Goals of the lecture: Weak Conjunctive Predicates

- Logic for global predicates
- Weak conjunctive algorithm

References: Garg and Waldecker 94
Logic for Global Predicates

- Three syntactic categories in the logic - `bool`, `lin` and `form`.

- Syntax: 
  \[ \text{form} ::= A: \text{lin} \mid E: \text{lin} \]
  \[ \text{lin} ::= \Diamond \text{lin} \mid \text{lin} \land \text{lin} \mid \neg \text{lin} \mid \text{bool} \]
  \[ \text{bool} ::= \text{a predicate over a global state} \]

- `bool`: boolean expression defined on a single global state of the system.
  - Example: if the global state has \((x = 3, y = 6)\), then the `bool (x \leq y)` is true.

- `lin` is a temporal formula defined over a global sequence.
  - `bool` is true in \(g\) if it is true in the last state of \(g\).
  - `\Diamond \text{lin}`: there exists a prefix of \(g\) such that \(\text{lin}\) is true for the prefix.
  - `\Box` and `\land`: duals of `\Diamond` and `\land`.

- A `form` is defined for a run and it is simply a `lin` qualified with universal \((A:)\) or existential \((E:)\) quantifier.
Semantics

\[ g \models \text{bool} \quad \text{iff} \quad g_m \models \text{bool}, \quad g_m = \text{last state in } g \]
\[ g \models \neg \text{lin} \quad \text{iff} \quad \neg g \models \text{lin} \]
\[ g \models \Diamond \text{lin} \quad \text{iff} \quad \exists i : g^i \models \text{lin} \]
\[ g \models \text{lin}_1 \rightarrow \text{lin}_2 \quad \text{iff} \quad \exists i < j : g^i \models \text{lin}_1 \land g^j \models \text{lin}_2 \]
\[ r \models \text{A:lin} \quad \text{iff} \quad \forall g : g \in \text{linear}(r) : g \models \text{lin} \]
\[ r \models \text{E:lin} \quad \text{iff} \quad \exists g : g \in \text{linear}(r) : g \models \text{lin} \]

- **strong** predicate: formulas starting with \( \text{A} \)
- **weak** predicate: formulas starting with \( \text{E} \).
Weak Conjunctive Predicates

A weak conjunctive predicate (WCP) is true for a given run if and only if there exists a global sequence consistent with that run in which all conjuncts are true in some global state.

- useful for bad or undesirable predicates
  - Example: the classical mutual exclusion problem.
- detect errors that may be hidden in some run due to race conditions.
Importance of Weak Conjunctive Predicates

- Sufficient for detection of any boolean expression of local predicates.
  - Example $x, y$ and $z$ are in three different processes. Then,
    \[
    E: \diamond even(x) \land ((y < 0) \lor (z > 6))
    \]
    \[
    \equiv
    \]
    \[
    E: \diamond (even(x) \land (y < 0)) \lor E: \diamond (even(x) \land (z > 6))
    \]

- The global predicate is satisfied by only a finite number of possible global states.
  - Example, $E : \diamond (x = y)$, $x$ and $y$ are in different processes. $(x = y)$ is not a *local* predicate. Assume that $x$ and $y$ can only take values \{0, 1\}.

- $A : \Box bool$ can be easily detected, why?
Conditions for Weak Conjunctive Predicates

- $LP_i$: a local predicate in the process $P_i$
- $LP_i(s)$: the predicate $LP_i$ is true in the state $s$.
- $s \in r[i]$: $s$ occurs in the sequence $r[i]$.

 can assume $k \leq n$ because $LP_i \land LP_j$ is just another local predicate if $LP_i$ and $LP_j$ belong to the same process.

**Theorem 1** $E$: $\diamond (LP_1 \land LP_2 \land \ldots \land LP_k)$ is true for a run $r$ iff for all $1 \leq i \leq m \exists s_i \in r[i]$ such that $LP_i$ is true in state $s_i$, and $s_i$ and $s_j$ are incomparable for $i \neq j$. 

A consistent cut
Detection of Weak Conjunctive Predicates: Centralized Algorithm

- One process serves as a checker.
- Other processes: non-checker processes. Each non-checker process maintains its local $lcmvector$ (last causal message vector).
- For $P_j$, $lcmvector[i] \ (i \neq j)$ is the message id of the most recent message from $P_i$ (to anybody) which has a causal relationship to $P_j$.
- $lcmvector[j]$ for the process $P_j$ is the next message id that $P_j$ will use.
**Algorithm: Non-checker processes**

```
var
    lcmvector: array [1..n] of integer;
    init \( \forall i : i \neq id : \text{lcmvector}[i] = 0; \)
    \( \text{lcmvector}[id] = 1; \)
    firstflag: boolean \textbf{init} true;
    local_pred: Boolean_Expression;

\( \square \) For sending \textbf{do}
    send \((\text{prog, lcmvector, \ldots});\)
    \( \text{lcmvector}[id]++ ; \text{firstflag}:=\text{true}; \)

\( \square \) Upon receive \((\text{prog, msg lcmvector, \ldots}) \textbf{do} \)
    \( \forall i : \text{lcmvector}[i]:=\text{max}(\text{lcmvector}[i], \text{msg lcmvector}[i]); \)

\( \square \) Upon \((\text{local_pred = true})\land \text{firstflag} \textbf{do} \)
    \( \text{firstflag} := \text{false}; \)
    send \((\text{dbg, lcmvector})\) to the checker process;
```
Optimization

Sufficient to send the lcm vector once after each message is sent irrespective of the number of messages received.

- \( local(s) \) denote that the local predicate is true in state \( s \).
- \( first(s) \): the local predicate is true for the first time since the most recently sent message.
- \( wcp(s_1, s_2, \ldots, s_m) \): if \( s_1, s_2, \ldots s_m \) are the states in different processes making the \( wcp \) true.

**Theorem 2** \( \exists s_1, \ldots, s_m : wcp(s_1, s_2, \ldots s_m) \iff \langle \exists s'_1, \ldots, s'_m : wcp(s'_1, s'_2, \ldots, s'_m) \land \forall i : 1 \leq i \leq m : first(s'_i) \rangle \)
Complexity

- Space complexity: the array $lcm\text{vector}$ and is $O(n)$.
- Message complexity is $O(m_s)$ where $m_s$ is the number of program messages sent.
  - In addition, program messages have to include time vectors.
- Time complexity
  - detection of local predicates
  - maintain time vectors ($O(n)/message$).
Checker Process

- Incoming debug messages from processes are enqueued in the appropriate queue.
  - assume that the checker process gets its message from any process in FIFO.

**Lemma 1** If the lcmvector at the head of one queue is less than the lcmvector at the head of any other queue, then the smaller lcmvector may be eliminated from further consideration in checking to see if the WCP is satisfied.
Formal Description

var

\( q_1 \ldots q_m \): queue of \( \text{lcm vector} \);

\text{changed, newchanged: set of } \{1,2,\ldots,m\}

\Box \text{Upon } \text{recv}(\text{elem}) \text{ from } P_k \text{ do}

\quad \text{insert}(q_k, \text{elem});

\quad \text{if (head}(q_k) = \text{elem}) \text{ then begin}

\quad \quad \text{changed} := \{k\};

\quad \quad \text{while (changed} \neq \emptyset) \text{ begin}

\quad \quad \quad \text{newchanged} := \{\};

\quad \quad \quad \text{for } i \text{ in changed, and } j \text{ in } \{1,2,\ldots,m\} \text{do}

\quad \quad \quad \quad \text{if } (\neg \text{empty}(q_i) \land \neg \text{empty}(q_j)) \text{ then begin}

\quad \quad \quad \quad \quad \text{begin}

\quad \quad \quad \quad \quad \text{end}

\quad \quad \quad \text{end}

\quad \quad \quad \text{end}

\quad \quad \text{end}

\quad \quad \text{end}

\quad \text{end}
Formal Description [Contd.]

\[
\text{if } \text{head}(q_i) < \text{head}(q_j) \text{ then}
\]
\[
\text{newchanged} := \text{newchanged} \cup \{i\};
\]
\[
\text{if } \text{head}(q_j) < \text{head}(q_i) \text{ then}
\]
\[
\text{newchanged} := \text{newchanged} \cup \{j\};
\]
\[
\text{end}; /* if */
\]
\[
\text{changed} := \text{newchanged};
\]
\[
\text{for } i \text{ in } \text{changed} \text{ do deletehead}(q_i);
\]
\[
\text{end}; /* while */
\]
\[
\text{if } \forall i : \neg \text{empty}(q_i) \text{ then } \text{found} := \text{true};
\]
\[
\text{end}; /* if */
\]

\text{changed}: \text{the set of indices for which the head of the queues have been updated.}

\textbf{Theorem 3} The above algorithm requires at most \(O(m^2p)\) comparisons.
Lower Bounds

Theorem 4 Any algorithm which determines whether there exists a set of incomparable vectors of size $m$ in $m$ chains of size at most $p$, makes at least $pm(m - 1)/2$ comparisons.

Proof:
Case 1: $p = 1$
Case 2: $p > 1$