Our goal is to design a program which computes the array $D$.

Assume a graph with vertices $0 \leq i, j \leq N$. For every pair of nodes, we are given a weight $W[i][j]$ which denotes the absence of an edge. It is required that $W[i][i] = 0$. The weight function $W$ can be an integer (possibly negative) or the special value $\infty$.

Under these conditions, it is possible to show that there exists a minimal weight array $D$, such that $D[i][j]$ is the minimal weight over all paths connecting $i$ to $j$.

Our goal is to design a program which computes the array $D$. A. Pnueli

Case Study: All-Points Shortest-Path Problem
A possible solution to the problem is given by the following Unity program:

Program $P_1$

\[
\begin{align*}
\left( D = p \right) & \iff \text{FP} \\
\text{Safety:} & \quad \left( \left( D = p \right) \iff \text{FP} \right) \\
\text{Liveness:} & \quad \diamond
\end{align*}
\]

The formal specification of this program consists of:

Program $P_1$

\[
\begin{align*}
\langle ([\ell, i] p + [\alpha, i] p)[\ell, i] p \rangle & =: [\ell, i] p :: \alpha, [\ell, i] p :: \square \\\n\langle [\ell, i] M1 = [\ell, i] p :: [\ell, i] p \rangle & =: [\ell, i] p :: \square \\
\end{align*}
\]
In order to prove the safety property, we use the following assertion:

\[ \langle ([\ell, i]p : \infty) \neq ([\ell, i]p : \ell, i+) \rangle = \text{sum} \]
\[ \infty = [\ell, i]p \text{ number of pairs } (\ell, i) \text{ for which } p \]

and the ranking function \((\text{sum}, \text{num})\) where
\[ ([\ell, i]p + [\ell, i]p, [\ell, i]p) \neq [\ell, i]p \]
\[ : (\ell, i) \phi \]

For the proof of liveness, we take for the helpful assertions
\[ \langle ([\ell, i]p + [\ell, i]p, [\ell, i]p) \text{ min } = [\ell, i]p : \phi \rangle \]
\[ \lor \]
\[ \equiv \text{FP} \]

The fixed point for program \(P_1\) is given by
\[ [\ell, i]p \supset [\ell, i]p \]

In order to prove the correctness of \(P_1\)

Proving the Correctness of \(P_1\)

A. Pnueli
It is not very difficult to prove that is an invariant of and that the program eventually terminates.

Proof Continued
When mapping program $P$ onto sequential architecture, we have to decide in what order should we activate the various assignments. In principle, we can traverse the 3 indices $i$, $j$, and $k$ in different orders. We show two different styles of expressing these scheduling decisions.

The first style is declarative and is based on presentation via equational scheme.

Current case, an optimal schedule is given by letting $i$ and $j$ vary faster than $k$. We show two different styles of expressing these scheduling decisions. In the current case, an optimal schedule is given by letting $i$ and $j$ vary faster than $k$. When mapping program $P$ onto sequential architecture, we have to decide in what order should we activate the various assignments. In principle, we can traverse the 3 indices $i$, $j$, and $k$ in different orders. We show two different styles of expressing these scheduling decisions.

The first style is declarative and is based on presentation via equational scheme.

When mapping program $P$ onto sequential architecture, we have to decide in what order should we activate the various assignments. In principle, we can traverse the 3 indices $i$, $j$, and $k$ in different orders.
Program $P_2$

\[
\text{declare } H : \text{array } [0::N+1; 0::N+1; 0::N+1]
\]

\[
\begin{align*}
\text{always } h_k i;j::H[i;j;0] = W[i;j] \\
h_k i;j::H[i;j;k+1] &= \min(H[i;j;k]; H[i;k;k] + H[k;j;k])
\end{align*}
\]

\[
d[i;j] = H[i;j;N]
\]

\text{end } P_2

Claim 3. For every $k \in 0..N$, $H[i;j;k]$ is the minimum length of all paths from $i$ to $j$ with intermediate nodes whose indices are smaller than $k$.

Proof: For $k = 0$, $H[0,0] = [0,0]W = 0$. For every $k \in 0..N$, $H[i;j;k]$ is the minimum length of all paths from $i$ to $j$ with intermediate nodes whose indices are smaller than $k$.

\[
\langle [N, ?, ?] H = [?, ?, ?] P :: ?, \parallel \rangle \quad \Box
\]

\[
\langle \langle [?, ?, ?] H + [?, ?, ?] H \cdot [?, ?, ?] H \rangle \text{min} = [1 + ?, ?, ?] H :: ?, \parallel \rangle :: ?, \Box \rangle
\]

\[
\langle [?, ?, ?] W = [0, ?, ?] H :: ?, \parallel \rangle \quad \Box
\]

Always

\[
\langle [N] H = 0, 0, \ldots, 0 \rangle :: N \rightarrow 1, 0::N \rightarrow 1, 0::N \rangle
data array $H$

\text{declare}$

Equational Scheme Program

Lecture 4: Shortest Path Program
A Program with Explicit Sequencing

If we were writing the program in a sequential language, we would probably produce the following Pascal-like program:

```pascal
program Floyd-Warshall;

{This can be encoded in Unity as follows:

\[(a \cdot n \cdot x) \oplus (a \cdot n \cdot x) =: (a \cdot n \cdot x) \quad \|
\]
\[(a \cdot x)p + [x \cdot n]p \cdot [a \cdot n]p) \parallel\parallel =: [a \cdot n]p
\]
\[0,0,0 = a \cdot n \cdot x \quad \parallel\parallel \langle [\ell \cdot ?] \parallel = [\ell \cdot ?]p :: \ell \cdot ? \parallel\rangle \]
\}
```

This can be encoded in Unity as follows:

```pascal
program Floyd-Warshall;

{This can be encoded in Unity as follows:

\[(a \cdot n \cdot x) \oplus (a \cdot n \cdot x) =: (a \cdot n \cdot x) \quad \|
\]
\[(a \cdot x)p + [x \cdot n]p \cdot [a \cdot n]p) \parallel\parallel =: [a \cdot n]p
\]
\[0,0,0 = a \cdot n \cdot x \quad \parallel\parallel \langle [\ell \cdot ?] \parallel = [\ell \cdot ?]p :: \ell \cdot ? \parallel\rangle \]
\}
```
This program can terminate in $N$ steps, if executed on a parallel synchronous architecture with $N^2 + I$ processes.

\begin{verbatim}
\textbf{Program} \texttt{ParallelFloyd-Warshall} \{ \\
\texttt{declare} \ k : \text{integer} \ \text{initially} \ 0 ; \\
\texttt{assign} \ d[i,j] :: d[i,j] = W[i,j] ; \\
\texttt{if} \ k < N \ { \\
\texttt{declare} \ \ k := k + 1 ; \\
\texttt{assign} \ d[i,j] :: d[i,j] = \min( d[i,j] , d[i,k] + d[k,j] ) \\
\}
\}
\end{verbatim}

The following \textit{Unity} program:

\begin{verbatim}
\textbf{Program} \texttt{ParallelFloyd-Warshall} \{ \\
\texttt{declare} \ k : \text{integer} \\
\texttt{assign} \ N > 1 \ \text{initially} \ 1 + 1 = : N \\
\texttt{assign} \ \langle [t,\gamma] p + [\gamma',\gamma] p + [t',\gamma] p \rangle := [t',\gamma] p :: \langle t,\gamma' \rangle p || \\
0 = \gamma || \langle [t,\gamma] \ W = [t',\gamma] p :: \langle t,\gamma' \rangle p || \\
\}
\end{verbatim}

We can collect together statements that can be executed in parallel. This leads to:

\textbf{Parallel Synchronous Architectures}
This leads to overall time of \( O(\log^2 N) \).

If we have \( N^2 \) processors, we can do better as follows:

Program 3 Steps with \( N^3 \) Processors

\[
\begin{align*}
\text{assign} & \quad p = [\ell', \ell']p :: \ell', \ell' \mid \\
[\ell', \ell']p + [\ell', \ell']p & \quad \text{initially} \\
[\ell', \ell']M & \quad \text{initially}
\end{align*}
\]

Claim 4. After the \( m \)th step, \( p = [\ell', \ell']p :: \ell', \ell' \mid \)

\( \langle [\ell', \ell']p + [\ell', \ell']p :: \ell', \ell' \rangle =: [\ell', \ell']p :: \ell', \ell' \mid \rangle \)

According to this claim, the program reaches a fixed point after \( \log N \) executions of the assignment. Each assignment can be done in \( \log N \) steps consisting of computing the \( N^2 \) values of \( d_{i;k} + d_{k;j} \) and computing the minimum for each \( i;j \) in time \( \log N \) using \( N \) processes.

This leads to overall time of \( O(\log^2 N) \).

Step with \( N^3 \) Processors

\[
\begin{align*}
\text{Program 4: Shortest Path Program}
\end{align*}
\]

\[
\begin{align*}
\text{Claim 4. After the } m \text{th step, } p = [\ell', \ell']p :: \ell', \ell' \mid \\
[\ell', \ell']p + [\ell', \ell']p & \quad \text{initially} \\
[\ell', \ell']M & \quad \text{initially}
\end{align*}
\]

If we have \( N^2 \) processors, we can do better as follows:
We consider a variant of program \( P_4 \), in which processors may progress at different rates.

**Asynchronous Shared-Memory Architectures**

We observe a property of the array \( H \), which is \( H_{i;j;k+1} \geq H_{i;j;k} \). Recall that the invariants of program \( P_4 \) was

\[
\begin{align*}
\forall i, j \in [0..N] \quad & d_{i;j} = H_{i;j;k} \quad \text{is the length of some path from } i \text{ to } j \\
\end{align*}
\]

For program \( P_6 \), this can be relaxed into

\[
\begin{align*}
\forall i, j \in [0..N] \quad & d_{i;j} = H_{i;j;k} \quad \text{is the length of some path from } i \text{ to } j \\
\forall i, j \in [0..N] \quad & H = H_{i;j;k+1} \geq H_{i;j;k} \quad \text{which is } H \\
\end{align*}
\]

End \( P_6 \)

Assign

Initially

Declare

Program \( P_6 \)

Different Rates.