Efficient Detection of Determinacy Races in Cilk Programs

Charles E. Leiserson
MIT Laboratory for Computer Science

Mingdong Feng
National University of Singapore and
MIT Laboratory for Computer Science
Determinacy Races

A Cilk program contains a determinacy race if two logically parallel threads access the same shared location, and one of the accesses is a write.

read x  read x

write x  write x

read/write race  write/write race
A Cilk Program with a Determinacy Race

```cilk
int x;

cilk void foo()
{
    x++;
    return;
}

cilk int main()
{
    x = 0;
    spawn foo();
    spawn foo();
    sync;
    printf("x is %d\n", x);
    return 1;
}
```

spawn tree
The Effect of a Determinacy Race

- A determinacy race can cause a Cilk program to behave *nondeterministically*.
- A determinacy race is usually a bug.
Races in N-queens Puzzle

cilk char *nqueens(char *board, int n, int row)
{
    char *new_board;
    ...
    new_board = malloc(row+1);
    memcpy(new_board, board, row);
    for (j = 0; j < n; j++)
    {
        ...
        new_board[row] = j;
        spawn nqueens(new_board, n, row+1);
        ...
    }
    sync;
    ...
}

row → 2

```
0 1 2 3
0 1 2 3
0 1 2 3
0 1 2 3
```

new_board
Races in N-queens Puzzle

cilk char *nqueens(char *board, int n, int row)
{
    char *new_board;
    ...
    new_board = malloc(row+1);
    memcpy(new_board, board, row);
    for (j = 0; j < n; j++)
    {
        new_board[row] = j;
        spawn nqueens(new_board, n, row+1);
        ...
    }
    sync;
    ...
}

Race between child reading & parent writing

Race (child)

Board (parent)
The Nondeterminator

Cilk program + Input data set

FAIL
Information localizing a determinacy race.

PASS
Every scheduling produces the same result.

A debugging tool, not a verifier.
Theorem. For a Cilk program that runs in \( T \) time serially and uses \( v \) shared-memory locations, the Nondeterminator runs in \( O(T \alpha(v,v)) \) time, where \( \alpha \) is Tarjan’s functional inverse of Ackermann’s function.

- The Nondeterminator is a serial program.
- As a practical matter, \( \alpha(v,v) \leq 4 \).
- The Nondeterminator uses \( O(v) \) space.
# Related Work

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time/Access</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>English-Hebrew</strong></td>
<td>$O(pt)$</td>
<td>$O(vt + \min(np,vtp))$</td>
</tr>
<tr>
<td>[Nudler/Rudolph 1986]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Task recycling</strong></td>
<td>$O(t)$</td>
<td>$O(vt + t^2)$</td>
</tr>
<tr>
<td>[Dinning/Schonberg 1990]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset-span labeling</strong></td>
<td>$O(p)$</td>
<td>$O(v + \min(np,vp))$</td>
</tr>
<tr>
<td>[Mellor-Crummey 1991]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SP-bags</strong></td>
<td>$O(\alpha(v,v))$</td>
<td>$O(v)$</td>
</tr>
<tr>
<td>[Feng/Leiserson 1996]</td>
<td>amortized</td>
<td></td>
</tr>
</tbody>
</table>

$n =$ number of threads  
$t =$ max number of parallel threads  
$v =$ number of shared locations  
$p =$ max depth of nested parallelism
Outline

• SERIES-PARALLEL DAGS
• TARJAN’S LEAST COMMON ANCESTORS ALGORITHM
• THE SP-BAGS ALGORITHM
• EMPIRICAL RESULTS
• CONCLUSION
• Series-Parallel Dags
Series-Parallel Dags

Base graph:
• source $s$
• sink $t$

Series composition:
• $s = s_1$
• $t_1 = s_2$
• $t = t_2$

Parallel composition:
• $s = s_1 = s_2$
• $t = t_1 = t_2$
A Cilk Dag is Series-Parallel

F: cilk int main()
{
  x = 0;
  F₁: spawn foo();
  e₁:
  F₂: spawn foo();
  e₂:
  sync;
  e₃: printf("%d", x);
  return 1;
}
Series-Parallel Parse Trees

Every series-parallel dag has a **parse tree**. The **least common ancestor** of two threads determines whether the threads are logically in series or in parallel.

- \( e < e' \) if \( \text{LCA}(e,e') \) is \( S \) and \( e \) is left of \( e' \).
- \( e \parallel e' \) if \( \text{LCA}(e,e') \) is \( P \).

A treewalk visits threads in serial execution order.
Canonical Cilk Parse Tree

```
e; spawn F; e; spawn F; ... e; sync;
```

```
F: S S ... S e
```

```
F:
```

```
e; sync block
```

```
e; return;
```
• Tarjan’s Least Common Ancestors Algorithm
Least Common Ancestors

**Definition.** The *least common ancestor* of two nodes in a rooted tree is the node on the path between them that is closest to the root.
Disjoint-Set Data Structure

$\Sigma$ is a collection of disjoint sets.

- $X, Y \in \Sigma$ implies $X \cap Y = \emptyset$.

Three operations:

- $\textbf{MAKE-SET}(e)$: $\Sigma \leftarrow \Sigma \cup \{\{e\}\}$.
- $\textbf{UNION}(X, Y)$: $\Sigma \leftarrow \Sigma - \{X, Y\} \cup \{X \cup Y\}$.
- $\textbf{FIND-SET}(e)$: returns $X \in \Sigma$ such that $e \in X$.

Any sequence of $m$ operations on $n$ sets can be performed in $O(m \alpha(m,n))$ time [Tarjan 1975].
Tarjan's LCA Algorithm

**Depth-first treewalk:**
- Visit node $v$: $S[v] \leftarrow \text{MAKE-SET}(v)$
- Return to $u$ from $v$: $S[u] \leftarrow \text{UNION}(S[u], S[v])$
- Encounter edge $(u,v)$ for the second time at $v$: $\text{LCA}(u,v) = \text{FIND-SET}(u)$
• The SP-Bags Algorithm
Shadow Spaces

Each shared-memory location $l$ has two corresponding shadow locations that are updated by the SP-bags algorithm as the Cilk program executes:

- **writer**[$l$]: ID of a procedure that wrote $l$.
- **reader**[$l$]: ID of a procedure that read $l$.
**S-Bags and P-Bags**

$S_F$ contains ID’s of previously executed descendants of $F$ that precede the current thread.

$P_F$ contains ID’s of previously executed descendants of $F$ that operate logically in parallel with the current thread.
The SP-Bags Algorithm

spawn procedure $F$: $S_F \leftarrow \text{MAKE-SET}(F); P_F \leftarrow \emptyset$

sync in a procedure $F$: $S_F \leftarrow \text{UNION}(S_F, P_F); P_F \leftarrow \emptyset$

return from $F'$ to $F$: $P_F \leftarrow \text{UNION}(P_F, S_F')$

write location $l$ by a procedure $F$:
  if $\text{FIND-SET}(\text{reader}[l])$ is a P-bag
    or $\text{FIND-SET}(\text{writer}[l])$ is a P-bag
  then a determinacy race exists
    $\text{writer}[l] \leftarrow F$

read location $l$ by a procedure $F$:
  if $\text{FIND-SET}(\text{writer}[l])$ is a P-bag
  then a determinacy race exists
  if $\text{FIND-SET}(\text{reader}[l])$ is an S-bag
  then $\text{reader}[l] \leftarrow F$
Correctness of SP-Bags

Lemma. Suppose threads $e_1$, $e_2$, and $e_3$ execute in order in the normal serial execution. Then

- $e_1 < e_2$ and $e_1 \parallel e_3$ implies $e_2 \parallel e_3$;
- (Pseudotransitivity)
  $e_1 \parallel e_2$ and $e_2 \parallel e_3$ implies $e_1 \parallel e_3$. 

\[ \\
\]