COL874: Advanced Compiler Techniques

Modules 181-185

Indrajit Banerjee

So far…

- Hoare triple notation
- Assertions and Invariants
- Verification conditions
- Verification conditions for sequence operator
- Verification conditions for if-then-else operator

Today's discussion…

- Transfer function graph (TFG) representation
- Sequencing with if-then-else operator
- The ternary operator
- Exponential paths problem
- Verification conditions for loops
- Floyd-Naur Proof method
- Hoare logic

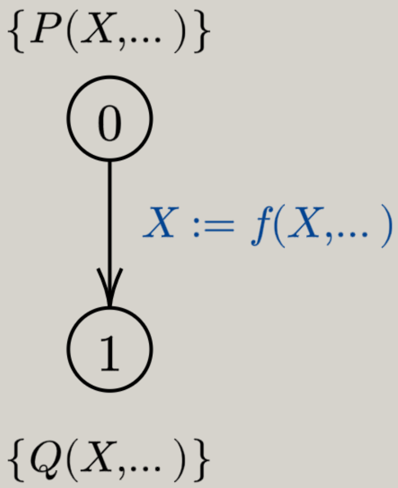
Transfer function graph (TFG) representation

- A graphical representation of a program.
- Each vertex represents a program point. This is where we want to prove assertions.
- Each edge represents a transfer function (e.g., skip, assignment) and a condition under which the edge is taken.

$\{ P(X,\ldots) \}$

$X := f(X,\ldots)$

$\{ Q(X,\ldots) \}$



Sequencing with if-then-else operator

{ P(X,...) }

if B(X,...) then

{ P1(X,...) }

X := f(X,...)

{ P2(X,...) }

else

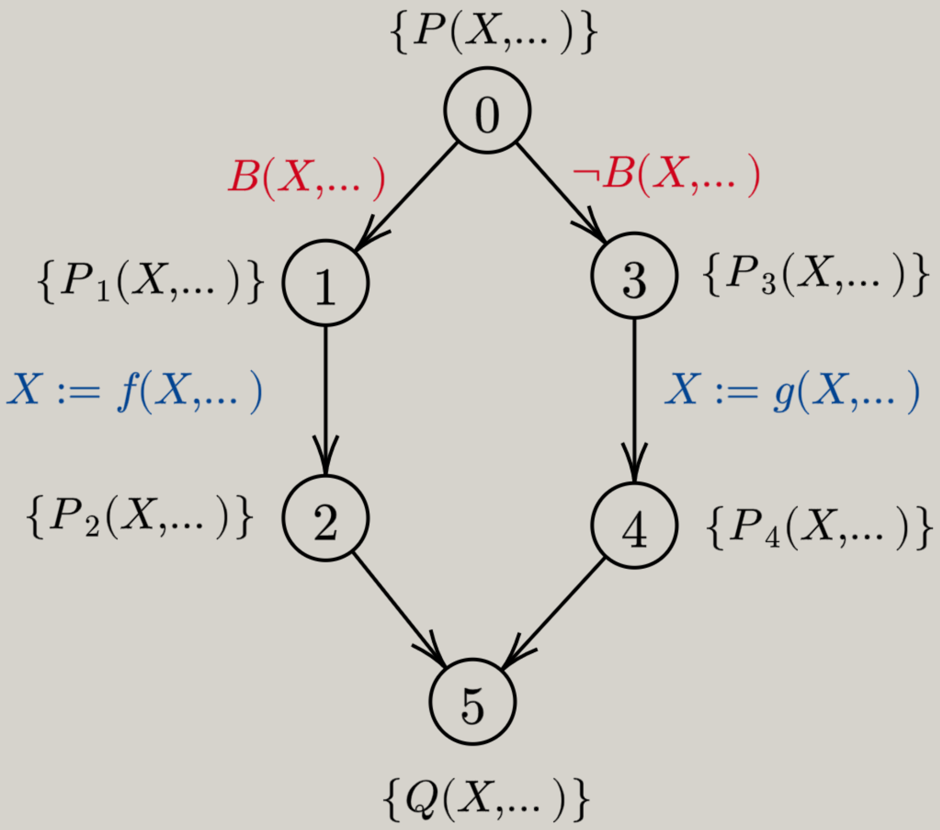
{ P3(X,...) }

X := g(X,...)

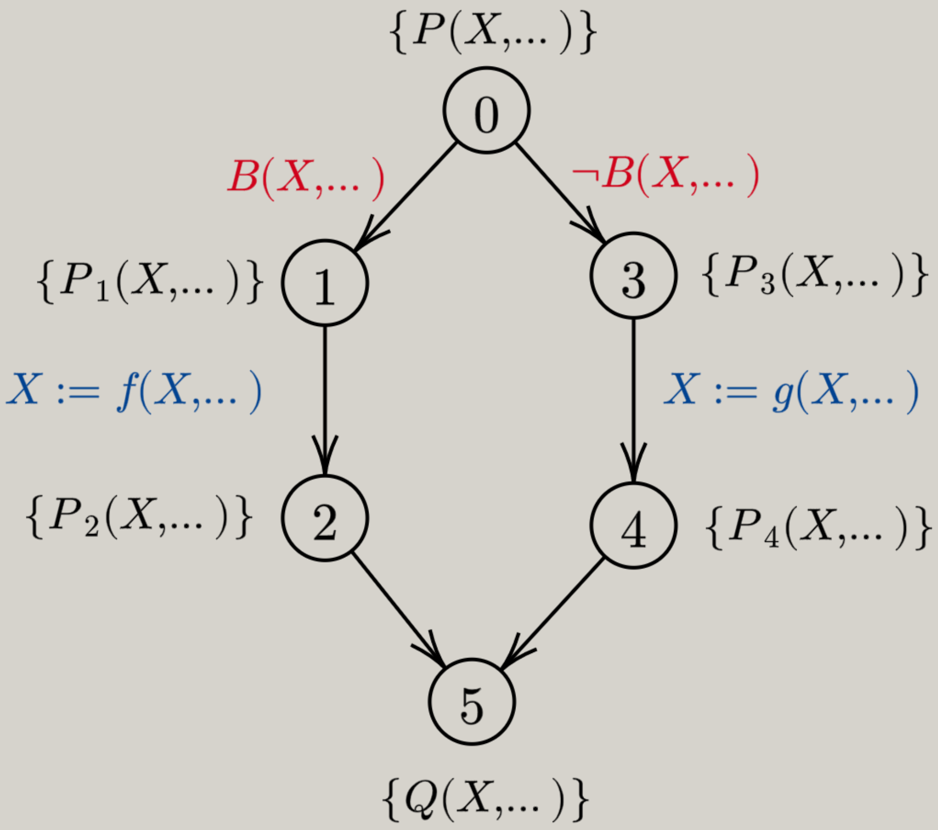
{ P4(X,...) }

endif

{ Q(X,...) }



Sequencing with if-then-else operator



$\{ P \wedge B \}$ skip $\{ P_1 \}$

$\{ P \wedge \neg B \}$ skip $\{ P_3 \}$

$\{ P_1 \}$ X := f(X,…) $\{ P_2 \}$

$\{ P_3 \}$ X := g(X,…) $\{ P_4 \}$

$\{ P_2 \}$ skip $\{ Q \}$

$\{ P_4 \}$ skip $\{ Q \}$

Sequencing with if-then-else operator

$\{ P \land B \}$ skip $\{ P_1 \}$ — $(P \land B) \Rightarrow P_1$

$\{ P \land \neg B \}$ skip $\{ P_3 \}$ — $(P \land \neg B) \Rightarrow P_3$

$\{ P_1 \}\ X := f(X,\ldots)\ \{ P_2 \}$ — $P_1 \Rightarrow P_2\ [X := f(X,\ldots)]$

$\{ P_3 \}\ X := g(X,\ldots)\ \{ P_4 \}$ — $P_3 \Rightarrow P_4\ [X := g(X,\ldots)]$

$\{ P_2 \}$ skip $\{ Q \}$ — $P_2 \Rightarrow Q$

$\{ P_4 \}$ skip $\{ Q \}$ — $P_4 \Rightarrow Q$

Choose P_1,P_2,P_3,P_4 as follows…

$P_2 = P_4 = Q$

$P_1 = P_2\ [X := f(X,\ldots)]$

$P_3 = P_4\ [X := g(X,\ldots)]$,

Verification conditions simplify to,

$(P \land B) \Rightarrow Q\ [X := f(X,\ldots)]$

$(P \land \neg B) \Rightarrow Q\ [X := g(X,\ldots)]$,

Define $(C\ ?\ A : B) \Leftrightarrow (C \Rightarrow A) \land (\neg C \Rightarrow B)$ further simplifying the verification conditions to,

$P \Rightarrow (B\ ?\ Q\ [X := f(X,\ldots) : Q\ [X := g(X,\ldots)]$

Sequencing with if-then-else operator

Define the ternary operator (B ? e1 : e2)
such that programs C_1 and C_2 are equivalent

Program C_1

```
if B then
        X := e1
else
        X := e2
endif
```

Program C_2

```
X := B ? e1 : e2
```

From if-then-else rule,

{ P } C_1 { Q } ⇔ P => (B ? Q[X := e1] : Q[X := e2])

From assignment rule,

{ P } C_2 { Q } ⇔ P => Q[X := B ? e1 : e2]

By definition, C_1 ⇔ C_2

Hence,

P => (B ? Q[X := e1] : Q[X := e2]) ⇔ P => Q[X := B ? e1 : e2]

B ? Q[X := e1] : Q[X := e2] ⇔ Q[X := B ? e1 : e2]

Exponential paths problem

{ P }

X := B_1 ? e1 : e2

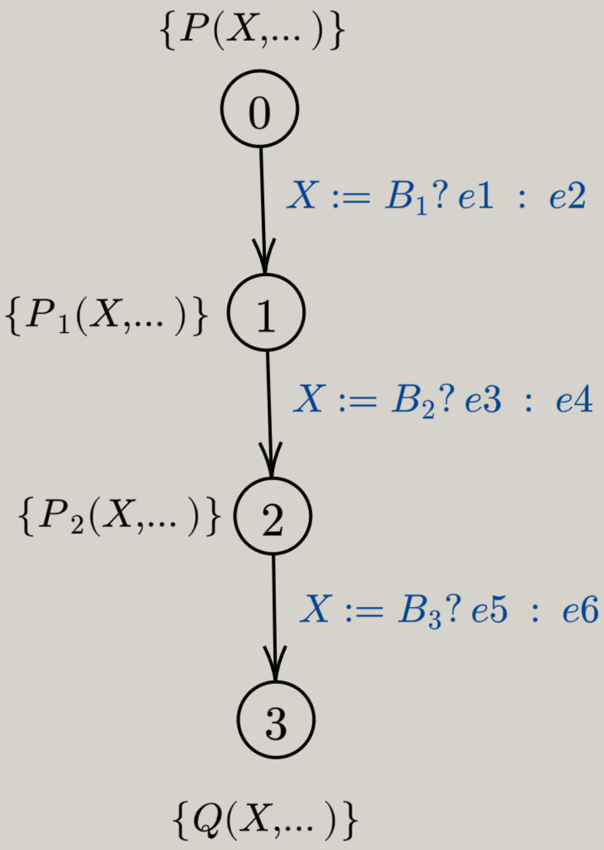
{ P_1 }

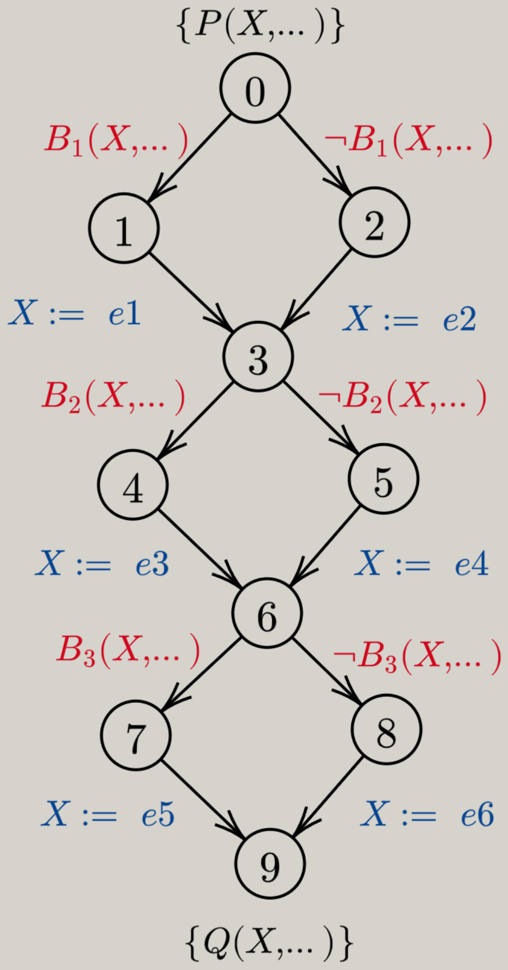
X := B_2 ? e3 : e4

{ P_2 }

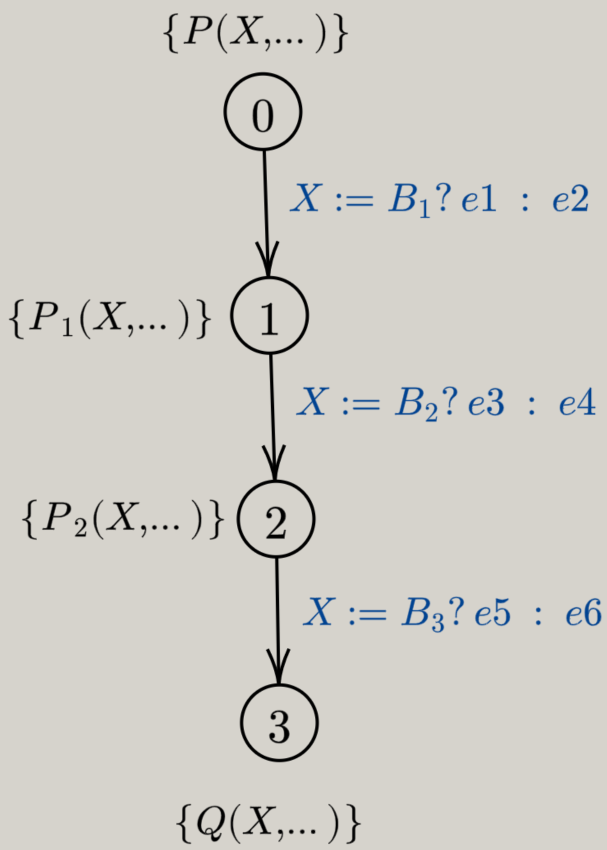
X := B_3 ? e5 : e6

{ Q }





Exponential paths problem



Combining verification conditions of assignment and sequencing,

$P \Rightarrow P_1[X := B_1?e1:e2]$

$P_1 \Rightarrow P_2[X := B_2?e3:e4]$

$P_2 \Rightarrow P_3[X := B_3?e5:e6]$

Choose P_1,P_2 as follows…

$P_1 = P_2[X := B_2?e3:e4]$

$P_2 = P_3[X := B_3?e5:e6]$

Verification condition simplifies to,

$P \Rightarrow P_1[X := B_3?e5:e6][X := B_2?e3:e4][X := B_1?e1:e2]$

Note: B_i's and ei's are also functions of X in general.

Size of the expression grows exponentially!

Verification conditions for loops

{ P(X,…) }

while B(X,…) {

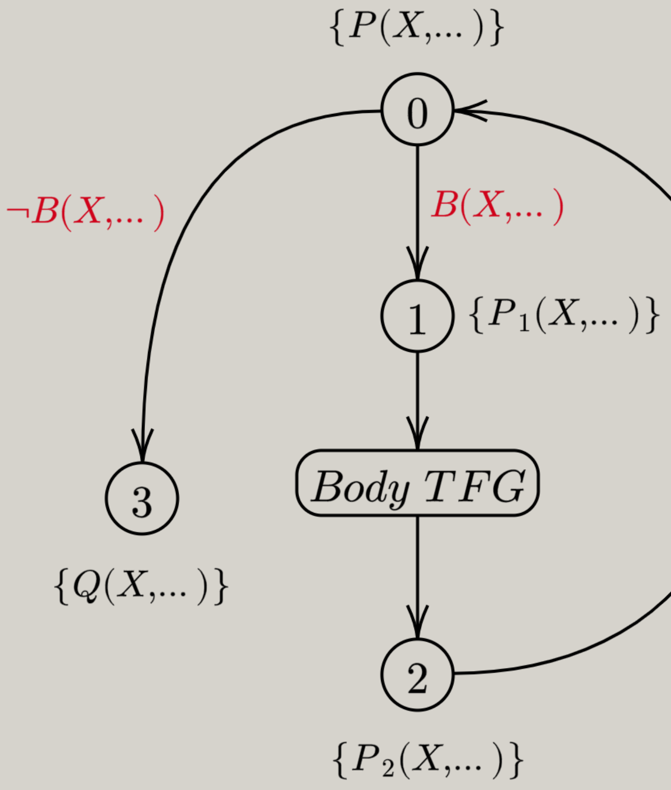
{ P1(X,…) }

Body

{ P2(X,…) }

}

{ Q(X,…) }



Hoare triple queries:

{ P ∧ B } skip { P_1 }

{ P ∧ ¬B } skip { Q }

{ P_1 } Body { P_2 }

{ P_2 } skip { P }

Verification conditions:

(P ∧ B) => P_1

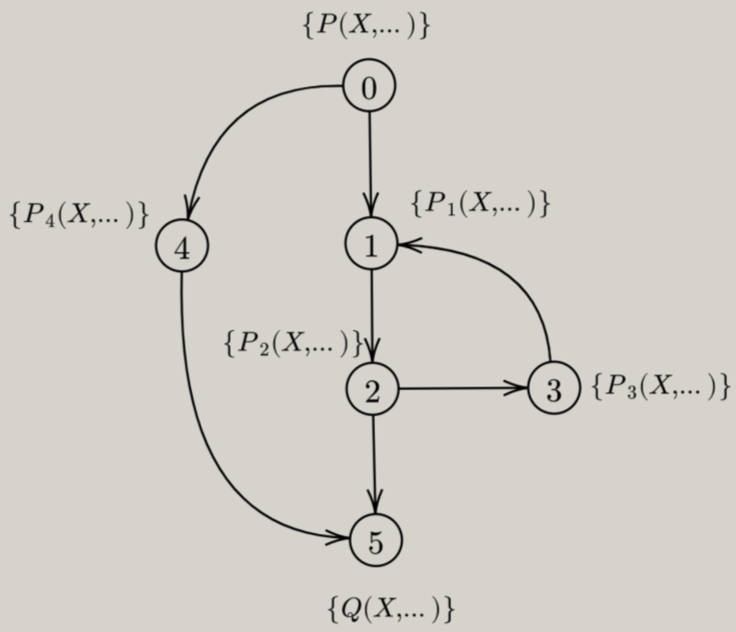
(P ∧ ¬B) => Q

Induction on Body

P_2 => P

Floyd-Naur Proof method

Proof of partial-correctness only. Does not prove termination!



Example TFG

Represent the program as a transfer function graph

Find assertion Pi at vertex i for all intermediate vertices of the graph

Construct a Hoare triple query { Pi ∧ B } f { Pj } for each edge (i, j) of the graph

Prove the verification condition corresponding to each query

Floyd-Naur Proof method example

$\{ X \geq 0 \}$

while $X \neq 0$ {

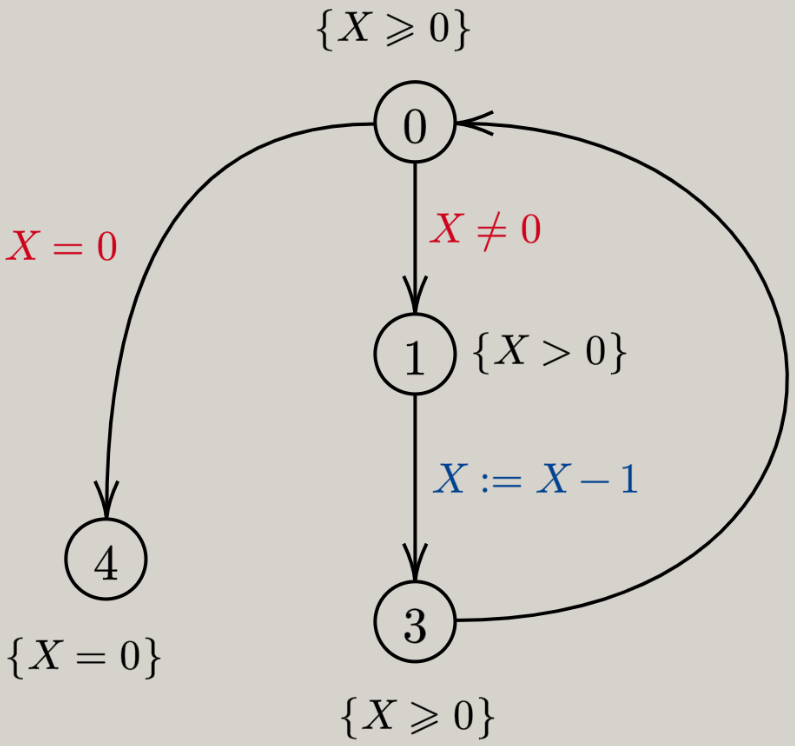
$\{ X > 0 \}$

$X := X - 1$

$\{ X \geq 0 \}$

}

$\{ X = 0 \}$



Hoare triple queries:

$\{ X \geq 0 \land X \neq 0 \}$ skip $\{ X > 0 \}$

$\{ X \geq 0 \land X = 0 \}$ skip $\{ X = 0 \}$

$\{ X > 0 \}$ $X := X - 1$ $\{ X \geq 0 \}$

$\{ X \geq 0 \}$ skip $\{ X \geq 0 \}$

Verification conditions:

$(X \geq 0 \land X \neq 0) \Rightarrow X > 0 \Leftrightarrow$ true

$(X \geq 0 \land X = 0) \Rightarrow X = 0 \Leftrightarrow$ true

$X > 0 \Rightarrow \{ X - 1 \geq 0 \} \Leftrightarrow$ true

$X \geq 0 \Rightarrow X \geq 0 \Leftrightarrow$ true

Hoare logic

Previous approach…

Given a Hoare triple query

↓

Lower it to a first order logic formula (verification conditions)

↓

Prove the first order logic formula

Hoare logic formulation…

Form an inference rule

↑

Abstract it using Hoare triples

↑

Given a first order logic formula which is always true i.e., a tautology

Hoare logic rules

- Assignment

{ P } x := e { Q } ⇔ P => Q [x := e]

We know that P => P is a tautology.

Substituting Q [x := e] for P,

{ Q [x := e] } x := e { Q } ⇔

Q [x := e] => Q [x := e] ⇔ true

$$\frac{}{\{ P [x := e] \}\ x := e\ \{ P \}} \quad (1)$$

- Composition

$$\frac{\{ P \}\ C_1\ \{ R \} \qquad \{ R \}\ C_2\ \{ Q \}}{\{ P \}\ C_1;C_2\ \{ Q \}} \quad (2)$$

- If-then-else

$$\frac{\{ P \land B \}\ C_1\ \{ Q \} \qquad \{ P \land \neg B \}\ C_2\ \{ Q \}}{\{ P \}\ \text{if } B \text{ then } C_1 \text{ else } C_2 \text{ endif}\ \{ Q \}} \quad (3)$$

- Consequence

$$\frac{P \Rightarrow P' \qquad \{ P' \}\ C\ \{ Q' \} \qquad Q' \Rightarrow Q}{\{ P \}\ C\ \{ Q \}} \quad (4)$$

Hoare logic example

$\{ x = x_0 \}$
if $x > 0$
    skip
else
    $x := -x$
endif
$\{ x = |x_0| \}$

- Apply Hoare logic rules backward starting from the required Hoare triple until all branches end in valid axiom (skip and assignment).
- Composition & consequence rules contain variables in premise that do not occur in conclusion.
- Skip and Consequence rule requires first order logic proof obligations.

$$\dfrac{\dfrac{\dfrac{\text{true}}{(x = x_0 \wedge x > 0) \Rightarrow x = |x_0|}}{\{ x = x_0 \wedge x > 0 \}\ \text{skip}\ \{ x = |x_0| \}} \qquad \dfrac{\dfrac{\text{true}}{(x = x_0 \wedge x \leq 0) \Rightarrow (-x = |x_0|)} \quad \dfrac{}{\{ -x = |x_0| \}\ x := -x\ \{ x = |x_0| \}}\ (1) \quad \dfrac{\text{true}}{(x = |x_0|) \Rightarrow (x = |x_0|)}}{\{ x = x_0 \wedge x \leq 0 \}\ x := -x\ \{ x = |x_0| \}}\ (4)}{\{ x = x_0 \}\ \text{if}\ x > 0\ \text{skip else}\ x := -x\ \text{endif}\ \{ x = |x_0| \}}\ (3)$$

Thank You