MaC
Monitoring and Checking at Runtime (Continue)

Presented By

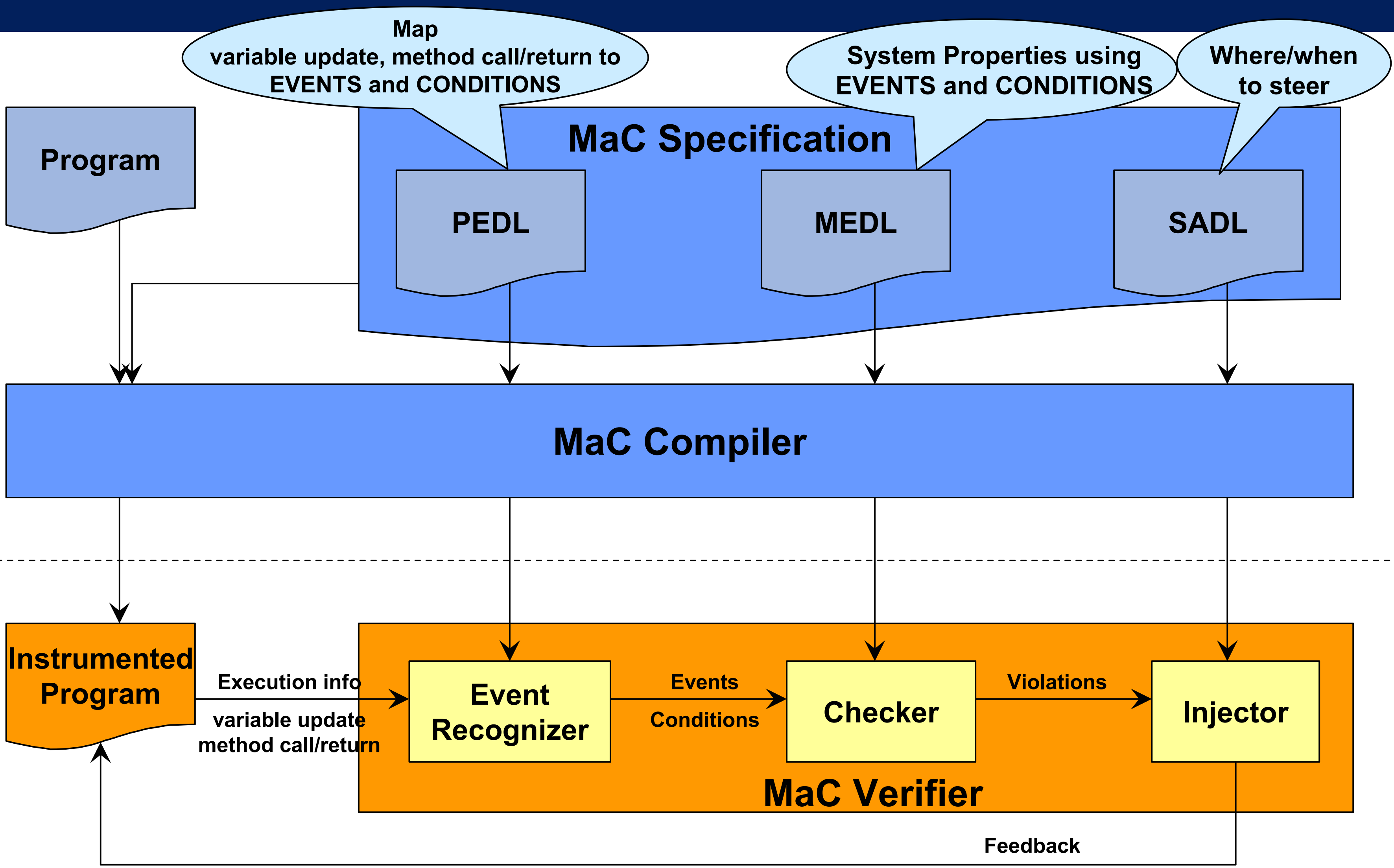
Usa Sammapun

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



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

- ***<u>Alarms</u>*:** events that must never occur

Penn
University *of* Pennsylvania

Conditions

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

- **c** - boolean expression
- **!c** - not c
- **c_1 || c_2** - or
- **c_1 && c_2** - and
- **c_1 -> c_2** - imply
- **defined(c)** - true when c is defined
- **[e_1, e_2)** - 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: $RE\ \ R\ \ \{\bar{E}\} = < R >,$
  - Grammar of R: $R ::= e \mid R.R \mid R{+}R \mid R*$
  - Relevant set $\{\bar{E}\}$: 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)

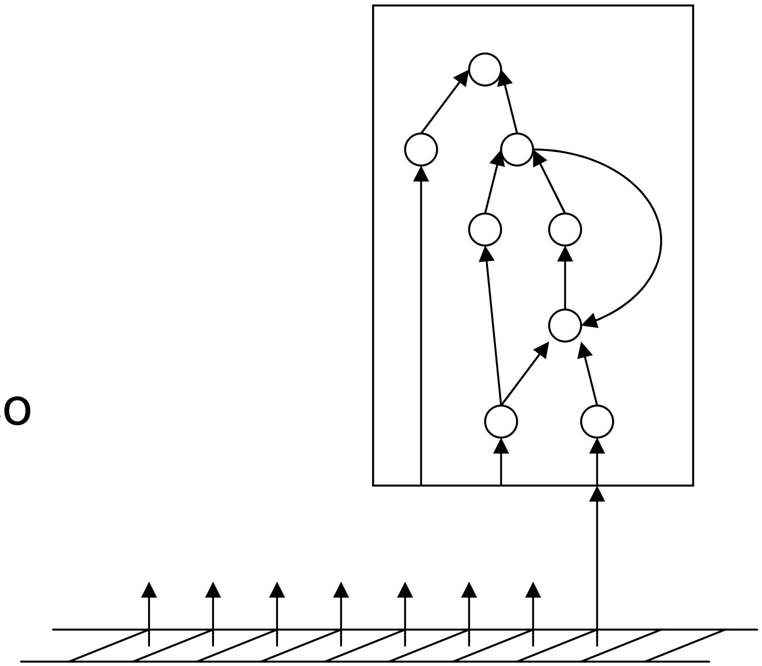
Penn
University *of* Pennsylvania

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?

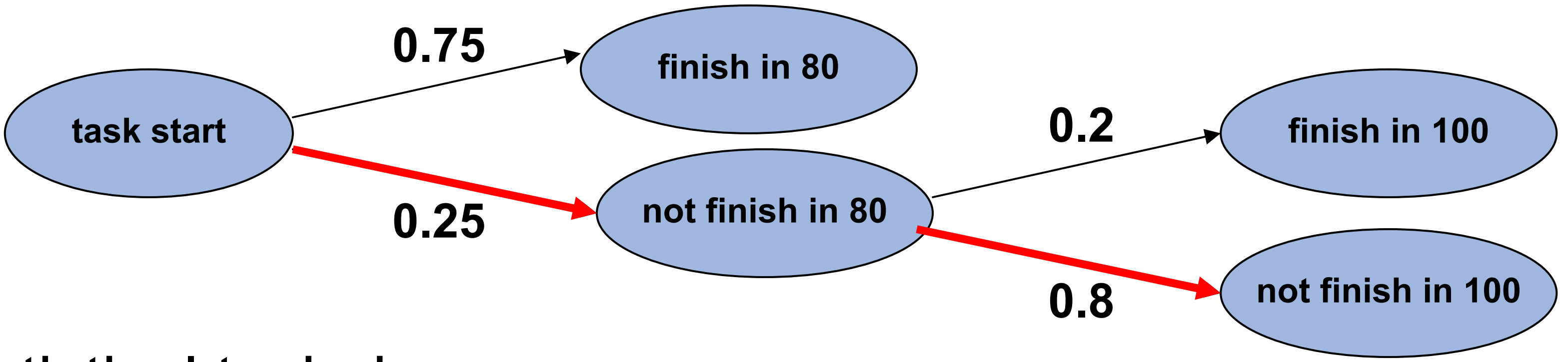
Simultaneous Events

- 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 the same RE
  - *a . (a || b) . b*
  - at state *i*, *a* occurs, then *(a || b)* also occurs
  - How do we order *a* and *(a || b)*



Probabilistic Properties

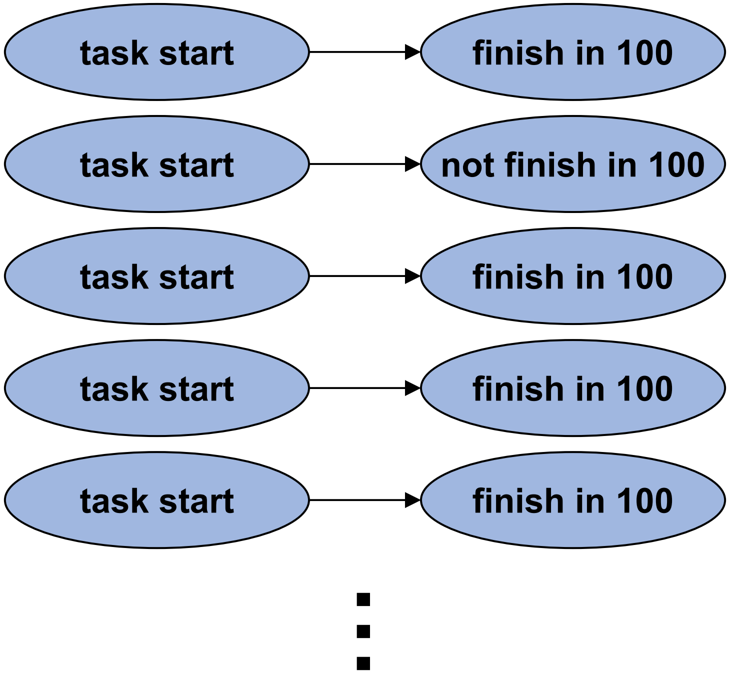
- Probability calculation
  - Numerical technique



  - Statistical technique
    1. Simulate
    2. Collect several samples
    3. Estimate probabilities

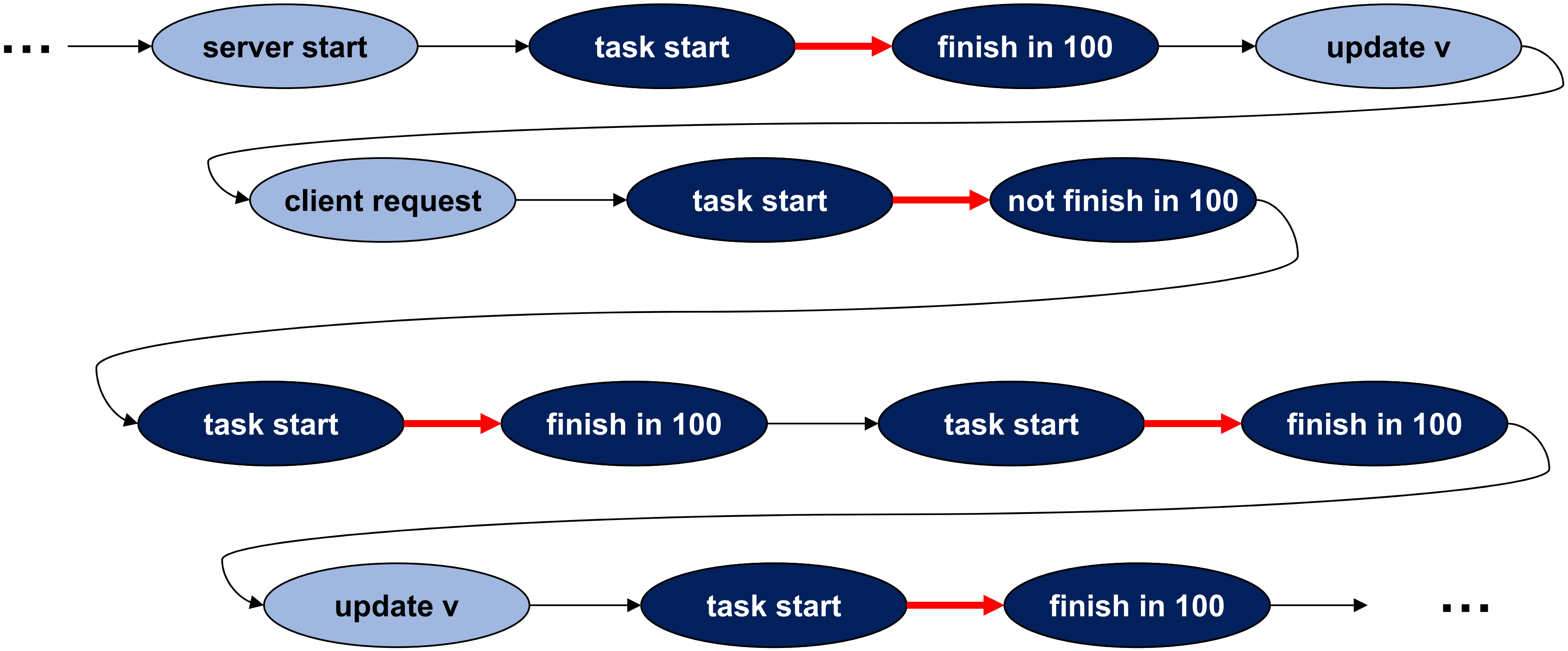
Statistical Technique

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



1. Simulate and 2. Collect Sample

- runtime verification – only one execution path



Penn
University of Pennsylvania

MaC Probabilistic Properties

- Experiment
  - An element that indicates a sub-path
    - e_{exp} *(previous example:* ***task start*** *)*
    - c_{exp}
- Probabilistic event $\odot = \{<, >, \leq, \geq, =, \neq\}$
  - $e \; prob(\odot p, e_{exp})$
- Probabilistic condition
  - $c \; prob(\odot p, c_{exp})$

Penn
UNIVERSITY of PENNSYLVANIA

Example

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

  event missDeadline = end([startT,endT)$_{\{\leq 100\}}$)
  alarm soft_rt_task = missDeadline prob(≥ 0.2, startT)

- A car velocity must be < 50mph with prob ≥ 0.9 in work zones

  property speed = (v < 50) prob(≥ 0.9, work_zone)

3. Estimating Probability

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

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

- A car velocity must be < 50mph with prob ≥ 0.9 in work zones
  → (v < 50) prob(≥ 0.9, work_zone)

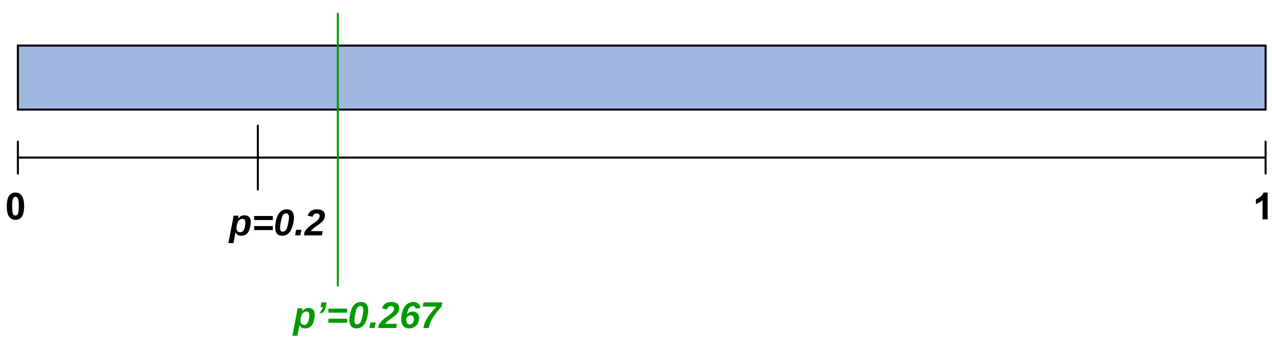
$$p'_{condition} = \frac{\#\text{states: } (v < 50) = \text{true}}{\#\text{states: work\_zone} = \text{true}}$$

Example

- task must not miss a deadline of 100 time units with probability ≥ 0.2
  - → **alarm soft_rt_task = missDeadline prob(≥ 0.2, startT)**

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

- → **# miss deadline events = 40**
- → **# 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

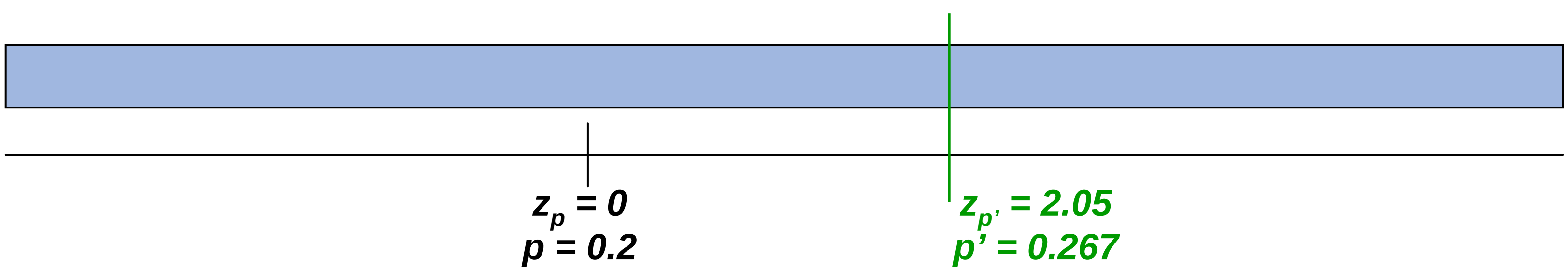
- Use z-score to calculate how far apart *p* and *p’* are

$$z = \frac{p' - p}{\sqrt{\frac{p(1-p)}{n}}}$$

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

- 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 ≥ 0.2
  - → **p = 0.2 p’ = 0.267**
  - → $z_{p'} = + 2.05$

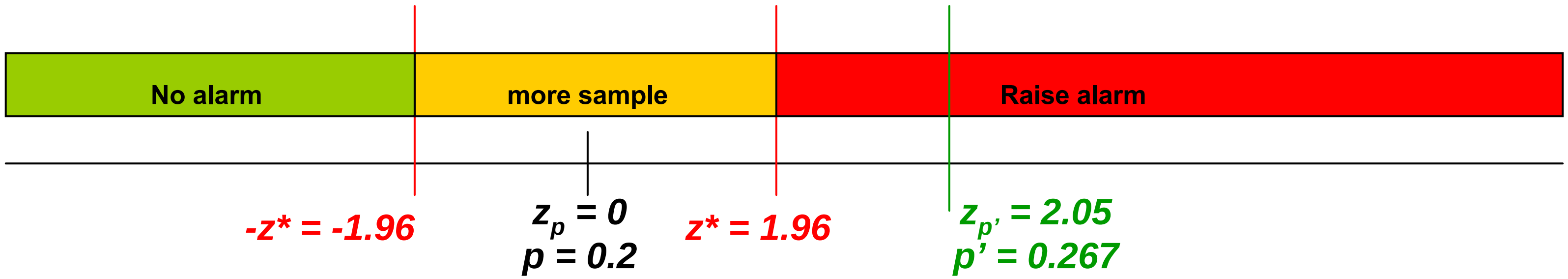


Continue…

- Given confidence interval (CI)
  - We calculate z-score z^* for CI

  (e.g. *CI = 95%* has $z^* = 1.96$)

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



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

Penn UNIVERSITY of PENNSYLVANIA