# CS 5090: Software Fault Tolerance – Characterizing Specifications

Ali Ebnenasir
Department of Computer Science
Michigan Technological University



MichiganTech

## Acknowledgement

- The contents of this Lecture are adapted from the following paper:

"**Defining Liveness**"
By: Bowen Alpern and Fred B. Schneider



MichiganTech

## Motivation

- Need a formal characterization for specifications
- Identify classes of specifications that can be captured by the formal characterization

MichiganTech

## Outline

- Basic Concepts
- Formal Definition of Safety
- Formal Definition of Liveness



MichiganTech

## Computations

- **Computation** of concurrent programs:
  An infinite sequence of states
  $\sigma = < s_1, s_2, s_3 , \dots >$
- **Transition** from $s_i$ to $s_{i+1}$ by a single atomic action
- **Stuttering**: Repeat the final state of a terminating computation to generate an infinite computation
- All program computations are infinite



MichiganTech

## Properties

- A **property** is a set of infinite sequence of states
  $\sigma_1 = < s_1, s_2, s_3 , \dots >$
  $\sigma_2 = < s'_1, s'_2, s'_3 , \dots >$
  …
  $\sigma_n = < s''_1, s''_2, s''_3 , \dots >$
- A property $\mathbb{P}$ **holds** for a program p (resp., p satisfies $\mathbb{P}$) **iff** all computations of p are contained in $\mathbb{P}$

## Classes of Properties

- **Safety**: nothing bad ever happens
  - Examples:
    - mutual exclusion: no more than one process will ever enter the critical section
    - Deadlock freedom
    - Partial correctness: violating a postcondition
    - *What is a safety property for a FIFO/LIFO?*
- **Liveness**: something good will eventually happen
  - Each trying process will eventually enter its critical section



## Formalizing Safety Properties

- S: set of program states
- $S^{\omega}$: set of infinite sequences of program states
  - Each element is a program computation
- $S^{*}$: set of finite sequences of program states
  - Each element is a **partial computation**; i.e., **computation prefix**
  - $\sigma_i$ denotes the computation prefix up to state $s_i$ in $\sigma$
- **$\sigma$ |=** $\mathbb{P}$ denotes that computation $\sigma$ is in property $\mathbb{P}$



## Formalizing Safety Properties

- If a safety property $\mathbb{P}$ does not hold for a computation $\sigma$, then in some finite state $s_i$ some bad thing happens
- If this bad thing happens, then the safety has been violated; no way to fix it

MichiganTech

## Formalizing Safety Properties

- **Definition**:

$\forall \sigma: \sigma \in S^{\omega} : (\sigma \not\models \mathbb{P} \Rightarrow$
$(\exists i: 0 \leq i : \forall \beta: \beta \in S^{\omega} : \sigma_i\beta \not\models \mathbb{P}) )$

- Observations:
  - There is a discrete point of time when the "bad thing" happens
  - Prohibits bad things from occurring; i.e., if it occurs there is no suffix that can fix it



MichiganTech

## Formalizing Liveness Properties

- Liveness: some good things eventually happen in a program computation
  - Examples:
    - **Starvation-freedom**: a process makes progress infinitely often
    - **Termination**: a program eventually terminates
    - **Guaranteed service:** every received request will eventually be serviced
- Any partial computation can be fixed to meet liveness. Why?
  - I.e., there is always hope that something good will happen



MichiganTech

## Formalizing Liveness Properties

- A partial computation $\alpha$ is **live** for a property $\mathbb{P}$ **iff** there exists a sequence of states $\beta$ s.t. $\alpha\beta \models \mathbb{P}$
- Liveness property: every partial computation is live
- $\mathbb{P}$ is a **liveness property** iff

  $\forall \alpha: \alpha \in S^{*} : (\exists \beta: \beta \in S^{\omega} : \alpha\beta \models \mathbb{P})$
- The above definition is the most general definition of liveness. Why?

## More Restrictive Liveness Definitions

- Uniform liveness:

$$\exists \beta : \beta \in S^{\omega} : (\forall \alpha : \alpha \in S^{*} : \alpha\beta \models \mathbb{P})$$

$\mathbb{P}$ is a uniform liveness property

- Absolute liveness:

$$(\exists \gamma : \gamma \in S^{\omega} : \gamma \models \mathbb{P}) \wedge (\forall \beta : \beta \in S^{\omega} : (\beta \models \mathbb{P}) \Rightarrow (\forall \alpha : \alpha \in S^{*} : \alpha\beta \models \mathbb{P}))$$

$\mathbb{P}$ is an absolute liveness property

- What is the relation between absolute and uniform liveness?



## Examples

- Leads-to properties:
  - A received request will eventually be serviced
  - $P \rightarrow Q$: If P holds in a state, then some state will eventually be reached where Q holds
- Does absolute liveness capture leads-to properties?



## Examples

- Consider the following property:
  - If P holds initially then eventually Q will become true and will continuously remain true.
  - Otherwise, Q will never become true
- Is this a liveness property? What is the good thing that should happen?
- Does uniform liveness capture this property?

MichiganTech

## Examples

- How about P $\mathcal{U}$ Q (P until Q)?
- Is the above property a liveness property?
- Is it a safety property?

Every property is an intersection of a safety and a liveness property