre gli assegnamenti, e l'unico che avrà un effetto sulla precondizione è $k := k + 1$.

$$\vdash \{p - (k + 1) < E_0\} \quad s := s + n; n := n + 2; k := k + 1 \quad \{p - k < E_0\}.$$

Osserviamo che $p - (k + 1) < E_0 \equiv p - k - 1 E_< 0 \equiv p - k \leq E_0$. Poiché $(i' \wedge k < p \wedge p - k = E_0) \rightarrow (p - k \leq E_0)$, possiamo ricavare la tripla del punto 2 applicando la regola di implicazione.

Esercizio 4
Calcolare la precondizione più debole per il programma qui sotto e la postcondizione $\{x\%2 = 0\}$. L'operatore binario % rappresenta il resto di una divisione intera (ad esempio, $5\%3 = 2$). Nel programma, l'operazione di divisione va intesa come divisione intera.

```
x := k;
if x % 2 = 0 then
    x = x / 2
else
    x = x - 1
endif
```

Soluzione
Vogliamo determinare tutti i valori iniziali di x tali che, dopo aver eseguito il programma, vale che $\{x\%2 = 0\}$. Inizialmente ad x viene assegnato il valore k.

Se k è pari, quindi $x\%2 = 0$, il programma esegue l'istruzione $x := x/2 = k/2$. La postcondizione chiede che x sia ancora divisibile per due, quindi che anche $k/2$ sia pari. Questo avviene se $k\%4 = 0$, e quindi inizialmente $x\%4 = 0$.

Se k è dispari, il programma esegue l'istruzione $x := x - 1 = k - 1$. Se k è dispari, $k - 1$ deve essere pari, quindi per ogni valore iniziale di x dispari la postcondizione è soddisfa.

La proposta di precondizione più debole che segue da questo ragionamento è $\{x\%4 = 0 \vee x\%2 = 1\}$.

Formalmente, la regola per trovare la precondizione più debole $wp(C, q)$, dove q è la postcondizione e C è della forma 'if B then F else D fi', è:

$$wp(C, q) = (B \wedge wp(F, q)) \vee (\neg B \wedge wp(D, q)).$$

Ricordiamo che applicando la regola di assegnamento ricaviamo sempre la precondizione più debole per la tripla. Sapendo questo e sostituendo le parti del nostro programma e la postcondizione desiserata nella formula, possiamo ottenere la precondizione più debole.