Predicate Transformers
(Based on [Dijkstra 1975; Gries 1981; Morgan 1994])

Yih-Kuen Tsay

Dept. of Information Management
National Taiwan University
The execution of a sequential program, if terminating, transforms the initial state into some final state.

If, for any given postcondition, we know the *weakest precondition* that guarantees termination of the program in a state satisfying the postcondition, then we have fully understood the meaning of the program.

For a program $S$ and a predicate (or an assertion) $Q$, let $wp(S, Q)$ denote the above weakest precondition.

Therefore, we can see a program as a *predicate transformer* $wp(S, \cdot)$, transforming a postcondition $Q$ (a predicate) into its weakest precondition $wp(S, Q)$.

Note: the weakest precondition is the weakest in the sense that it identifies all the desired initial states and nothing else.
Notational Conventions

⇒ vs. →

A ⇒ B (A entails B) states a relation between two formulae A and B: in every state, if A is true then B is true.

A → B is a formula. When “A → B” stands alone, it usually means A → B is true in every state (model).

≡ vs. ↔

A ≡ B (A is equivalent to B) states a relation between two formulae A and B: in every state, if A is true if and only if B is true.

A ↔ B is a formula. When “A ↔ B” stands alone, it usually means A ↔ B is true in every state (model).
Hoare Triples in Terms of $wp$

When total correctness is meant, $\{P\} \ S \ \{Q\}$ is simply another notation for $P \Rightarrow wp(S, Q)$.

In effect, $wp$ provides a semantic foundation for the Hoare logic.

The precondition $P$ here may be as weak as $wp(S, Q)$, but often a stronger and easier-to-find $P$ is all that is needed.
Properties of $wp$

Fundamental Properties (Axioms):

- **Law of the Excluded Miracle**: $wp(S, false) \equiv false$
- **Distributivity of Conjunction**:
  $wp(S, Q_1) \land wp(S, Q_2) \equiv wp(S, Q_1 \land Q_2)$
- **Distributivity of Disjunction** for deterministic $S$:
  $wp(S, Q_1) \lor wp(S, Q_2) \equiv wp(S, Q_1 \lor Q_2)$

Derived Properties:

- **Law of Monotonicity**: if $Q_1 \Rightarrow Q_2$, then
  $wp(S, Q_1) \Rightarrow wp(S, Q_2)$
- **Distributivity of Disjunction** for nondeterministic $S$:
  $wp(S, Q_1) \lor wp(S, Q_2) \Rightarrow wp(S, Q_1 \lor Q_2)$
Predicate Calculation

Equivalence is preserved by substituting equals for equals

Example:

\[(A \lor B) \rightarrow C\]

\[\equiv \{ A \rightarrow B \equiv \neg A \lor B \}\]

\[\neg (A \lor B) \lor C\]

\[\equiv \{ \text{de Morgan’s law} \}\]

\[\neg A \land \neg B \lor C\]

\[\equiv \{ \text{distributive law} \}\]

\[\neg A \lor C' \land (\neg B \lor C')\]

\[\equiv \{ A \rightarrow B \equiv \neg A \lor B \}\]

\[(A \rightarrow C') \land (B \rightarrow C)\]
Predicate Calculation (cont.)

Entailment distributes over conjunction, disjunction, quantification, and the consequence of an implication.

Example:

\[ \forall x (A \rightarrow B) \land \forall x A \]
\[ \Rightarrow \{ \forall x (A \rightarrow B) \Rightarrow (\forall x A \rightarrow \forall x B) \} \]
\[ (\forall x A \rightarrow \forall x B) \land \forall x A \]
\[ \equiv (\neg \forall x A \lor \forall x B) \land \forall x A \]
\[ \equiv (\neg \forall x A \land \forall x A) \lor (\forall x B \land \forall x A) \]
\[ \equiv \{ \neg A \land A \equiv false \} \]
\[ false \lor (\forall x B \land \forall x A) \]
\[ \equiv \{ false \lor A \equiv A \} \]
\[ \forall x B \land \forall x A \]
\[ \Rightarrow \forall x B \]
Some Laws for Predicate Calculation

- Equivalence is commutative and associative
  \[ A \leftrightarrow B \equiv B \leftrightarrow A \]
  \[ A \leftrightarrow (B \leftrightarrow C) \equiv (A \leftrightarrow B) \leftrightarrow C \]

- False or \( A \) is equivalent to \( A \) or false, which is equivalent to \( A \)

- Negated \( A \) and \( A \) is equivalent to false

- \( A \) implies \( B \) is equivalent to not \( A \) or \( B \)

- \( A \) implies false is equivalent to \( \neg A \)

- \( A \) or \( B \) implies \( C \) is equivalent to \( A \) or \( C \) and \( B \) or \( C \)

- \( A \) implies \( B \) or \( C \) is equivalent to \( A \) and \( B \) implies \( C \)

- \( A \) implies \( B \) is equivalent to \( A \leftrightarrow (A \land B) \)

- \( A \land B \) implies \( A \)
Some Laws for Predicate Calculation (cont.)

- $\forall x (x = E \rightarrow A) \equiv A[E/x] \equiv \exists x (x = E \land A)$, if $x$ is not free in $E$.
- $\forall x (A \land B) \equiv \forall x A \land \forall x B$
- $\forall x (A \rightarrow B) \Rightarrow \forall x A \rightarrow \forall x B$
- $\forall x (A \rightarrow B) \equiv A \rightarrow \forall x B$, if $x$ is not free in $A$.
- $\exists x (A \land B) \equiv A \land \exists x B$, if $x$ is not free in $A$. 
“Extreme” Programs

- \( wp(\text{skip}, Q) \triangleq Q \)
- \( wp(\text{choose } x, x \in \text{Dom}(x)) \triangleq \text{true} \)
- \( wp(\text{choose } x, Q) \triangleq Q, \text{ if } x \text{ is not free in } Q \)
- \( wp(\text{abort}, Q) \triangleq \text{false} \)
The Assignment Statement

🌐 Syntax: \( x := E \)

Note: this becomes a multiple assignment, if we view \( x \) as a list of distinct variables and \( E \) as a list of expressions.

🌐 Semantics: \( \wp(x := E, Q) \triangleq Q[E/x] \).
Sequencing

Syntax: $S_1; S_2$

Semantics: $wp(S_1; S_2, Q) \triangleq wp(S_1, wp(S_2, Q))$. 
Abbreviation of Conjunctions/Disjunctions

Conjunction:
- **Original Form:** $B_1 \land B_2 \land \cdots \land B_n$
- **Abbreviation:** $\forall i : 1 \leq i \leq n : B_i$

Disjunction:
- **Original Form:** $B_1 \lor B_2 \lor \cdots \lor B_n$
- **Abbreviation:** $\exists i : 1 \leq i \leq n : B_i$

This applies to conjunctions/disjunctions of first-order formulae, Hoare triples, etc.
The Alternative Statement

✿ Syntax:

```
IF: if \( B_1 \rightarrow S_1 \)
   \( \| B_2 \rightarrow S_2 \)
   \ldots
   \( \| B_n \rightarrow S_n \)
fi
```

Each of the “\( B_i \rightarrow S_i \)”s is a guarded command, where \( B_i \) is the guard (a boolean expression) and \( S_i \) the command (body).

✿ Informal description: One of the guarded commands, whose guard evaluates to true, is nondeterministically selected and the corresponding command executed. If none of the guards evaluates to true, then the execution aborts.
The Alternative Statement (cont.)

 Syntax:

\[
\text{IF: } \textbf{if } B_1 \rightarrow S_1 \\
\quad \mid B_2 \rightarrow S_2 \\
\quad \quad \cdots \\
\quad \mid B_n \rightarrow S_n \\
\textbf{fi}
\]

 Semantics:

\[
wp(\text{IF, } Q) \overset{\Delta}{=} (\exists i : 1 \leq i \leq n : B_i) \\
\quad \land (\forall i : 1 \leq i \leq n : B_i \rightarrow wp(S_i, Q))
\]

 The case of simple IF:

\[
wp(\textbf{if } B \rightarrow S \textbf{fi}, Q) \overset{\Delta}{=} B \land (B \rightarrow wp(S, Q))
\]
The Alternative Statement (cont.)

Suppose there exists a predicate $P$ such that

1. $P \Rightarrow (\exists i: 1 \leq i \leq n : B_i)$ and
2. $\forall i : 1 \leq i \leq n : P \land B_i \Rightarrow \text{wp}(S_i, Q)$.

Then $P \Rightarrow \text{wp}(\text{IF}, Q)$.

The less obvious part is $P \Rightarrow (\forall i : 1 \leq i \leq n : B_i \rightarrow \text{wp}(S_i, Q))$.

\[
\forall i : 1 \leq i \leq n : (P \land B_i) \rightarrow \text{wp}(S_i, Q)
\]

\[
\equiv \forall i : 1 \leq i \leq n : P \rightarrow (B_i \rightarrow \text{wp}(S_i, Q))
\]

\[
\equiv P \rightarrow (\forall i : 1 \leq i \leq n : B_i \rightarrow \text{wp}(S_i, Q))
\]
Inference rule in the Hoare logic:

\[
P \Rightarrow (\exists i : 1 \leq i \leq n : B_i) \quad \forall i : 1 \leq i \leq n : \{ P \land B_i \} \ S_i \ \{ Q \}
\]

\[
\{ P \} \text{ IF : if } B_1 \rightarrow S_1 \mid \cdots \mid B_n \rightarrow S_n \ \text{fi } \{ Q \}
\]

This rule follows from the preceding theorem.

The case of simple \textbf{IF}:

\[
P \Rightarrow B \quad \{ P \land B \} \ S \ \{ Q \}
\]

\[
\{ P \} \ \text{if } B \rightarrow S \ \text{fi } \{ Q \}
\]
The Iterative Statement

🔮 Syntax:

```
DO: do B_1 → S_1
    | B_2 → S_2
    ...
    | B_n → S_n
od
```

Each of the “$B_i \rightarrow S_i$”s is a guarded command.

🔮 Informal description: Choose (nondeterministically) a guard $B_i$ that evaluates to true and execute the corresponding command $S_i$. If none of the guards evaluates to true, then the execution terminates.

🔮 The usual “while $B$ do $S$ od” can be defined as this simple while-loop: “do $B \rightarrow S$ od”.
Let \( BB \) denote \( \exists i : 1 \leq i \leq n : B_i \), i.e., \( B_1 \lor B_2 \lor \ldots \lor B_n \).

The \textbf{DO} statement is equivalent to

\[
\textbf{do } BB \rightarrow \textbf{if } B_1 \rightarrow S_1 \\
\hspace{1cm} \textbf{if } B_2 \rightarrow S_2 \\
\hspace{2cm} \ldots \\
\hspace{3cm} \textbf{if } B_n \rightarrow S_n \\
\textbf{if } \textbf{od}
\]

or simply \textbf{do } BB \rightarrow \textbf{IF } \textbf{od}.

This suggests that we could have got by with just the simple \textit{while}-loop.
The Iterative Statement (cont.)

Again, let $BB$ denote $\exists i : 1 \leq i \leq n : B_i$.

Let $H_k(Q), k \geq 0$, be defined as follows.

\[
\begin{align*}
H_0(Q) & \triangleq \neg BB \land Q \\
H_k(Q) & \triangleq H_0(Q) \lor wp(IF, H_{k-1}(Q)) \quad \text{for } k > 0
\end{align*}
\]

The predicate $H_0(Q)$ represents the set of states where execution of $DO$ terminates immediately (0 iteration).

The predicate $H_k(Q), \text{ for } k > 0$, represents the set of states where execution of $DO$ terminates after at most $k$ iterations.

Semantics of $DO$:

\[wp(DO, Q) \triangleq (\exists k : 0 \leq k : H_k(Q))\]
A More Useful Theorem for \( \text{DO} \)

Suppose there exist a predicate \( P \) and an integer-valued expression \( t \) such that

1. \( \forall i : 1 \leq i \leq n : P \land B_i \Rightarrow wp(S_i, P) \),
2. \( P \Rightarrow (t \geq 0) \), and
3. \( \forall i : 1 \leq i \leq n : P \land B_i \land (t = t_0) \Rightarrow wp(S_i, t < t_0) \),

where \( t_0 \) is a rigid variable.

Then \( P \Rightarrow wp(\text{DO}, P \land \neg \text{BB}) \).

\[
\begin{align*}
P & \equiv P \land (\exists k : 0 \leq k : t \leq k) \quad (t \text{ is finite}) \\
& \equiv \exists k : 0 \leq k : P \land t \leq k \quad (k \text{ is not free in } P) \\
& \Rightarrow \exists k : 0 \leq k : H_k(P \land \neg \text{BB}) \quad (P \land t \leq k \Rightarrow H_k(P \land \neg \text{BB})) \\
& \equiv wp(\text{DO}, P \land \neg \text{BB}) \quad (\text{def. of DO})
\end{align*}
\]
A More Useful Theorem for \texttt{DO} (cont.)

- Proof of $P \land t \leq k \Rightarrow H_k(P \land \neg \text{BB})$ is by induction on $k$.
- Will do this for the case of simple \texttt{DO}.
A Simplified Theorem for Simple

Suppose there exist a predicate $P$ and an integer-valued expression $t$ such that

1. $P \land B \implies wp(S, P)$,
2. $P \implies (t \geq 0)$, and
3. $P \land B \land (t = t_0) \implies wp(S, t < t_0)$, where $t_0$ is a rigid variable.

Then $P \implies wp(\text{do } B \rightarrow S \text{ od}, P \land \neg B)$.

This is to be contrasted by

\[
\begin{align*}
\{ P \land B \} & S \{ P \} & \{ P \land B \land t = Z \} & S \{ t < Z \} & P \implies (t \geq 0) \\
\{ P \} & \text{while } B \text{ do } S \text{ od } \{ P \land \neg B \}
\end{align*}
\]
Proof of $P \land t \leq k \Rightarrow H_k(P \land \neg B)$ is by induction on $k$.

Recall, for simple DO,

\[
\begin{align*}
H_0(Q) & \triangleq \neg B \land Q \\
H_k(Q) & \triangleq H_0(Q) \lor \text{wp}(\text{if } B \rightarrow S \text{ fi}, H_{k-1}(Q)) \quad \text{for } k > 0
\end{align*}
\]
Base case: \( P \land t \leq 0 \Rightarrow H_0(P \land \neg B) \), which is equivalent to \( P \land t \leq 0 \Rightarrow P \land \neg B \).

Since \( P \Rightarrow (t \geq 0) \), it suffices to show that \( P \land t = 0 \Rightarrow \neg B \).

\[
\begin{align*}
P \land t = 0 &\land B \\
\equiv &\ (P \land B) \land (P \land B \land t = 0) \\
\Rightarrow &\ wp(S, P) \land wp(S, t < 0) \\
\equiv &\ wp(S, P \land t < 0) \\
\equiv &\ wp(S, false) \\
\equiv &\ false
\end{align*}
\]
Inductive step \((k > 0)\): \(P \land t \leq k \Rightarrow H_k(P \land \neg B)\), i.e.,
\(P \land t \leq k \Rightarrow H_0(P \land \neg B) \lor wp(\text{if } B \rightarrow S \text{ fi}, H_{k-1}(P \land \neg B))\).

Split \(P \land t \leq k\) into three cases:
- \(P \land (t \leq k - 1)\)
- \(P \land B \land (t = k)\)
  \[\Rightarrow B \land (B \rightarrow wp(S, P)) \land B \land (B \rightarrow wp(S, t < k))\]
  \[\Rightarrow wp(\text{if } B \rightarrow S \text{ fi}, P) \land wp(\text{if } B \rightarrow S \text{ fi}, t < k)\]
  \[\equiv wp(\text{if } B \rightarrow S \text{ fi}, P \land t < k)\]
  \[\equiv wp(\text{if } B \rightarrow S \text{ fi}, P \land (t \leq k - 1))\]
  \[\Rightarrow wp(\text{if } B \rightarrow S \text{ fi}, H_{k-1}(P \land \neg B))\]
  \[\Rightarrow H_0(P \land \neg B) \lor wp(\text{if } B \rightarrow S \text{ fi}, H_{k-1}(P \land \neg B))\]
- \(P \land \neg B \land (t = k)\)
Refinement

Syntax:

\[ \text{prog}_1 \sqsubseteq \text{prog}_2 \]

which is read as “\( \text{prog}_1 \) is refined by \( \text{prog}_2 \)” or “\( \text{prog}_2 \) refines \( \text{prog}_1 \)” (\( \text{prog}_2 \sqsupseteq \text{prog}_1 \)).

Informal description: intuitively, the refinement relation conveys the concept of program \( \text{prog}_2 \) being better than \( \text{prog}_1 \). Program \( \text{prog}_2 \) is better in the sense that it is more accurate, applies in more situations, or runs more efficiently.

A program may be derived through a series of refinement steps.
Specifications

- **Syntax:**

  \[ w : [pre, post] \]

  where \( pre \) is the precondition, \( post \) is postcondition, and the “\( w \)” part is called the **frame**.

- **Informal description:** the specification describes an abstract program such that if the initial state satisfies the precondition \( pre \), then it changes only variables listed in the frame and terminates in a final state satisfying the postcondition \( post \).

- **Examples:**

  - \( y : [0 \leq x \leq 9, y^2 = x] \)
  - \( y : [0 \leq x, y^2 = x \land y \geq 0] \)
Some Laws for Refinement

**Strengthen Postcondition**: If $post' \Rightarrow post$, then

$$w : [pre, post] \sqsubseteq w : [pre, post']$$

Example:

$$y : [0 \leq x \leq 9, y^2 = x] \sqsubseteq y : [0 \leq x \leq 9, y^2 = x \land y \geq 0]$$

**Weaken Precondition**: If $pre \Rightarrow pre'$, then

$$w : [pre, post] \sqsubseteq w : [pre', post]$$

Example:

$$y : [0 \leq x \leq 9, y^2 = x \land y \geq 0] \sqsubseteq y : [0 \leq x, y^2 = x \land y \geq 0]$$

**Combining the two refinements**,

$$y : [0 \leq x \leq 9, y^2 = x] \sqsubseteq y : [0 \leq x, y^2 = x \land y \geq 0]$$
assignment: If $\text{pre} \Rightarrow \text{post}[E/x]$, then

$$w, x : [\text{pre}, \text{post}] \sqsubseteq x := E$$

Note: $w$ may (but not necessarily) be changed.

sequential composition: For any predicate $\text{mid}$,

$$w : [\text{pre}, \text{post}] \sqsubseteq w : [\text{pre}, \text{mid}] ; w : [\text{mid}, \text{post}]$$
Semantics of Specification

- Syntax: \( w : [pre, post] \)
- Semantics:

\[
wp(w : [pre, post], Q) \triangleq pre \land (\forall w(post \rightarrow Q))[v/v_0]
\]

where the substitution \([v/v_0]\) replaces all “initial” variables, i.e., \(v_0\), by corresponding final variables. Note: initial variables \(v_0\) do not occur in \(Q\).

- Example: \( wp(x := x \pm 1, Q) \equiv Q[x + 1/x] \land Q[x - 1/x] \)
\[ \text{Semantics of Specification (cont.)} \]

\[
wp(x := x \pm 1, Q) \\
\equiv \ wp(x : [\text{true}, x = x_0 + 1 \lor x = x_0 - 1], Q) \\
\equiv \ \{ \text{def. of specification} \} \\
\text{true} \land \forall x((x = x_0 + 1 \lor x = x_0 - 1) \rightarrow Q)[x/x_0] \\
\equiv \ \forall x((x = x_0 + 1 \rightarrow Q) \land (x = x_0 - 1 \rightarrow Q))[x/x_0] \\
\equiv \ (\forall x(x = x_0 + 1 \rightarrow Q) \land \forall x(x = x_0 - 1 \rightarrow Q))[x/x_0] \\
\equiv \ \forall x(x = x_0 + 1 \rightarrow Q)[x/x_0] \land \forall x(x = x_0 - 1 \rightarrow Q)[x/x_0] \\
\equiv \ \{ \forall x(x = E \rightarrow A) \equiv A[E/x] \} \\
(Q[x_0 + 1/x])[x/x_0] \land (Q[x_0 - 1/x])[x/x_0] \\
\equiv \ \{ \text{Q does not contain } x_0 \} \\
Q[x + 1/x] \land Q[x - 1/x] \]
Semantics of Refinement

🌍 Syntax: \( \text{prog}_1 \subseteq \text{prog}_2 \)

🌍 Semantics: for all \( Q \),

\[
wp(\text{prog}_1, Q) \Rightarrow wp(\text{prog}_2, Q)
\]

🌍 Examples:

☀️ \( x := x \pm 1 \subseteq x := x + 1 \)

\[
wp(x := x \pm 1, Q) \\
\equiv Q[x + 1/x] \land Q[x - 1/x] \\
\Rightarrow Q[x + 1/x] \\
\equiv wp(x := x + 1, Q)
\]

☀️ \( x := x \pm 1 \subseteq x := x - 1 \)
References