CS 233
SOFTWARE DEVELOPMENT AND VALIDATION FOR MEDICAL CYBER PHYSICAL SYSTEMS

智能医疗仪器软件的设计与验证

1

CS 233
SOFTWARE DEVELOPMENT AND VALIDATION FOR MEDICAL CYBER PHYSICAL SYSTEMS

智能医疗仪器软件的设计与验证

2

CYBER-PHYSICAL SYSTEMS(CPS)

- A mechanism that is monitored and controlled by computer-based algorithms
- Two trends
  - Autonomy
  - Connectivity

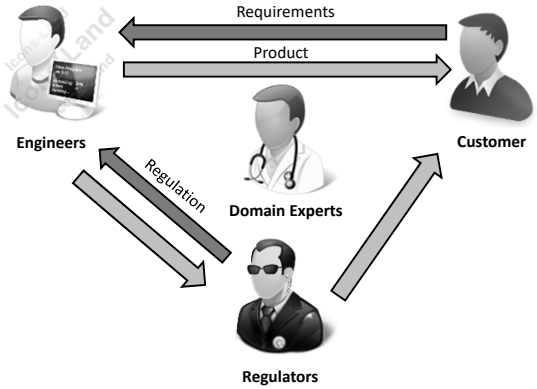
Automotive

Industrial Control

Medical

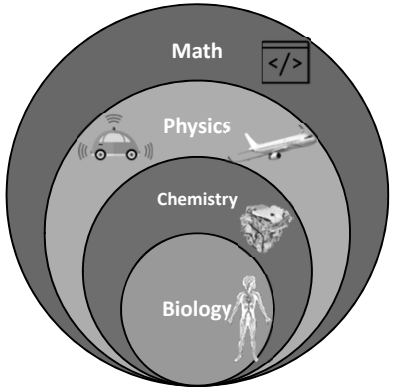
3

STAKEHOLDERS FOR CPS



4

Relationships among domains

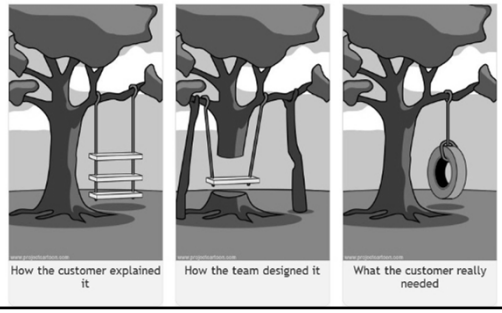


- The domains with more constraints are also less understood
- Software is in math domain
  - Amazon negative quantity bug
- The domain specific constraints have to be encoded or considered in the software

5

Challenge

- Impossible to list the constraints explicitly
- How to effectively communicate with domain experts?



6

"The Expert"

7

Angry, right?

- People look stupid in the video because the constraints in their field are known to everyone
- What if we change the subject to medical?
- We will all experience some of the roles in this video at some time…

8

CS 233
Software Development and Validation for Medical Cyber Physical Systems

智能医疗仪器软件的设计与验证

Medical Devices

FDA Definition：

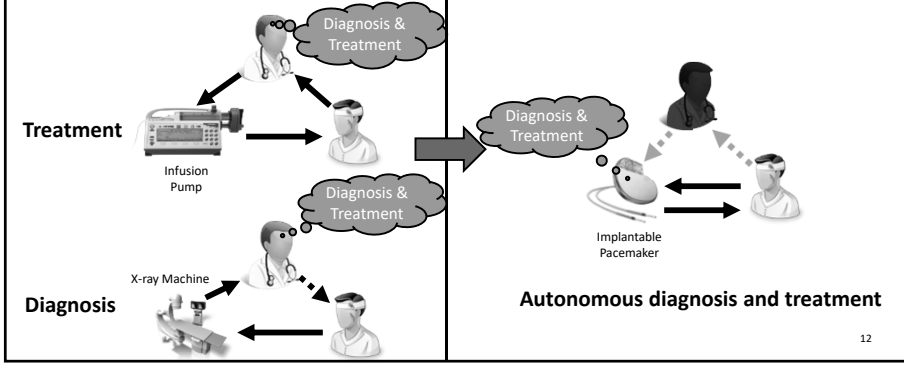
An instrument, apparatus, implement, machine, contrivance, implant, in vitro reagent, or other similar or related article, including a component part, or accessory which is: recognized in the official National Formulary, or the United States Pharmacopoeia, or any supplement to them, intended for use in the diagnosis of disease or other conditions, or in the cure, mitigation, treatment, or prevention of disease, in man or other animals, or intended to affect the structure or any function of the body of man or other animals, and which does not achieve any of it's primary intended purposes through chemical action within or on the body of man or other animals and which is not dependent upon being metabolized for the achievement of any of its primary intended purposes.

Medical Device Classifications

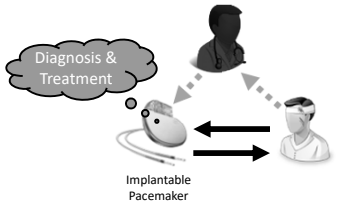
- Classified according to its risk
  - Class 1: present minimal potential harm to the user
    - i.e. Elastic bandages, dental floss
  - Class 2: more complex with higher risk
    - i.e. pregnancy testing kits and powered wheelchairs
  - Class 3: present a potential unreasonable risk of illness or injury to the patient
    - i.e. replacement heart valves, implantable pacemakers

Medical Cyber-Physical Systems

- Traditional Medical Devices
- Medical Cyber-Physical Systems



Challenges for Medical CPS



- Variability
  - Everyone is different
  - Infeasible to consider all possible situations during development
- Low observability
  - Less invasiveness
  - More ambiguities

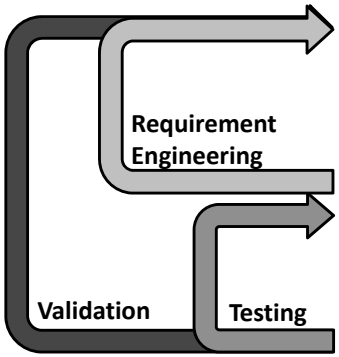
13

CS 233
Software Development and Validation for Medical Cyber Physical Systems

智能医疗仪器软件的设计与验证

14

Software Development Process



- Requirements
  - i.e. the vehicle should not crash with any other objects
- Specifications
  - i.e. when the vehicle detects an object on its path, apply brake
- Implementation

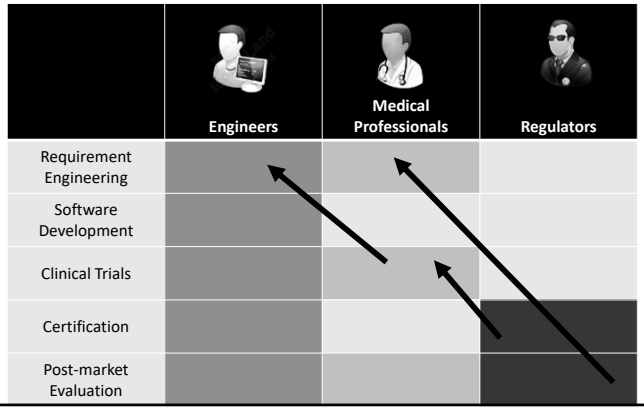
15

Physiological Requirements

- Does the device always deliver therapy when needed?
- Does the device always not deliver therapy when unnecessary?
- Behavior Coverage
- Physiological Context

16

Medical Device Life Cycle - Traditional



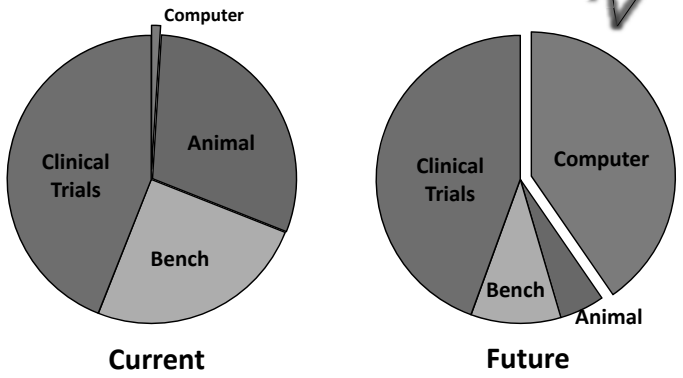
17

Medical Device Recalls

- 2011-2015: **More than 1,400,000 medical devices were recalled due to software malfunctions**
  - 190,000 of which can cause serious injury to the patients*
- Can we find these bugs earlier?

* Software-Related Recalls of Health Information Technology and Other Medical Devices: Implications for FDA Regulation of Digital Health

Validation of Medical Devices

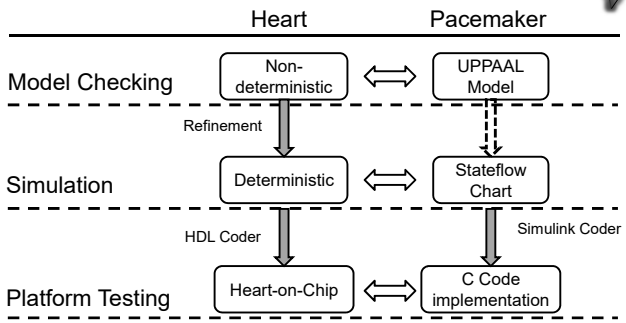


19

Let the models catch the bugs before the patients do

20

VERIFIED MODELS TO VERIFIED CODE



21

MEDICAL DEVICE LIFE CYCLE – MODEL-BASED

	Engineers	Models	Medical Professionals	Regulators
Requirement Engineering		Counter-examples		
Software Development		Closed-loop Simulation/Testing		
Clinical Trials		In-silico Pre-clinical trials		
Certification		Model-based safety evidence		
Post-market Evaluation		Reconstruct physiological conditions		

Early involvement and early feedback

22

CURRICULUM

- Develop a pacemaker algorithm and prove its safety and efficacy
  - Phase 1: Understand and model cardiac electrophysiology
  - Phase 2: Develop pacemaker software models
  - Midterm exam
  - Phase 3: Model-based software validation
  - Phase 4: Medical device certification
  - Final Presentation & Report

23

PHASE 1: UNDERSTAND AND MODEL CARDIAC ELECTROPHYSIOLOGY

- Physiological knowledge
  - Why do we need a pacemaker?
  - What are the physiological conditions that a pacemaker may encounter?
  - What is considered as an appropriate therapy?
- Modeling
  - What are the different roles of models?
  - What is considered as a good model for an application?
  - Develop models of electrophysiological conditions

24

PHASE 2: DEVELOP PACEMAKER SOFTWARE MODELS

- Learn the general principles of pacemaker algorithms
- Develop pacemaker models using existing pacemaker specifications
  - Extra credit: creative improvements on existing specifications
- Simulate the pacemaker model with heart models

25

MID-TERM EXAM

26

PHASE 3: MODEL-BASED SOFTWARE VALIDATION

- Model-based testing of pacemaker algorithm
- Model checking of pacemaker model
  - How to use the heart models to cover all possible heart conditions that a pacemaker may encounter?
  - How to interpret the counter-examples?
    - Valid vs. invalid
    - Appropriate vs. inappropriate
- How to maintain traceability from model to code?

27

PHASE 4: MEDICAL DEVICE CERTIFICATION

- Why do we need certification?
- What is considered as sufficient evidence?
- How to organize evidence into a convincing argument?

28

Final Presentation & Report

- I will serve as medical expert and regulator
- You need to demonstrate that your pacemaker algorithm is safe and effective
- In ENGLISH!

29

What you will learn during this course

- State of the art software development and validation
- Experience the whole software life cycle
- Limitations of different techniques

- What are real-world problems?
- Systems Engineering
- The power of inter-disciplinary

- Modeling methodologies
- Validation techniques
- Effective communication

30

What are real-world problems?

- There are no perfect solutions!
  - There are trade-offs
    - Game among stakeholders
  - There are legacy solutions
    - Improve on existing solutions rather than starting from scratch

31

System Engineering

- 1 million settlers on mars

Elon Musk

	Founded	Status	Valuation or market cap	Industry
SPACEX	2002	Private	$25B	Aerospace
TESLA	2003	Public	$49B	Electric vehicles
NEURALINK	2016	Private	$47M	Neurotechnology
THE BORING COMPANY	2016	Private	Unknown	Tunnel infrastructure

PitchBook

32

SpaceX

- Reduce the transportation cost per person from 10 billion dollars to 200 thousands dollars (1/50,000)
  - Recycle the rockets
  - Fuel the spaceship in orbit
  - Produce fuel on mars

33

The power of inter-disciplinary

- If you want something extraordinary, you have two paths:
  1. Become the best at one specific thing.
  2. Become very good (top 25%) at two or more things.

Scott Adams

34

Logistics

- Homework: 4*10%
- Midterm: 20%
- Final Presentation: 40%
- Office hour: TBD

35

Reading Material

- Electrical activities of the heart (35 pg)

36

Lecture 2: Cardiac Electrophysiology

Announcement

- Piazza: https://piazza.com/shanghaitech.edu.cn/fall2019/cs233
- Office hour
  - Wed 5pm-6pm, 2-302A, starting next week.
- TA: Guangyao Chen
  - Office hour: Fri 5pm-6pm, 1B-101, starting next week.
  - chengy2@shanghaitech.edu.cn

Terminologies

For our interest in medical device software in this text, let us focus on the first sentence. The FDA is responsible for "assuring the safety, efficacy, and security" of medical devices. To expand slightly on this, the FDA's responsibility is to assure:

- *Efficacy:* The device is effective in achieving the claims that are made for its intended and expected use;
- *Safety:* The device does not expose the patient, user, or other bystanders to undue risk from its use;
- *Security:* The device has been designed to protect patients, users, and bystanders from both unintentional and malicious misuse of the device.

What do we want the device to achieve?

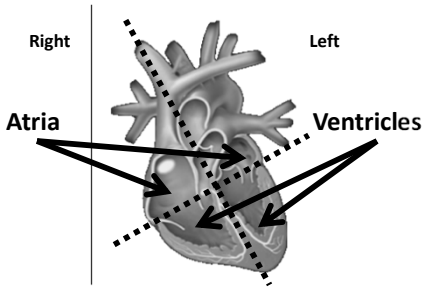
- Let's define $V(H)$ as the quantification of how good the heart condition H is.
- We want to achieve: for all heart conditions H :

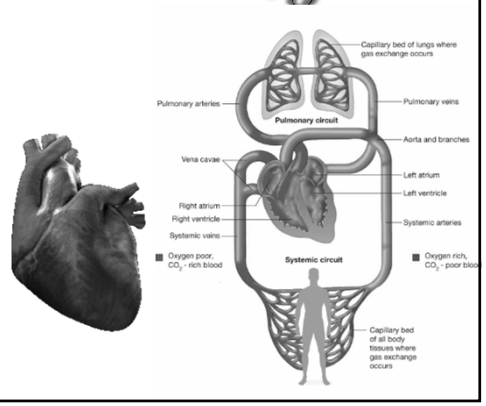
$$V(H + device) \geq V(H)$$

  - Efficacy: For "intended use", improve the heart condition as intended
  - Safety: For other heart conditions, don't make it worse!
- In this lecture, we want to identify the function $V()$

Heart Anatomy

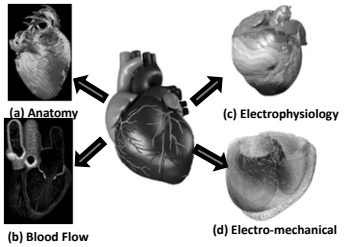
- 4 heart chambers





From different perspectives

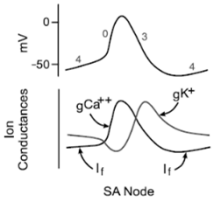
- Different level of abstractions
- Serve different purposes

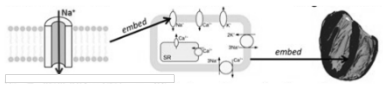


6

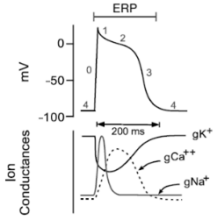
Tissue Types

- Pacemaker Tissue
  - Periodically self depolarizes



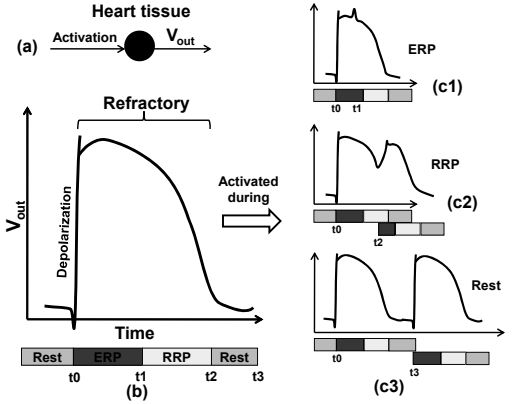


- Muscle Tissue (Myocardium)
  - Triggered depolarization
    - By nearby tissue
    - By external voltage



7

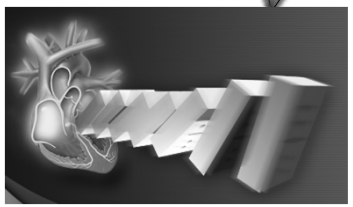
Refractory Periods



- ERP: Effective Refractory Period
- RRP: Relative Refractory Period
- A cell needs to "recover" before it can be depolarized again
- A cell can be partially depolarized
  - Shortened ERP
  - Increased conduction delay

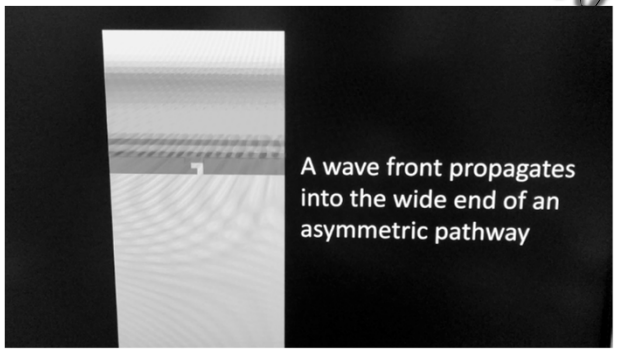
Conduction of Electrical Signals

- Cells are connected via gap junctions etc
- Once the current exceed the activation threshold, the cell is activated



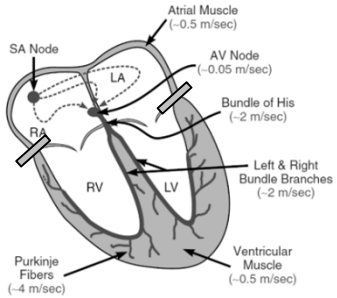
9

Uni-directional Blocks



10

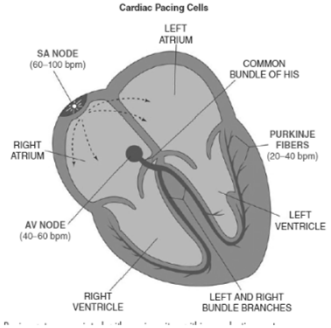
Electrical Conduction System of the Heart



- Muscles have slow conduction velocity
- AV node has very long delay
- Purkinje fibers has fast conduction velocity to ensure simultaneous ventricular muscle contractions

11

Other Pacemaker Tissue

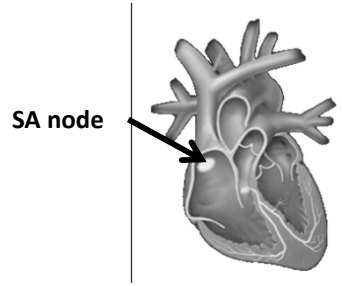


- In normal heart, electrical signal generation is dominated by the SA node
- Other tissue is depolarized by signals from the SA node before they self-depolarize
- There are exceptions

12

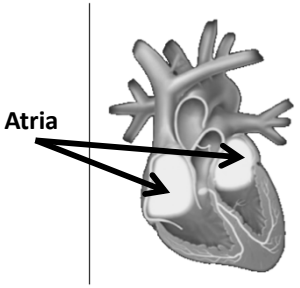
Impulse Formation

- Natural pacemaker: Periodically generates electrical impulses to initialize heart beats



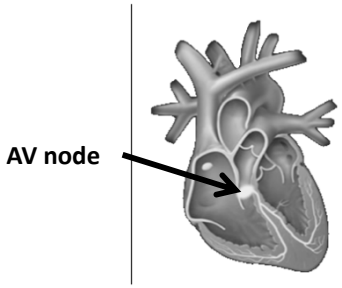
Atrial Contraction

- An impulse first triggers muscle contractions in the atria, pushing blood into the ventricles



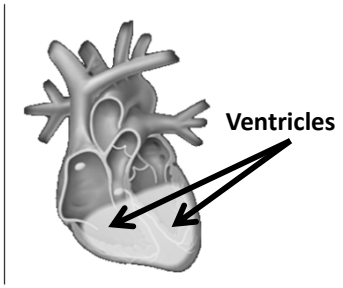
Delay At AV Node

- Delay at AV node which allows the ventricles to fill fully

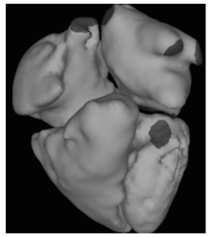


Ventricular Contraction

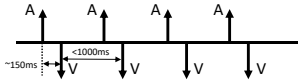
- Strong muscle contractions pump blood out of the heart



Normal Sinus Rhythm



- Electrical activities originate from the SA node
- The heart rate is around 60-100bpm, which is sufficient during resting



17

Key physiological requirements

- Ventricular rate
  - In order to satisfy the minimum blood/oxygen demand
- Synchrony
  - Maintain efficient blood flow
    - A-V synchrony
    - V-V synchrony
    - Local synchrony

18

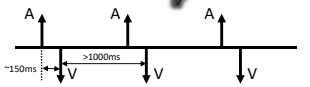
Cardiac Arrhythmia

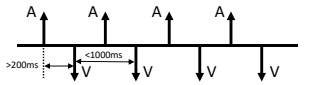
- Bradycardia
  - Inadequate heart rate
  - Symptom: Dizziness, feint
  - Pacemakers are designed to treat Bradycardia
- Tachycardia
  - Excessive heart rate
  - Symptom: Palpitation, shortness of breath, chest pain
  - Pacemaker should not further increase the heart rate

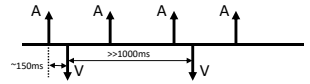
19

Bradycardia

- Slow generation
  - Slow atrial and ventricular rate
- Delayed conduction
  - Affecting synchrony
- Blocked conduction
  - Missed ventricular contractions







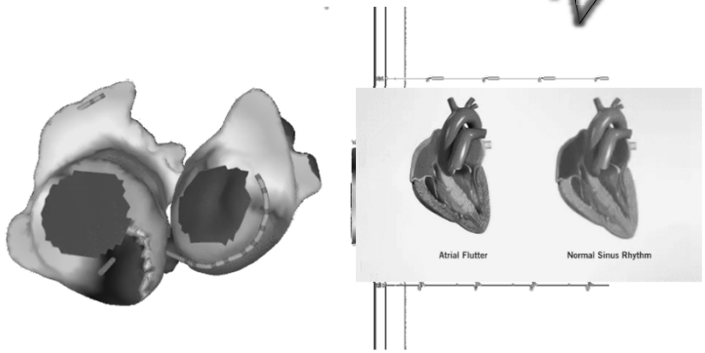
20

Mechanisms for tachycardia

- Focal
  - Some tissue other than the SA node generates fast electrical events
- Reentry
  - Self-sustained electrical activations circling within an electrical circuit

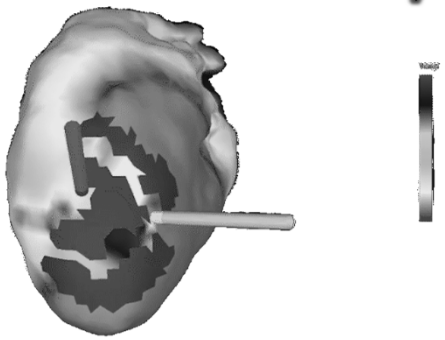
21

SVT: Atrial Flutter



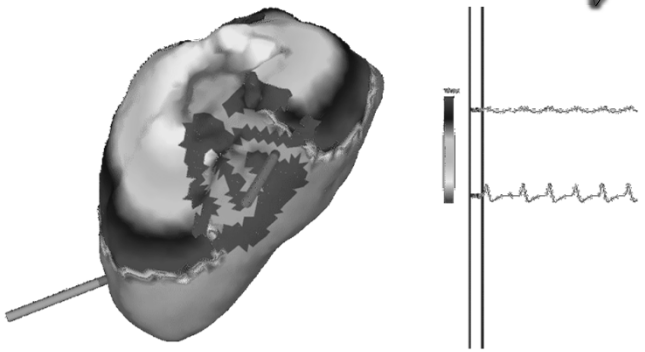
22

VT: Case 1



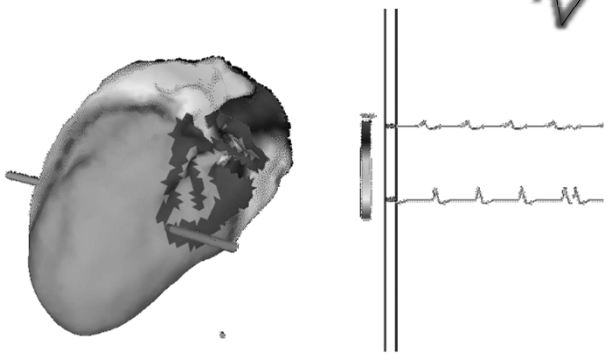
23

VT: Case 2



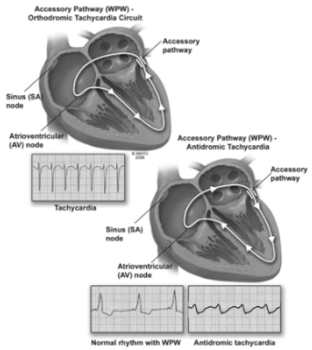
24

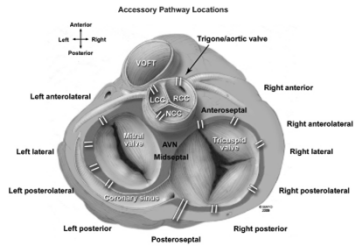
VT: Case 3



25

Atrial-ventricular reentry tachycardia (AVRT)

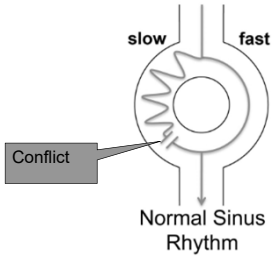




26

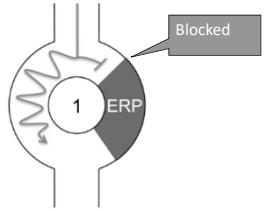
Mechanism: Reentry Tachycardia

- Equivalent to the fast pathway alone



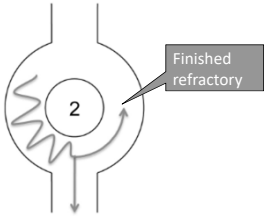
Mechanism: Reentry Tachycardia

- Premature activation blocked in the fast pathway



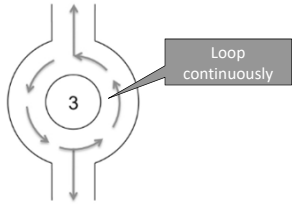
MECHANISM: REENTRY TACHYCARDIA

- Retrograde conduction through fast pathway creating echo beat

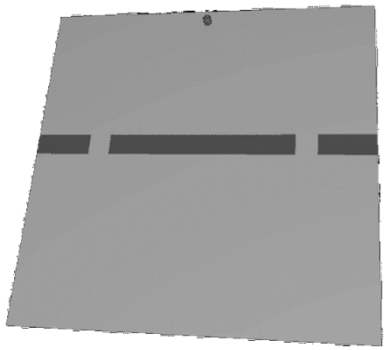


MECHANISM: REENTRY TACHYCARDIA

- Reentry induced and maintain high atrial and ventricle rate

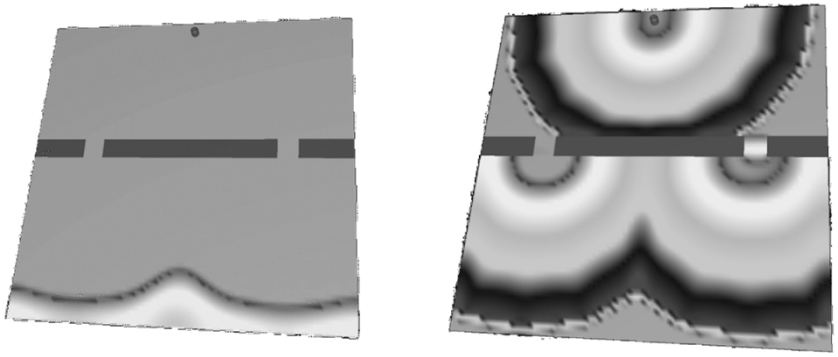


REENTRY MECHANISM: DIFFERENCE IN REFRACTORY PERIOD



31

REENTRY INDUCTION IN SLOW MOTION

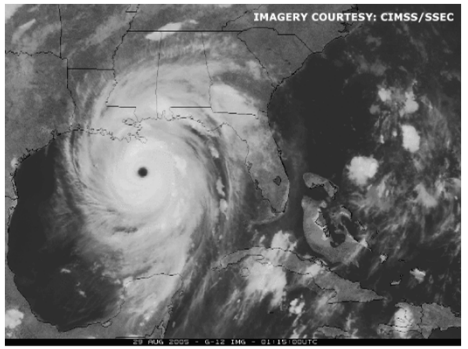


FIBRILLATION

- Reentry without anatomical pathways
- The location of the "circuit" can change over time

33

FIBRILLATION IN ANOTHER DOMAIN



34

FIBRILLATION

- Multiple moving centers

35

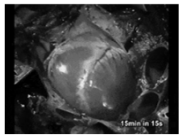
TACHYCARDIA

- SupraVentricular Tachycardia (SVT)
  - Fast electrical signals originate from above the ventricles
  - Not fatal, but may develop into more serious conditions
- Ventricular Tachycardia
  - Fast electrical signals originate from the ventricles
  - Fatal, can cause death in minutes

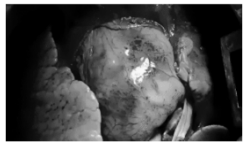
36

VENTRICULAR TACHYCARDIA

Ventricular Tachycardia
Regularly irregular



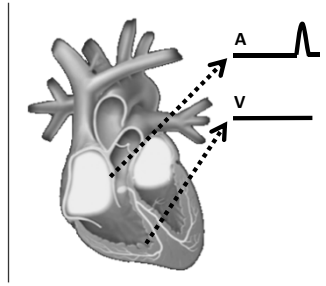
Ventricular Fibrillation
Irregularly irregular



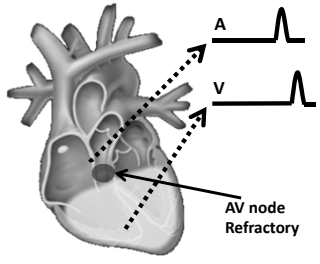
Death

37

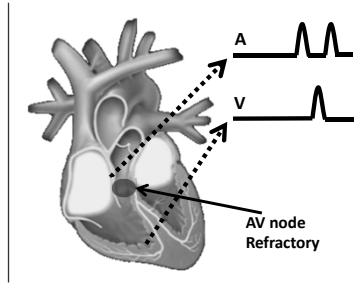
WHY SVT IS NOT FATAL?



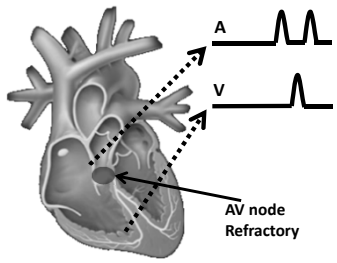
WHY SVT IS NOT FATAL?



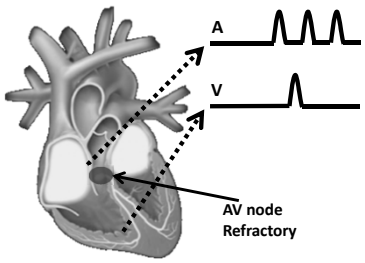
WHY SVT IS NOT FATAL?



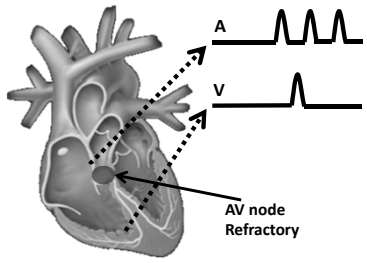
WHY SVT IS NOT FATAL?



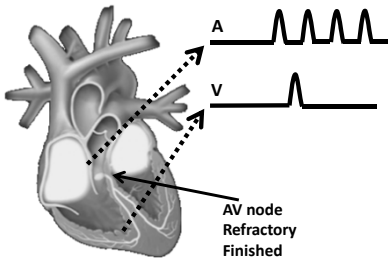
WHY SVT IS NOT FATAL?



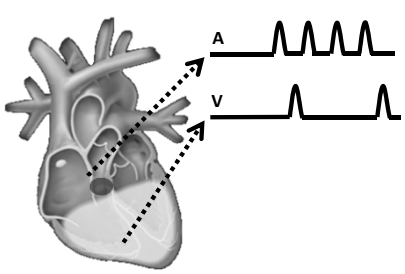
WHY SVT IS NOT FATAL?



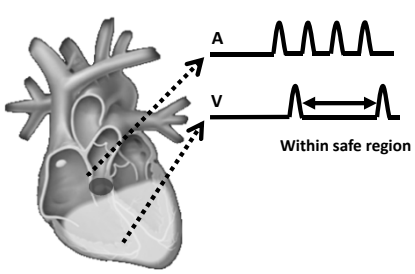
WHY SVT IS NOT FATAL?



WHY SVT IS NOT FATAL?



WHY SVT IS NOT FATAL?



WHY SVT IS NOT FATAL?

- The AV node can block some very fast electrical events
- The ventricles still maintain coordinated contractions

47

THE V(H) FUNCTION

- Roughly
  - NSR > SVT ~= Bradycardia > VT > VF

48

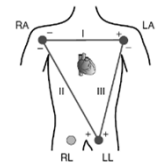
Diagnosing arrhythmia

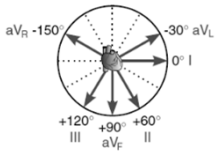
- Electrocardiogram (ECG) (EKG)
- Electrophysiological (EP) Testing

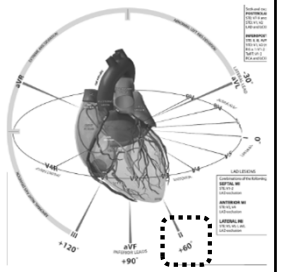
49

Electrocardiogram (ECG)

- 12-lead ECG monitors how electrical activities in the heart map to different vectors
- The most commonly used ECG signal is Lead II

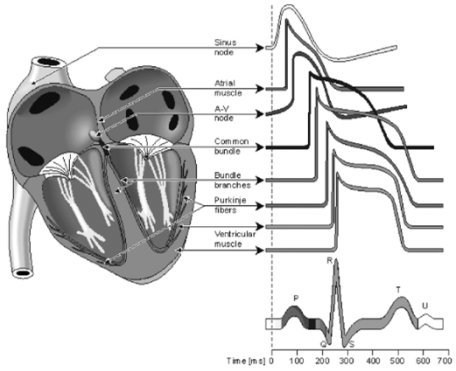






50

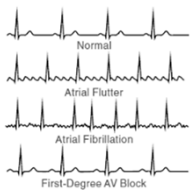
How local electrical activities map to ECG signal

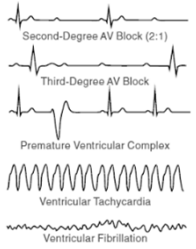


51

Diagnosing common arrhythmia using ECG

- Cannot tell exactly where the anomalies like reentry circuits are

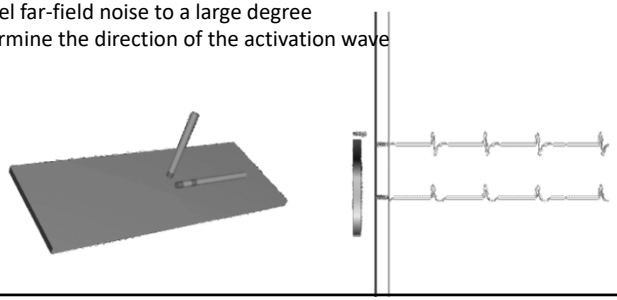




52

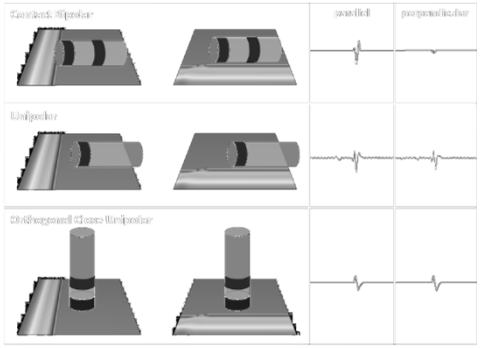
ELECTROGRAM (EGM)

- Approaching and departing of electrical waves
- Unipolar EGM are subjected to far-field noise
- Bi-polar EGM can
  - Cancel far-field noise to a large degree
  - Determine the direction of the activation wave



53

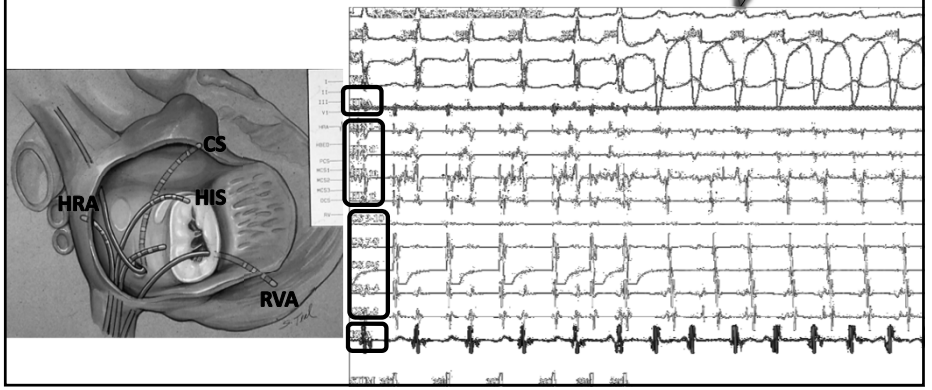
EGM MORPHOLOGY WITH DIFFERENT CATHETER POSE



54

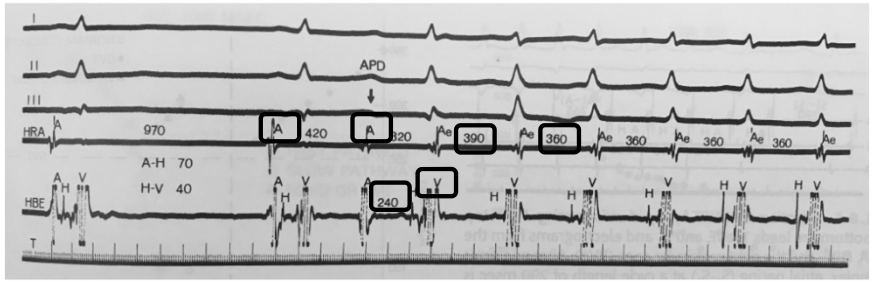
ELECTROPHYSIOLOGICAL (EP) TESTING

ELECTROGRAM (EGM) SIGNALS

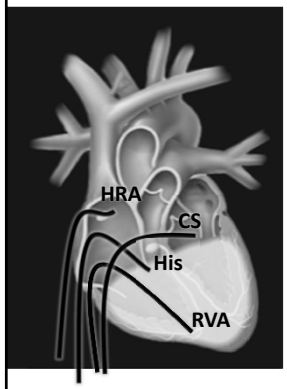


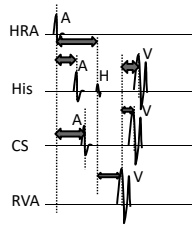
Electrogram Annotation

- Event type & origin
- Delays between events



Baseline NSR Electrogram Annotation



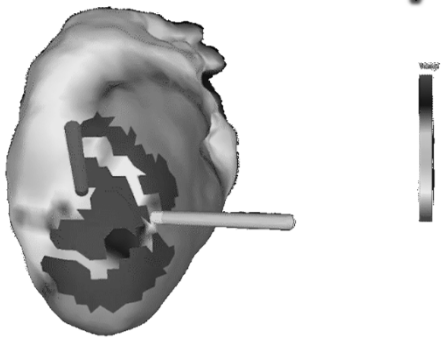


Diagnosing Arrhythmia

- Electrocardiogram (ECG) (EKG)
  - Global view of the electrical activities
  - Difficult to map to local anomalies (Inverse problem)
- Electrophysiological (EP) Testing
  - Monitors local electrical activities
  - The number of locations that can be monitored at the same time is limited

59

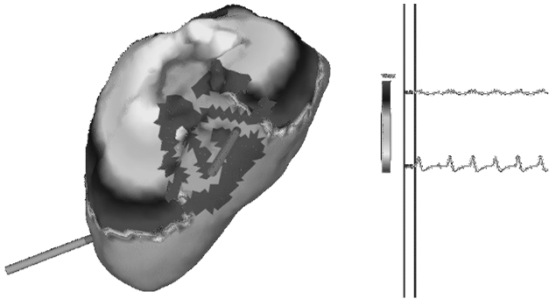
Ablation: Treating Tachycardia



60

ABLATION: TREATING TACHYCARDIA

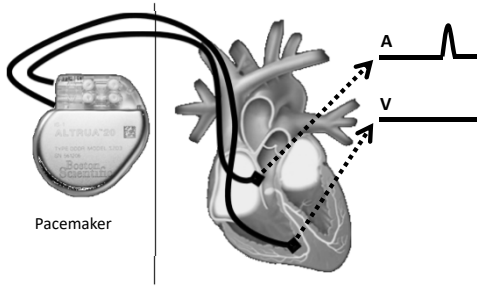
This will fail as the ablation catheter is at a part of the circuit which is irrelevant



61

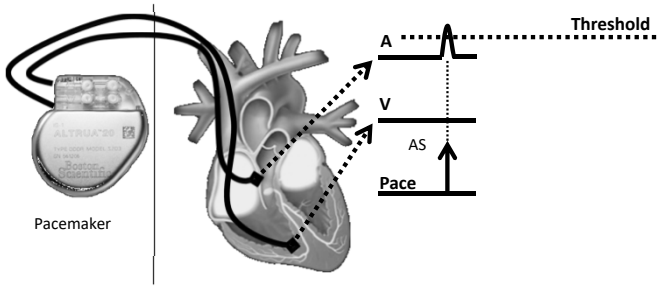
ELECTROGRAM (EGM) SIGNALS

- Pacemaker leads sense voltage changes corresponding to electrical activities



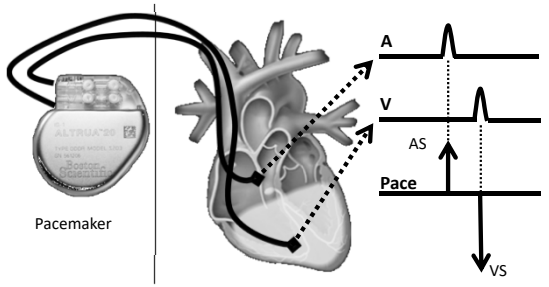
ATRIAL SENSING (AS)

- Generate sensed event when signal above threshold



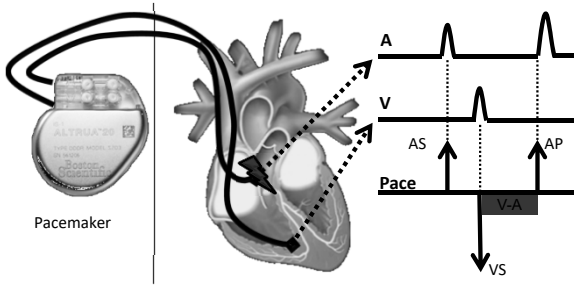
VENTRICULAR SENSE (VS)

- Same for ventricular channel



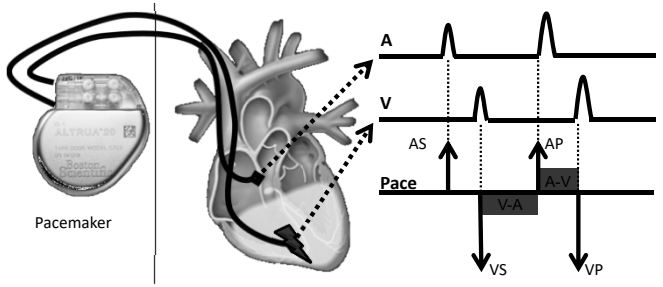
V-A DEADLINE & ATRIAL PACING (AP)

- Pace atrium when no AS within deadline (track ventricular event)



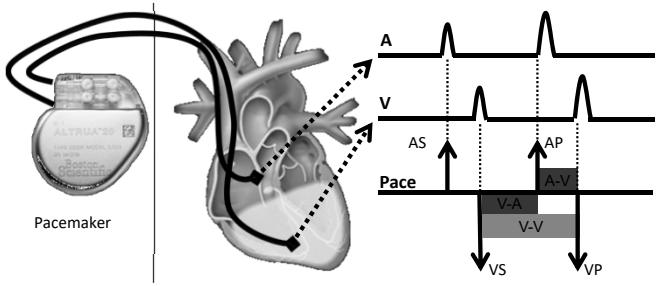
A-V DEADLINE AND VENTRICULAR PACING (VP)

- Pace ventricle if no VS happen within deadline (track atrial event)



MAINTAIN MINIMUM HEART RATE

- Maximum interval between two ventricular events (max(V-A)+max(A-V))



NEXT LECTURE

- How to collect more information via EP study?
- What is a good model for an application?

68

READING MATERIAL

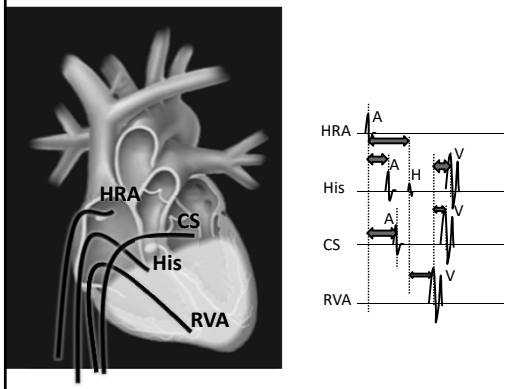
- R2: EP Study Basics
- R3: Mathematical modeling (by Mette Sofie Olufsen)

- For heart simulations and tutorials you can check:
  - http:www.visibleep.com

69

Lecture 3: EP Testing and Modeling Basics

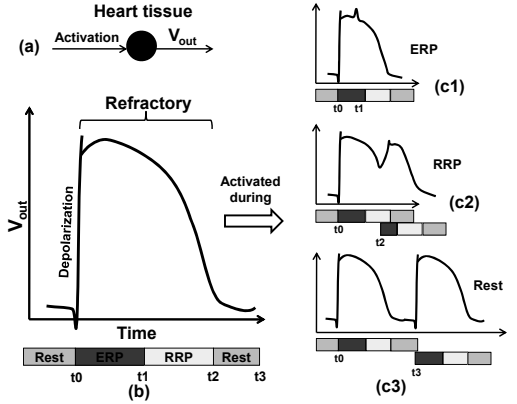
What we can learn from this EGM?



How to get new information?

- Passive observations provide us very little information
- What if we would like to know
  - The ERP of certain tissue
  - Are there additional pathways?
  - Where electrical events can be blocked along a pathway

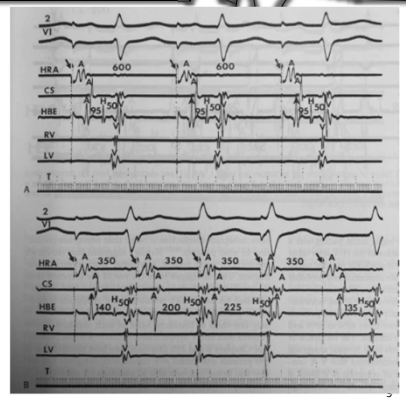
Refractory Periods



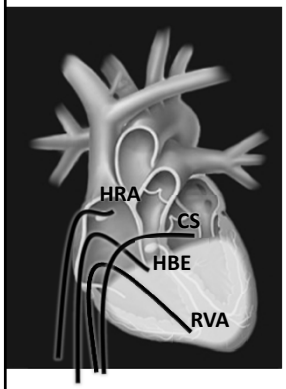
- ERP: Effective Refractory Period
- RRP: Relative Refractory Period
- A cell needs to "recover" before it can be depolarized again
- A cell can be partially depolarized
  - Shortened ERP
  - Increased conduction delay

WENCKEBACH AV BLOCK

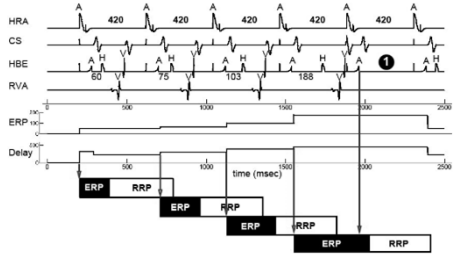
- A: Each pacing event provides the same information
- B: AV_ERP+RRP>350ms
- AV_ERP<350



WENCKEBACH AV BLOCK (SIMULATION)



- Gradually increased conduction delay and ERP result in blocked A-V conduction



6

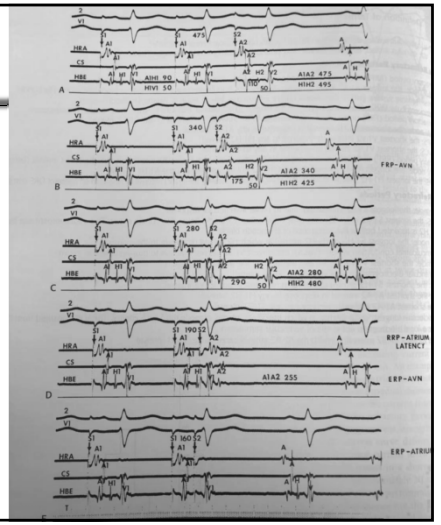
EXTRASTIMULI

- Stabilize the heart to a known state
  - A sequence of pacing stimuli with long cycle length (S1)
  - Faster than intrinsic heart rate, but shorter than most ERP+RRP
- A quick pacing stimulus with short cycle length (S2)

7

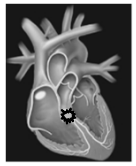
EXTRASTIMULI FROM HRA

- Gradually decrease the cycle length of S2
- A: AV_ERP+RRP>=475
- B: AV_ERP<340
- C: AV_ERP<280, SA_ERP+RRP>190
- D: 190<AV_ERP<280, SA_ERP+RRP>190
- E: SA_ERP>160

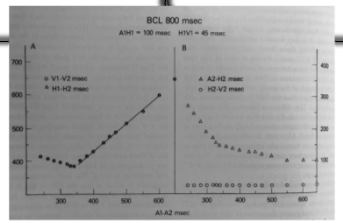


USING EXTRASTIMULI TO DISTINGUISH DIFFERENT HEART CONDITIONS

Type I

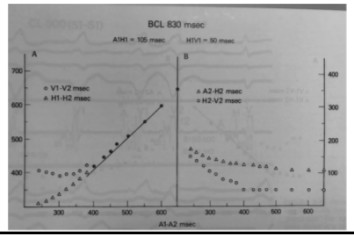


- AV_ERP+RRP=350ms

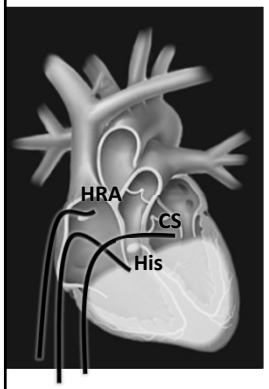


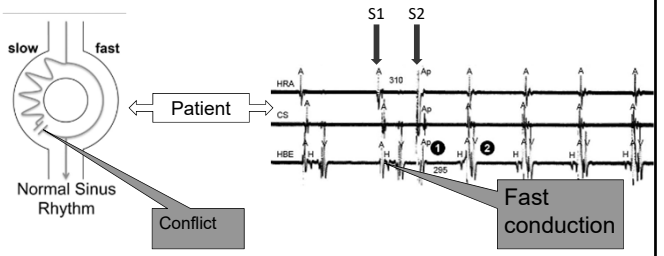
Type II

- RBB_ERP+RRP=400ms

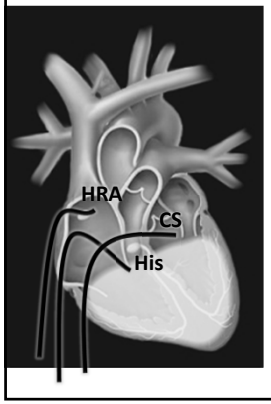


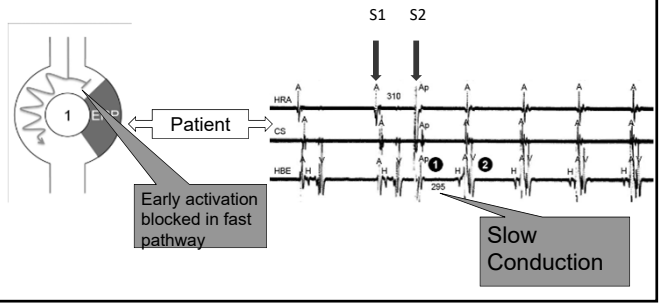
INDUCING AVNRT



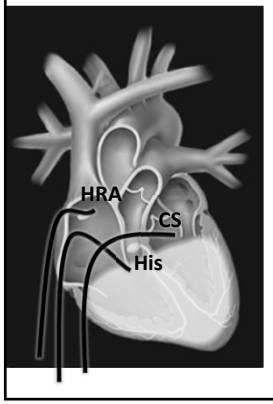


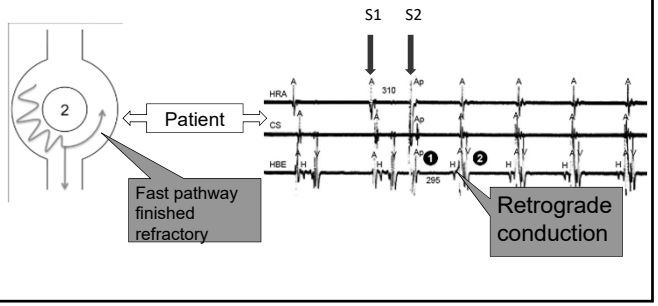
INDUCING AVNRT





INDUCING AVNRT





Modeling Cardiac Electrophysiology

Why do we need models?

- Prediction
  - We know the low-level mechanisms but we want to understand how they affect higher-level behaviors
  - Use simulation instead of testing on the real system
- Explain the data
  - Make assumptions and use our knowledge to explain mechanisms that we don't understand
- Classification
  - i.e. definitions, machine learning algorithms

What are models?

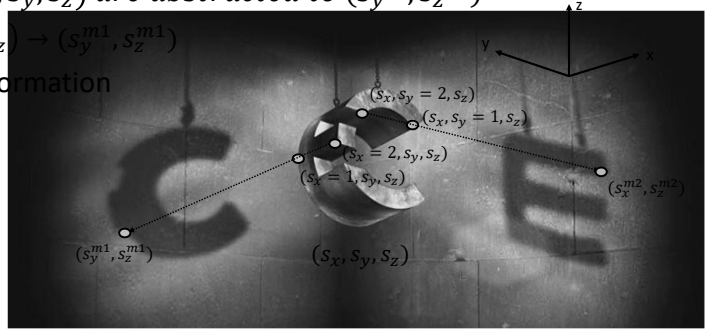
- A system: (S, I, T, O)
  - S: States $s_1, s_2 \dots s_n$
  - I: Inputs (could be $\emptyset$)
  - T: Transitions $S \times I \times S$
  - O: Observations $f(S_o), S_o \subseteq S$

⇒

- Model of the system (S^m, I^m, T^m)
  - S^m: Abstraction/interpolation of S
    - Much fewer state variables
  - I^m: abstraction of I (could be $\emptyset$)
  - T^m: Transitions $S^m \times I^m \times S^m$

Abstraction – removal of state variables

- States (s_x, s_y, s_z) *are abstracted to* (s_y^{m1}, s_z^{m1})
  - $(s_x, s_y, s_z) \rightarrow (s_y^{m1}, s_z^{m1})$
- Loss of information



Chernin Entertainment

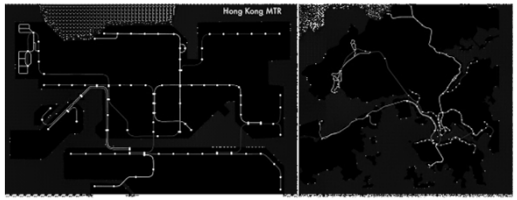
Abstraction: Approximation of state variable values

- Irrational numbers
  - $\pi \approx 3.1415$
  - $\sqrt{2} \approx 1.414$
- Approximation is another way of abstraction

17

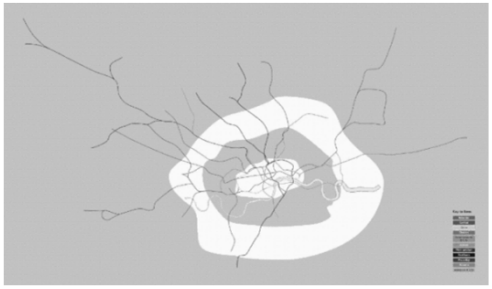
Interpolation: extracting interpretable information

- Locational information -> topological information
- $S^m = f(S_p), S_p \subseteq S$



18

More Interpolation: London MTR



19

What is considered as a "good" model?

- Accuracy
  - All models are wrong!
  - Error accumulates over time
  - Initial condition of the model cannot be determined due to limited observability
- Generality
  - The capability to explain not only training data, but also testing data
- Identifiability
  - Model parameters can be identified from data
- Interpretability
  - S^m are meaningful and interpretable by human

20

Newton vs. Einstein

- Newtonian physics is suitable for macro level objects at low speed
- $L = L_0\sqrt{1 - \frac{v^2}{c^2}}$
- A model can only be "good" within the context of its designated application
- The definition of "goodness" is changing over time

21

Modeling methodologies

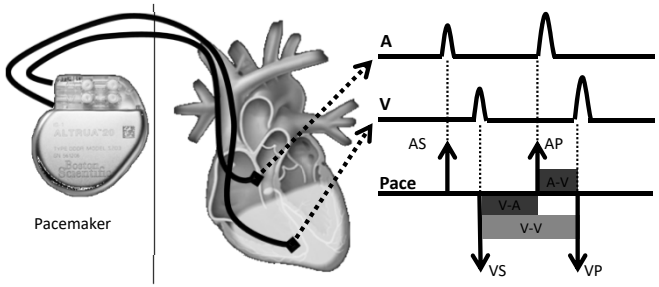
- Bottom-up modeling
  - "White-box" model
  - Using first principles
  - Pros:
    - Interpretable
    - Convincing
  - Cons:
    - State space explosion
    - Difficult to be general
    - Low identifiability

- Data driven models (i.e. Neural networks)
  - "Black-box" model
  - From observable data
  - Pros:
    - No need to know domain knowledge
  - Cons:
    - Large and uninterpretable S^m
    - Depends highly on the quality and quantity of data

22

Implantable Pacemaker

- Maximum interval between two ventricular events (max(V-A)+max(A-V))



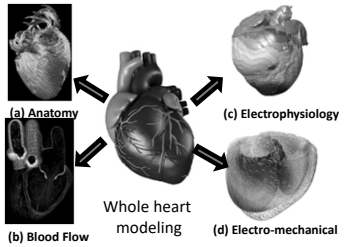
Physiological Requirements

- Does the device always deliver therapy when needed?
- Does the device always not deliver therapy when unnecessary?
- Behavior Coverage (Generality)
- Physiological Context (Interpretability)

24

Existing heart models

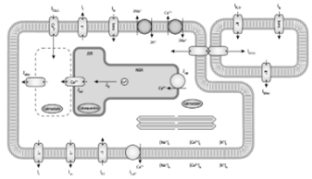
- Researchers created heart models from different perspectives
- Only electrophysiological model is suitable for pacemaker evaluation



25

Whole heart Electrophysiological Model

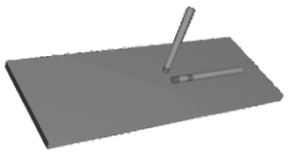
- Too complex with unnecessary details
- Infeasible to identify parameters from data

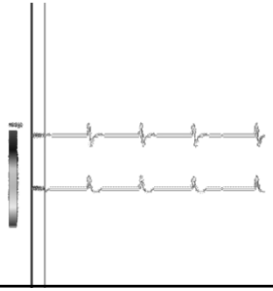


26

Electrogram (EGM)

- Fixed locations
- Bi-polar EGM
  - Cancel far-field noise to a large degree
  - Lose local action potential information

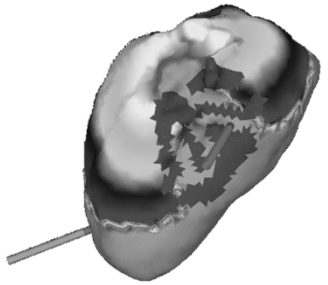


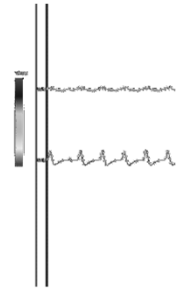


27

State equivalence

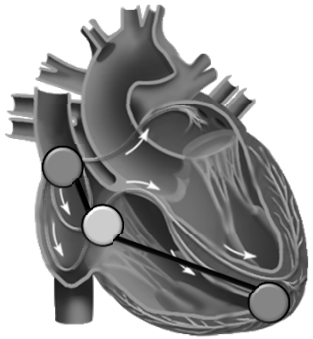
- From the pacemaker perspective, a reentry circuit is equivalent to a node firing electrical events at high frequency





28

Electrophysiological Behavioral Model

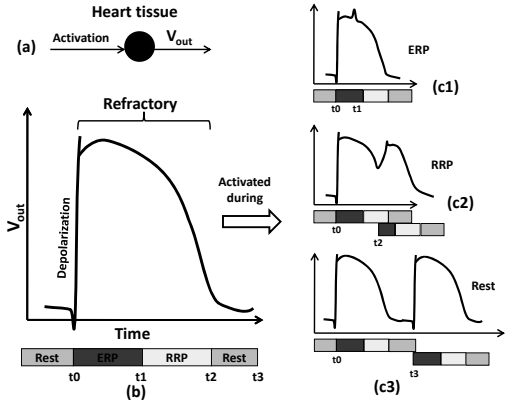


Electrical Conduction System

- Tissue interacting with the pacemaker
- Tissue or structure that affects the observable behaviors
- How the electrical signal conduct among the tissue

29

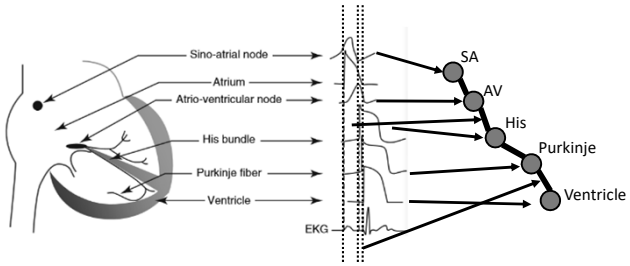
Refractory Periods



- ERP: Effective Refractory Period
- RRP: Relative Refractory Period
- A cell needs to "recover" before it can be depolarized again
- A cell can be partially depolarized
  - Shortened ERP
  - Increased conduction delay

Our Approach: Clinical EP Heart Modeling

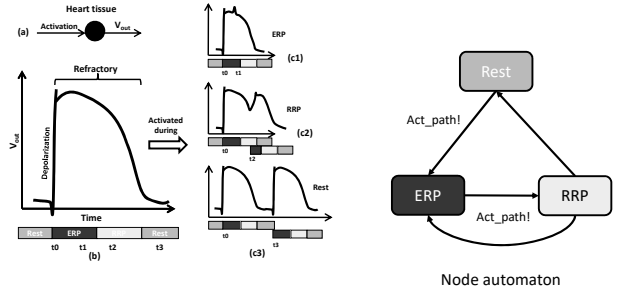
- Nodes: Electrical events generation and blocking
- Paths: Conduction delays between nodes



31

Node Automata

- Generation and blocking of electrical events

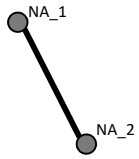


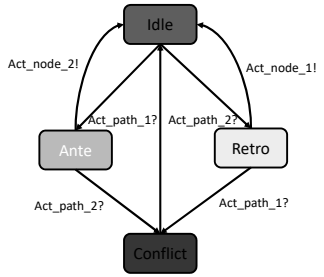
Node automaton

32

PATH AUTOMATA

- Conduction delay between nodes



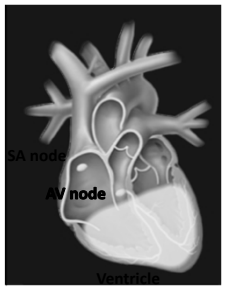


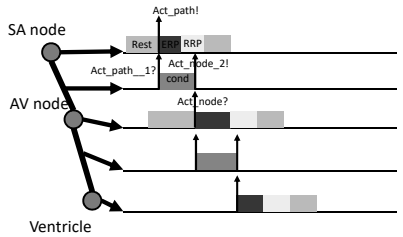
Path automaton

33

EXAMPLE

- VHM execution correspond to heart beat





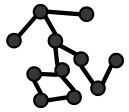
34

NETWORK OF TIMED AUTOMATA

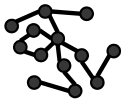
Represent variety of heart conditions using different topologies and timing parameters

Normal Sinus Rhythm

Atrial Flutter



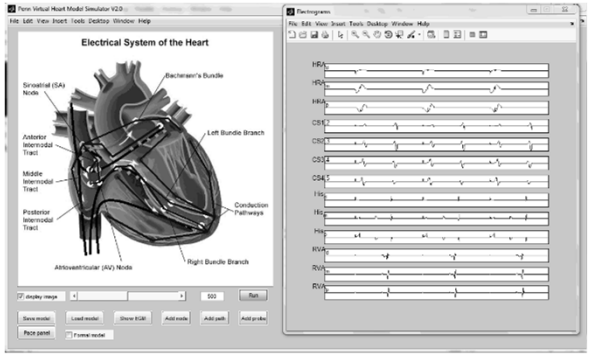
Ventricle Tachycardia



AV Nodal Reentry

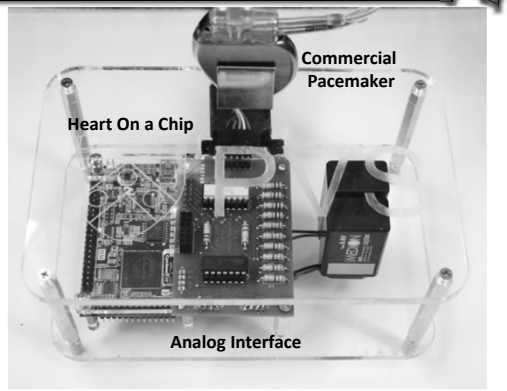
35

HEART MODEL SIMULATION

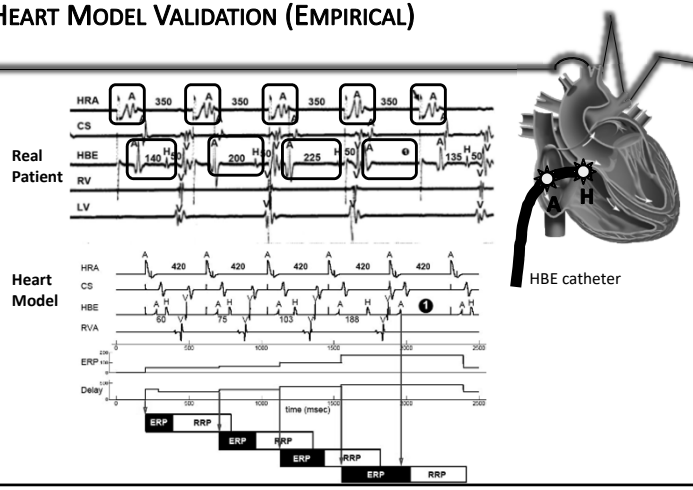


36

HEART ON A CHIP PLATFORM



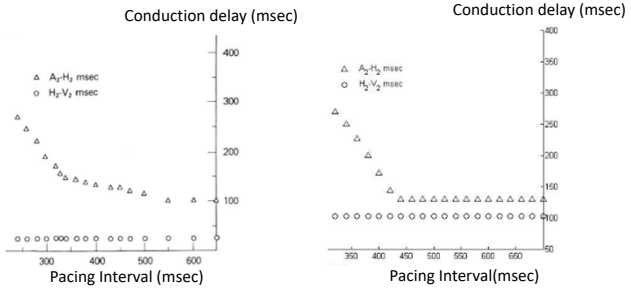
HEART MODEL VALIDATION (EMPIRICAL)



38

HEART MODEL VALIDATION (QUANTITATIVE)

Atria-to-ventricle conduction delay vs. pacing interval



Real Patient

Heart Model Simulation

39

HOW GOOD IS OUR MODEL?

- Accuracy
  - Condition-level accuracy vs. patient-level accuracy
- Generality
  - Can be used for modeling a large number of arrhythmia
- Identifiability
  - Compatible with existing EP testing techniques
- Classification
  - Capable of distinguishing different arrhythmia

40

READING MATERIAL

- R4: Pacemaker introduction
- https://matlabacademy.mathworks.com/

41

HW 1 Out

- Due on Mon Sep 30th, at the end of the day
- Please use Matlab R2019a or higher

Homework Policy

- Homework should be done individually!!!
- Submit on Blackboard
- Deadline: 23:59 of specified date. (okay if submitted within 15min after)
- Late Policy: Late homework submission will receive penalties.
  - Homework submitted 72hr after the deadline will get 0 pt.
  - Every 24hr after the deadline, the grade will have a 20% penalty.
    - i.e. if you gained 8/10 points in a homework, with 1 day late submission you will only gain 6.4/10 points.
- Late pass:
  - Each student is given a "Late pass" which can be used ONCE on a homework submission. The late pass extends ONE homework deadline for 24hr, but remember you will still receive penalties if you miss the Extended deadline.

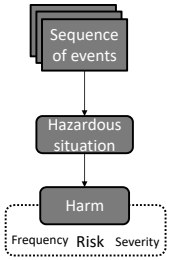
Lecture 6: Risk Management

What is risk management?

- There are undesired situations that we don't want to be in
  - i.e. an order was missed and the customer was not happy
- What can lead to those undesired situations?
- Can we avoid these undesired situations?
- If not, can we reduce potential damage?

TERMINOLOGIES

- Harm/impact: Loss when running the system in certain situations
  - Financial/time loss
  - Life/health loss
- Hazard: A potential source of harm
- Hazardous situation: circumstance in which people or property are exposed to one or more hazard(s).
- Risk: Combination of the probability of occurrence of harm and the severity of that harm



TYPES OF RISKS

- **Efficacy risk:** The system fails to do what it was intended to do
- **Safety risk:** People & properties may be harmed under normal use of the system
- **Security risk:** The system is prone to unintentional and malicious misuse which can cause harm

EXAMPLE: INFUSION PUMP

- Deliver drug with programmed dosage
- Hazard: Pump won't stop
- Hazardous situation: over-delivery of drug
- Sequence of events:
  1. Unexpected user programming sequence
  2. Corrupted limit value
  3. Pump runs beyond programmed limit
- Harm:
  1. Discomfort
  2. Tissue/organ damage
  3. Death

Infusion Pump

EXAMPLE: PILOTING AN AIRCRAFT

- Hazard: Pilot fatigue
- Hazardous situation: aircraft control lost/delayed
- Sequence of events:
  1. Pilot did not sleep well last night
  2. Pilot forget to lower the landing gear before landing
  3. Aircraft landed without landing gear down
- Harm:
  1. Aircraft severely damaged
  2. Crash

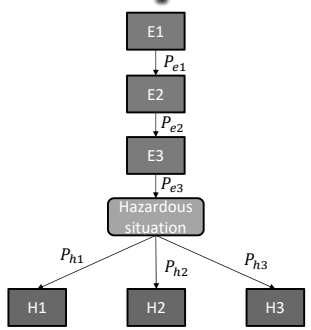
Risk Management process

1. Risk analysis
   – Analyze the frequency and severity of harms
2. Risk evaluation
   – Determine what is acceptable risk
3. Risk control
   – How to prevent hazard and/or reduce harm?
4. Residual risk acceptability
   – Are there new risks introduced by control measures?

Risk analysis

- Hazard: Pump won't stop
- Hazardous situation: over delivery of drug
- Sequence of events:
  1. Unexpected user programming sequence
  2. Corrupted limit value
  3. Pump runs beyond programmed limit
- Harm:
  1. Discomfort — **Minor**
  2. Tissue/organ damage — **Major**
  3. Death — **Catastrophic**

$$P_{death} = P_{e1} \times P_{e2} \times P_{e3} \times P_{h3}$$

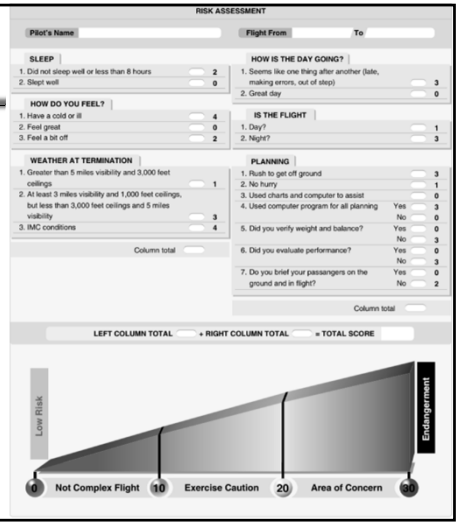


Risk Analysis methods

- Expert opinion
  - Solicitation of views from those deemed to be most knowledgeable
- Data diving
  - Use of data to assess impact and frequency over a specified time period as a proxy for likelihood
- Stochastic
  - Have some prior knowledge
  - Application of statistical models (e.g., Monte Carlo simulation) that capture the essentials of the underlying behavior
- Cultural
  - Reliance on the inherent subjective assessment of both the nature of events and their likelihood and impact
- Often used in combinations

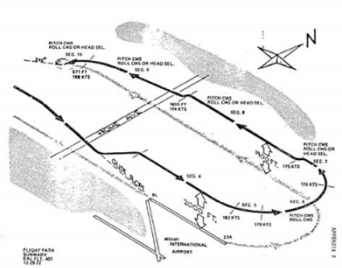
Example

- Pilot self evaluation
  - Expert opinion
- Considering both the frequency and harm of potential risks

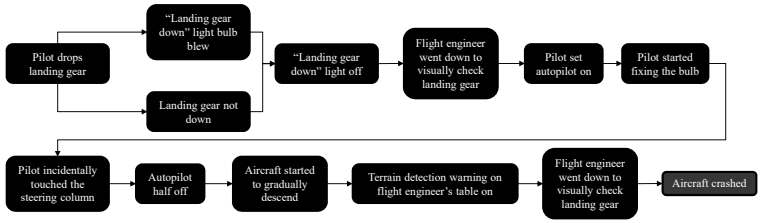


SMALL & RARE HAZARDS CAN LEAD TO FATAL HARM

- Eastern Airlines Flight 401, 1972



SEQUENCE OF EVENTS



RISK ACCEPTABILITY

- Acceptable
- Unacceptable
- ALARP (As Low As Reasonably Practicable)

Probability / Severity	Frequent	Probable	Occasional	Remote	Improbable
Catastrophic	IN	IN	IN	H	M
Critical	IN	IN	H	M	L
Serious	H	H	M	L	T
Minor	M	M	L	T	T
Negligible	M	L	T	T	T

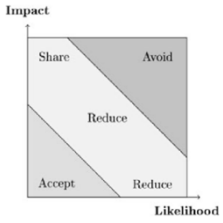
LEGEND: T = Tolerable L = Low M = Medium H = High IN = Intolerable

Probability	Description	Severity	Consequence
Frequent	Not surprised, will occur several times (Frequency per year > 1)	Catastrophic	Greater than 6 month slip in schedule, greater than 10% cost overrun, greater than 10% reduction in product functionality
Probable	Occurs repeatedly/ an event to be expected (Frequency per year 1-10^{-1})	Critical	Less than 6 month slip in schedule, less than 10% cost overrun, less than 10% reduction in product functionality
Occasional	Could occur some time (Frequency per year 10^{-1} - 10^{-2})	Serious	Less than 3 month slip in schedule, less than 5% cost overrun, less than 5% reduction in product functionality
Remote	Unlikely though conceivable (Frequency per year 10^{-2} - 10^{-4})	Minor	Less than 1 month slip in schedule, less than 2% cost overrun, less than 2% reduction in product functionality
Improbable	So unlikely that probability is close to zero (Frequency per year 10^{-4} - 10^{-5})	Negligible	Negligible impact on program

Table 2-1 Sample Risk Matrix

ACTIONS TOWARDS DIFFERENT RISKS

- Accept
  - Appropriate for low risk exposure. No specific action will be undertaken to mitigate or manage the risk. Instead a contingency plan will be developed to tackle the eventuality that the risk may be realized
- Reduce
  - Appropriate for medium negative risk. Actions will be undertaken to reduce either the impact or the likelihood of occurrence of the event
- Share/transfer
  - Appropriate for low frequency high impact positive risk where the efforts to manage the risk alone may not be warranted owing to the relative unlikelihood of it occurring. Measures will be undertaken (with others) to share the risks and rewards
- Avoid
  - Appropriate for high negative risk. The activities that give rise to the risk will not be undertaken



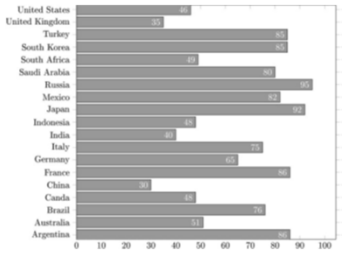
How to draw the line - Attitude towards risks

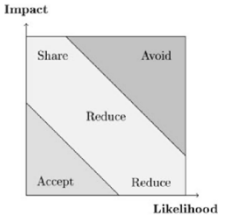
- *Risk-averse*:
  - Preference of secure payoffs, common sense and facts over theories. Propensity to over-react to threats and under-react to opportunities.
- *Risk-seeking*:
  - Preference towards speculation and unafraid to take action. Propensity to underestimate threats and overestimate opportunities.
- *Risk-tolerance*:
  - Indifference towards uncertainty that lends itself to reactive rather than proactive measures. Propensity to fail to appreciate importance of threats and opportunities alike.
- *Risk-neutral*:
  - Impartial attitude towards risk and act in the interests of significant benefits. Propensity to focus on the longer term.

*Murray-Webster and Hillson (2008)

Uncertainty Avoidance Index (UAI)

- Extent to which the members of a culture feel threatened by ambiguous or unknown situations and have created beliefs and institutions that try to avoid these" (Hofstede 2013)



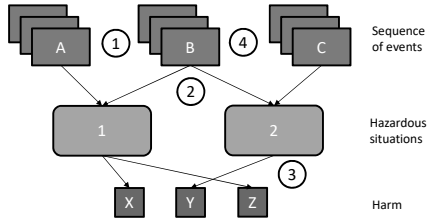


Risk Control

- The activity in which decisions are made and measures implemented in which risks are reduced to, or maintained within specified levels
- Risk Control Measures (RCM)

Risk Control Measures (RCM)

1. Preventive measures
   - i.e. a cover on emergency stop button
2. Corrective measures
   - i.e. watchdog
3. Mitigating measures
   - i.e. fuse
4. Informational control
   - i.e. warning, personnel training



	Reduce probability	Reduce severity
Preventive	Yes	No
Corrective	Possibly	Possibly
Mitigating	No	Yes
Informational	Yes	No

Residual Risk Evaluation

- Whether the benefit of further measures outweigh the overall residual risk
- Treat remaining risk as new risk and see whether it's acceptable
- Principle: ALARP (As Low As Reasonably Practicable)

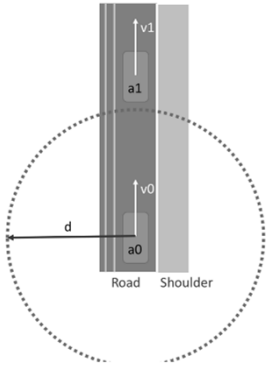
RCM Example

- Mostly preventive

Hazards	Pre Mitigation Likelihood	Severity	Outcome	Mitigation	Post mitigation Likelihood	Severity	Outcome
Poor visibility (smoke)	Frequent	Catastrophic	High	Determine effectiveness of the operation (risk vs. benefit) and discontinue if warranted. Limit number of aircraft in operating area. Increase vertical/horizontal separation of aircraft.	Occasional	Catastrophic	High
Wake turbulence and speed differential (SEATs)	Frequent	Critical	High	Use show me or chase profile. Use lead profile only when necessary. Performance maneuvers (e.g., Steep turns and pushovers) should be communicated to other aircraft. SEAT performance (speed) needs to be pre-determined in order to set the correct drop speed.	Occasional	Critical	Serious
Weather (turbulence/wind/thunderstorms)	Frequent	Critical	High	Adjust tactics or shut down air ops. Increase vertical/horizontal separation of aircraft. Utilize human aided technology (weather radar, etc.). Encourage dispatch to obtain/communicate weather information. Utilize and share pilot reports of severe weather.	Occasional	Critical	Serious
Fuel management	Occasional	Critical	Serious	Monitor fuel quantities. Follow fuel transfer procedures. Pre-flight the aircraft. Plan the flight; know refueling locations. Query other aircraft.	Remote	Critical	Medium
Density altitude	Frequent	Catastrophic	High	Relocate aircraft. Consult performance charts. Download fuel.	Remote	Catastrophic	Serious
Poor engine performance (single/twin, turbine/recip).	Occasional	Catastrophic	High	Avoid high density altitudes. Download cargo/fuel load. Relocate to favorable location. Alter the mission. Upgrade the aircraft. Ensure aircraft is appropriate for the mission. Perform pre-flight planning.	Remote	Catastrophic	Serious

Prioritize Risks

1. Avoid Collision
2. Driving on shoulder prohibited
3. Avoid hard brakes

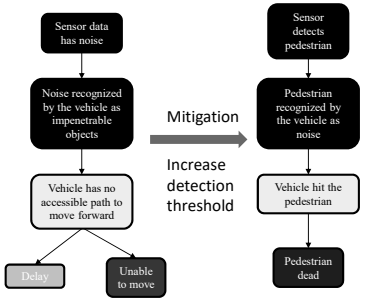


Balancing among Risks

- Sometimes, mitigating Risk 1 will make Risk 2 more likely
- Uber Autonomous Vehicle Accident

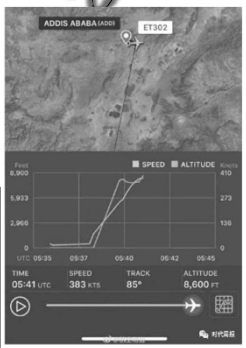
Uber Autonomous Vehicle Accident

- High frequency * serious harm
- Low frequency * catastrophic harm
- False-positive vs. false-negative
  - Always balance between the two



Ethiopian airline crash

- Flight ET302 from Addis Ababa for Nairobi
- Crashed on March 10, 2019
- Crashed 6 mins after takeoff
- All 157 people on board died
- Boeing 737-Max 8

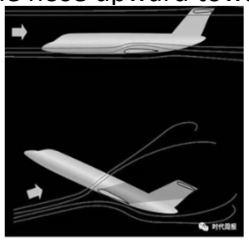


Similar Crash

- Lion Airline JT610 Oct 29, 2018
- Crashed 12 mins after takeoff
- All 189 people on board died

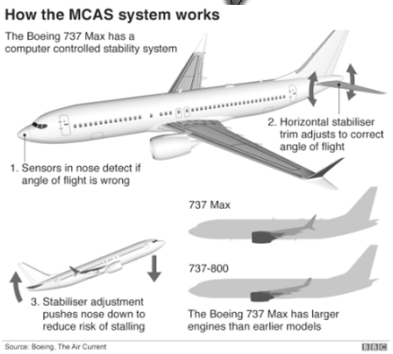
Boeing 737 Max 8/9

- The Max's engines were bigger and mounted farther forward on its wings
- A configuration that could push the nose upward toward a stall in certain circumstances.



Maneuvering Characteristics Augmentation System (MCAS)

- MCAS to automatically push the nose down to prevent stalling
- Two sensors mounted at the head of the aircraft
- MCAS activates if just one of two sensors says the nose is too high.

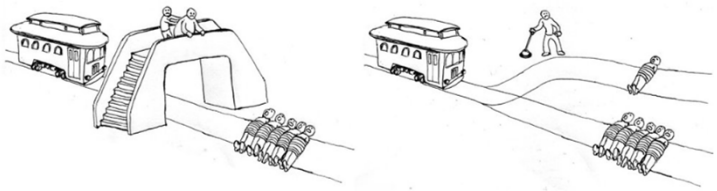


Boeing's "solution"

- Provided instructions to the pilots on how to turn off MCAS

Ethnic considerations when balancing risks

- Choices have to be made
- Invaluable things have to be priced
- For medical applications: Institutional Review Board (IRB)

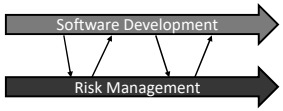


Risk management vs. Finding bugs vs. Validation

- Risk analysis
  - Finding flaws in design that may cause harm
- Finding bugs
  - Discrepancies when converting design into implementation
  - Verification
- Validation
  - Evaluate whether the design meets the requirements of customers

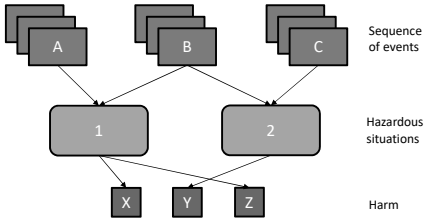
When to do risk management?

- Short answer: always
- Parallel with software development
- Identify & control & document & review new risks as development goes



Identify sequences of events that can lead to hazardous situations

- Hazardous situations are relatively easy to identify
- What are the sequences of events that can lead to the hazardous situation?
- What's the frequency of hazardous situations?



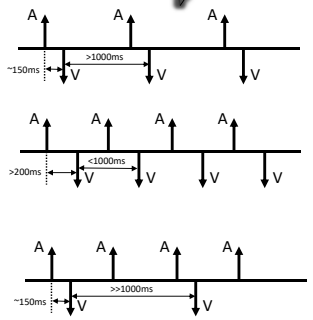
Next Lecture

- History of Pacemakers
- Reading Material: R4: Pacemaker Step-by-step

Lecture 5: Pacemaker Algorithm

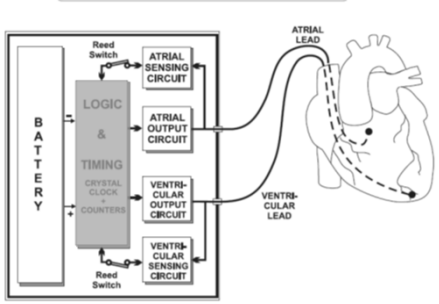
Bradycardia

- Ventricular rate slower than 60bpm
  - Slow generation
    - Slow atrial and ventricular rate
  - Delayed conduction
    - Affecting synchrony
  - Blocked conduction
    - Missed ventricular contractions



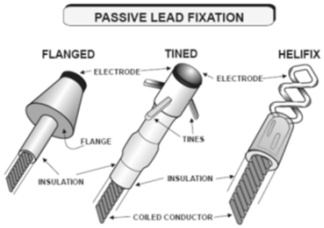
Pacemaker Architecture

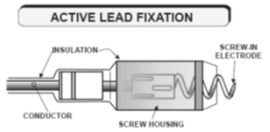
- We focus on the logic part



Lead Fixations

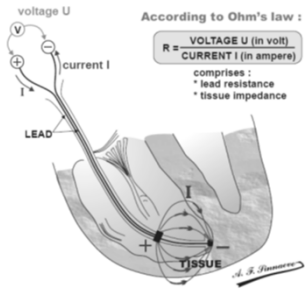
- Ensure stable contact with cardiac tissue





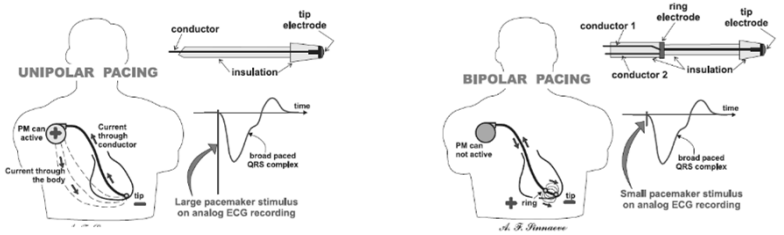
Pacemaker Pacing

- Electrical current triggers tissue depolarization



5

Unipolar vs. Bipolar Pacing



6

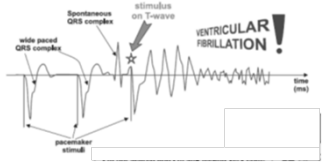
Pacemaker Modes

	I	II	III
Category	**Chambers Paced**	**Chambers Sensed**	**Response To Sensing**
Letters	O – None A – Atrium V – Ventricle D – Dual	O – None A – Atrium V – Ventricle D - Dual	O – None T – Triggered I – Inhibited D – Tracked

- VVI: Sense and pace in the ventricle only, inhibit pacing when sensed
- DDD: Sense and pace in both chambers, pace according to events in another chamber (tracked)

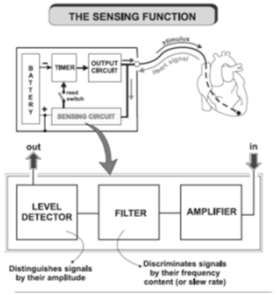
The History of Pacemakers

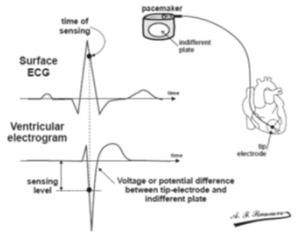
- VOO pacemaker
  - No sensing, pace in the ventricle with preset rate
- Pace on T wave may cause Ventricular Fibrillation



8

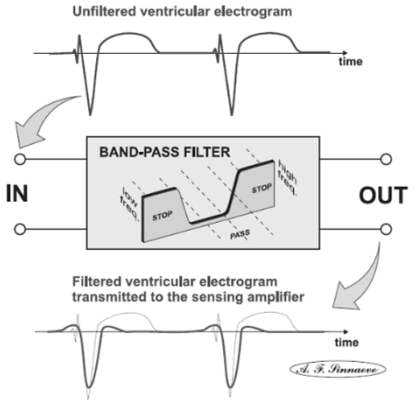
Introducing Sensing





9

Sensing Filter



10

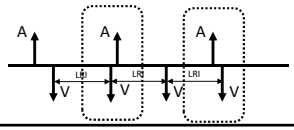
Limitations of pacemaker sensing

- Our goal: Identify heart chamber contractions
- Our assumption: Sensed atrial events reflect physiological need
- Reality: We only know that at a particular time, whether the sensed voltage at the lead is larger than the sensing threshold
  - We don't know whether an event corresponds to chamber contraction
    - Noise
  - We don't know where the signal is coming from
    - From the SA node?
    - From a reentry circuit?

11

Introducing of sensing

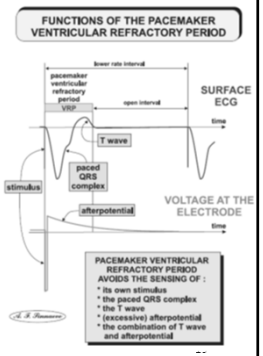
- VVI pacemaker
  - Sense and pace in the ventricle, inhibit pacing when event sensed
  - Lower Rate Interval (LRI): minimum interval between two consecutive ventricular events
- Pacemaker Syndrome
  - Atrial and ventricular contractions are too close to each other
  - No A-V synchrony



12

NOISE THAT MAY AFFECT DETECTION OF REAL HEART BEAT

- ERP: there should be no new activations
- Ventricular Refractory Period (VRP)
  - Ignore voltage above threshold (no sensed events)



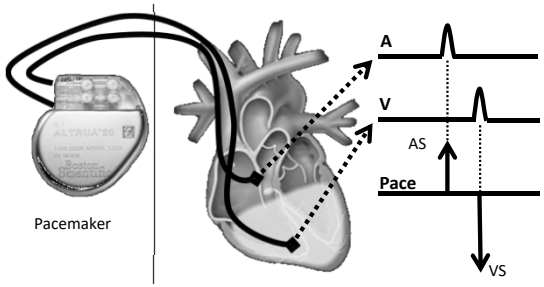
INTRODUCING THE ATRIAL LEAD

- DDD pacemaker
  - Sense and pace both atria and ventricles
  - Track: The timing of atrial event depends on the ventricular event, and vice versa
  - Lower Rate Interval (LRI): The minimum ventricular rate
    - Atrial Escape Interval (AEI): the minimum delay between a ventricular event and an atrial event
    - Atrial-Ventricular Interval (AVI): the minimum delay between an atrial event and a ventricular event
  - Upper Rate Interval (URI): The maximum paced ventricular rate

14

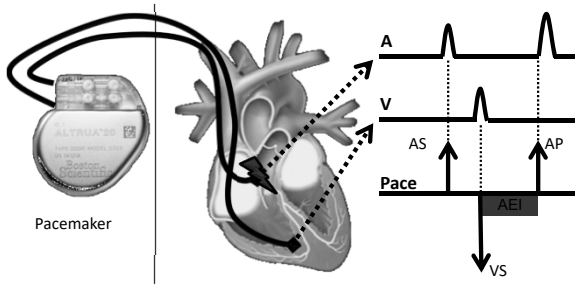
VENTRICULAR SENSE (VS)

- Same for ventricular channel



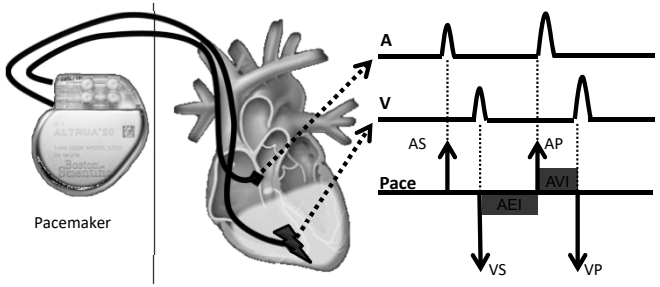
V-A DEADLINE AND ATRIAL PACING (AP)

- Pace atrium when no AS within deadline (track ventricular event)



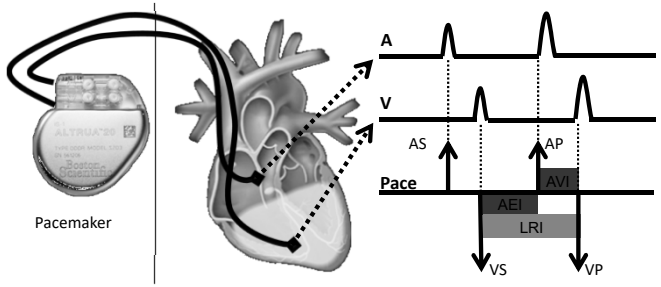
A-V DEADLINE AND VENTRICULAR PACING (VP)

- Pace ventricle if no VS happen within deadline (track atrial event)



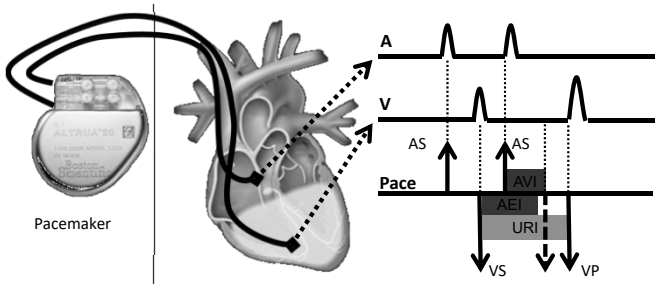
V-V DEADLINE

- Maximum interval between two ventricular events (max(V-A)+max(A-V))



MINIMUM VENTRICULAR PACING INTERVAL

- Upper Rate Limit (scheduled early VP (dashed) is delayed)



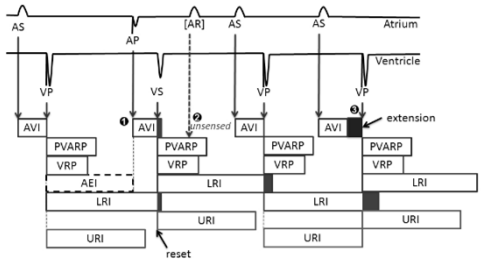
NOISE FILTERING

- Ventricular Refractory Period (VRP)
- Atrial Refractory Period (ARP)?
  - AVI period is also ARP
- Cross-talk between atrial channel and the ventricular channel
  - Ventricular events are strong enough to be sensed in the atrial channel
    - Post-Ventricular Atrial Refractory Period (PVARP): Inhibit atrial sensing after a ventricular event

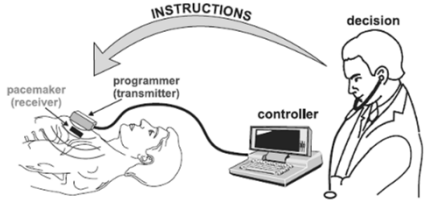
20

DDD Timing Cycles

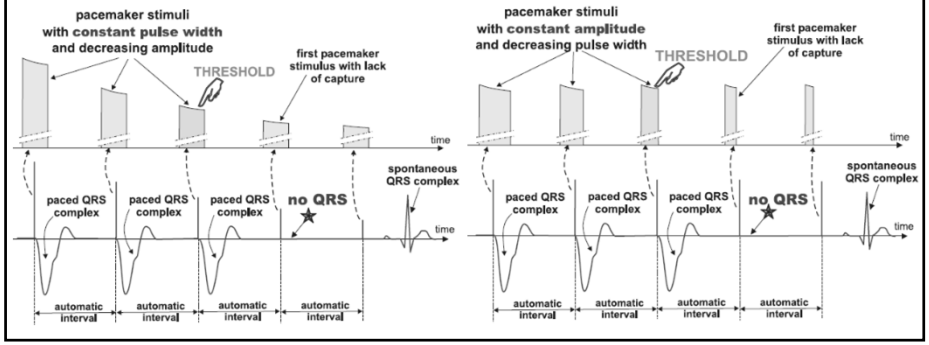
- 5 basic timing cycles



Pacemaker Programming



Adjusting Pacing Threshold



How good is the DDD pacemaker?

- We want to achieve: for all heart conditions H :

$$V(H + device) \geq V(H)$$

  - Efficacy: For"intended use", improve the heart condition as intended
  - Safety: For other heart conditions, don't make it worse!

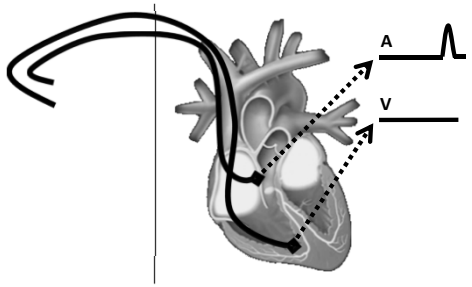
Pacemaker Mediated Tachycardia (PMT)

- Normal heart rate -> tachycardia with pacemaker
- SVT -> VT with pacemaker
- Atrial Tachycardia Response (ATR)
- Endless-Loop Tachycardia (ELT)

25

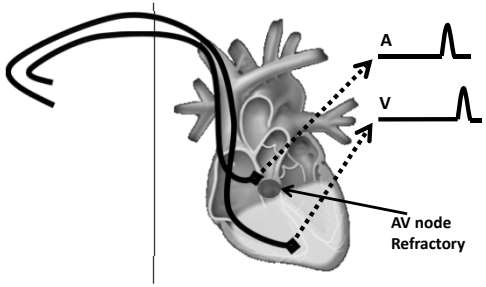
Atrial Tachycardia Response: Open-loop

- Atrial Tachycardia: Abnormally fast atrial rate



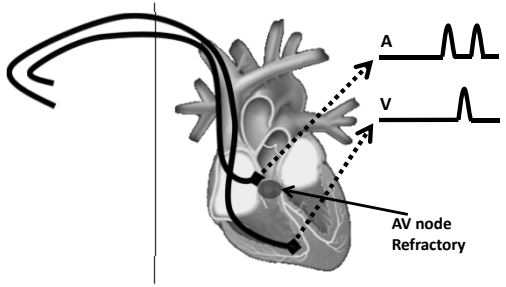
Atrial Tachycardia Response: Open-loop

- AV node enters Refractory period



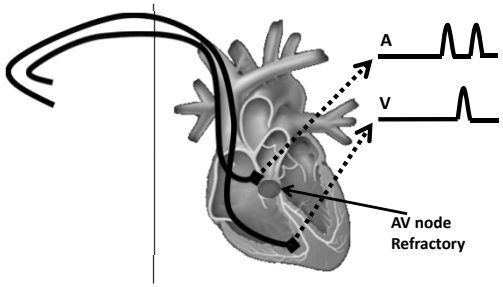
Atrial Tachycardia Response: Open-loop

- Another early atrial contraction



Atrial Tachycardia Response: Open-loop

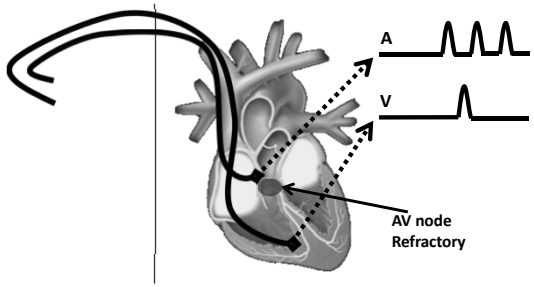
- Blocked by AV node during refractory



Medical Systems

Atrial Tachycardia Response: Open-loop

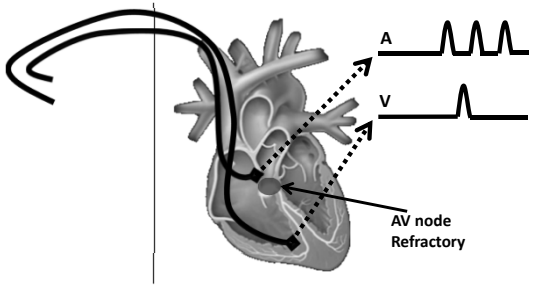
- Third atrial contraction



Medical Systems

Atrial Tachycardia Response: Open-loop

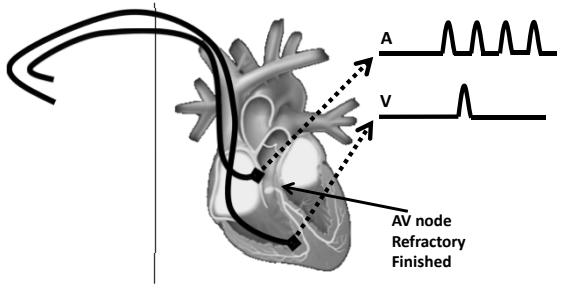
- Blocked by AV node again



Medical Systems

Atrial Tachycardia Response: Open-loop

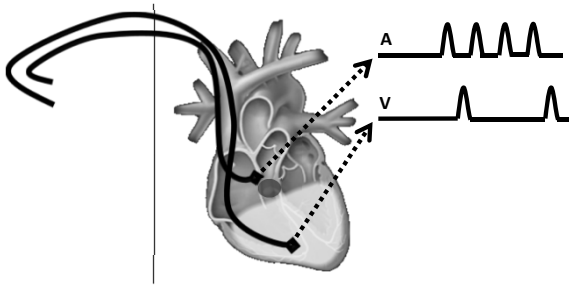
- AV node refractory finished



Medical Systems

ATRIAL TACHYCARDIA RESPONSE: OPEN-LOOP

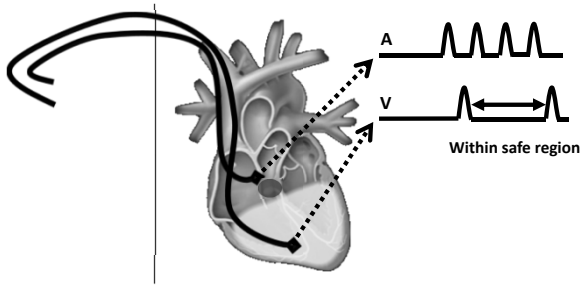
- 3:1 A-V conduction



Medical Systems

ATRIAL TACHYCARDIA RESPONSE: OPEN-LOOP

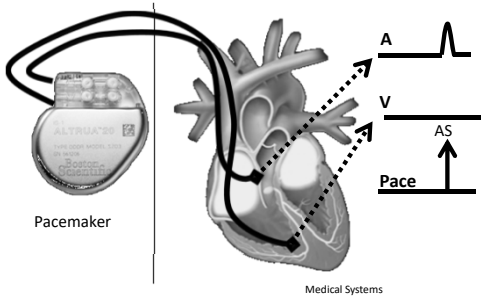
- Normal Ventricular rate in contrast to fast atrial rate



Medical Systems

ATRIAL TACHYCARDIA RESPONSE: CLOSED-LOOP

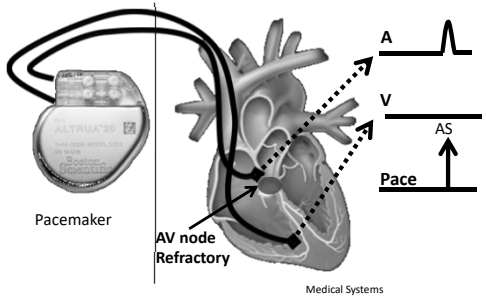
- Inappropriately increase ventricular rate



Medical Systems

ATRIAL TACHYCARDIA RESPONSE: CLOSED-LOOP

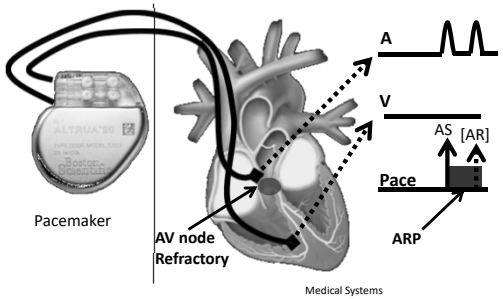
- Mimic **refractory** property of heart tissue



Medical Systems

ATRIAL TACHYCARDIA RESPONSE: CLOSED-LOOP

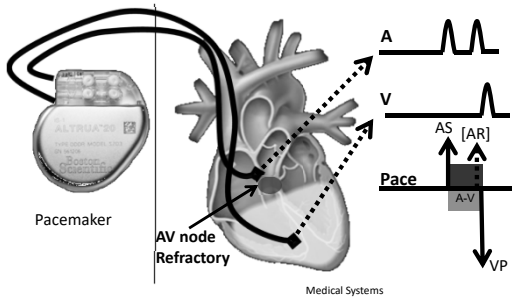
- Some of the fast atrial events are filtered



Medical Systems

ATRIAL TACHYCARDIA RESPONSE: CLOSED-LOOP

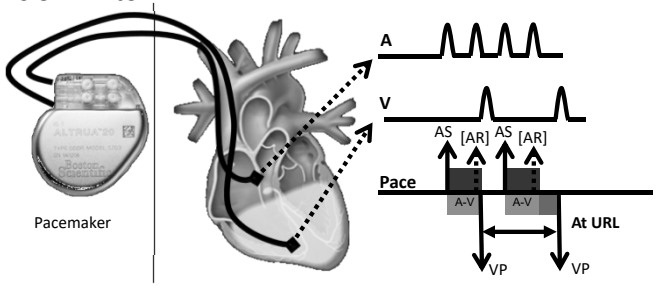
- VP when AV node in Refractory
- AV node function **bypassed**



Medical Systems

ATRIAL TACHYCARDIA RESPONSE: CLOSED-LOOP

- Ventricular rate at Upper Rate Limit
- Turns SVT into VT



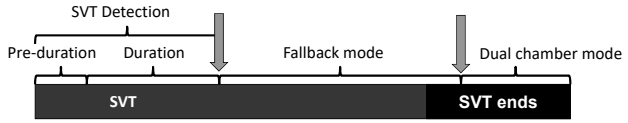
Medical Systems

ATRIAL TACHYCARDIA RESPONSE (ATR) TERMINATION ALGORITHM

- Detection of SVT
- Switch to single chamber pacing mode to detach A-V synchrony
- Switch back to dual chamber mode after AVT terminates

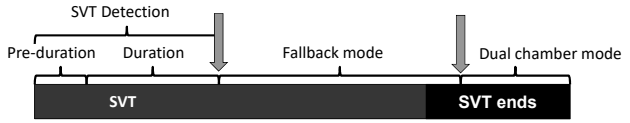
ATR: SVT Detection

- Pre-duration:
  - A timer measures the time between atrial events in and out of Refractory Period (AS, AR)
  - A counter +1 if the timer value is shorter than sensing threshold and -1 otherwise
  - When the counter value reaches 8, start *Duration*



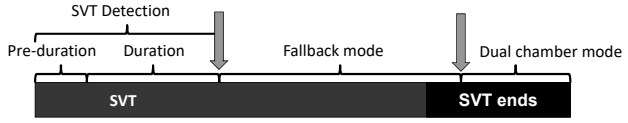
ATR: SVT Detection

- Duration:
  - The duration confirms the detection of SVT
  - If after Duration the counter value is still positive, the pacemaker switch to Fallback mode
  - If the counter value reaches 0 during Duration, the algorithm will exit Duration and reset the counter



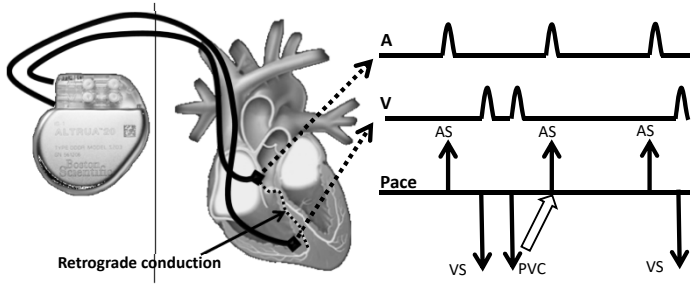
ATR: Fallback mode

- Fallback mode:
  - Single chamber mode which only sense and pace in the ventricle
  - Detach ventricular rate from fast atrial rate
  - At any time the counter reaches 0, the pacemaker assume the termination of SVT and switch back to dual chamber mode



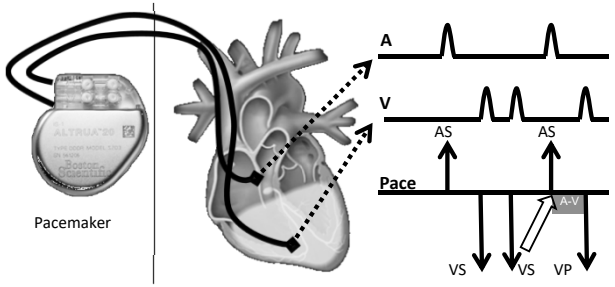
Endless Loop Tachycardia (ELT)

- Open-loop Behavior (ODO pacemaker)
  - Random heart event (rare) trigger retrograde conduction



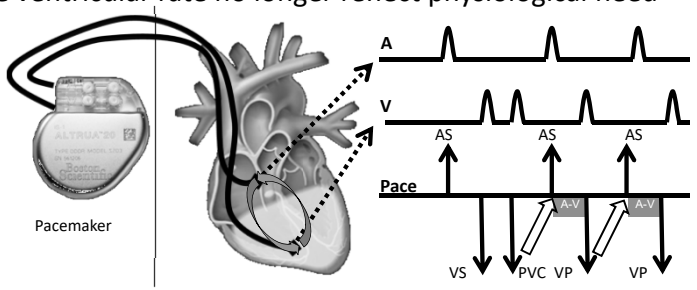
ENDLESS LOOP TACHYCARDIA (ELT)

- VP after A-V deadline



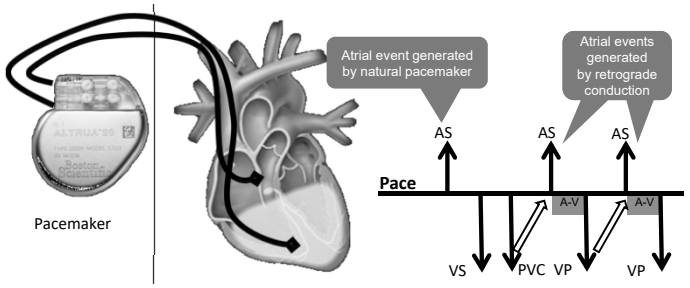
ENDLESS LOOP TACHYCARDIA (ELT)

- The VP-AS-VP pattern persists, causing ELT
- The ventricular rate no longer reflect physiological need

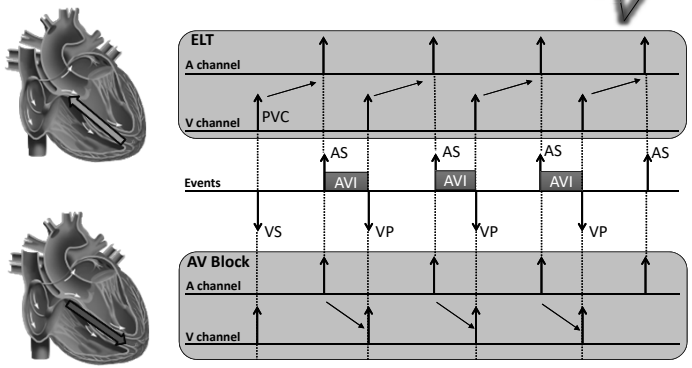


LIMITED SENSING CAPABILITY

- Same input sequence may correspond to different heart conditions

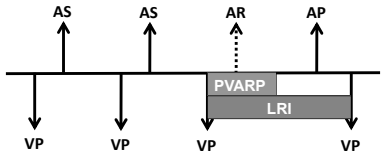


ENDLESS LOOP TACHYCARDIA (ELT)



ELT Detection & Termination

- V-A conduction delay is mostly fixed
- Monitor VP-AS interval when the ventricular rate is at Upper Rate Limit
- If 16 consecutive intervals are similar, extend PVARP to 500ms for one time



What went wrong during PMT?

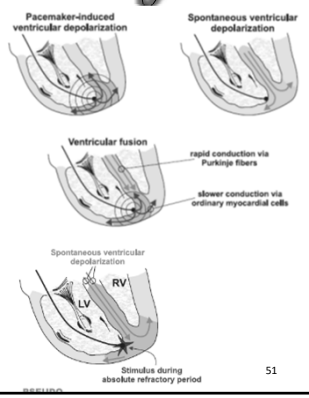
- Ventricular rate no longer reflect and satisfy physiological need
- DDD pacemaker introduced a virtual pathway between atria and ventricles
- Atrial Tachycardia Response
  - The pacemaker cannot distinguish atrial events from the SA node and the ones from atrial tachycardia
- Endless Loop Tachycardia (ELT)
  - The pacemaker cannot distinguish atrial events from the SA node and the ones from retrograde V-A conductions

50

Prioritized physiological requirements

1. Maintain appropriate ventricular rate
2. Maintain A-V synchrony
3. Encourage intrinsic electrical generation/conduction

- We can sacrifice a low priority requirement if only by doing so we can achieve a higher priority requirement



51

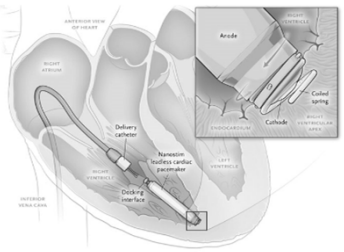
Are the termination algorithms correct?

- We will find out in the following lectures/homeworks.

52

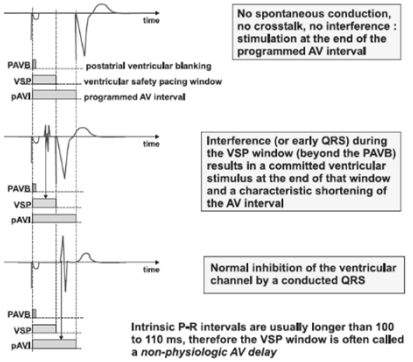
Other fun pacemaker functions

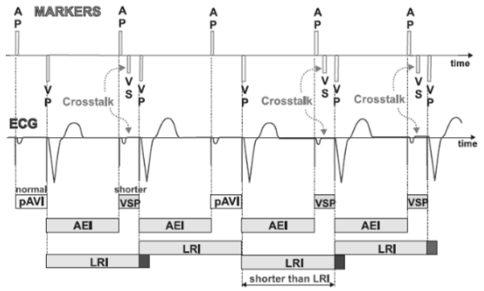
- Rate-adaptive pacemaker (DDDR)
  - Adapt pacing rate according to perceived physiological need
    - Accelerometer
    - Respiratory rate
- Pacemaker with 3 leads
  - Cardiac Resynchronization Therapy (CRT) Devices
  - Treat and prevent heart failure
- Leadless pacemaker



53

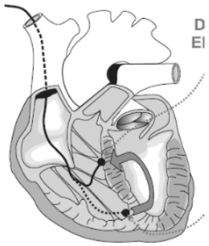
Ventricular Safety Pacing

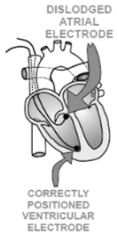


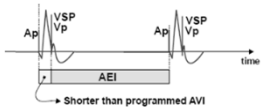


54

Weird Pacemaker Malfunctions

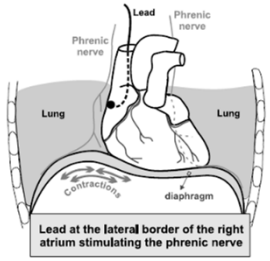






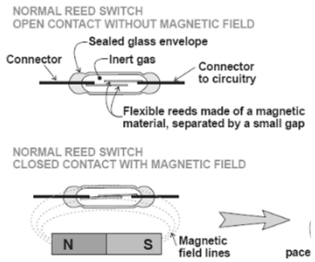
55

Hiccups caused by pacemaker



56

OBSOLETE: REED SWITCH



57

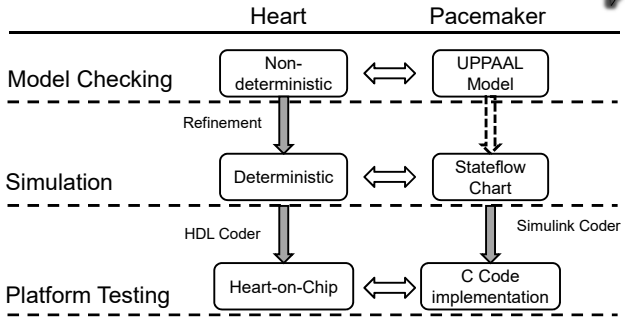
READING MATERIALS

- http://www.uppaal.org/
- Uppaal 4.0 : Small Tutorial∗

58

LECTURE 6: MODEL CHECKING

VERIFIED MODELS TO VERIFIED CODE



FORMAL VERIFICATION

Grand challenge:

Automate verification as much as possible !

ANALYSIS TECHNIQUES

- Dynamic Analysis (runtime)
  - Execute the system, possibly multiple times with different inputs
  - Check if every execution meets the desired requirement
- Static Analysis (design time)
  - Analyze the source code or the model for possible bugs
- Trade-offs
  - Dynamic analysis is incomplete, but accurate (checks real system, and bugs discovered are real bugs
  - Static analysis can catch design bugs early !
  - Many static analysis techniques are not scalable (solution: analyze approximate versions, can lead to false warnings)

Verification Methods

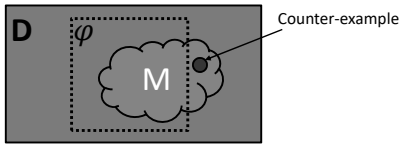
- Simulation
  - Simulate the model, possibly multiple times with different inputs
  - Easy to implement, scalable, but no correctness guarantees
- Proof based
  - Construct a proof that system satisfies the invariant
  - Requires manual effort (partial automation possible)
- State-space analysis (Model checking)
  - Algorithm explores "all" reachable states to check invariants
  - Not scalable, but current tools can analyze many real-world designs (relies on many interesting theoretical advances)

Different Requirements

- Safety
  - A system always stays within "good' states (i.e. a nothing bad ever happens)
  - Leader election: it is never the case that two nodes consider them to be leaders
  - Collision avoidance: Distance between two cars is always greater than some minimum threshold
- Liveness
  - System eventually attains its goal
  - Leader election: Each node eventually makes a decision
  - Cruise controller: Actual speed eventually equals desired speed
  - A car will always eventually reach its destination

Model Checking

- A domain D representing the state space of a model
- The reachable state space M for the model
- Define a subset of the state space as property φ
- Explore the whole reachable state space of a model for property violations
- Widely used in semi-conductor industries for verifying chip design

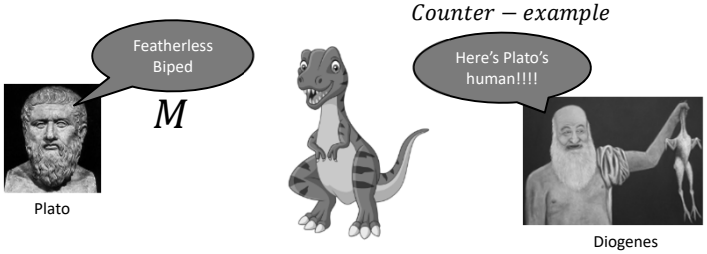


Plato vs. Diogenes

- The definition of "*human*"

φ All living creatures D

$Counter - example$



Plato

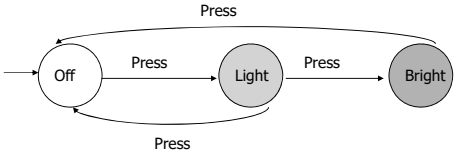
Diogenes

CHALLENGE

- State space explosion
  - Not every model can be model checked!!
  - i.e. Real-number (continuous) state space
  - Complex dynamics between states
- Solution
  - Simple yet expressive formalisms
  - Symbolic states/executions
  - Model abstraction/approximation

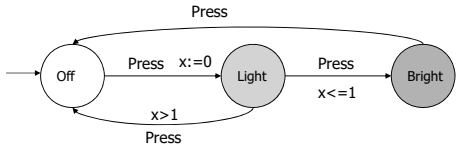
9

SIMPLE LIGHT CONTROL



WANT: if press is issued twice quickly
then the light will get brighter; otherwise the light is turned off.

SIMPLE LIGHT CONTROL

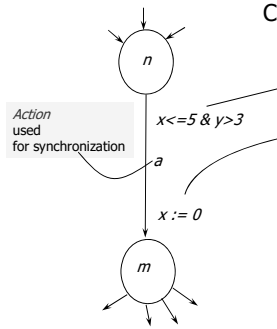


Solution: Add a real-valued clock x

Adding continuous variables to state machines

TIMED AUTOMATA

Clocks: *x, y*



Guard
Boolean combination of comparisons with Integer/rational bounds

Action
used
for synchronization

Reset
Action performed on clocks

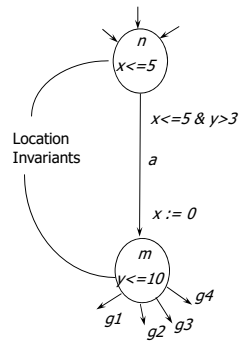
State
(*location* , *x*=v , *y*=u) where v,u are in R

Transitions

(*n* , *x*=2.4 , *y*=3.1415) $\xrightarrow{a}$ (*m* , *x*=0 , *y*=3.1415)

(*n* , *x*=2.4 , *y*=3.1415) $\xrightarrow{wait(1.1)}$ (*n* , *x*=3.5 , *y*=4.2415)

ADDING INVARIANTS



Clocks: *x, y*

Transitions

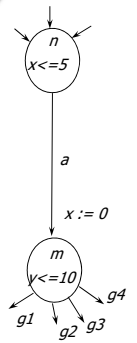
(*n* , *x*=2.4 , *y*=3.1415) ~~wait(3.2)~~ → (*n* , *x*=3.5 , *y*=4.2415)

(*n* , *x*=2.4 , *y*=3.1415) wait(1.1) → (*n* , *x*=3.5 , *y*=4.2415)

Invariants ensure progress!!

TIMED AUTOMATA: SYNTAX

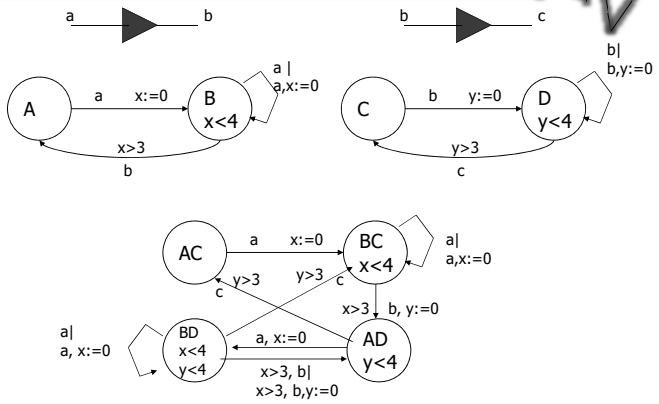
- A finite set *V* of locations
- A subset V^0 of initial locations
- A finite set Σ of labels (alphabet)
- A finite set *X* of clocks
- Invariant *Inv(l)* for each location: (clock constraint over *X*)
- A finite set *E* of edges. Each edge has
  - source location *l*, target location *l'*
  - label *a* in Σ (ε labels also allowed)
  - guard *g* (a clock constraint over *X*)
  - a subset λ of clocks to be reset



TIMED AUTOMATA: SEMANTICS

- For a timed automaton *A*, define an infinite-state transition system *S(A)*
- States *Q*: a state *q* is a pair *(l,v)*, where *l* is a location, and *v* is a clock vector, mapping clocks in *X* to *R*, satisfying *Inv(l)*
- *(l,v)* is initial state if *l* is in V^0 and *v(x)=0*
- Elapse of time transitions: for each nonnegative real number *d*, *(l,v)-d->(l,v+d)* if both *v* and *v+d* satisfy *Inv(l)*
- Location switch transitions: *(l,v)-α->(l',v')* if there is an edge *(l,α,g,λ,l')* such that *v* satisfies *g* and *v'=v[λ:=0]*

PRODUCT CONSTRUCTION

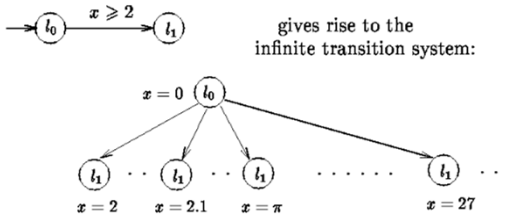


Model Checking: Forward Reachability

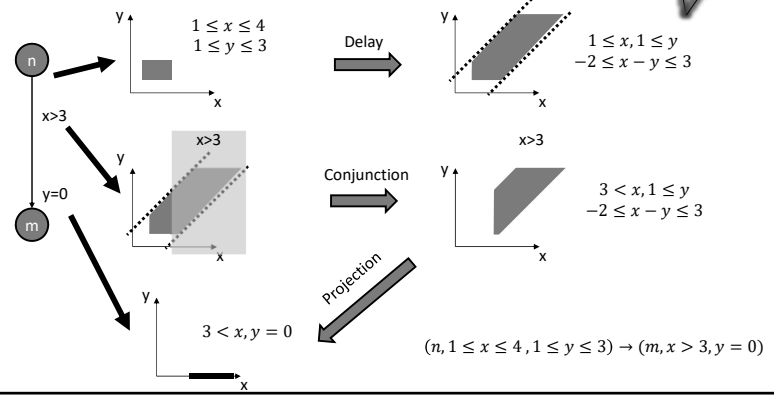
- Given a timed automata and a property φ
- R:=I
- Repeat
  - If R intersects $\neg\varphi$, report "yes"
  - Else if R contains Post(R), report "no"
  - Else R := R union Post(R)

17

Reachability for Timed Automata



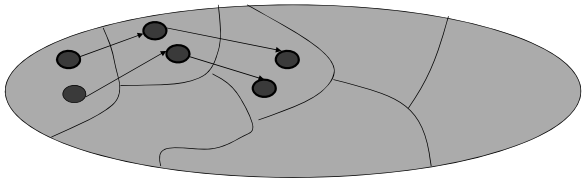
Symbolic States/Executions



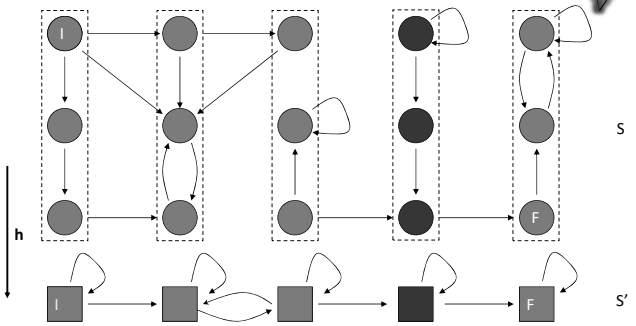
19

Finite Partitioning

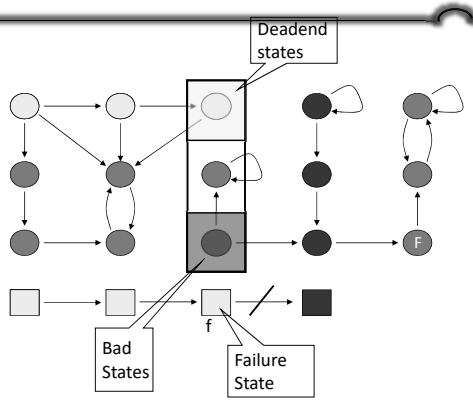
Goal: To partition state-space into finitely many equivalence classes so that equivalent states exhibit similar behaviors



EXISTENTIAL ABSTRACTION (OVER-APPROXIMATION)



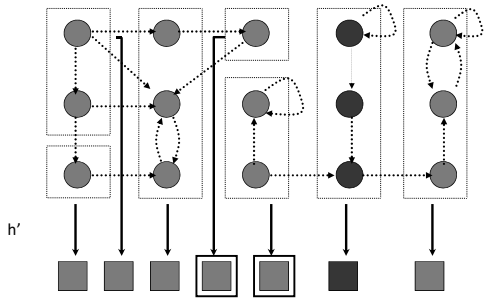
WHY SPURIOUS COUNTEREXAMPLE?



REFINEMENT

- Problem: Deadend and Bad States are in the same abstract state.
- Solution: Refine abstraction function.
- The sets of Deadend and Bad states should be separated into different abstract states.

REFINEMENT

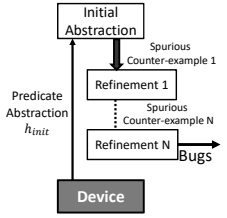


Refinement : h'

Counter-Example-Guided Abstraction and Refinement (CEGAR)

- Obtain initial abstraction

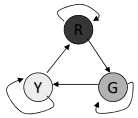
1. Model checking
2. Property satisfied -> no bugs
3. Property unsatisfied -> counter-examples
4. Check whether the CE is spurious
5. If not, bug found
6. If yes, refine the model and start from 1 again



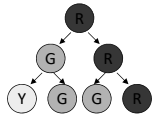
25

Computational Tree Logic (CTL)

- CTL is a logic used to express properties for model checking
- CTL is useful because there is an efficient technique to check it
- A temporal logic is a logic which can express aspects of time
- CTL makes statements about the computational tree of a state machine



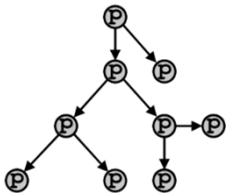
Traffic light FSM



Computational tree for FSM

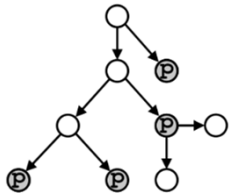
Temporal Computational Tree Logic (TCTL) Properties

- **A[] p** "Always globally p"
- For each (all) execution path p holds for all the states of the path



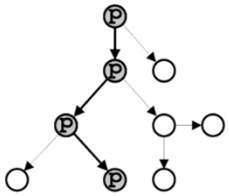
Properties

- **A<> p** "Always eventually p"
- For each (all) execution path p holds for at least one state of the path



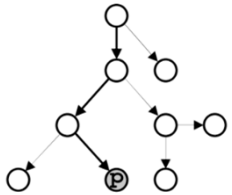
PROPERTIES

- **E[] p** "Exists globally p" meaning there is an execution path in which p holds for all the states of the path



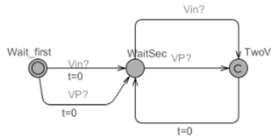
PROPERTIES

- **E<> p** "Exists eventually p" meaning there is an execution path in which p eventually (in some state of the path) holds



MONITORS

- Sometimes we need assistance to express our requirements
- The minimum ventricular rate should be 60bpm
- The maximum interval between two ventricular events (VS,VP) should be no larger than 1000ms
- A[] (PM.TwoV imply PM.t<=1000)



31

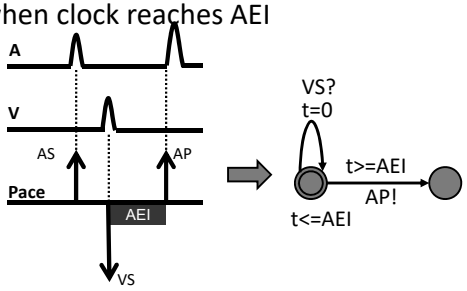
PHYSIOLOGICAL REQUIREMENTS

- Does the device always deliver therapy when needed?
- Does the device always not deliver therapy when unnecessary?
- Behavior Coverage (Generality)
- Physiological Context (Interpretability)

32

Modeling Timing Delays With UPPAAL

- Reset the clock when VS occurs
- Send AP when clock reaches AEI



Specification **Timed Automaton**

33

What should we cover for pacemaker?

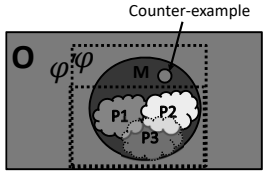
- Heart conditions
- Individual variabilities
- Noise

- How to use finite number of heart models to cover infinite number of heart conditions?

34

Capture Physiological Variability With Over-approximation

- Properties satisfied by M are also satisfied by P1, P2
- Behaviors not exist in P1, P2 may also be physiologically-valid
- Is this a valid counter-example?
- Need a framework to provide context for counter-examples



35

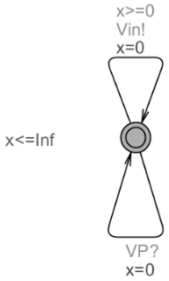
Observable Behavioral Space of a Pacemaker

- Atrial channel: Boolean
- Ventricular channel: Boolean
- Time: Real number
- The combination of the factors above

36

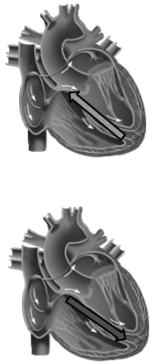
RANDOM HEART MODEL

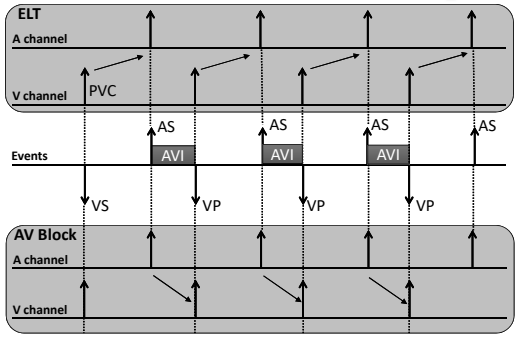
- Atrial and ventricular events can happen at any time
  - Intervals between two consecutive events are between [0,inf]



37

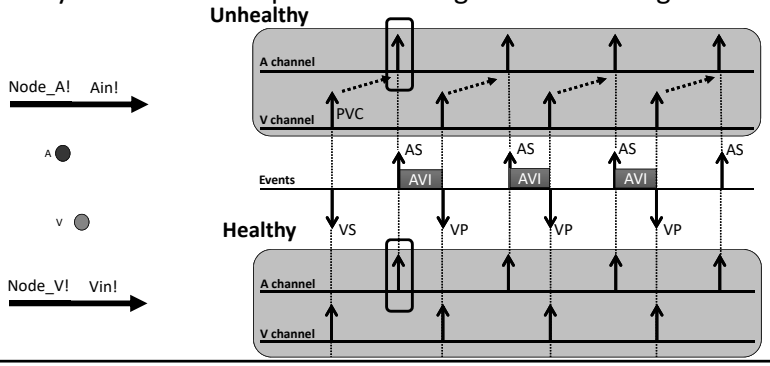
CANNOT DISTINGUISH CERTAIN SCENARIOS





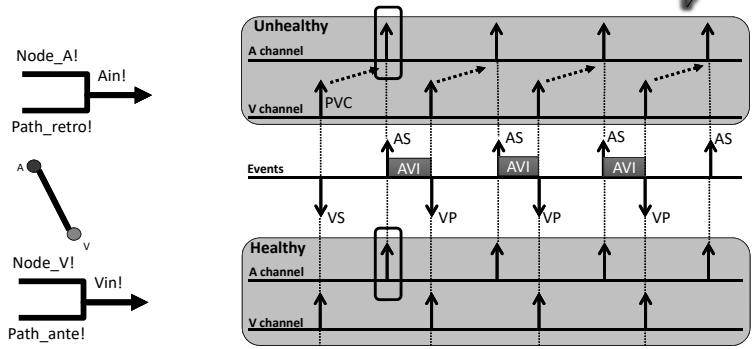
AMBIGUITY WITH ABSTRACT MODEL

May introduce false-positives during model checking



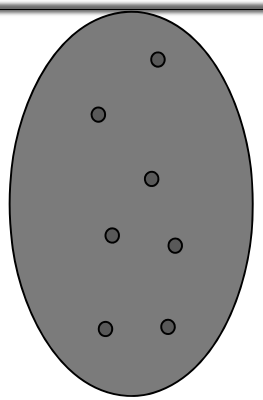
39

HEART MODEL REFINEMENT TO RESOLVE AMBIGUITY

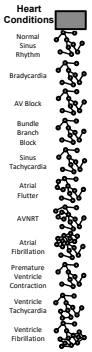


40

Initial Heart Models



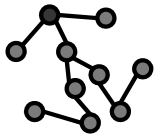
Observable Behavior Space

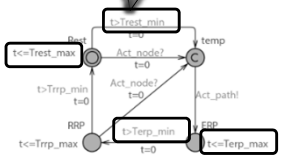


41

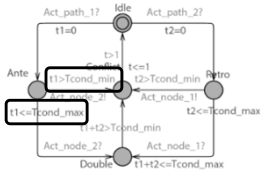
Heart models for closed-loop model checking

- Capture variability with non-determinism
  - Rest period (Heart Rate)
  - ERP period (Blocking)
  - Conduction delay





Node automata

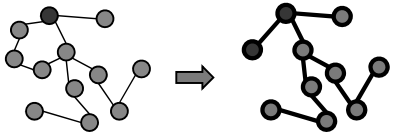


Path automata

42

Physiological Abstraction Rules

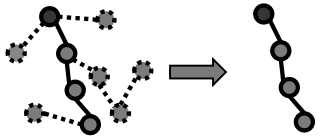
- R1: Convert reentry circuits to self-depolarization nodes



43

Physiological Abstraction Rules

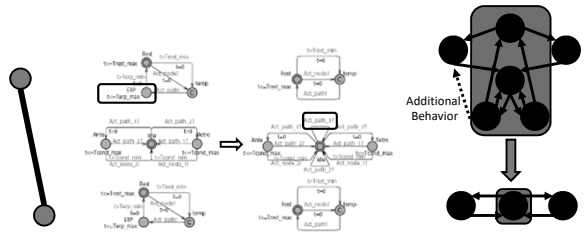
- R2: Ignore structures that do not affect observable behaviors



44

PHYSIOLOGICAL ABSTRACTION RULES: SIMPLIFY

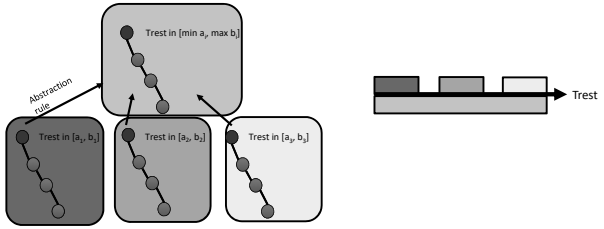
- R6: Simplify the model to increase non-determinism for more behavior coverage



Will ignore activations at specific time **May ignore activations at any time**

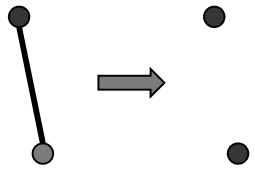
PHYSIOLOGICAL ABSTRACTION RULES: COMBINE

- R4: Combine models of different heart conditions for more behavior coverage

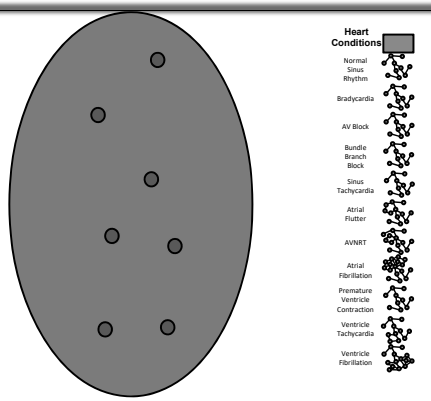


PHYSIOLOGICAL ABSTRACTION RULES

- R7: Use self-depolarization to replace conducted depolarization

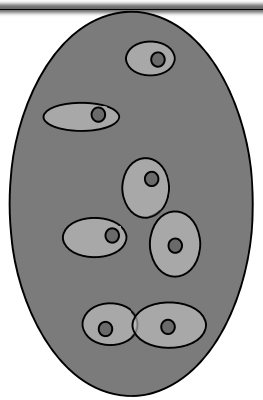


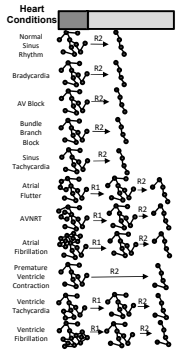
INITIAL HEART MODELS



Observable Behavior Space

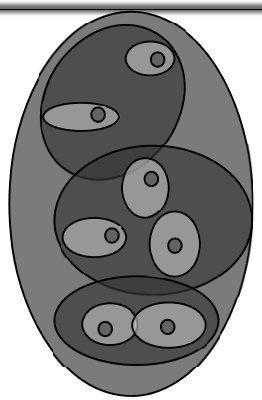
Abstraction Tree: Heart Model Abstraction

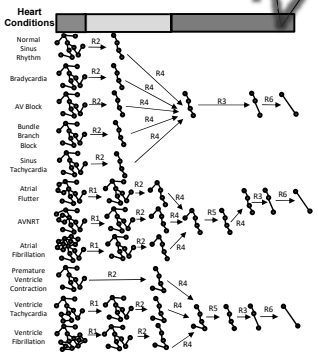




Observable Behavior Space

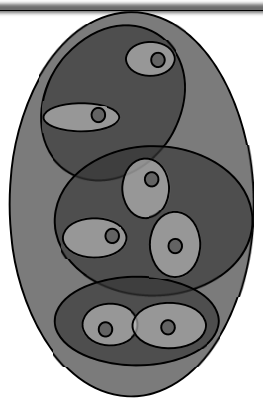
Abstraction Tree: Heart Model Abstraction

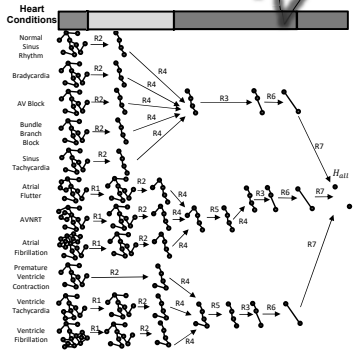




Observable Behavior Space

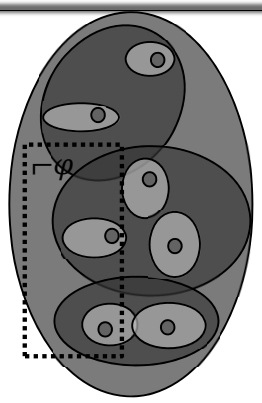
Abstraction Tree: Heart Model Abstraction

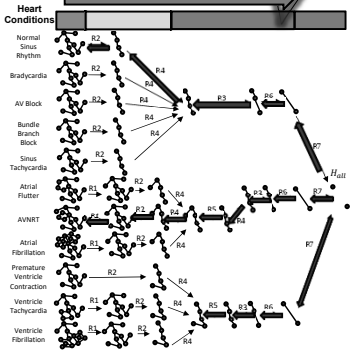




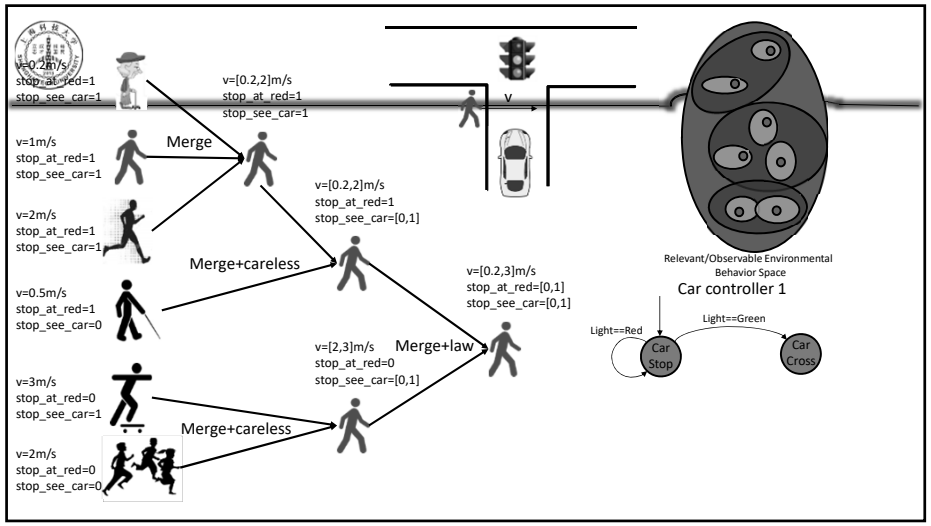
Observable Behavior Space

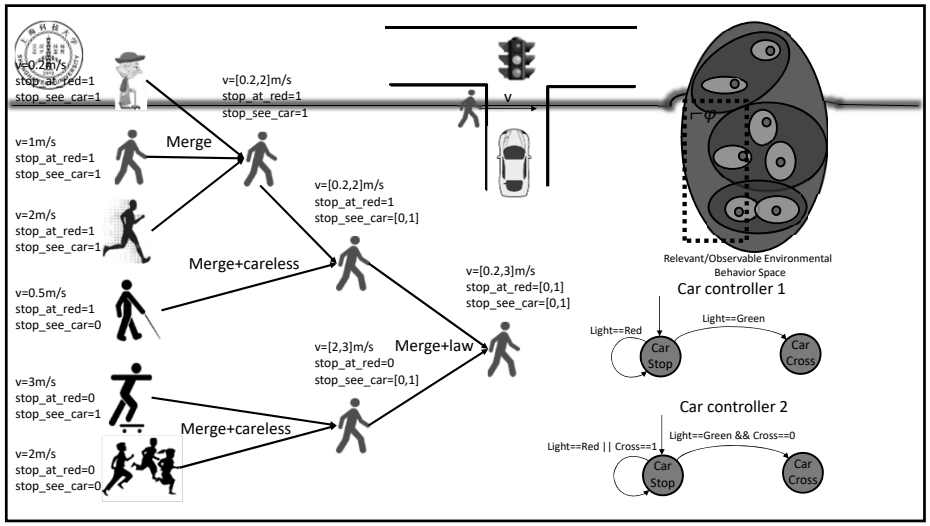
Abstraction Tree: Heart Model Refinement





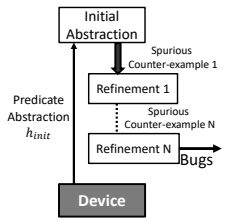
Observable Behavior Space

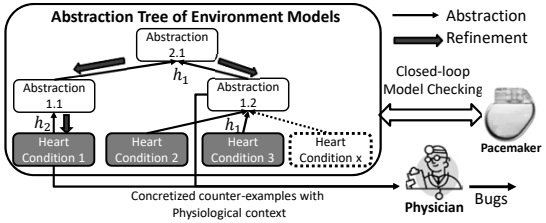




ABSTRACTION & REFINEMENT

- Abstracting the system vs. abstracting the environment
- No (single) ground truth to check whether the CE is spurious





COMING NEXT:

- UPPAAL Tutorial

56

UPPAAL TUTORIAL

Partially referenced Prof. Insup Lee's course at UPenn

UPPAAL??!!

- Model checking tool for Timed-automata
- Developed by Uppsala University and Aalborg University

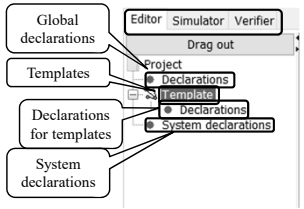
- **SWE**den + **DEN**mark = SWEDEN
  - REJECTED
- swe**DEN** + den**MARK** = DENMARK
  - REJECTED
- **UPP**sala + **AAL**borg = UPPAAL
  - ACCEPTED

UPPAAL TOOL PARTS

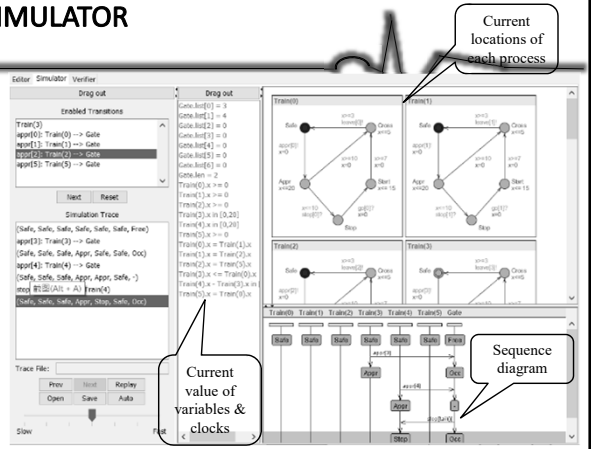
- Graphical user interface (GUI)
  - Used for modeling, simulation, and verification. Uses the verification server for simulation and verification.
- Verification server
  - Used for simulation and verification. In simulation, it is used to compute successor states.
- A command line tool
  - A stand-alone verifier, appropriate for e.g. batch verifications.

UPPAAL GUI: EDITOR

- Global declarations
  - Accessible to all system processes
- Templates
  - Can be parameterized
- Declarations for templates
  - Only accessible locally
- System declarations
  - Declare processes and system composition

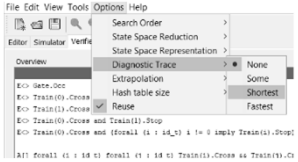


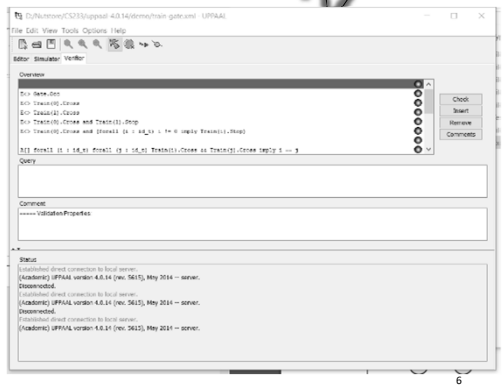
UPPAAL GUI: SIMULATOR



UPPAAL GUI: VERIFIER

- Turn on diagnostic trace to view counter-examples in the simulator





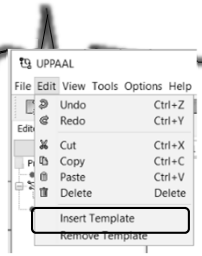
UPPAAL SYNTAX

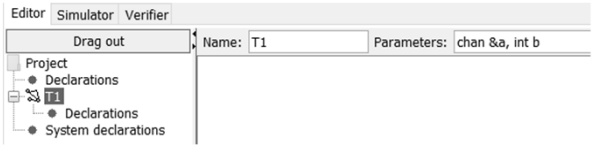
UPPAAL SYNTAX

- Global declarations
  - Clocks:
    - clock x1,…xn;
  - Data variables
    - int n1,…; integer with default domain
    - Int[l,u] n1,… ; integer with domain defined by [l,u]
    - Int n1[m],…; array with elements n1[0] to n1[m-1]
  - Channels
    - Chan a, … ;
    - Urgent chan b … ;
    - Broadcast chan c … ;
  - Constants
    - Const int c1=n1;

UPPAAL Syntax (cont.)

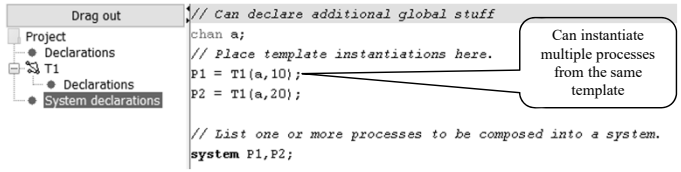
- Template
  - Names should be unique
  - Channel declaration has a “&” in front





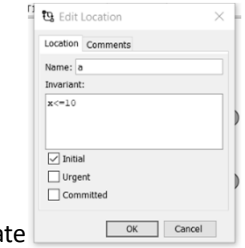
UPPAAL Syntax (cont.)

- System declaration



UPPAAL Syntax: Locations

- Initial Location
  - Only one per template
- Urgent Location
- Committed Location
- Invariant
  - Conditions that need to be satisfied when in state



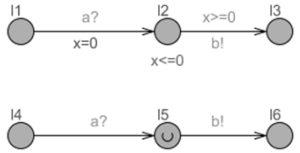
UPPAAL Syntax: Edges

- Select
  - Defines multiple parameterized transitions
- Guard
  - Condition under which the edge is enabled
- Sync
  - a! for sending message
  - a? for receiving message
- Update
  - Actions taken during the transition

UPPAAL SEMANTICS

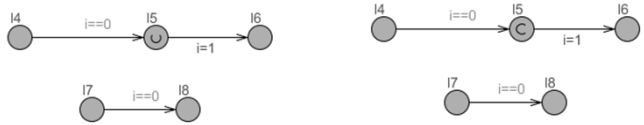
URGENT LOCATION

- No time pass in an urgent location
- The two automata is equivalent
- Save a clock thus reduce state space



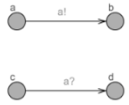
COMMITTED LOCATION

- Urgent location still allows interleaving
  - l7 -> l8 can happen before l5->l6, although no time has passed
- Committed states are stronger than urgent locations
- If multiple committed states reached at the same time, the transitions will interleave
- Reduce interleaving thus reduce complexity



URGENT CHANNELS

- Urgent channel definition
  - Urgent chan a;
- a! sent as soon as location a and c are reached
- No clock guard is allowed on the transitions with urgent channel components

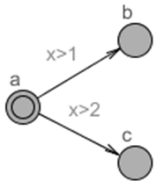


BROADCAST CHANNELS

- Synchronize multiple processes
- If receiving channel a? is enabled, the transition must be taken
- Can send without any receivers ready

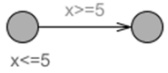
NON-DETERMINISM

- Transitions with guard evaluates true are only enabled
- Used to model variabilities of system environment
- Location a does not have an invariant, thus can stay forever
  - $x \in [0,1]$: location a
  - $x \in (1,2]$: location a or b
  - $x \in (2,\infty]$: location a or b or c



WANT TO DO SOMETHING AT A PARTICULAR TIME?

- The transition will take place when x==5

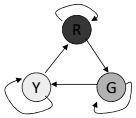


DEADLOCK

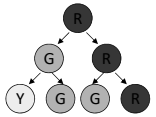
- No enabled transitions
- Common deadlock scenarios
  - Cannot enter a state
  - State invariant about to expire and no enabled transition available
  - Committed state does not have enabled outgoing transition

Computational Tree Logic (CTL)

- CTL is a logic used to express properties for model checking
- CTL is useful because there is an efficient technique to check it
- A temporal logic is a logic which can express aspects of time
- CTL makes statements about the computational tree of a state machine



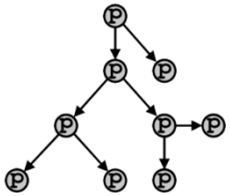
Traffic light FSM



Computational tree for FSM

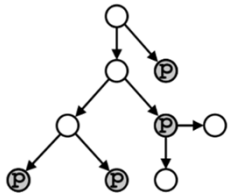
Temporal Computational Tree Logic (TCTL) Properties

- **A[] p** "Always globally p"
- For each (all) execution path p holds for all the states of the path



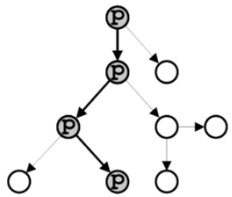
Properties

- **A<> p** "Always eventually p"
- For each (all) execution path p holds for at least one state of the path



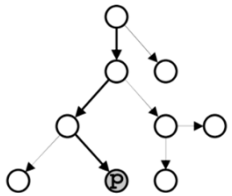
Properties

- **E[] p** "Exists globally p" meaning there is an execution path in which p holds for all the states of the path



PROPERTIES

- **E<> p** "Exists eventually p" meaning there is an execution path in which p eventually (in some state of the path) holds

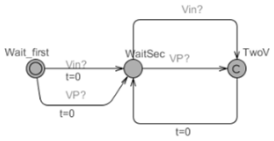


UPPAAL TCTL PROPERTIES

- A[]p, A<>p, E<>p, E[]p, p-->p'(p imply p')
- p can be
  - Location of a process: a.l
  - Data guard
  - Clock guard
  - p and p'
  - p or p'
  - Not p
  - p imply p'

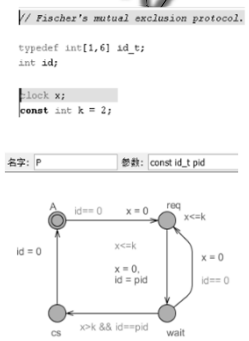
MONITORS

- Sometimes we need assistance to express our requirements
- The maximum interval between two ventricular events (Vin,VP) should be no larger than 1000ms
- A[] (PM.TwoV imply PM.t<=1000)



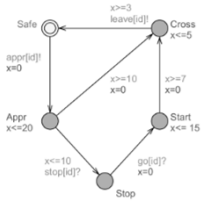
EXAMPLE: MUTUAL EXCLUSION PROTOCOL

- Only one component can access the critical section (cs) which represents resource
- Use global variable id to communicate with each other
- If there are two components at critical section together, their id must be the same
  - A[] forall (i:id_t) forall (j:id_t) P(i).cs && P(j).cs imply i == j



EXAMPLE: TRAIN-GATE

- Each train has an id
- Each train can approach the gate at any time
- Approaching takes 10-20 sec
- The gate controller can stop a train within 10 sec after its approaching, otherwise the train will cross
- After receive a GO signal, the train will start within 7-15 sec
- Crossing takes 3-5 sec

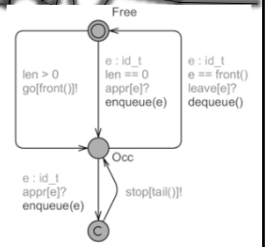


EXAMPLE: TRAIN-GATE (CONT.)

- Gate controller maintains a queue
- If queue empty and a train approaches, gate stay occupied
- If the gate is occupied and a train approaches, stop the last one in queue
- If the train at the front of the queue leaves, remove it from the queue
- If the gate is free and there are trains in queue, let the front one go

```
typedef int[0,N-1] id_t;

// Put an element at the end of the queue
void enqueue(id_t element)
{
        list[len++] = element;
}
```

```
// Remove the front element of the queue
void dequeue()
{
        int i = 0;
        len -= 1;
        while (i < len)
        {
                list[i] = list[i + 1];
                i++;
        }
        list[i] = 0;
}
```




```
// Returns the front element of the queue
id_t front()
{
    return list[0];
}

// Returns the last element of the queue
id_t tail()
{
    return list[len - 1];
}
```


EXAMPLE: TRAIN-GATE (CONT.)

- E<> Train(0).Cross
  - Train 0 can eventually cross
- E<> Train(0).Cross and Train(1).Stop
  - Train 0 can be crossing bridge while Train 1 is waiting to cross
- E<> Train(0).Cross and (forall (i:id-t) i != 0 imply Train(i).Stop)
  - Train 0 can cross bridge while the other trains are waiting to cross
- A[] Gate.list[N] == 0
  - There can never be N elements in the queue
- A[] forall (i:id-t) forall (j:id-t) Train(i).Cross && Train(j).Cross imply i == j
  - There is never more than one train crossing the bridge
- Train(1).Appr -> Train(1).Cross
  - Whenever a train approaches the bridge, it will eventually cross
- A[] not deadlock
  - The system is deadlock-free

HW 2 OUT

- Due 10/18 at the end of the day

HW 1 Review

Grading Criteria

- 10 pts total
- 1-3: 1pt each
  - Need to plot the result
- 4: 2pts
- 5: 5pts
  - 5.1, 5.2: 1pt each
    - Identify correct pathway: 0.2
    - Answer contains relative conduction speed or signal cancellation: 0.8
  - 5.3: 3pts
    - Logic: 2pts
    - Rate: 1pt

Common Problems

- Unlink Library

When you try to modify a copied component, Simulink might protest and ask whether you want to "unlink" the component from the library. Do that.

- AV block problem

Use blocks `Node`, and `Path` to create a model which has a fast atrial rate, but such that the AV node is blocking 1 out of every 2 atrial beats, thus keeping a 2:1 ratio between atria and ventricles.

Submission Format

- Please submit .zip file with name: HWx_YourName.zip

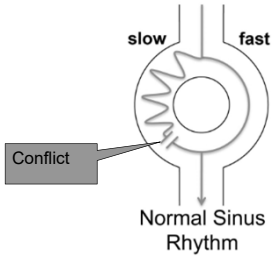
Submission

A .zip file containing the following files:

- NPNAntegradeOnly.slx
- Bradycardia.slx
- Tachycardia.slx
- AVblock.slx
- Reentry.slx

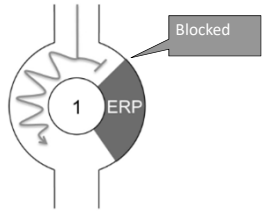
MECHANISM: REENTRY TACHYCARDIA

- Equivalent to the fast pathway alone



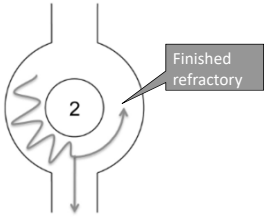
MECHANISM: REENTRY TACHYCARDIA

- Premature activation blocked in the fast pathway



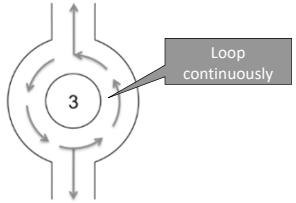
MECHANISM: REENTRY TACHYCARDIA

- Retrograde conduction through fast pathway creating echo beat

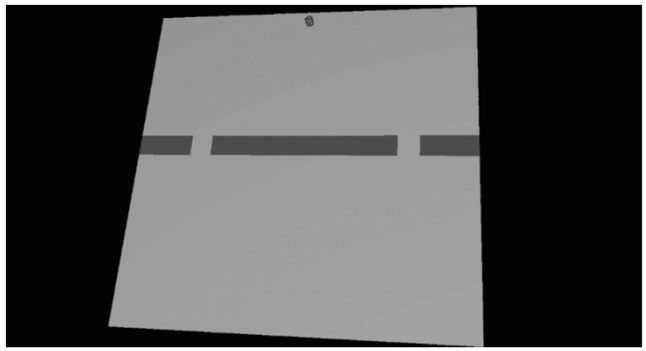


MECHANISM: REENTRY TACHYCARDIA

- Reentry induced and maintain high atrial and ventricle rate



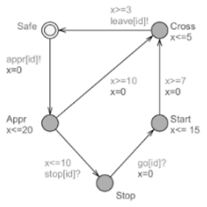
Reentry Mechanism: difference in refractory period



UPPAAL Tutorial (Cont)

Example: Train-Gate

- Each train has an id
- Each train can approach the gate at any time
- Approaching takes 10-20 sec
- The gate controller can stop a train within 10 sec after its approaching, otherwise the train will cross
- After receive a GO signal, the train will start within 7-15 sec
- Crossing takes 3-5 sec

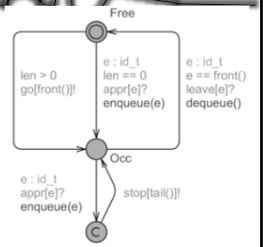


Example: Train-Gate (cont.)

- Gate controller maintains a queue
- If queue empty and a train approaches, gate stay occupied
- If the gate is occupied and a train approaches, stop the last one in queue
- If the train at the front of the queue leaves, remove it from the queue
- If the gate is free and there are trains in queue, let the front one go

```
typedef int[0,N-1] id_t;

// Put an element at the end of the queue
void enqueue(id_t element)
{
        list[len++] = element;
}
```

```
// Remove the front element of the queue
void dequeue()
{
        int i = 0;
        len -= 1;
        while (i < len)
        {
                list[i] = list[i + 1];
                i++;
        }
        list[i] = 0;
}
```

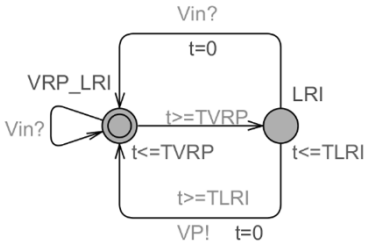



```
// Returns the front element of the queue
id_t front()
{
    return list[0];
}

// Returns the last element of the queue
id_t tail()
{
    return list[len - 1];
}
```

EXAMPLE: TRAIN-GATE (CONT.)

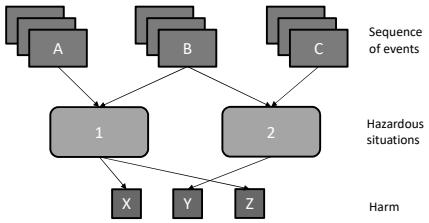
- E<> Train(0).Cross
  - Train 0 can eventually cross
- E<> Train(0).Cross and Train(1).Stop
  - Train 0 can be crossing bridge while Train 1 is waiting to cross
- E<> Train(0).Cross and (forall (i:id-t) i != 0 imply Train(i).Stop)
  - Train 0 can cross bridge while the other trains are waiting to cross
- A[] Gate.list[N] == 0
  - There can never be N elements in the queue
- A[] forall (i:id-t) forall (j:id-t) Train(i).Cross && Train(j).Cross imply i == j
  - There is never more than one train crossing the bridge
- Train(1).Appr -> Train(1).Cross
  - Whenever a train approaches the bridge, it will eventually cross
- A[] not deadlock
  - The system is deadlock-free

VVI EXAMPLE



IDENTIFY SEQUENCES OF EVENTS THAT CAN LEAD TO HAZARDOUS SITUATIONS

- Hazardous situations are relatively easy to identify
- What are the sequences of events that can lead to the hazardous situation?
- What's the frequency of hazardous situations?



QUANTITATIVE ANALYSIS

So far we only answered Yes/No questions

- There is need for quantitative verification
  - Quantify uncertainty
    - How often does bad events happen?
  - Quantify performance
    - What's the minimum battery consumption?
- There are tools available to evaluate
  - Probability
  - Cost/reward

UPPAAL SMC

- Statistical Model Checking (SMC)
  - Non-exhaustive evaluation of the model's state space
  - Through statistical simulations within certain time bound
- Statistical Timed Automata
  - Resolve non-determinism with stochastic behaviors
  - Based on Monte Carlo Simulation

Monte Carlo Simulation

Suppose you timed 20 athletes running the 100-yard dash and tallied the information into the four time intervals below.

You then count the tallies and make a frequency distribution.

Then convert the frequencies into percentages.

Finally, use the percentages to develop the random number intervals.

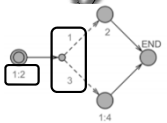
Seconds	Tallies	Frequency	%	RN Intervals
0-5.99	𝍸	4	20	01-20
6-6.99	𝍸 𝍸	10	50	21-70
7-7.99	𝍸	4	20	71-90
8 or more	||	2	10	91-100

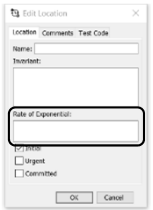
Monte Carlo Simulation: NBA Draft

- 14 ping pong balls numbered 1 through 14 are placed in a drum.
  - C_{14}^{4}=1,001
- Prior to the Lottery, 1,000 combinations are assigned to the Lottery teams based on their order of finish during the regular season.
  - The worst team has 250 combinations (25% chance for No.1 pick)
- 4 balls are drawn from the drum with a combination
- The team that has been assigned that combination will receive the number one pick.
- The four balls are placed back in the drum and the process is repeated to determine the number two and three picks.

UPPAAL SMC: New Syntax

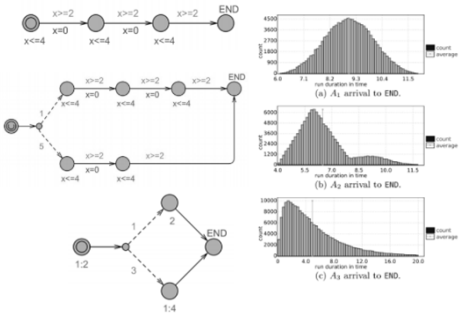
- Probabilistic transition
  - Resolves nondeterminism
  - i.e. ¼ chance going up, ¾ chance going down
- Rate of Exponential
  - "How eager you want to exit the state"





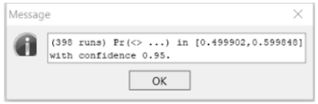
UPPAAL SMC: Semantics

- The time it takes to reach END
- Uniform distribution
  - Transition out at time 2 and time 3 are equal
- Probabilistic transition
- Exponential distribution



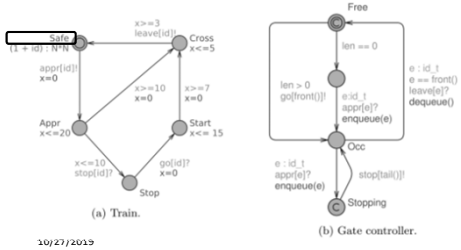
UPPAAL SMC New Queries

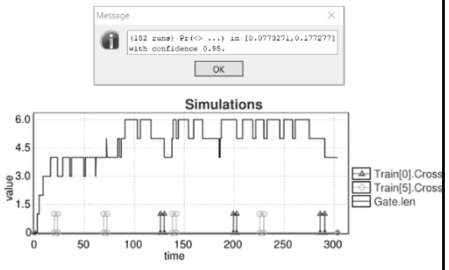
- Simulation
  - simulate N [<=bound] { E1,..,Ek }
- Probability Estimation
  - Pr[bound](<> psi)
- Hypothesis Testing
  - Pr[bound](psi) >= p_0
- Probability Comparison
  - Pr[bound1](psi1) >= Pr[bound2](psi2)
- Expected min/max for certain expression
  - E[bound ; N] (min/max: expr)



Example: Stochastic Train-Gate

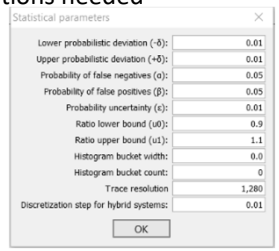
- Train with larger id is more eager to start the approach
- simulate 1 [<=300] { Train(0).Cross, Train(5).Cross, Gate.len}
- Pr[<=300](<> Gate.len < 3)





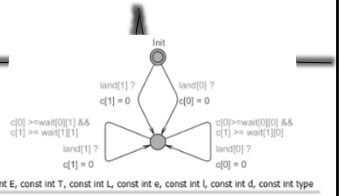
Stochastic Parameters

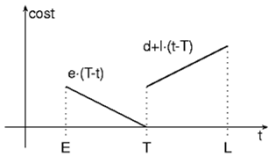
- δ, α, β: hypothesis testing
- ε: uncertainty for the output
  - The smaller the range, the more simulations needed
- $u0, u1$: for probability comparison



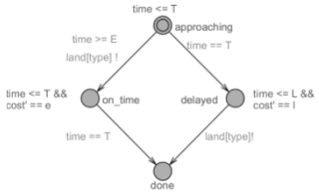
UPPAAL CORA: Cost Optimal Reachability Analysis

- Linearly priced timed automata (LPTA)
- Add cost/reward to each location
- Calculates the path with minimum cost
- Can be used to model power consumption, etc





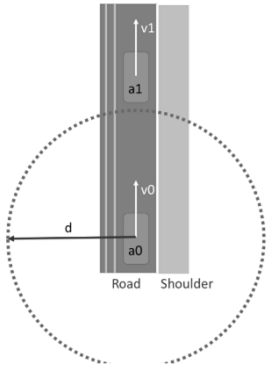
E earliest landing time
T target (cruise) landing time
L latest landing time
e early cost rate
l late cost rate
d late penalty



Controller Synthesis

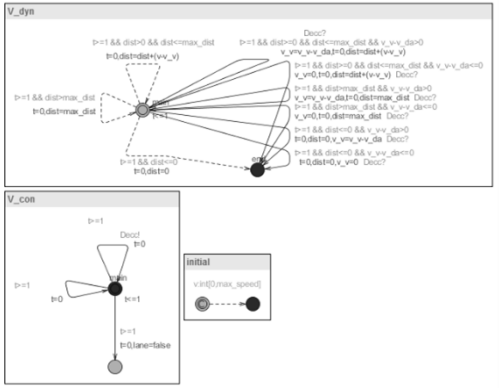
- Synthesize a controller that satisfy the requirement
- Two player game: Controller vs. Environment
- Return the winning strategy for controller

Toy Example

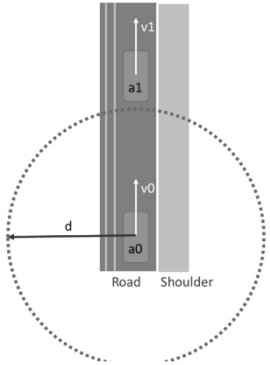


R1: No collision
R2: No driving on shoulder
R3: No hard braking

UPPAAL TIGA



NAÏVE SOLUTION



R1: No collision
R2: No driving on shoulder
R3: No hard braking

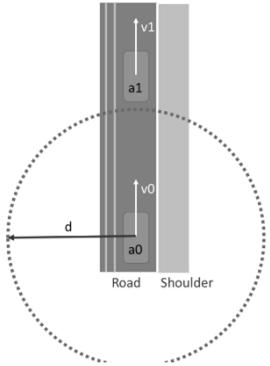
A: v1=[0,v_max]
G: R1 R2 R3

WINNING STRATEGY

- Change lane to avoid collision

```
Strategy to avoid losing:

State: ( V_dyn.main V_con.main initial._id4 ) v_v=4 lane=1 dist=6 v=0
When you are in (V_dyn.t==1 && V_dyn.t==V_con.t && V_con.t==1), take transition
V_con.main->V_con._id0 ( t >= 1, tau, t := 0, lane := 0 )

State: ( V_dyn.main V_con.main initial._id4 ) v_v=4 lane=1 dist=2 v=0
When you are in (V_dyn.t-V_con.t==-1 && V_con.t==1), take transition V_con.main-
>V_con._id0 ( t >= 1, tau, t := 0, lane := 0 )
```


MODEL/STRATEGY REFINEMENT



R1: No collision
R2: No driving on shoulder
R3: No hard braking

A: v1=[0,v_max]
G: R1 R2 R3

A: v1=[0,v_b]
G: R1 R2 R3

A: v1=[v_b,v_max]
G: R1 R2 R3

A: v1=[v_b,v_g]
G: R1 R2 R3

A: v1=[v_g,v_max]
G: R1 R2 R3

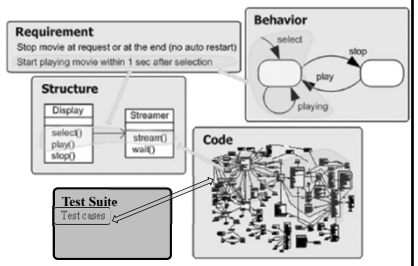
REFERENCE

- Downlaod
  - www.uppaal.org
- Tutorials
  - On the same webpage
  - Recommended:
    - UPPAAL 4.0: Small Tutorial.
    - Uppaal SMC Tutorial

Lecture 9: Traceability & Stateflow

What is traceability?

- We would like to make sure that
  - All requirements are implemented
  - All implementations are necessary
- Trace artifacts
  - Requirements, models, code, etc.
- Trace link
  - Association between two trace artifacts
  - Type: Refinement, Abstraction, Implementation, etc.
- Trace granularity: component level, statement level, etc.
- Trace quality: completeness, correctness, etc.



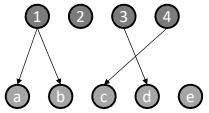
Traceability Activities

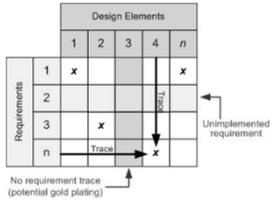
- Trace Creation
  - Establish *trace link* between a *source artifact* and a *target artifact*
  - Traceability document

- Trace Validation
  - Between requirements and model: Model checking
  - Between concept model and implementation model: Model translation
  - Between model and code: Conformance testing
- Trace Maintenance
  - Update trace when modification happened

Traceability Matrix

- Try to avoid orphan nodes



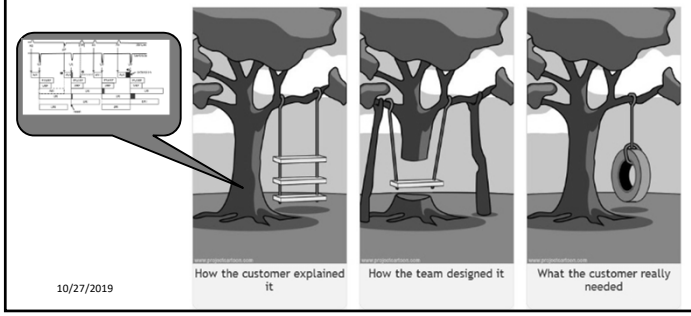


WHY DO WE NEED TO ENSURE TRACEABILITY?

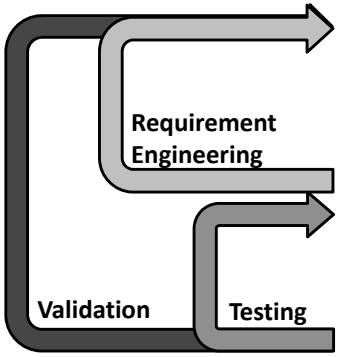
- Software life cycle involves more than one person
- They need to be on the same page
  - Why we are developing this software?
- People tend to focus on their job at hand, and forget why they are doing it.

CHALLENGE

- Impossible to list the constraints explicitly
- How to effectively communicate with domain experts?

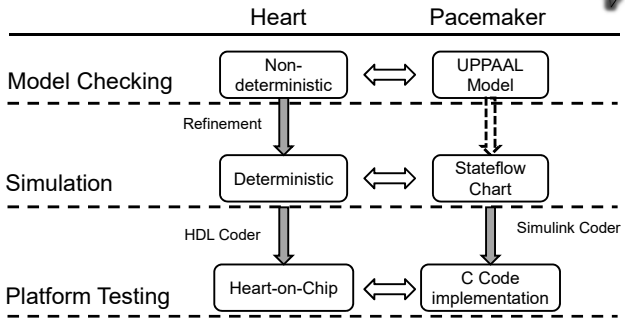


SOFTWARE DEVELOPMENT PROCESS

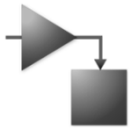


- Requirements
  - i.e. the vehicle should not crash with any other objects
- Specifications
  - i.e. when the vehicle detects an object on its path, apply brake
- Implementation

F[redacted]ERIFIED MODELS TO VERIFIED CODE



SIMULINK & STATEFLOW

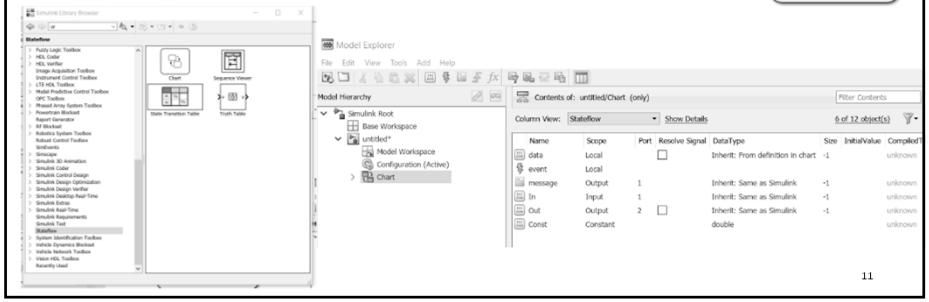


SIMULINK & STATEFLOW

- A graphical modeling/programming language
- Developed by Mathworks
  - Highly integrated with Matlab
- Has a full model-based design toolchain
- Widely adapted by system/software developers
- Rich expressiveness

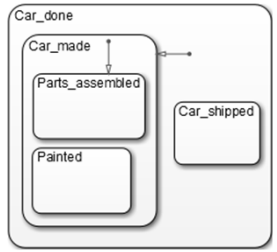
MODEL EXPLORER

- Define and configure input/output, event/message, va



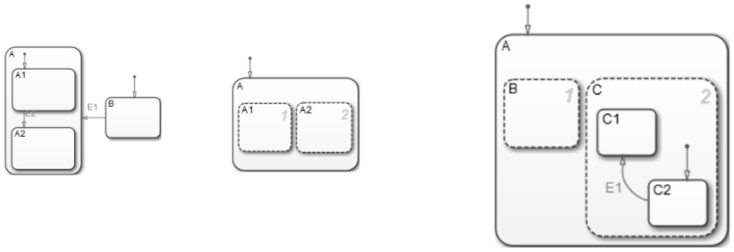
STATE HIERARCHY

- Object oriented modeling
  - /Car_done
  - /Car_done.Car_made
  - /Car_done.Car_shipped
  - /Car_done.Car_made.Parts_assembled
  - /Car_done.Car_made.Painted



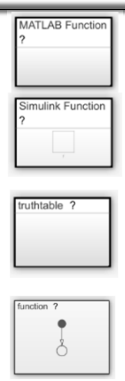
STATE COMPOSITIONS

- "OR"/exclusive composition
- "AND"/parallel composition



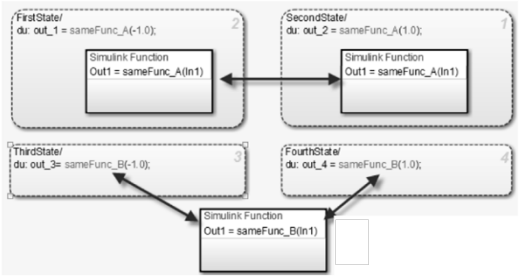
EMBEDDED FUNCTIONS

- Matlab function
- Simulink function
- Truthtable
- Graphical function



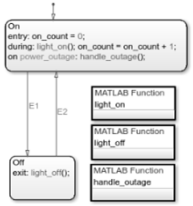
FUNCTION AVAILABILITY

- Only available to states at the same level



STATE ACTIONS

- entry: (en:) entry actions
  - Action occurs on a time step when the state becomes active.
- during: (du:) during actions
  - Action occurs on a time step when the state is already active and the chart does not transition out of the state.
- exit: (ex:) exit actions
  - Action occurs on a time step when the chart transitions out of the state.
- on event_name: on event_name actions
- on message_name: on message_name actions



REDUCE REDUNDANCY

```
en:
  fcn1();
  fcn2();
du: fcn1();
ex: fcn1();
```

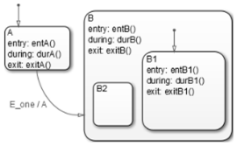
```
en, du, ex: fcn1();
en: fcn2();
```


TRANSITION

- event_or_message[condition]{condition_action}/transition_action
- Condition
  - Boolean expression that specifies that a transition path is valid if the expression is true; part of a transition label
- Condition actions
  - Executes after the condition for the transition is evaluated as true, but before the transition to the destination is determined to be valid
- Transition actions
  - Executes after the transition to the destination is determined to be valid

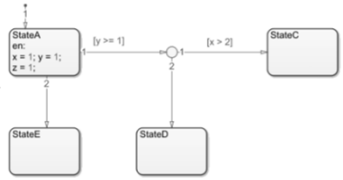
DEFAULT TRANSITIONS

- State A is active. Event E_one occurs and awakens the chart
- State A exit actions (exitA()) execute and complete.
- State A is marked inactive.
- The transition action, A, is executed and completed.
- State B is marked active.
- State B entry actions (entB()) execute and complete.
- State B detects a valid default transition to state B.B1.
- State B.B1 is marked active.
- State B.B1 entry actions (entB1()) execute and complete.
- The chart goes back to sleep.



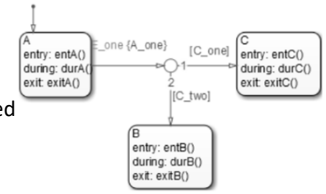
EXECUTION ORDER

- Transition 1 from StateA is marked for evaluation.
- Transition 1 from StateA has a condition.
- The condition is true.
- The destination of transition 1 from StateA is not a state.
- The junction does have outgoing transitions.
- Transition 1 from the junction is marked for evaluation.
- Transition 1 from the junction has a condition.
- The condition is false.
- Transition 2 from the junction is marked for evaluation.
- Transition 2 from the junction does not have a condition.
- The destination of transition 2 from the junction is a state (StateD).
- StateD is marked for entry, and StateA is marked for exit.



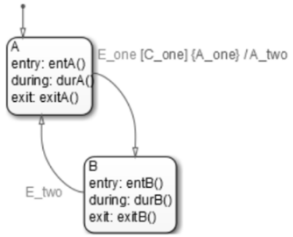
CONDITION ACTION BEHAVIOR

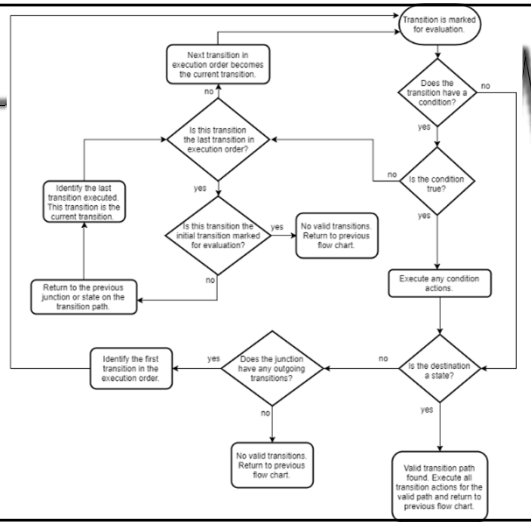
- E_one happened when state A is active. C_one and C_two are false
- A valid transition segment from state A to a connective junction is detected.
- The condition action A_one is immediately executed and completed. State A is still active.
- No complete transitions is valid.
- State A during actions (durA()) execute and complete.
- State A remains active.
- The chart goes back to sleep.



CONDITION AND TRANSITION ACTION BEHAVIOR

- E_one happened and awaked the chart.
- The condition C_one is true. The condition action A_one is immediately executed.
- State A is still active.
- State A exit actions (ExitA()) execute and complete.
- State A is marked inactive.
- The transition action A_two is executed.
- State B is marked active.
- State B entry actions (entB()) execute.
- The chart goes back to sleep.

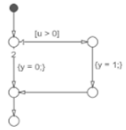




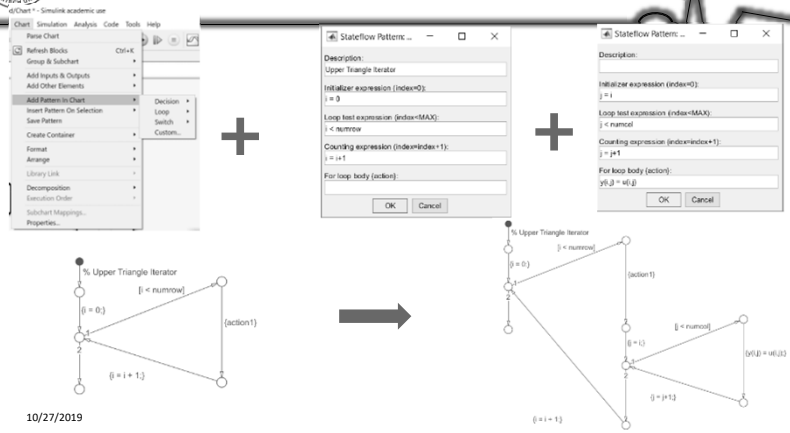
FLOW CHART

- No time consumption during execution
- Can be used for graphical function definition

```
if u > 0
  y = 1;
else
  y = 0;
end
```

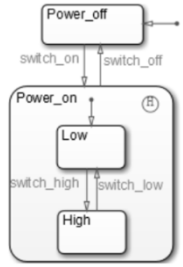


ADD PATTERN IN CHART



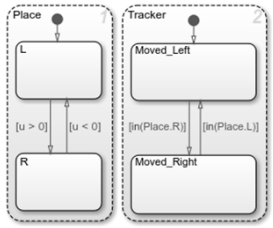
HISTORY JUNCTION

- Restores the state that is on the same level of the composite state as the history state itself
- If the system was switched off when the system was at the "High" state, when the system is switched back on, it will start from the "High" state



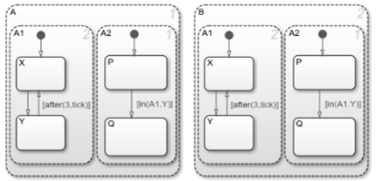
CHECK STATE ACTIVITY BY USING IN() OPERATOR

- We can use in() operator to reference status of other parallel states
  - Return 1 if the referenced state is also active
- Starting point
  - If in state action, start from the containing state
  - If on transition, start from the parent state
- Search up the state hierarchy until the chart level is reached



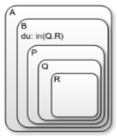
IN() OPERATOR EXAMPLE

- In(A1.Y) in both A and B only find local copies of A1.Y
- Because at the chart level, there is no A1.Y

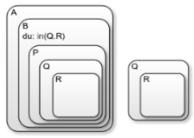


IN() OPERATOR EXAMPLE (CONT.)

No match

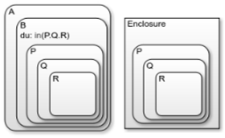


In(P.Q.R) will do Wrong match



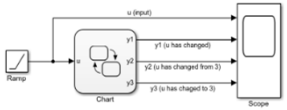
In(B.P.Q.R) will do Multiple match

Use enclosure to ensure local match

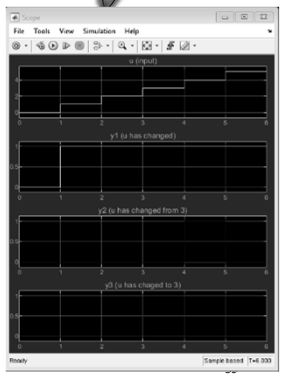


DETECT DATA CHANGE

- hasChanged(u)
  - Detects changes in data value from the beginning of the last time step to the beginning of the current time step.
- hasChangedFrom(u,v)
  - Detects changes in data value from a specified value at the beginning of the last time step to a different value at the beginning of the current time step.
- hasChangedTo(u,v)
  - Detects changes in data value to a specified value at the beginning of the current time step from a different value at the beginning of the last time step.




```
{
y1 = hasChanged(u);
y2 = hasChangedFrom(u,3);
y3 = hasChangedTo(u,3);
}
```

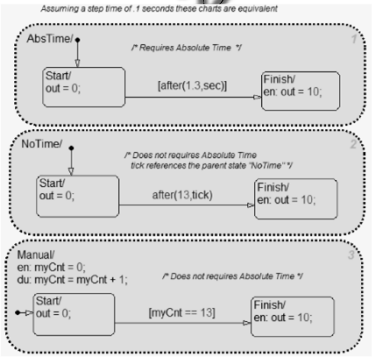


TEMPORAL LOGIC

- E is the base event and n is a positive integer
- after(n,E)
  - Returns true if n units of simulation time have elapsed since activation of the associated state.
- before(n,E)
  - Returns true if fewer than n units of simulation time have elapsed since activation of the associated state.
- every(n,E)
  - Returns true every n units of simulation time since activation of the associated state.
- temporalCount(sec)
  - Counts and returns the number of specified simulation time (sec) that have elapsed since activation of the associated state.

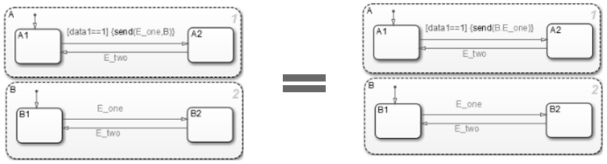
COUNTER VS. TEMPORAL LOGIC

- Simulation time, not real world time



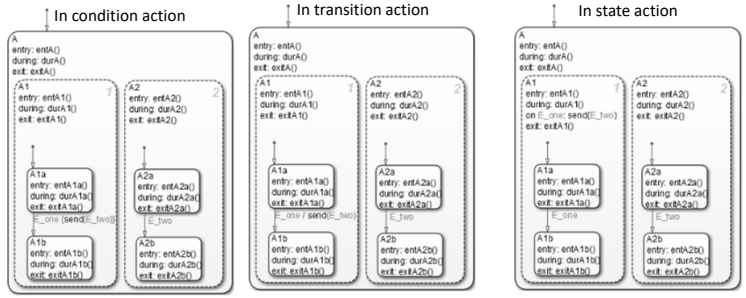
Broadcast Local Events to Synchronize Parallel States

- send(event_name,state_name)
- event_name is broadcast to its owning state (state_name) and any offspring of that state in the hierarchy.
- The receiving state must be active during the event broadcast.
- An action in one chart cannot broadcast events to states in another chart.



Broadcasting events to parallel states

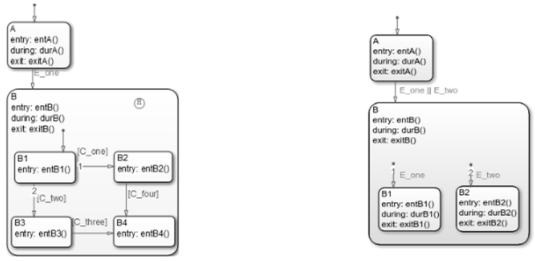
- Parallel substates A.A1.A1a and A.A2.A2a are active. Event E_one occurs and awakens the chart



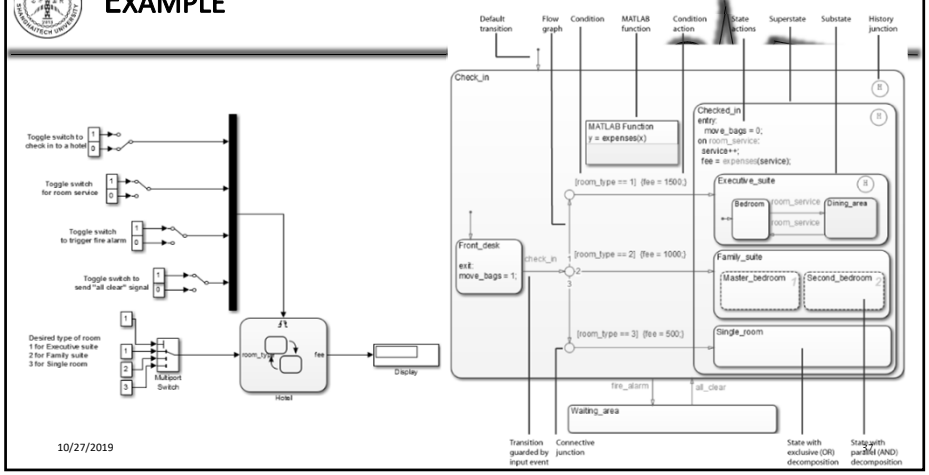
Implicit Events

- change(data_name) or chg(data_name)
  - generates a local event when writing a value to the variable data_name
  - Data_name has to be at chart level or lower
- enter(state_name) or en(state_name)
  - generates a local event when the specified state_name is entered
- exit(state_name) or ex(state_name)
  - generates a local event when the specified state_name is exited
- Tick/wakeup
  - generates a local event when the chart of the action being evaluated awakens

Example



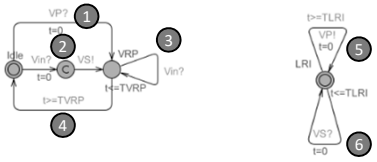
EXAMPLE

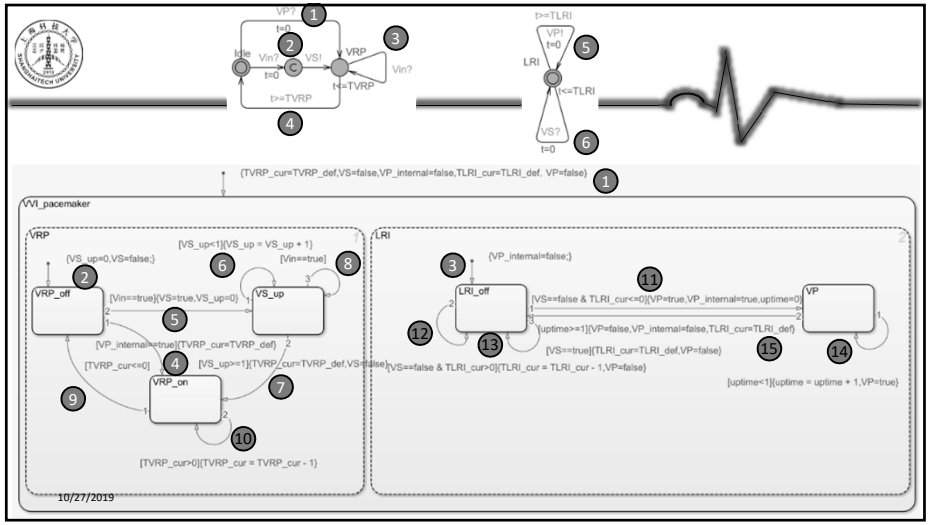


MODELING TIPS

- Use signals of the same data type for input events
- Use a default transition to mark the first state to become active among exclusive (OR) states
- Use condition actions instead of transition actions whenever possible
- Use explicit ordering to control the testing order of a group of outgoing transitions
- Use MATLAB functions for performing numerical computations in a chart
- Do not use history junctions in states with parallel (AND) decomposition

UPPAAL MODEL





MATLAB CODE

```
% Parameters
Param.TLRI_def=1000;
Param.TVRP_def=100;

% States and initialization %%% T1
States.TLRI_cur=Param.TLRI_def;
States.TVRP_cur=Param.TVRP_def;
States.Cur_stateVRP='VRP_off'; %%% T2
States.VS_up=0;
States.VS=0;
States.Cur_stateLRI='LRI_off'; %%% T3
States.VP_internal=0;

% Default input
Ain=0;
Vin=0;

% Output
VP=0; %%% T1

% Global Clock
t=0;
data=[];
% Running the Pacemaker
while t<=2500
    t=t+1;
    % Your pacemaker function here:
    [States,VP]=HW3_ShihaoJiang_PM(Ain,Vin,States,Param);
end
```

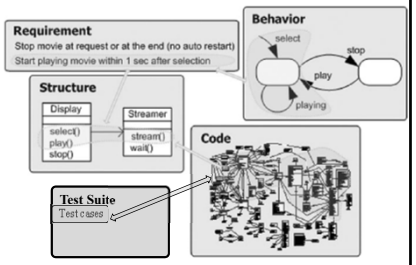
```
function [States,VP]=HW3_ShihaoJiang_PM(Ain,Vin,States,Param)

switch States.Cur_stateVRP
    case "VRP_off"
        if States.VP_internal==1 %%% T4
            States.TVRP_cur=Param.TVRP_def;
            States.Cur_stateVRP="VRP_on";
        else if Vin==1 %%% T5
                States.VS=1;
                States.VS_up=0;
                States.Cur_stateVRP="VS_up";
            else
            end
        end
    case "VS_up"
        if States.VS_up<1 %%% T6
            States.VS_up=States.VS_up+1;
            States.Cur_stateVRP="VS_up";
        else if States.VS_up>=1 %%% T7
                States.TVRP_cur=Param.TVRP_def;
                States.VS=0;
                States.Cur_stateVRP="VRP_up";
            else if Vin==true %%% T8
                    States.Cur_stateVRP="VS_up";
                end
            end
        end
    case "VRP_on"
        if States.TVRP_cur<=0 %%% T9
            States.Cur_stateVRP="VRP_off";
        else if States.TVRP_cur>0 %%% T10
                States.TVRP_cur=States.TVRP_cur-1;
                States.Cur_stateVRP="VRP_off";
            end
        end
end
```

```
switch States.Cur_stateLRI
    case "LRI_off"
        if States.VS==0 && States.TLRI_cur<=0 %%% T11
            VP=1;
            States.VP_internal=1;
            States.uptime=0;
            States.Cur_stateLRI="VP";
        else if States.VS==0 && States.TLRI_cur>0 %%% T12
                States.TLRI_cur=States.TLRI_cur-1;
                States.Cur_stateLRI="LRI_off";
                VP=0;
            else if States.VS==1 %%% T13
                    States.TLRI_cur=Param.TLRI_def;
                    VP=0;
                end
            end
        end

    case "VP"
        if States.uptime<1 %%% T14
            States.uptime=States.uptime+1;
            States.Cur_stateLRI="VP";
            VP=1;
        else if States.uptime>=1 %%% T15
                VP=0;
                States.VP_internal=0;
                States.TLRI_cur=Param.TLRI_def;
                States.Cur_stateLRI="LRI_off";
            end
        end

end
```

IN THE LAST LECTURE…

WHAT IS TRACEABILITY?

- We would like to make sure that
  - All requirements are implemented
  - All implementations are necessary
- Trace artifacts
  - Requirements, models, code, etc.
- Trace link
  - Association between two trace artifacts
  - Type: Refinement, Abstraction, Implementation, etc.
- Trace granularity: component level, statement level, etc.
- Trace quality: completeness, correctness, etc.



WHAT WE SHOULD KNOW ABOUT TRACEABILITY

- Traceability is not just traceability documentation
- Traceability is a requirement, just like safety and correctness
- Will be one of the most important judging criteria for final projects

LECTURE 10: TESTING

WHY?

- Validation: whether the system satisfy the requirement
  - Functional Testing
- Verification: whether the system conform with the specification
  - Conformance Testing

WHO AND WHEN?

- Testing is performed by test engineers after the initial development
- Test design engineers may be different from test engineers
- The test engineers should be able to do it without participating the development process

SOFTWARE CHARACTERISTICS

- Functional characteristics
- Non-functional characteristics
- Non-functional tests should be performed together with functional tests

EARLY TESTING

- The cost of finding (and fixing) a defect
  - In the design phase is “1”.
  - In the coding phase, multiplied by “10”
  - During the test phase (system test or acceptance test), multiplied by “100”.
  - In production (by the customer or by a user), multiplied by “1,000”.

INDEPENDENCE LEVEL

- Same person
- Pair programming
- Independent tester from the same team
- From the same company
- Third party

TESTING MENTALITY

- A developer will try to find *one* solution to a particular problem, and will design the program based on the solution envisioned;
- A tester will try to identify *all* possible failures that might have been forgotten.
- A developer usually has two hats: the developer's and the tester's
- Our brain has a tendency to hide defects because it is looking for usual patterns.
- Need to have different perspectives

TESTING IS NOT A DESTRUCTIVE ACTIVITY

- The defects were not introduced by testers, but developers
- It is not personal, it is just business
  - Point out defects should not look bad for developers
  - need good relational skills

TEST CASES AND TEST SUITE

- Test a software using a set of carefully designed test cases:
  - The set of all test cases is called the test suite

TEST CASES AND TEST SUITE

- A test case is a triplet [I,S,O]:
  - I is the data to be input to the system,
  - S is the state of the system at which the data is input,
  - O is the expected output from the system.

13

DESIGN OF TEST CASES

- Exhaustive testing of any non-trivial system is impractical:
  - input data domain is extremely large.
- Design an optimal test suite:
  - of reasonable size
  - to uncover as many errors as possible.

14

DESIGN OF TEST CASES

- If test cases are selected randomly:
  - Many test cases do not contribute to the significance of the test suite,
  - Do not detect errors not already detected by other test cases in the suite.
- The number of test cases in a randomly selected test suite:
  - Not an indication of the effectiveness of the testing.

15

DESIGN OF TEST CASES

- Testing a system using a large number of randomly selected test cases:
  - does not mean that many errors in the system will be uncovered.
- Consider an example:
  - finding the maximum of two integers x and y.

16

Design of Test Cases

- If (x>y) max = x;
  else max = x;
- The code has a simple error:
- test suite {(x=3,y=2);(x=2,y=3)} can detect the error,
- a larger test suite {(x=3,y=2);(x=4,y=3); (x=5,y=1)} does not detect the error.

17

International Standard

- ISO/IEC/IEEE 29119-4 Software and Systems Engineering – Software Testing – Part 4: Test Techniques
- Available on Blackboard

18

Test Design (TD) Process

- TD1: Identify feature set
- TD2: Derive test conditions
- TD3: Derive test coverage items
- TD4: Derive test cases
- TD5: Assemble test sets
- TD6: Derive test procedures

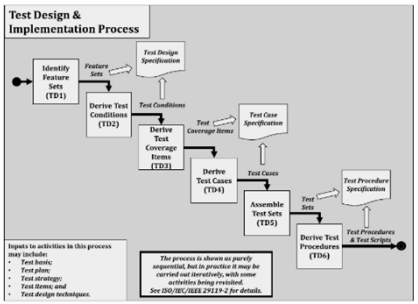


Figure 1 — ISO/IEC/IEEE 29119-2 Test Design and Implementation Process

19

Test Design Techniques

- Specification-based Testing
  - Black-box Testing
- Structure-based Testing
  - White-box Testing
- Experience-based Testing

20

SPECIFICATION-BASED TESTING

- Equivalence Partitioning
- State Transition Testing
- Scenario Testing

21

EQUIVALENCE PARTITIONING

22

EXAMPLE: EQUIVALENCE PARTITIONING

- Homework (HW) 25pt
- Exam 75pt
- Specification: A function *generate_grading*
  - HW+Exam>=70 -> 'A'
  - 50<= HW+Exam<70 -> 'B'
  - 30<= HW+Exam<50 -> 'C'
  - HW+Exam<30 -> 'D'
  - Invalid inputs -> 'FM'

23

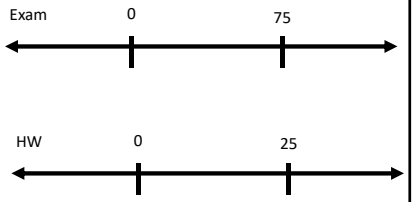
STEP 1: IDENTIFY FEATURE SETS (TD1)

- FS1: generate_grading function

24

Step 2: Derive Test Conditions (TD2)

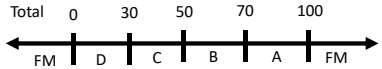
- Input Partitions
  - Valid
    - TCOND1: 0<=Exam<=75
    - TCOND2: 0<=HW<=25
  - Invalid
    - TCOND3: Exam<0
    - TCOND4: Exam>75
    - TCOND5: HW<0
    - TCOND6: HW>25
    - TCOND7: non-integer Exam input
    - TCOND8: non-integer HW input



25

Step 2: Derive Test Conditions (TD2) (CONT)

- Output Partitions
  - TCOND9: 'A' induced by 70<=Total<=100
  - TCOND10: 'B' induced by 50<=Total<70
  - TCOND11: 'C' induced by 30<=Total<50
  - TCOND12: 'D' induced by 0<=Total<30
  - TCOND13: 'Fault Message' (FM) induced by Total>100
  - TCOND14: 'Fault Message' (FM) induced by Total<0
  - TCOND15: 'Fault Message' (FM) induced by non-integer inputs



26

Step 3: Derive Test Coverage Items (TD3)

- Specify a test coverage item (TCOVER) for each test condition (TCOND)
  - TCOVER1: 0<=Exam<=75 (for TCOND1)
  - TCOVER2: 0<=HW<=25 (for TCOND2)
  - …

27

Step 4: Derive Test Cases (TD4)

- Attempt to "hit" Test Coverage Items
  - One-to-One: One test case for EACH Test Coverage item
    - More test cases but more intuitive
  - Minimized: Each test case may exercise more than one Test Coverage Items
    - Less test cases

28

Step 4: Derive Test Cases (TD4) One-to-one

- Test cases for input Exam
- Test cases for input HW
- Test cases for non-integer inputs
- Test cases for valid output

29

Step 4: Derive Test Cases (TD4) One-to-one

Test cases for input Exam

Test Case	1	2	3
Input (Exam)	60	-10	93
Input (HW)	15	15	15
Total	75	5	108
Test Coverage Item	TCOVER1	TCOVER3	TCOVER4
Partition Tested	0<=Exam<=75	Exam<0	Exam>75
Expected Output	'A'	'FM'	'FM'

30

Step 4: Derive Test Cases (TD4) Minimized

Test Case	1	2	3	4
Input (Exam)	60	50	35	19
Input (HW)	20	16	10	8
Total	80	66	45	27
Test Coverage Item	TCOVER1 TCOVER2 TCOVER9	TCOVER1 TCOVER2 TCOVER10	TCOVER1 TCOVER2 TCOVER11	TCOVER1 TCOVER2 TCOVER12
Partition Exam	0<=Exam<=75	0<=Exam<=75	0<=Exam<=75	0<=Exam<=75
Partition HW	0<=HW<=25	0<=HW<=25	0<=HW<=25	0<=HW<=25
Partition Total	70<=Total<=100	50<=Total<70	30<=Total<50	0<=Total<30
Expected Output	'A'	'B'	'C'	'D'

31

Step 5: Assemble Test Sets (TD5)

- Arrange test cases sequences for efficient testing

32

TEST COVERAGE MEASUREMENT

- $Coverage = \left(\frac{N}{T} \times 100\right)\%$
  - N is the number of test coverage items covered by test cases
  - T is the number of identified test coverage items
- Coverage is only measured for a particular criteria
- Coverage criteria has strength

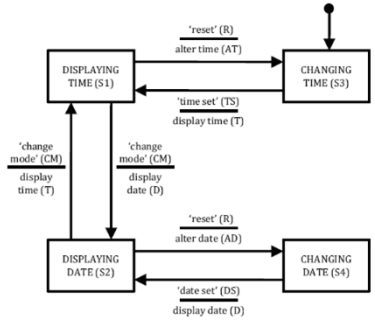
33

STATE TRANSITION TESTING

34

EXAMPLE: MANAGE DISPLAY

- A function: manage_display_changes
- 4 inputs
  - Change Mode (CM)
  - Reset (R)
  - Time Set (TS)
  - Date Set (DS)



35

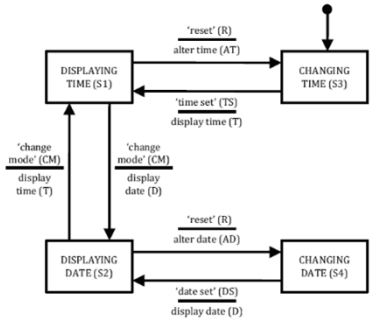
STEP 1: IDENTIFY FEATURE SETS (TD1)

- FS1: manage_display_changes

36

STEP 2: DERIVE TEST CONDITIONS (TD2)

- The state model is the test condition



STEP 3: DERIVE TEST COVERAGE ITEMS

- All states
  - Test cases should visit all states in the model
- Single transition (0-switch coverage)
  - Only valid transitions
- All transitions
  - Both valid and invalid transitions
- Multiple transitions (N-switch coverage)
  - Valid sequences of N+1 transitions in the state model

Table B.31 — State table for manage_display_changes

	CM	R	TS	DS
S1	S2/D	S3/AT	S1/–	S1/–
S2	S1/T	S4/AD	S2/–	S2/–
S3	S3/–	S3/–	S1/T	S3/–
S4	S4/–	S4/–	S4/–	S2/D

STEP 3: DERIVE TEST COVERAGE ITEMS (TD3)

- TCOVER1: S1 to S2 with input CM (valid)
- TCOVER2: S1 to S3 with input R (valid)
- …

Table B.31 — State table for manage_display_changes

	CM	R	TS	DS
S1	S2/D	S3/AT	S1/–	S1/–
S2	S1/T	S4/AD	S2/–	S2/–
S3	S3/–	S3/–	S1/T	S3/–
S4	S4/–	S4/–	S4/–	S2/D

STEP 4: DERIVE VALID TEST CASES (TD4) 0-SWITCH TEST CASES

- 0-switch test cases
- Invalid test cases should not cause state changes

Table B.31 — State table for manage_display_changes

	CM	R	TS	DS
S1	S2/D	S3/AT	S1/–	S1/–
S2	S1/T	S4/AD	S2/–	S2/–
S3	S3/–	S3/–	S1/T	S3/–
S4	S4/–	S4/–	S4/–	S2/D

Table B.33 — 0-switch test cases for manage_display_changes

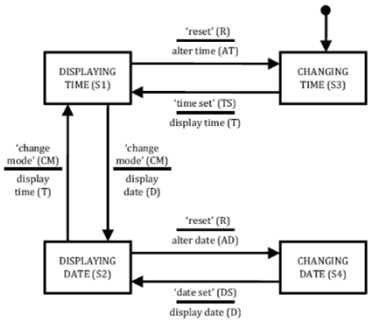
Test Case	1	2	3	4	5	6
Start State	S1	S1	S2	S2	S3	S4
Input	CM	R	CM	R	TS	DS
Expected Output	D	AT	T	AD	T	D
Finish State	S2	S3	S1	S4	S1	S2
Test Coverage Item	1	2	5	6	11	16

STEP 4: DERIVE VALID TEST CASES (TD4)
1-SWITCH TEST CASES

- TCOVER 17: S1 to S2 to S1 with inputs CM and CM
- …

Table B.35 — 1-switch test cases for manage_display_changes

Test Case	17	18	19	20	21	22	23	24	25	26
Start State	S1	S1	S1	S3	S3	S2	S2	S2	S4	S4
Input	CM	CM	R	TS	TS	CM	CM	R	DS	DS
Expected Output	D	D	AT	T	T	T	T	AD	D	D
Next State	S2	S2	S3	S1	S1	S1	S1	S4	S2	S2
Input	CM	R	TS	CM	R	CM	R	DS	CM	R
Expected Output	T	AD	T	D	AT	D	AT	D	T	AD
Finish State	S1	S4	S1	S2	S3	S2	S3	S2	S1	S4
Test Coverage Item	17	18	19	20	21	22	23	24	25	26



STEP 5: ASSEMBLE TEST SETS

- TS1: 0 switch test cases - Test cases 1,2,3,4,5,6
- More efficient if rearranged to 5,1,4,6,3,2
  - The finish state of test case n is the start state of test case n+1

Table B.33 — 0-switch test cases for manage_display_changes

Test Case	1	2	3	4	5	6
Start State	S1	S1	S2	S2	S3	S4
Input	CM	R	CM	R	TS	DS
Expected Output	D	AT	T	AD	T	D
Finish State	S2	S3	S1	S4	S1	S2
Test Coverage Item	1	2	5	6	11	16

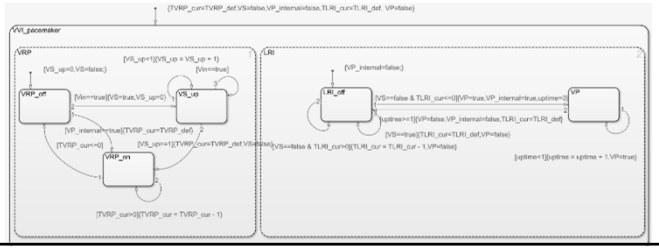
42

THE PACEMAKER EXAMPLE

43

VVI PACEMAKER

- Step 1: Identify Feature Sets (TD1)
  - The VVI_pacemaker function
- Step 2: Derive Test Conditions (TD2)
  - VVI_pacemaker model

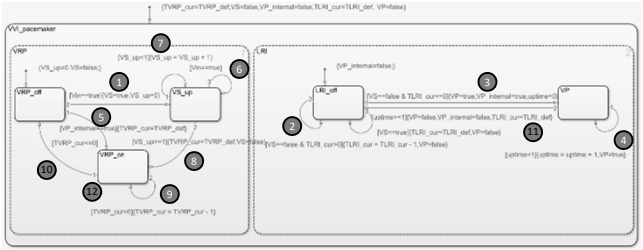


44

STEP 3: DERIVE TEST COVERAGE ITEMS (TD3)

- TCOVER1: S1 to S3 with Vin
- …

State	Index
VRP_off & LRI_off	S1
VRP_off & VP	S2
VS_up & LRI_off	S3
VS_up & VP	S4
VRP_on & LRI_off	S5
VRP_on & VP	S6



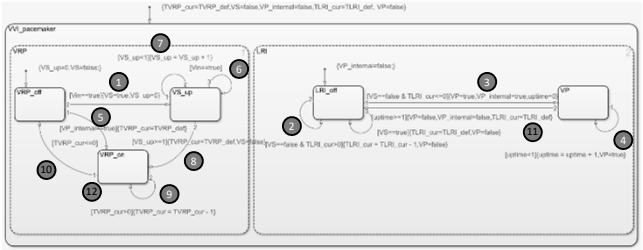
State\Input	Vin	Time
S1	S3 (1)	S1 (2)
S1	S3	S2/VP (3)
S2	S2/VP (4)	S6/VP (5)
S3	S3 (6)	S3 (7)
S3	S3	S5 (8)
S5	S5 (12)	S1 (10)
S5	S5	S5 (9)
S6	S6/VP	S5 (11)

45

STEP 4: DERIVE TEST CASES (TD4)

Test Case	1	2	3	4	5	6	7	8	9	10	11
Start state	S1	S1	S3	S3	S3	S5	S5	S5	S1	S2	S6
Input	200ms	Vin	1ms	Vin	1ms	50ms	Vin	50ms	900ms	1ms	1ms
Expected output	/	/	/	/	/	/	/	/	VP	VP	VP
Finish State	S1	S3	S3	S3	S5	S5	S5	S1	S2	S6	S5
Test Coverage Item	2	1	7	6	8	9	12	10	3	4, 5	11

State	Index
VRP_off & LRI_off	S1
VRP_off & VP	S2
VS_up & LRI_off	S3
VS_up & VP	S4
VRP_on & LRI_off	S5
VRP_on & VP	S6



State\Input	Vin	Time
S1	S3 (1)	S1 (2)
S1	S3	S2/VP (3)
S2	S2/VP (4)	S6/VP (5)
S3	S3 (6)	S3 (7)
S3	S3	S5 (8)
S5	S5 (12)	S1 (10)
S5	S5	S5 (9)
S6	S6/VP	S5 (11)

46

TEST COVERAGE MEASUREMENT

- $Coverage = \left(\frac{N}{T} \times 100\right)\%$
  - $Coverage = \left(\frac{12}{12} \times 100\right)\% = 100\%$

47

FIND A GOOD TEST CASE USING UPPAAL

- Assign each transition a Boolean variable t_n=false
- When transition i is taken, assign t_i=true
- E<> (t_1=true && t_2=true &&….)
- Use the counter-example/evidence to specify test cases

48