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:

  form ::= A: lin | E: lin
  lin ::= $\diamond$ lin | lin $\wedge$ lin | $\neg$lin | bool
  bool ::= 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.
  - $\diamond$ lin: there exists a prefix of g such that lin is true for the prefix.
  - $\Box$ and $\vee$: duals of $\diamond$ and $\wedge$.
- A $form$ is defined for a run and it is simply a lin qualified with universal $(\underline{A}:)$ or existential $(\underline{E}:)$ quantifier.

Semantics

$$
\begin{array}{ll}
g \models bool & \textbf{iff } g_m \models bool,\ g_m = \text{last state in } g \\
g \models \neg lin & \textbf{iff } \neg g \models lin \\
g \models \Diamond lin & \textbf{iff } \exists i : g^i \models lin \\
g \models lin_1 \hookrightarrow lin_2 & \textbf{iff } \exists i < j : g^i \models lin_1 \wedge g^j \models lin_2 \\
r \models \underline{\mathsf{A}}{:}lin & \textbf{iff } \forall g : g \in linear(r) : g \models lin \\
r \models \underline{\mathsf{E}}{:}lin & \textbf{iff } \exists g : g \in linear(r) : g \models lin
\end{array}
$$

- $strong$ predicate: formulas starting with $\underline{\mathsf{A}}$
- $weak$ predicate: formulas starting with $\underline{\mathsf{E}}$.

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,
    $\underline{\mathrm{E}}{:}\Diamond even(x) \wedge ((y < 0) \vee (z > 6))$
    $\equiv$
    $\underline{\mathrm{E}}{:}\Diamond(even(x) \wedge (y < 0)) \vee \underline{\mathrm{E}}{:}\Diamond(even(x) \wedge (z > 6))$

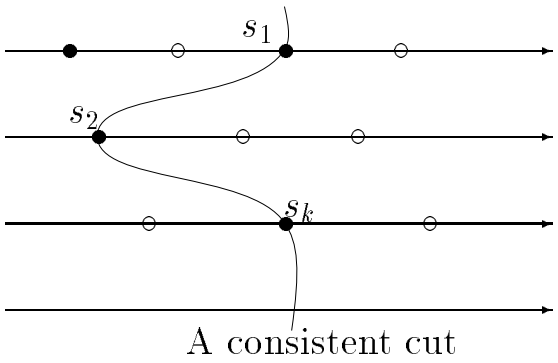
- the global predicate is satisfied by only a finite number of possible global states.
  - Example, $\underline{E} : \Diamond(x = y)$, x and y are in different processes. $(x = y)$ is not a $local$ predicate Assume that x and y can only take values $\{0, 1\}$.
- $\underline{A} : \Box bool$ can be easily 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 $r[i]$.

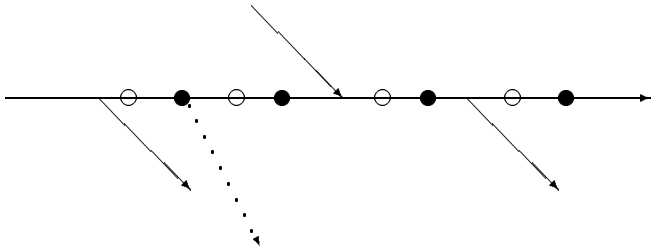
can assume $k \leq n$ because $LP_i \wedge LP_j$ is just another local predicate if LP_i and LP_j belong to the same process.

Theorem 1 *$\underline{E}$: $\diamond(LP_1 \wedge LP_2 \wedge \ldots LP_k)$ 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 $i \neq j$.*



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 $\forall i : i \neq id$:lcmvector$[i] = 0$;

lcmvector[id] $= 1$;

firstflag: boolean **init** true;

local_pred: Boolean_Expression;

□ **For** sending **do**

send (prog, lcmvector, . . .);

lcmvector[id]++ ; firstflag:=true;

□ **Upon** receive (prog, msg_lcmvector, . . .) **do**

$\forall i$:lcmvector[i]:=max(lcmvector[i], msg_lcmvector[i]);

□ **Upon** (local_pred = true)$\wedge$ firstflag **do**

firstflag := false;

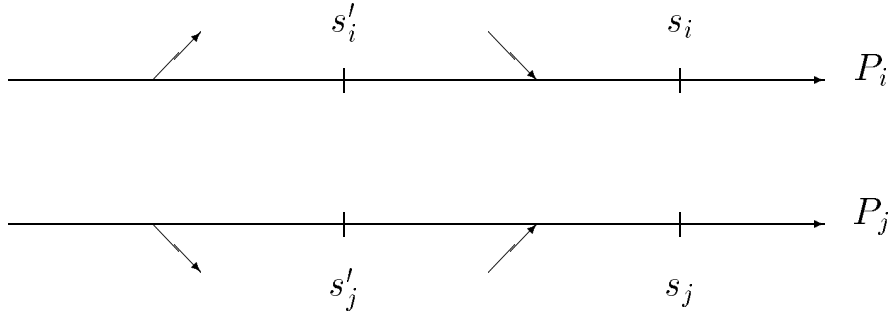
send (dbg, lcmvector) to the checker process;

Optimization

Sufficient to send the lcmvector 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.
- $wcp(s_1, s_2, ..., s_m)$: if $s_1, s_2, ...s_m$ are the states in different processes making the wcp true.

Theorem 2 $\exists s_1, ..., s_m : wcp(s_1, s_2, ...s_m) \Leftrightarrow$
$\langle \exists s'_1, .., s'_m : wcp(s'_1, s'_2, ..., s'_m) \wedge \forall i : 1 \leq i \leq m : first(s'_i) \rangle$



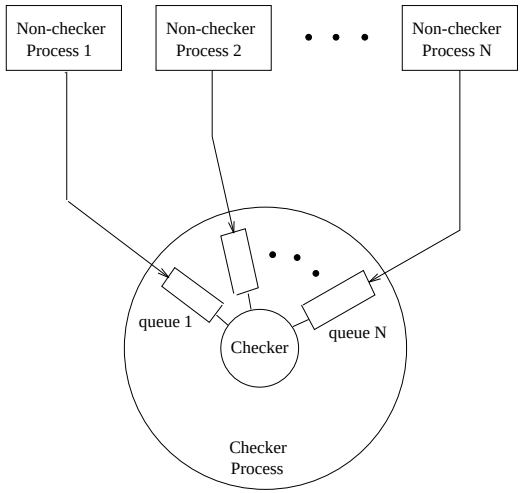
Complexity

- Space complexity: the array $lcmvector$ 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 $(O(n)/\text{message})$.

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 lcmvector;
changed, newchanged: set of {1,2,...,m}
□ **Upon** recv(elem) from P_k **do**
    insert(q_k, elem);
    if (head(q_k) = elem) **then begin**
        changed := { k };
        while (changed $\neq \phi$) **begin**
            newchanged := {};
            for i **in** changed, and j **in** {1,2,...,m }**do**
            if $(\neg empty(q_i) \wedge \neg empty(q_j))$ **then**
            begin

Formal Description [Contd.]

```
            if head(q_i) < head(q_j) then
                newchanged:=newchanged ∪ {i};
            if head(q_j) < head(q_i) then
                newchanged:=newchanged ∪ {j};
        end; /* if */
        changed := newchanged;
        for i in changed do deletehead(q_i);
    end;/* while */
    if ∀i : ¬empty(q_i) then found:=true;
end; /* if */
```

$changed$: the set of indices for which the head of the queues have been updated.

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: $p > 1$