

# Modelli della concorrenza - Dispense

Alessandro Vasquez

November 21, 2016

# Indice

<b>I</b>	<b>Correttezza dei programmi sequenziali</b>	<b>4</b>
<b>1</b>	<b>Logica di Hoare</b>	<b>5</b>
1.1	Premessa . . . . .	5
1.2	Correttezza parziale . . . . .	6
1.2.1	Assiomi e regole di inferenza . . . . .	6
1.2.2	Regole di derivazione ottenibili induttivamente . . . . .	7
1.3	Correttezza totale . . . . .	8
1.3.1	Definizioni e notazioni . . . . .	8
1.3.2	Determinare la preconditione . . . . .	9
<b>II</b>	<b>Modelli della concorrenza</b>	<b>11</b>
<b>2</b>	<b>Introduzione</b>	<b>12</b>
2.1	Sintassi . . . . .	12
2.2	Semantica . . . . .	13
2.2.1	Semantica operativa strutturale . . . . .	13
2.2.2	Equivalenza rispetto alle tracce . . . . .	14
2.2.3	Bisimulazione . . . . .	15

# Introduzione

Sia dato un programma  $P$ . Ci si vuole convincere del suo buon funzionamento. Vi sono tre diverse strade che si posso intraprendere:

- Sperimentazione
- Metodo assiomatico
- Model-checking

Questo corso tratterà le ultime due metodologie. Più precisamente, si dividerà in tre parti:

1. Correttezza dei programmi sequenziali
2. Modelli della concorrenza
3. Model-checking

Si riporta ora un esempio che introduce all'applicazione del metodo assiomatico.

**Esempio 0.0.1.** Sia data la seguente funzione, ove  $v$  è un vettore di interi e  $n$  la sua dimensione:

---

```
1      int f (int n, int v[]){
2          int x = v[0];
3          int h = 1;
4          while (h < n){
5              if (x < v[h]){
6                  x = v[h];
7                  h = h+1;
8              }
9          }
10         return x;
11     }
```

---

A una prima analisi la funzione sembra ordinare il vettore in modo decrescente. Più formalmente, si può assumere che i requisiti della funzione siano i seguenti:

$$\begin{aligned} \forall i \in \{0, \dots, n-1\} & : v[i] \leq x, \\ \exists i \in \{0, \dots, n-1\} & : x = v[i]. \end{aligned} \tag{1}$$

Si vuole dimostrare tali requisiti per induzione sulla dimensione del vettore  $v$ . Sia data quindi la seguente ipotesi induttiva:

$$Hp\ ind : \begin{cases} \forall i \in \{0, \dots, h-1\} : v[i] \leq x, \\ \exists i \in \{0, \dots, h-1\} : x = v[i]. \end{cases}$$

È necessario effettuare il passo induttivo e dimostrare tali proprietà per i casi base.

Formalizzando l'esempio 0.0.1 si ottiene la tripla:

$$\{n > 0\} \ P \ \{\alpha\}.$$

Dove  $n > 1$  è chiamata preconditione,  $P$  è il programma o la funzione e  $\alpha$  è la postcondizione che consiste dei requisiti esplicitati in (1).

Tale formalismo sarà ripreso nel capitolo 1.

## Part I

# Correttezza dei programmi sequenziali

# Chapter 1

## Logica di Hoare

### 1.1 Premessa

D'ora in avanti gli esempi saranno applicati a programmi scritti con un linguaggio giocattolo la cui grammatica è la seguente:

```
E ::= n | x | (-E) | (E+E) | (E*E) | ...
B ::= true | false | ( NOT B ) | ( B AND B ) | ( B OR B ) | ( E < E ) |
    ( E = E )
C ::= X := E | skip | C;C | if B then C else C fi | while B do C
    od
```

E rappresenta le variabili semplici quali costanti intere o variabili di programma.

B rappresenta le espressioni booleane.

C rappresenta i comandi ammessi dal linguaggio.

**Esempio 1.1.1.** Sia dato il programma  $P$ :

---

```
1   y:=1; z:=0;
2   while z!=x do
3       z:=z+1;
4       y:=y*z
5   od
```

---

A una prima occhiata  $P$  sembra calcolare  $x!$  in  $y$ . Supponendo vera tale ipotesi, si scriverà

$$\{x \geq 0\} P \{y = x!\}.$$

Tale espressione è nota come Tripla di Hoare. In questo caso, se si esegue  $P$  nello stato iniziale determinato dalla preconditione  $\{x \geq 0\}$  e si ottiene la postcondizione  $\{y = x!\}$ , si dirà che la tripla è valida. Il sistema delle triple

di Hoare equivale al sistema di contratti tra programmatori (DBC, Design By Contract).

**Definizione 1.1.1.** La "fotografia" dei valori delle variabili di un programma in un certo momento viene detta **stato di un programma**. Tale "fotografia" può essere interpretata come un vettore contenente i valori di tutte le variabili del programma.

## 1.2 Correttezza parziale

Riprendendo le strutture delle triple, si vuole ora costruire un sistema formale basato sulla deduzione naturale e sui seguenti assiomi.

### 1.2.1 Assiomi e regole di inferenza

**Istruzione vuota** Sia data una formula proposizionale  $\alpha$ . Allora:

$$\frac{}{\{\alpha\} \text{ skip } \{\alpha\}}$$

**Assegnamento** Sia data una formula proposizionale  $\alpha$ , una variabile  $x$  e un'espressione  $E$ . Allora:

$$\frac{}{\{\alpha \left[ \frac{E}{x} \right]\} x := E \{\alpha\}}$$

**Conseguenza** Siano date le formule proposizionali  $p$ ,  $q$ ,  $p_1$ ,  $q_1$  e il programma  $P$ . Allora:

$$\frac{p_1 \rightarrow p \quad \{p\} P \{q\}}{\{p_1\} P \{q\}} \quad \frac{\{p\} P \{q\} \quad q \rightarrow q_1}{\{p\} P \{q_1\}}$$

Dalle ultime due deduzioni si ottiene induttivamente la seguente:

$$\frac{p_1 \rightarrow p \quad \{p\} P \{q\} \quad q \rightarrow q_1}{\{p_1\} P \{q_1\}}$$

**Sequenza** Siano date le formule proposizionali  $p$ ,  $q$ ,  $r$  e le istruzioni  $C_1$ ,  $C_2$ . Allora:

$$\frac{\{p\} C_1 \{q\} \quad \{q\} C_2 \{r\}}{\{p\} C_1 ; C_2 \{r\}}$$

La regola di sequenza è utile nei cicli il cui corpo presenta istruzioni diverse che coinvolgono le stesse variabili.

**Iterazione** Siano dati un ciclo iterativo, la relativa condizione  $B$ , una sua invariante  $i$ , e il corpo del ciclo  $C$ . Allora:

$$\frac{\{i \wedge B\} C \{i\}}{\{i\} \text{ while } B \text{ do } C \text{ od } \{i \wedge \neg B\}}$$

### 1.2.2 Regole di derivazione ottenibili induttivamente

Siano date le formule proposizionali  $p$ ,  $q$ ,  $r$  e il programma  $P$ . Allora:

$$\frac{\{p\} P \{q\} \quad \{r\} P \{q\}}{\{p \vee r\} P \{q\}}$$

Nel caso in cui  $r = \neg p$ , la preconditione nella conclusione della deduzione diventa una tautologia:

$$\frac{\{p\} P \{q\} \quad \{\neg p\} P \{q\}}{\{p \vee \neg p\} P \{q\}}$$

**Controllo di flusso** Siano date le formule proposizionali  $p$ ,  $q$ . Siano dati inoltre una condizione  $B$ , un blocco  $P$  (eseguito solo se  $B$  è verificata) e un blocco  $Q$  (eseguito altrimenti). Sia infine  $R$  il blocco che include l'istruzione di controllo su  $B$  e i blocchi  $P$  e  $Q$ . Vale la seguente deduzione:

$$\frac{\{p \wedge B\} P \{q\} \quad \{p \wedge \neg B\} Q \{q\}}{\{p\} R \{q\}}$$



## 1.3 Correttezza totale

Si vuole verificare la terminazione di un programma dato. Poiché la non-terminazione può avvenire solo in presenza di cicli, ci si concentrerà sullo studio di programmi iterativi.

Si supponga di avere un programma  $P$  che presenta un ciclo while  $W$  il cui corpo è denotato con  $C$ .

Si supponga di aver determinato un invariante  $i$  del ciclo. Per dimostrare che  $P$  termina è necessario determinare una espressione  $E$  e un invariante  $p$ , indipendente da  $E$  e da  $i$ , tali che:

1.  $p \rightarrow E \geq 0$ ,
2.  $\vdash_p \{p \wedge B \wedge E = K\} C \{E < K\}$ .

### 1.3.1 Definizioni e notazioni

**Definizione 1.3.1.** Sia  $V$  l'insieme delle variabili di un programma  $P$ . Uno stato  $\sigma$  di  $P$  è una funzione

$$\sigma : V \rightarrow \mathbb{Z}.$$

**Definizione 1.3.2.** L'insieme di tutti gli stati di un programma è l'insieme:

$$\Sigma\{\sigma : V \rightarrow \mathbb{Z}\}.$$

**Definizione 1.3.3.** L'insieme di tutte le formule proposizionali costruite a partire da  $V$  si chiama  $\Pi$ .

Dati  $\sigma \in \Sigma$  e  $p \in \Pi$ , si dirà che la formula  $p$  è valida nello stato  $\sigma$  usando la notazione

$$\sigma \models p.$$

**Definizione 1.3.4.** L'insieme di tutte le possibili asserzioni vere per lo stato  $\sigma$  è l'insieme

$$t(\sigma) = \{p \in \Pi : \sigma \models p\}.$$

**Definizione 1.3.5.** L'insieme degli stati che rendono vera la formula  $p$  è l'insieme

$$m(p) = \{\sigma \in \Sigma : \sigma \models p\}.$$

Tale insieme è definito come l'estensione della formula  $p$ .

**Lemma 1.3.1.** Siano  $S \subseteq \Sigma$  e  $F \subseteq \Pi$ . Allora

$$t(S) = \{p \in \Pi : \forall s \in S : s \models p\} = \bigcap_{s \in S} t(s),$$

$$m(F) = \{\sigma \in \Sigma : \forall p \in F : \sigma \models p\} = \bigcap_{p \in F} m(p).$$

**Lemma 1.3.2.** Siano  $A, B \in \Sigma$  tali che  $A \subseteq B$ . Allora

$$t(A) \supseteq t(B).$$

**Lemma 1.3.3.** Siano  $A, B \in \Pi$  tali che  $A \subseteq B$ . Allora

$$m(A) \supseteq m(B).$$

**Lemma 1.3.4.** Siano  $p, q$  formule proposizionali. Grazie alla definizione ricorsiva di formula proposizionale, valgono le seguenti proprietà:

- $m(\neg p) = \Sigma \setminus m(p)$
- $m(p \vee q) = m(p) \cup m(q)$
- $m(p \wedge q) = m(p) \cap m(q)$
- $m(p \rightarrow q) = (\Sigma \setminus m(p)) \cup m(q)$

in cui l'ultima uguaglianza è data dal fatto che  $p \rightarrow q \equiv \neg p \vee q$ <sup>1</sup>.

### 1.3.2 Determinare la preconditione

Si supponga di avere una tripla di Hoare priva della preconditione:  $P\{q\}$ . Si rende necessario trovare un metodo per determinarne la preconditione. Tramite l'assioma dell'assegnamento è possibile ottenere la preconditione più debole possibile<sup>2</sup>.

**Esempio 1.3.1.** Si supponga di avere la seguente tupla:

$$C_1; C_2\{q\}.$$

Tramite l'assioma dell'assegnamento applicato a  $C_2\{q\}$  posso ottenere la tripla

$$\vdash \{wp(C_2, q)\} C_2 \{q\}^3.$$

Applicando poi l'assioma di assegnamento e l'assioma di sequenza a  $C_1$ , ottengo la tripla

$$\{wp(C_1, wp(C_2, q))\} C_1; C_2 \{q\},$$

nella quale compare la più debole preconditione della tupla iniziale.

**Esempio 1.3.2.** Sia dato un generico programma iterativo  $P\{q\}$ :

---

```

1  while B do
2  C
3  od

```

---

<sup>1</sup>Si veda la definizione di  $\beta$ -formula.

<sup>2</sup>La preconditione che pone meno vincoli allo stato iniziale del programma

<sup>3</sup>wp = weakest precondition

Si vuole determinare la preconditione più debole. Vi possono essere due possibili condizioni:  $B$  è vera o no.

1. Se  $B$  è falsa prima del ciclo allora il while viene saltato e  $P$  equivale a un'istruzione vuota. La preconditione in questo caso deve coincidere con la postcondizione:

$$\{\neg B \wedge q\}.$$

2. Se  $B$  è vera prima del ciclo allora è possibile pensare di provare ad eseguire una volta il corpo del ciclo subito prima del ciclo stesso:

---

```

1      C
2      while B do
3          C
4      od

```

---

Posso quindi estrarre la preconditione più debole di  $C$ :

$$wp(C, q) = (B \wedge wp(C; P, q)).$$

Mettendo insieme le due preconditioni ottengo:

$$wp(P, q) = \{(\neg B \wedge q) \vee (B \wedge wp(C; P, q))\}.$$

Il risultato dell'esempio 1.3.2 potrebbe sembrare insoddisfacente: si è giunti a una formula più complessa di quella iniziale.

In effetti il calcolo di  $wp$  per un programma iterativo non porta meccanicamente a un risultato: tale calcolo non definisce un algoritmo, la soluzione può essere determinata grazie a metodologie euristiche informalmente giustificate. Tuttavia, nei programmi non iterativi il calcolo di  $wp$  può essere usato meccanicamente per determinare una soluzione.

## **Part II**

# **Modelli della concorrenza**

## Chapter 2

# Introduzione

Si vuole definire un formalismo per l'analisi di correttezza di programmi concorrenti.

### 2.1 Sintassi

Saranno ora definiti gli elementi di base del formalismo che si vuole definire. Siano  $\mathcal{K}^1$  l'insieme di tutti i possibili processi e  $Act$  l'insieme di tutte le possibili azioni:

$$Act = \mathcal{A} \cup \overline{\mathcal{A}} \cup \{\tau\},$$

dove  $\overline{\mathcal{A}} = \{\bar{a} \mid a \in \mathcal{A}\}$  è l'insieme delle coazioni di  $a$ .

**Definizione 2.1.1.** Si definisce espressione CCS una qualunque delle seguenti:

- Nil (il processo vuoto, denotato con  $\emptyset$ )
- $K$  (qualche nome di un processo qualunque:  $K \in \mathcal{K}$ )
- $\alpha.P$ , dove  $\alpha \in Act$
- $\Sigma_{i \in I} P_i$
- $P_1 \mid P_2$  (composizione parallela)
- $P \setminus L$ , ove  $L \subseteq \mathcal{A}$ , ad esempio  $P \setminus \{a, c\}$  significa che  $P$  non può interagire con l'ambiente usando una qualunque tra le azioni  $a, c, \bar{a}, \bar{c}$
- $P[f]$ , dove  $f$  è una funzione  $f : Act \rightarrow Act$  tale che:

$$f(\tau) = \tau,$$

$$f(\bar{a}) = \overline{f(a)}$$

---

<sup>1</sup>Talvolta ci si riferirà a tale insieme come:  $P_{CCS}$ .

La definizione 2.1.1 è ottenuta per induzione strutturale sui processi  $P_i$ .

**Lemma 2.1.1.** La coazione di una coazione  $a$  è  $a$  stessa:  $\overline{\overline{a}} = a$ .

**Definizione 2.1.2.** Una specifica o programma CCS sarà un insieme di equazioni che specificano per ogni processo qual è l'espressione CCS che specifica il suo comportamento. L'insieme di equazioni sarà del tipo:

$$K = P, \quad K \in \mathcal{K}.$$

Per ognuna di tali equazioni valgono le seguenti proprietà:

- per ogni  $K$  viene definita una sola equazione
- è ammessa la ricorsione, ad es.  $K = a.\bar{b}.K$  che equivale a dire che il processo  $K$  esegue  $a$ , poi  $\bar{b}$  e poi si comporta come  $K$  stesso

Si vuole definire ora un sistema di precedenza tra le operazioni. In questo corso vale il seguente sistema:

1.  $\setminus$ ;  $L$ ;
2.  $P[f]$ ;
3.  $\alpha.P$ ;
4.  $|$ ;
5.  $+$ .

Le operazioni sono elencate per ordine decrescente di priorità.

**Esempio 2.1.1.** L'equazione

$$R + a.P \mid b.Q \setminus L$$

equivale all'equazione

$$R + ( (a.P) \mid (b.(Q \setminus L)) ).$$

## 2.2 Semantica

Si vuole passare dal sistema CCS precedentemente definito a un labelled transition systems (LTS).

### 2.2.1 Semantica operativa strutturale

Per introdurre un nuovo LTS, è necessario fissare un insieme di regole di inferenza, come è stato fatto nella logica di Hoare. Tali regole presentano espressioni CCS nelle loro proposizioni.

### Regole di base

$$\frac{}{\alpha.P \xrightarrow{\alpha} P}$$

### Regola della somma

$$\frac{\alpha.P_i \xrightarrow{\alpha} P_j}{\sum_{i \in I} \alpha.P_i \xrightarrow{\alpha} P_j} \quad j \in I$$

### Regole della composizione parallela

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \quad \alpha \in Act \qquad \frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'} \quad \alpha \in Act$$

Mettendo insieme le ultime due, si ottiene la seguente:

$$\frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P' \mid Q'} \quad \alpha \in Act$$

### Regola della restrizione

$$\frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \quad \alpha, \bar{\alpha} \notin L$$

### Regola delle etichettature

$$\frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(a)} P'[f]}$$

### Regola della sostituzione

$$\frac{P \xrightarrow{\alpha} P'}{K \xrightarrow{\alpha} K'} \quad P = K$$

## 2.2.2 Equivalenza rispetto alle tracce

**Definizione 2.2.1.** Dato un processo  $P$ , si definisce "tracce di  $P$ " l'insieme:

$$Tracce(P) = \{w \in A_{CCS}^* : P \xrightarrow{w} P'\},$$

dove  $P \xrightarrow{w} P'$  equivale a dire che:

$$\exists P_1, \dots, P_n \in P_{CCS} : P \xrightarrow{x_1} P_1 \xrightarrow{x_2} P_2 \dots \xrightarrow{x_n} P_n = P',$$

con  $w = x_1 \dots x_n : x_i \in A_{CCS}$ .

Nel caso in cui  $n = 0$  avrò che  $P = P'$ .

Due processi  $P, Q$  si diranno *equivalenti rispetto alle tracce* sse  $Tracce(P) = Tracce(Q)$ . Tale fatto è denotato come:

$$P \sim^T Q.$$

Tale relazione di equivalenza non è sufficiente ai nostri scopi: non distinguendo le *tau*-transizioni tale relazione è troppo debole per i nostri scopi. Sarà necessario introdurre un'altra relazione.

### 2.2.3 Bisimulazione

#### Bisimulazione forte

**Definizione 2.2.2.** Sia  $P \in P_{CCS}$ . Sia inoltre  $\mathcal{R} \subseteq P_{CCS} \times P_{CCS}$  una relazione binaria.  $\mathcal{R}$  è detta di bisimulazione forte sse  $\forall P, Q \in P_{CCS}, P \mathcal{R} Q$  sse:

- $\forall \alpha \in A_{CCS}, P \xrightarrow{\alpha} P'$  allora  $\exists Q' : Q \xrightarrow{\alpha} Q' \wedge P' \mathcal{R} Q'$ ,
- $\forall \alpha \in A_{CCS}, Q \xrightarrow{\alpha} Q'$  allora  $\exists P' : P \xrightarrow{\alpha} P' \wedge P' \mathcal{R} Q'$ .

$P$  e  $Q$  si diranno allora bisimili e si denoteranno nel seguente modo:

$$P \sim^{Bis} Q.$$

**Definizione 2.2.3.** La relazione  $\sim^{Bis}$  è definita nel seguente modo:

$$\sim^{Bis} = \bigcup \{ \mathcal{R} : \mathcal{R} \text{ è relazione di bisimulazione} \}.$$

$\sim^{Bis}$  è una relazione di equivalenza.

Si può dire che due processi  $P, Q \in P_{CCS}$  sono bisimili sse  $\forall \alpha \in A_{CCS}$ :

- $\forall P' : P \xrightarrow{\alpha} P', \exists Q' : Q \xrightarrow{\alpha} Q' \wedge P' \sim^{Bis} Q'$ ,
- $\forall Q' : Q \xrightarrow{\alpha} Q', \exists P' : P \xrightarrow{\alpha} P' \wedge P' \sim^{Bis} Q'$ .

Si introducono ora alcune regole di equivalenza rispetto alla bisimulazione.

Siano  $P, Q \in P_{CCS} \wedge P \sim^{Bis} Q$ , allora:

- $\alpha.P \sim^{Bis} \alpha.Q, \forall \alpha \in A_{CCS}$
- $P + Z \sim^{Bis} Q + Z, \forall Z \in P_{CCS}$
- $P \mid Z \sim^{Bis} Q \mid Z, \forall Z \in P_{CCS}$
- $P \setminus L \sim^{Bis} Q \setminus L, \forall L \in A_{CCS}$
- $P[f] \sim^{Bis} Q[f], \forall f : A_{CCS} \rightarrow A_{CCS}$

Da tali regole è possibile concludere che  $\sim^{Bis}$  è una relazione di congruenza.



### Bisimulazione debole

Si supponga di voler astrarre la relazione di bisimulazione dalle  $\tau$ -transizioni. Ciò è necessario nell'ottica di astrarre l'osservazione delle transizioni da quelle non osservabili (appunto, le  $\tau$ -transizioni).

È necessario introdurre una regola che possa astrarre dalle  $\tau$ -transizioni.

**Definizione 2.2.4.** La relazione  $\Rightarrow$  è definita come segue:

$$\forall \alpha \in A_{CCS}, P \xRightarrow{\alpha} P' \text{ sse } P \xrightarrow{\tau^*} \xrightarrow{\alpha} \xrightarrow{\tau^*} P', \forall P, P' \in P_{CCS}.$$

La relazione  $' \Rightarrow'$  è dunque definita come segue:

$$' \Rightarrow' \subseteq P_{CCS} \times A_{CCS} \times P_{CCS}.$$

**Definizione 2.2.5.** Le *tracce osservabili* sono definite come l'insieme:

$$L(P) = \{w \in (A \cup \bar{A})^* : P \xRightarrow{w}\}.$$

**Definizione 2.2.6.** Sia data una relazione  $\mathcal{R} \subseteq P_{CCS} \times P_{CCS}$ .  $\mathcal{R}$  è detta *bisimulazione debole* sse:

- se  $PRQ$ , allora  $\forall \alpha \in A_{CCS}$ <sup>2</sup> se  $P \xrightarrow{\alpha} P'$ , allora  $\exists Q' : Q \xRightarrow{\alpha} Q' \wedge P' \mathcal{R} Q'$ ,
- se  $Q \xrightarrow{\alpha} Q'$ , allora  $P \xRightarrow{\alpha} P' \wedge P' \mathcal{R} Q'$ .

**Lemma 2.2.1.** Vale la seguente relazione:

$$P \xrightarrow{\tau} \wedge P \xRightarrow{\tau^*} P'.$$

È dunque possibile che  $P'$  coincida con  $P$ .

Allora si dirà che

$$P \approx^{Bis} Q \text{ sse } \exists \mathcal{R} \text{ di bisimulazione debole tale che } PRQ.$$

**Definizione 2.2.7.** La relazione  $\approx^{Bis}$  può essere definita come segue:

$$\approx^{Bis} = \bigcup \{ \mathcal{R} \mid \mathcal{R} \text{ è relazione di bisimulazione debole} \}.$$

Come per la bisimulazione forte, è possibile dire che due processi  $P, Q \in P_{CCS}$  sono debolmente bisimili sse:

- $\forall \alpha \in A_{CCS}, \forall P' : P \xrightarrow{\alpha} P', \exists Q' : Q \xRightarrow{\alpha} Q' \wedge P' \approx^{Bis} Q'$ ,
- $\forall \alpha \in A_{CCS}, \forall Q' : Q \xrightarrow{\alpha} Q', \exists P' : P \xRightarrow{\alpha} P' \wedge P' \approx^{Bis} Q'$ .

---

<sup>2</sup>Si ricorda che  $\tau \in A_{CCS}$ .