MaC
Monitoring and Checking at Runtime
(Continue)

Presented By
Usa Sammapun
CIS 700 Oct 12, 2005
Recap: MaC

- Runtime verification technique
  - Ensures the current program execution follows its formal requirements at run-time
MaC Verifier and Language

MaC Verification System

Program → MaC Specification → MaC Compiler → Instrumented Program

MaC Specification

- PEDL
- MEDL
- SADL

MaC Compiler

Instrumented Program

Event Recognizer → Checker → Injector

Event Recognizer

Variable update, method call/return to EVENTS and CONDITIONS

Checker

System Properties using EVENTS and CONDITIONS

Injector

Where/when to steer

Feedback

Variable update, method call/return to EVENTS and CONDITIONS

System Properties using EVENTS and CONDITIONS

Where/when to steer

Event

Conditions

Violations
Events

- **e** - variable update, start/end method
- **e1 || e2** - or
- **e1 && e2** - and
- **start(c)** - instant when condition c becomes true
- **end(c)** - instant when condition c becomes false
- **e when c** - e occurs when condition c is true

**Alarms**: events that must never occur
Conditions

- Conditions interpreted over 3 values: true, false and undefined.

- c - boolean expression
- !c - not c
- c₁ || c₂ - or
- c₁ && c₂ - and
- c₁ -> c₂ - imply
- defined(c) - true when c is defined
- [e₁, e₂) - interval

Safety Properties: conditions that must always hold true
Current Work

- Timing properties: \([e_1, e_2] \leq d\) \[e_1, e_2] < d \[e_1, e_2] = d\)

- Regular expressions

- Probabilistic properties

- Dynamic MaC
Regular Expressions in MEDL

► MEDL is based on temporal logic

► Regular expressions (RE) may be better
  – Engineers understand them
  – More concise than TL for temporal ordering

► RE ranges over MaC events
  – event a,b,c
  – a.b*.c
Challenges

► When to accept several possible inputs (ab*c*)
  – Shortest input
  – Longest input
  – All input

► Identify which events are relevant

► Overlapping RE

► Simultaneous events
Identify which events are relevant

- An unexpected event fails the RE check
- Trace may contain “irrelevant” events, which should not make RE fail
Example: no sends after read

open.send*.read*.close

**Which traces should be accepted or rejected?**
- open.send.read.close accept
- open.send.read.send.close reject
- open.send.send.read continue
- open.send.delete ?reject
- open.send.chdir.close ?accept

RE fileaccess{open,send,close,delete} =
open.send*.read*.close
MaC with Regular Expressions

- **Regular expression over events**
  - Statement: $RE R \{\tilde{E}\} = \langle R \rangle$, 
  - Grammar of $R$: $R ::= e | R.R | R+R | R*$
  - Relevant set $\{\tilde{E}\}$: contribute to RE failure

- **RE are neither events nor conditions**
  - Events associated with RE $R$:
    - startR($R$), success($R$), fail($R$)

- alarm badAccess = fail(fileaccess)
Overlapping RE

- Property: open.send*.read*.close
- Trace:
  - Actual: open open send read send read
  - We see: open open send read send read
- Cannot distinguish between two overlapping instances; events miss attribution
  - What is the right way to index events?
Simultaneous Events

- Checker operates on a stream of observations
  - Observations are primitive events that reflect change of system state

- One primitive event can trigger different other events

- What if those events are in the same RE
  - $a \cdot (a \parallel b) \cdot b$
  - at state $i$, $a$ occurs, then $(a \parallel b)$ also occurs
  - How do we order $a$ and $(a \parallel b)$
Probabilistic Properties

- Probability calculation
  - Numerical technique
  - Statistical technique
    1. Simulate
    2. Collect several samples
    3. Estimate probabilities
Statistical Technique

usually, we 1) execute for X times, 2) use them as samples, and 3) estimate probabilities

```
<table>
<thead>
<tr>
<th>task start</th>
<th>finish in 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>task start</td>
<td>not finish in 100</td>
</tr>
<tr>
<td>task start</td>
<td>finish in 100</td>
</tr>
<tr>
<td>task start</td>
<td>finish in 100</td>
</tr>
<tr>
<td>task start</td>
<td>finish in 100</td>
</tr>
</tbody>
</table>
```
1. Simulate and 2. Collect Sample

- runtime verification – only one execution path
MaC Probabilistic Properties

- **Experiment**
  - An element that indicates a sub-path
    - $e_{exp}$ (previous example: *task start*)
    - $c_{exp}$

- **Probabilistic event**
  - $e\ prob(\oplus p, e_{exp})$

- **Probabilistic condition**
  - $c\ prob(\oplus p, c_{exp})$
Example

- A soft real-time task must not miss a deadline of 100 time units with probability $\geq 0.2$

  
  \[
  \text{event missDeadline} = \text{end}([\text{startT, endT}]_{\leq 100})
  \]

  \[
  \text{alarm soft\_rt\_task} = \text{missDeadline} \quad \text{prob}(\geq 0.2, \text{startT})
  \]

- A car velocity must be $< 50$mph with prob $\geq 0.9$ in work zones

  
  \[
  \text{property speed} = (v < 50) \quad \text{prob}(\geq 0.9, \text{work\_zone})
  \]
3. Estimating Probability

- Estimate probability from program execution
  - compute experimental probability $p'_{\text{condition}}$ and $p'_{\text{event}}$
  - **Condition**: $c \ \text{prob}( < p, c_{\text{exp}})$
  - **Event**: $e \ \text{prob}( < p, e_{\text{exp}})$

\[
p'_{\text{condition}} = \frac{|S_i \text{ s.t. } c = \text{true}|}{|S_i \text{ s.t. } c_{\text{exp}} = \text{true}|}
\]
\[
p'_{\text{event}} = \frac{|\text{occurrences of } e|}{|\text{occurrences of } e_{\text{exp}}|}
\]

- A car velocity must be $< 50$mph with prob $\geq 0.9$ in work zones
  $\rightarrow (v < 50) \ \text{prob}(\geq 0.9, \text{work\_zone})$

\[
p'_{\text{condition}} = \frac{\#\text{states: } (v < 50) = \text{true}}{\#\text{states: } \text{work\_zone} = \text{true}}
\]
Example

- task must not miss a deadline of 100 time units with probability $\geq 0.2$
  - alarm soft_rt_task = missDeadline prob($\geq 0.2$, startT)

$$p'_{event} = \frac{\# \text{ miss deadline events}}{\# \text{ task start events}}$$

- $\# \text{ miss deadline events} = 40$
- $\# \text{ startT (task start events)} = 150$
- $p' = 40 / 150 = 0.267$
Statistical Hypothesis Testing

► Given
  – Probability estimation
  – Confidence interval (CI) e.g. CI = 95%

► Statistical Hypothesis Testing
  – Satisfied
  – Not satisfied
  – Need more sample
Probability Estimation: Z-Score

- Use z-score to calculate how far apart $p$ and $p'$ are
  \[ z = \frac{p' - p}{\sqrt{\frac{p(1-p)}{n}}} \]

  For event, $n = |\text{occurrences of } e_{\text{exp}}|$
  For condition, $n = |S_i \text{ s.t. } c_{\text{exp}} = \text{true}|$

- Sign of $z$ says which direction
  + $z$ says $p' > p$
  - $z$ says $p' < p$

- Value of $z$ says how far apart $p'$ and $p$

- Task must not miss a deadline of 100 time units with probability $\geq 0.2$
  \[ p = 0.2 \quad p' = 0.267 \]
  \[ z_{p'} = +2.05 \]
Given confidence interval (CI)
- We calculate z-score $z^*$ for CI
  (e.g. CI = 95% has $z^* = 1.96$)

Decide: $\text{alarm soft\_rt\_task} = \text{missDeadline prob(}\geq 0.2, \text{startT)}$
- no alarm: $zp' < -z^*$ [ means $p' < p$ with confidence CI ]
- raise alarm: $zp' > z^*$ [ means $p' > p$ with confidence CI ]
- more sample: $-z^* < zp' < z^*$ [ means $p' \approx p$, either action wouldn’t cause serious error ]

<table>
<thead>
<tr>
<th>No alarm</th>
<th>more sample</th>
<th>Raise alarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>-z* = -1.96</td>
<td>$zp = 0$ $p = 0.2$</td>
<td>$z^* = 1.96$ $zp' = 2.05$ $p' = 0.267$</td>
</tr>
</tbody>
</table>
Dynamic MaC

► From fixed to dynamic object sets

► What if tasks can be added dynamically?
  – The set of events and conditions changes dynamically
    • Events and conditions are parameterized

► Example: Client

```
event clientReq(ID i) = startM(Client.request()) { clientReq.i = Client.id; }
condition clientValid(ID i) = [clientReq(i), clientDropped(i)];
```

► Special event that add or remove an object in the object set