COMP 522
Modelling and Simulation
"**model** everything"

Hans Vangheluwe

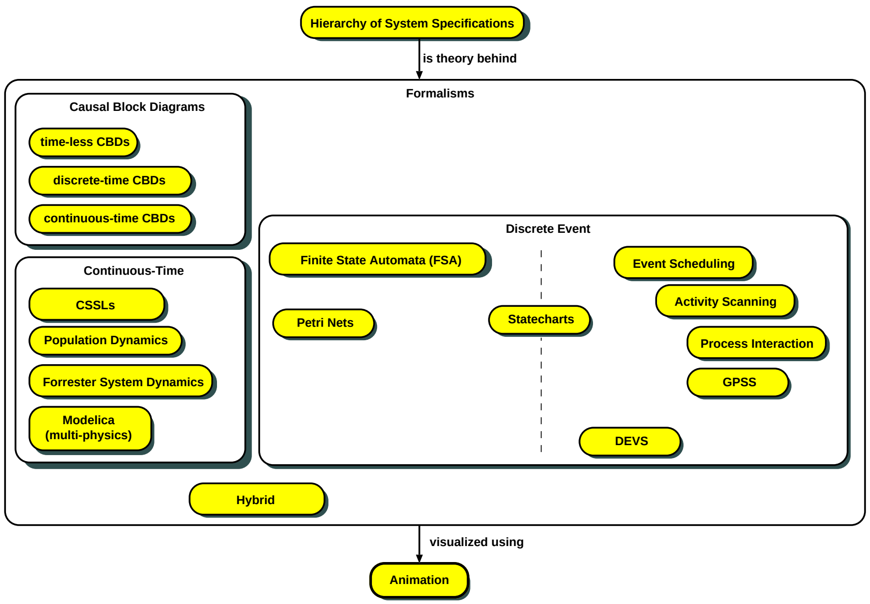
Modelling, Simulation and Design Lab (MSDL)
School of Computer Science, McGill University, Montréal, Canada

Overview

1. Course Description
2. Practical Information
3. Why/When Modelling and Simulation?
4. Complex Systems
5. Modelling and Simulation Concepts

Course Description

The course presents the **generic** (tool and application domain independent) **concepts** of modelling and simulation of complex dynamic systems. By the end of this course, you should have a deep understanding of these concepts using a **variety of formalisms**.



Strengths and weaknesses of different formalisms will be explained. This will allow you to choose the **most appropriate formalism(s)** for a given problem.

Course Description (ctd.)

You will learn to **build** modelling and simulation (software) systems. This will give you ample background to understand and **use existing** modelling and simulation systems.

The course presents general modelling and simulation principles by applying them to **concrete** problems in various application domains: software process modelling and simulation, reactive systems design such as complex graphical user interfaces, population dynamics analysis, traffic analysis, supermarket queueing, . . .

Practical Information

Main Reference (public!):

- `moncs.cs.mcgill.ca/people/hv/teaching/MS/`
- you will upload your project here!

Assignment submission, discussions: WebCT

- `www.mcgill.ca/mycourses/`

Need help ?

- Talk to me after class or make an appointment
- Come and see me during my office hours
  Monday 16:00 - 18:00 in MC328
- Use the discussion forum in WebCT (no direct e-mail!)
- Arrange to meet one of the TAs (assignment/project-specific)
- Assignments/projects are never fully specified ! Give feedback !

COMP522
Who's Who

Hans Vangheluwe
hv@cs.mcgill.ca

Reehan Shaikh
Reehan.Shaikh@cs.mcgill.ca

Eugene Syriani
esyria@cs.mcgill.ca

What are the pre-requisites ?

- COMP 251 (data structures and algorithms),
- COMP 302 (programming languages and paradigms),
- COMP 350 (numerical computing).

. . . or equivalent (see me).

Note:

- most assignment/project programming in Python (where appropriate)
- no prior knowledge required, but read Tutorial at `www.python.org`

Undergraduate or Graduate course ?

- Challenging course (work load)
- graduate “flavour” (independent thinking/work)
- some of the highest grades ever were obtained by ugrads

How is evaluation done ?

- 60% on assignments.
- 30% on the project (work, correctness, presentation).
- 10% on a mini-quizzes after each theory subject (in-next-class).

Together, assignments, mini-quizzes and project
cover the entire course.
Hence, there is **no final exam**.

Assignment/project rules of the game ?

- Completely in HTML form: requirements, design, code, discussion.
- Assignments: submit via WebCT.
- Project: on course website.
- Coding in Python `www.python.org` (where appropriate).
- Some assignments (and projects) in teams of 2.
  Clearly describe work distribution !
- Original work, some presented in class.
- Respect deadlines (or do more work to compensate).
- Alternate subjects may be proposed.

Assignments cover these topics

1. A Causal Block Diagram simulation tool.
2. Petri Net modelling, simulation and analysis.
3. Statechart modelling, simulation and software synthesis.
4. Event Scheduling simulation tool.
5. DEVS modelling and simulation of computer architecture.
6. GPSS (process interaction) model of a production (queueing) process.

Project

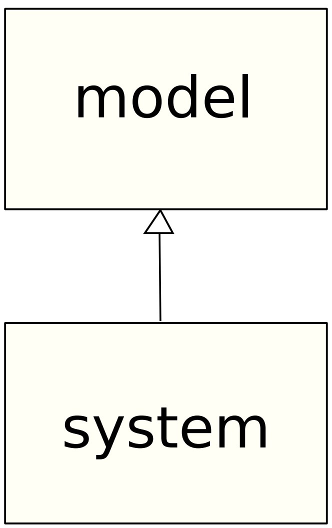
- For a formalism of choice (possibly construct your own):
  build a modelling and/or simulation environment.
- Using an existing modelling/simulation system:
  study a specific problem
  (games, user interfaces, physical systems, . . .).

Have a look at the course website for examples from previous years.
Examples: dead-reckoning in distributed games, SimCity, world dynamics, hybrid systems, solar car, dependable systems, TCP/IP, . . .

Questions?

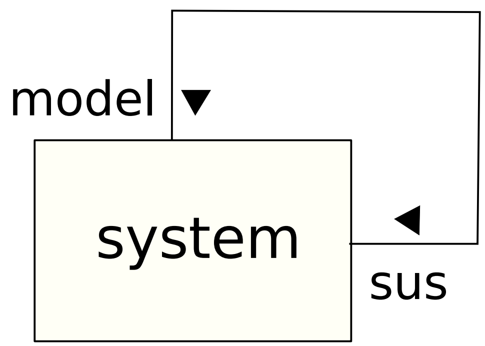
Jean Bézivin

Everything is a model !

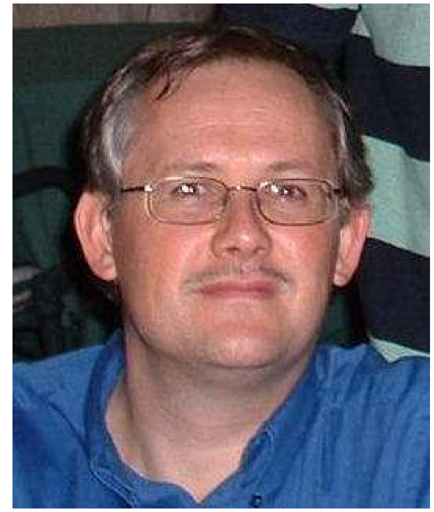


Jean-Marie Favre

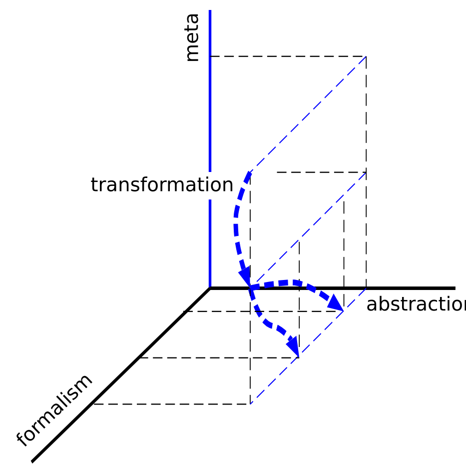
Nothing is a model !



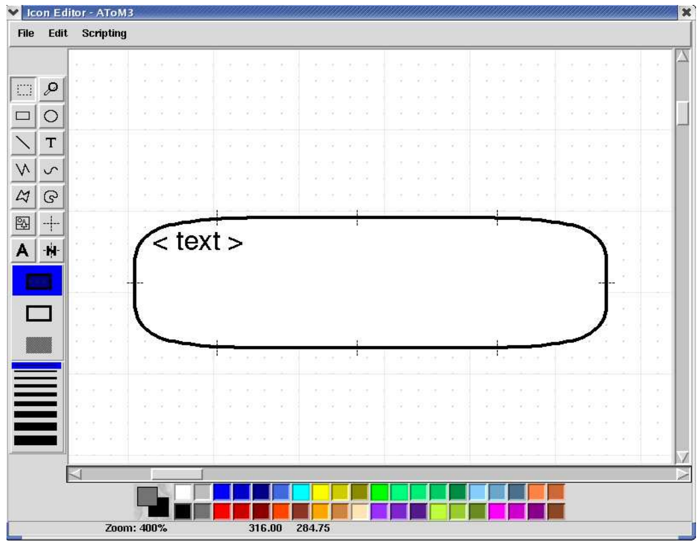
Hans Vangheluwe



Model everything !



A Variety of Complex Systems . . .



Need to be **modelled**

- at most appropriate **level of abstraction**
- in most appropriate **formalism(s)**

Why simulation? ... when too costly/dangerous

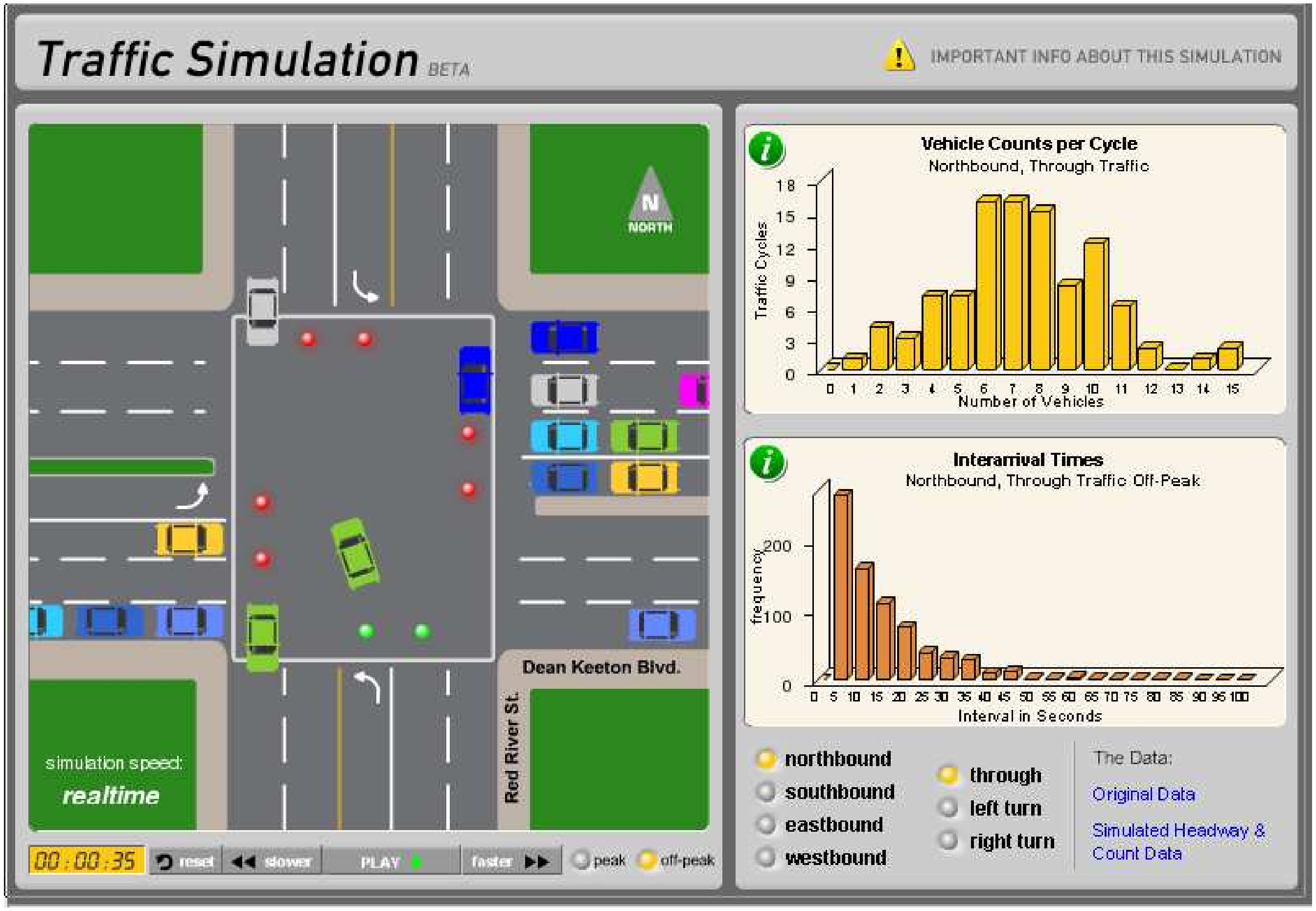
analysis ↔ design

Why simulation? . . . real experiment not ethical

training, physical simulation

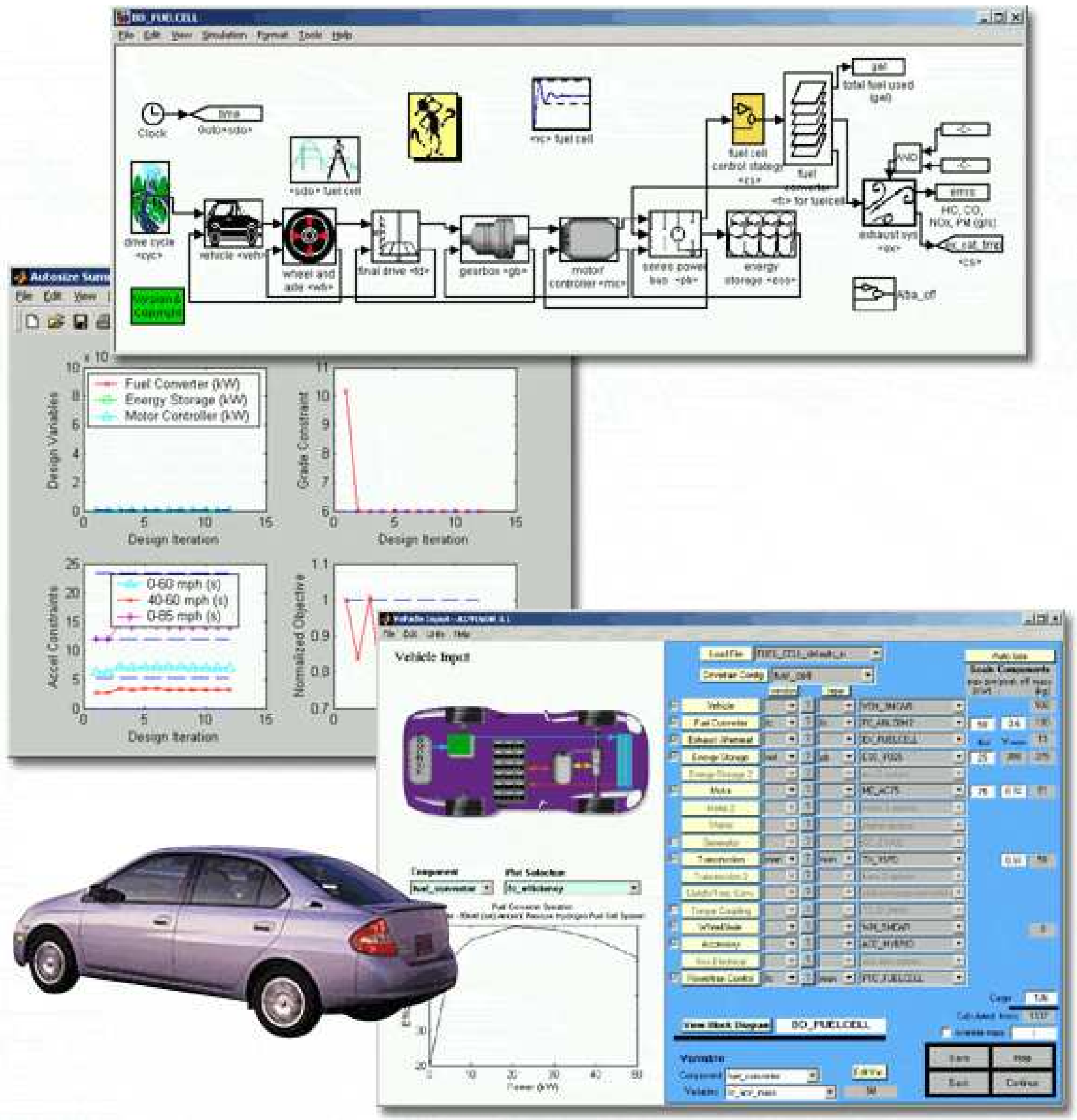
Simulation . . . evaluate alternatives

Simulation . . . evaluate alternatives

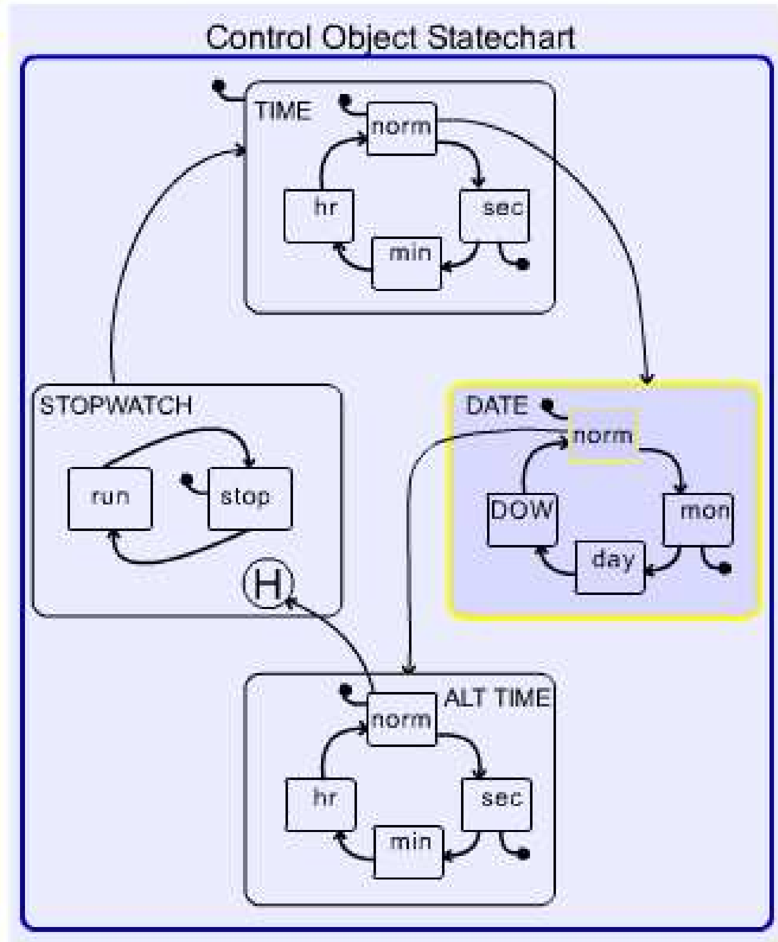


www.engr.utexas.edu/trafficSims/

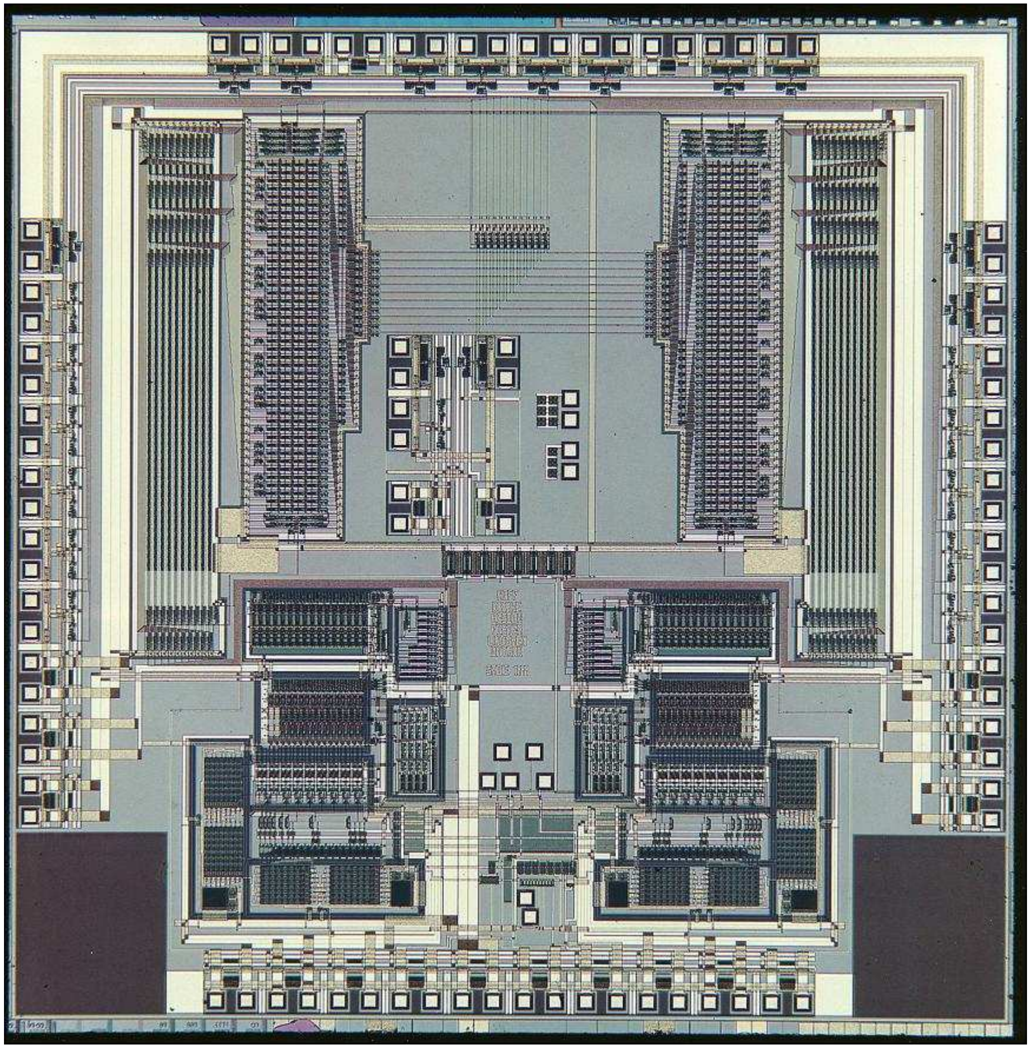
Simulation . . . “Do it Right the First Time”



Modelling/Simulation ...and code/app Synthesis



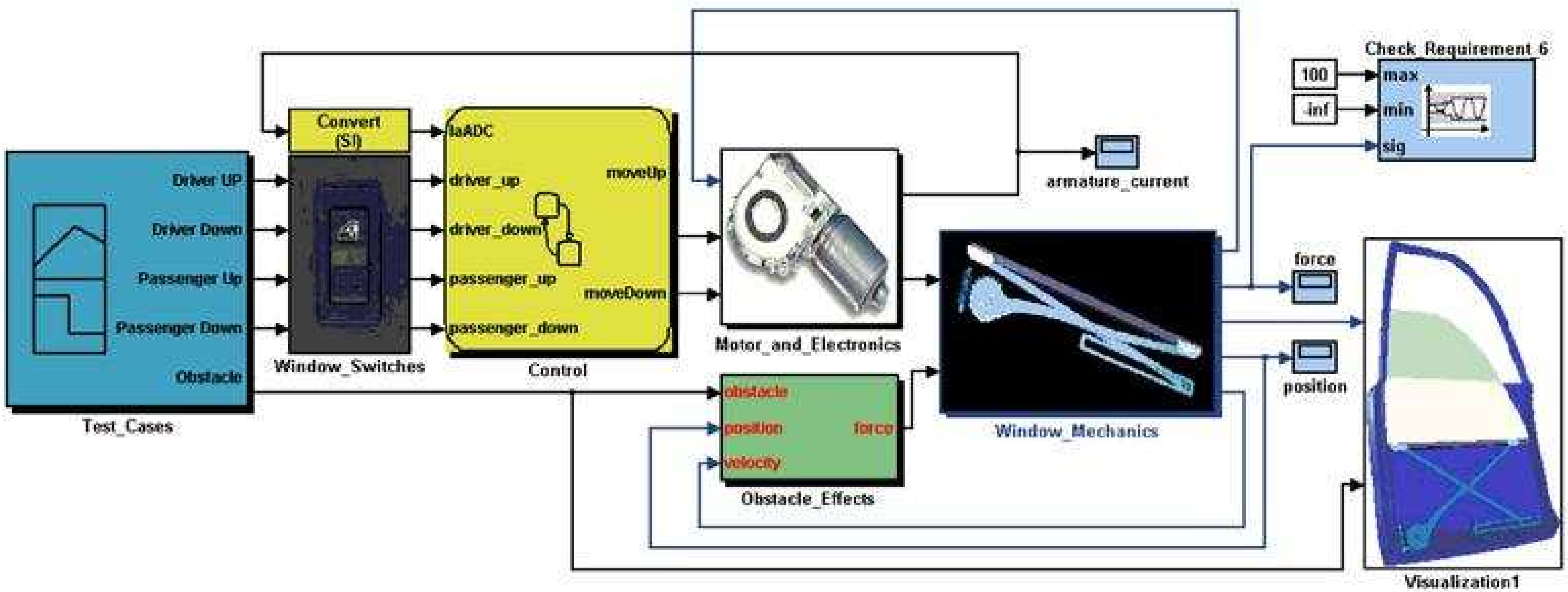
Complex Systems: complexity due to . . .



large number of components

Complex Systems: complexity due to . . .

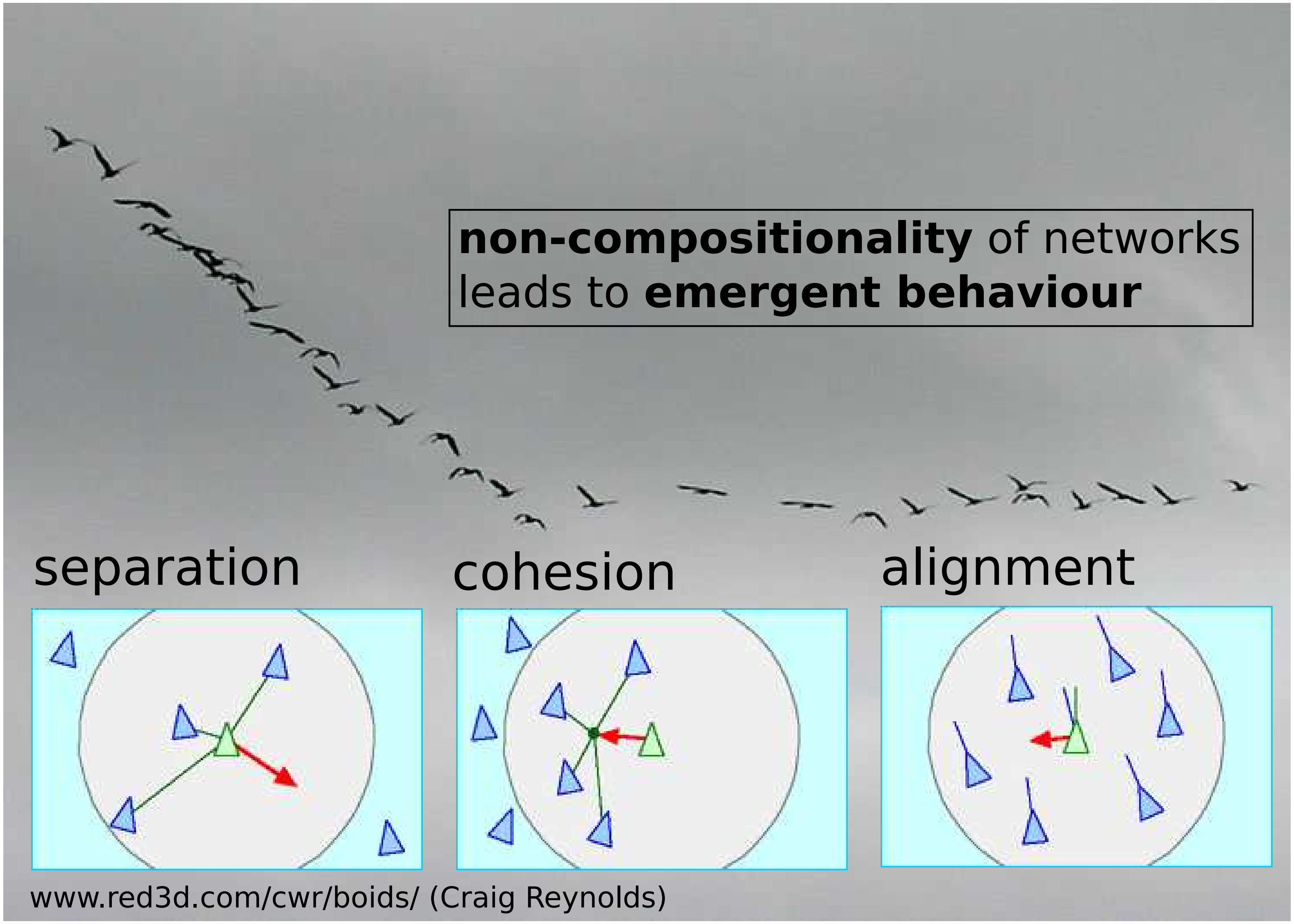
Complex Systems: complexity due to ...



www.mathworks.com/products/demos/simulink/PowerWindow/html/PowerWindow1.html

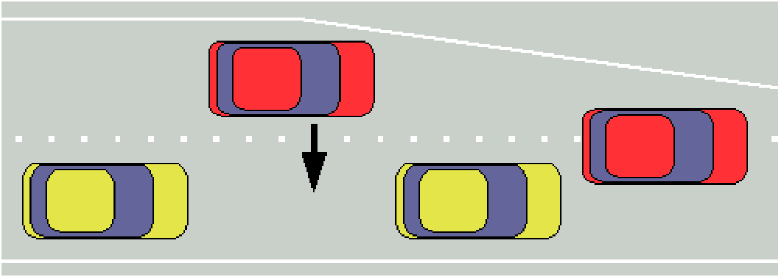
heterogeneity of components

Complex Systems: complexity due to . . .



Modelling/Simulating Complex Systems . . .

- at the most appropriate **level of abstraction**



- using the most appropriate **formalism(s)**
  Ordinary Differential Equations, Petri Nets, Bond Graphs, Statecharts, Forrester System Dynamics, CSP, Queueing Networks, . . .

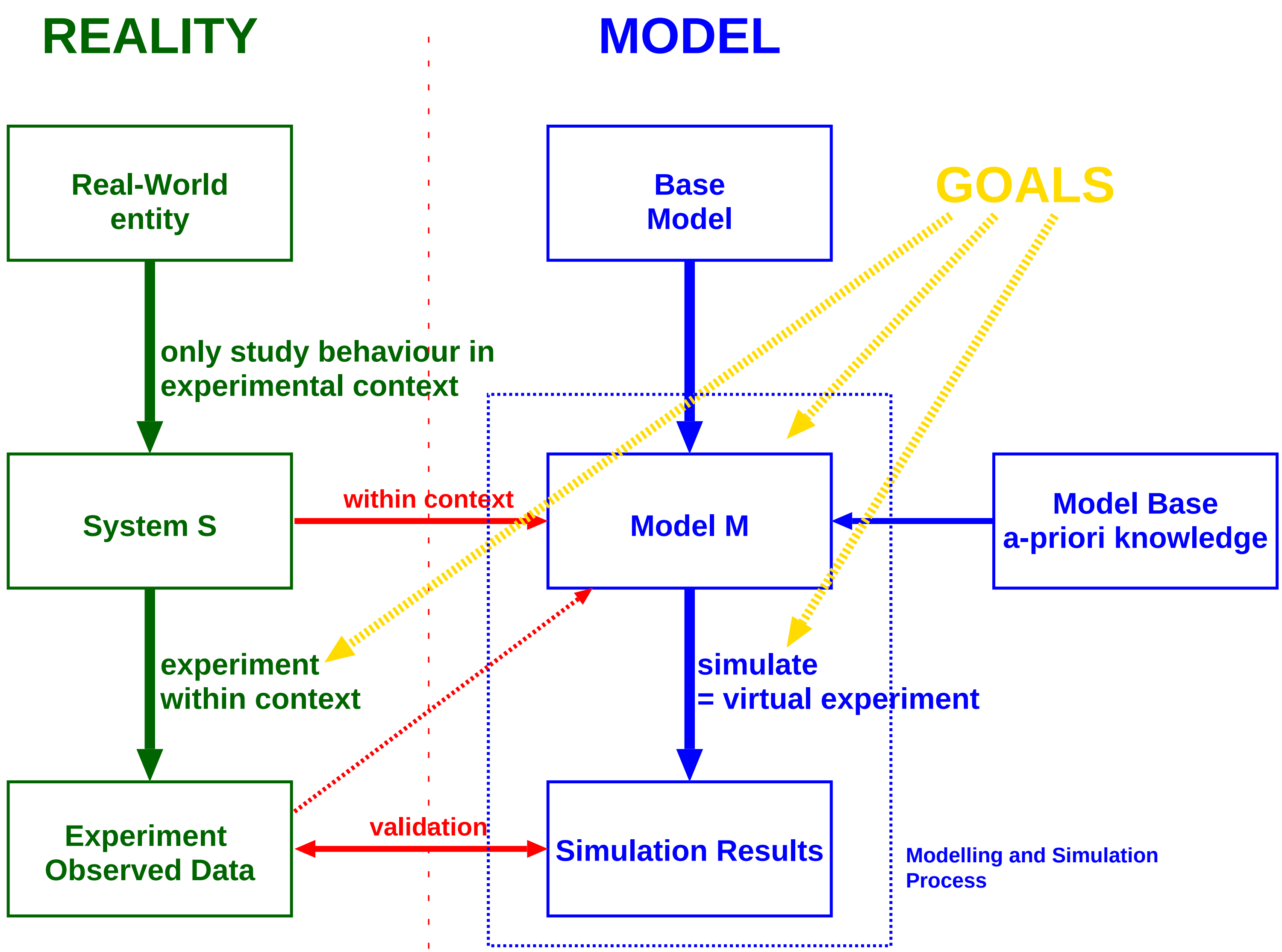
COMP 522: Modelling and Simulation

- . . . to study (static/dynamic) **structure** and (dynamic) **behaviour**
- . . . for **analysis** and **design** of **complex** systems
- . . . for different **application domains**:
  computer networks, software design, traffic control, software engineering, biology, physics, chemistry, management, . . .
- . . . implemented using Computer Science

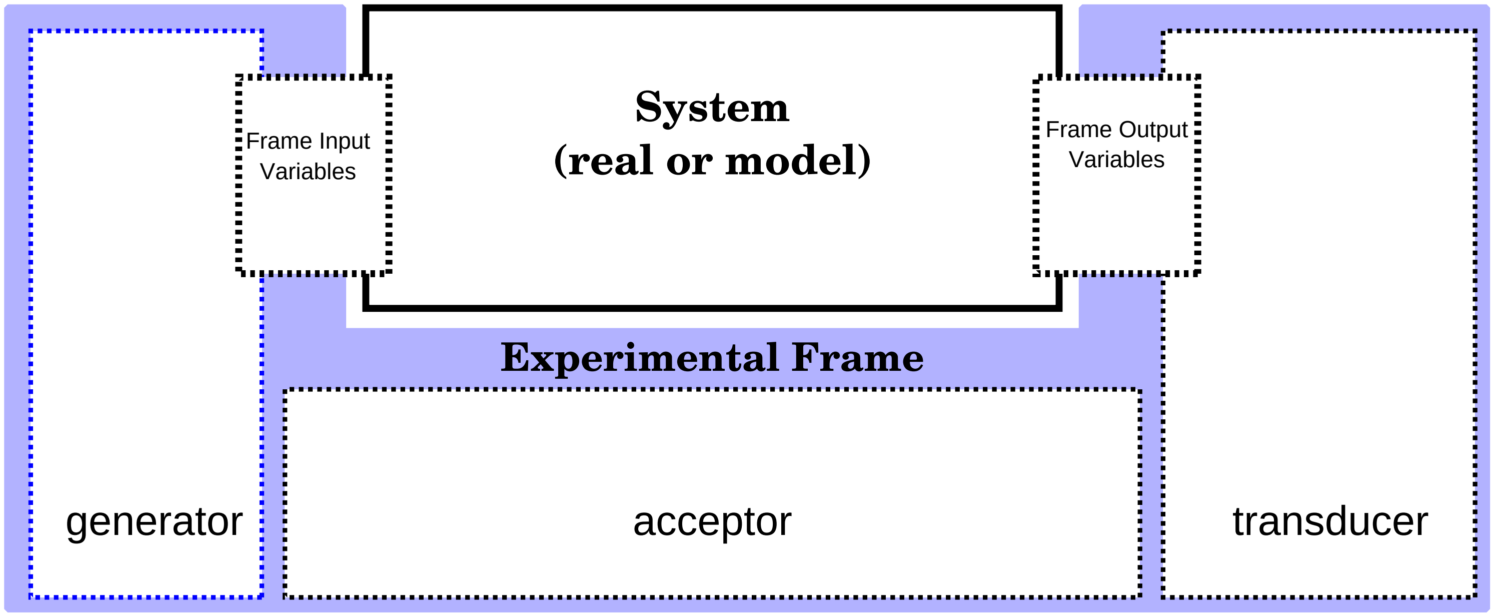
What is Modelling and Simulation ?

- **Modelling**: represent/re-use/exchange *knowledge* about system *structure* and *behaviour*
- **Simulation**: to *accurately* and *efficiently emulate* real behaviour

Modelling and Simulation Concepts



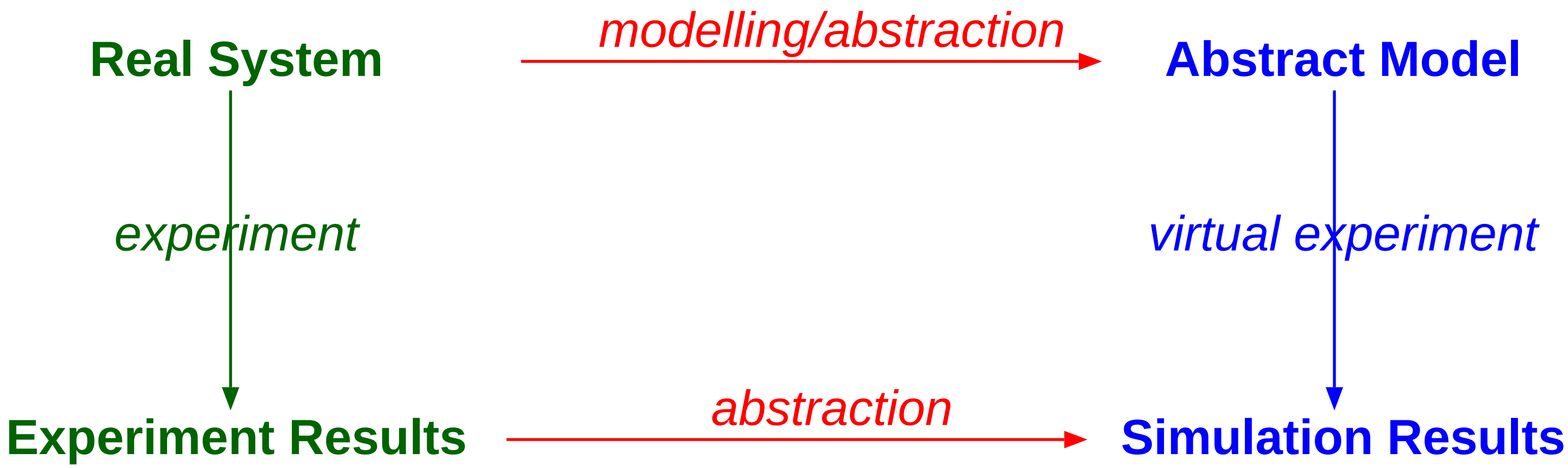
Experimental Frame (Zeigler)



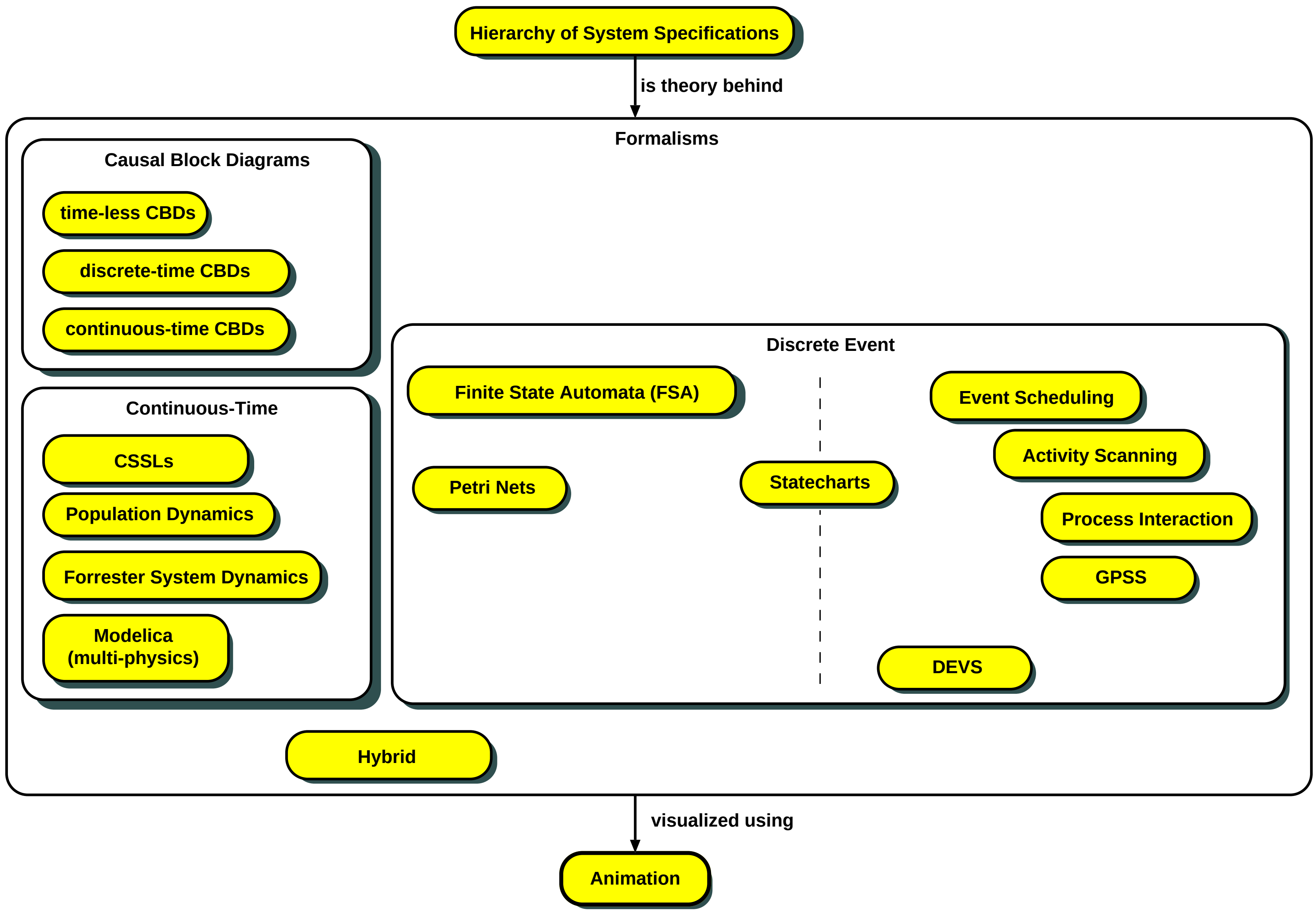
- set of all "contexts" in which model is valid
- includes experiment descriptions: parameters, initial conditions

$\sim$ re-use, testing

Behaviour morphism

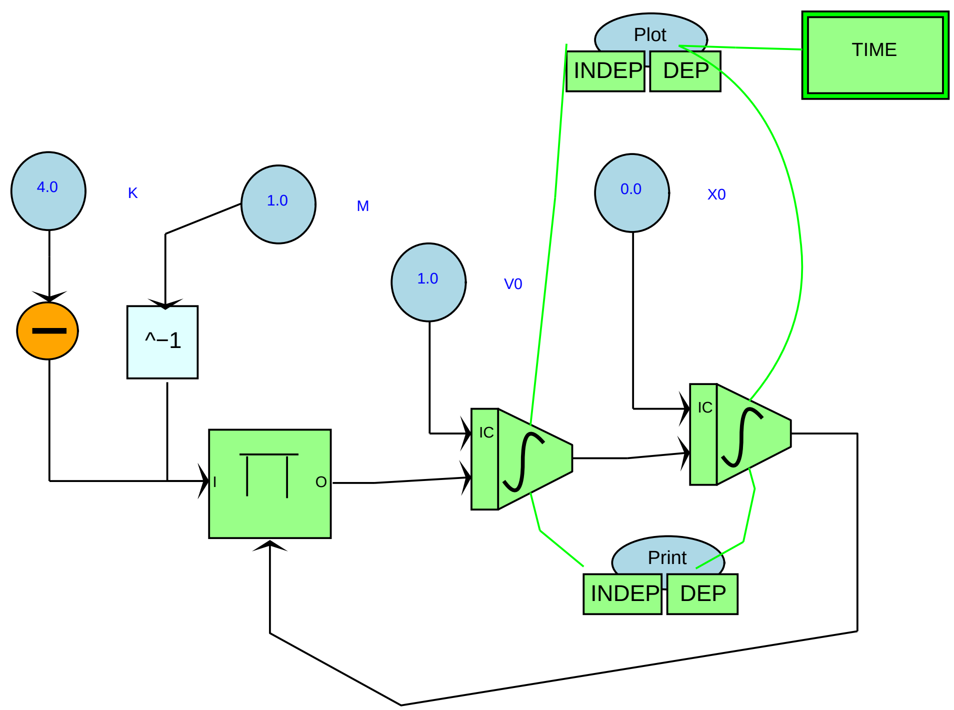


Which topics does the course cover ?



Which topics does the course cover ?

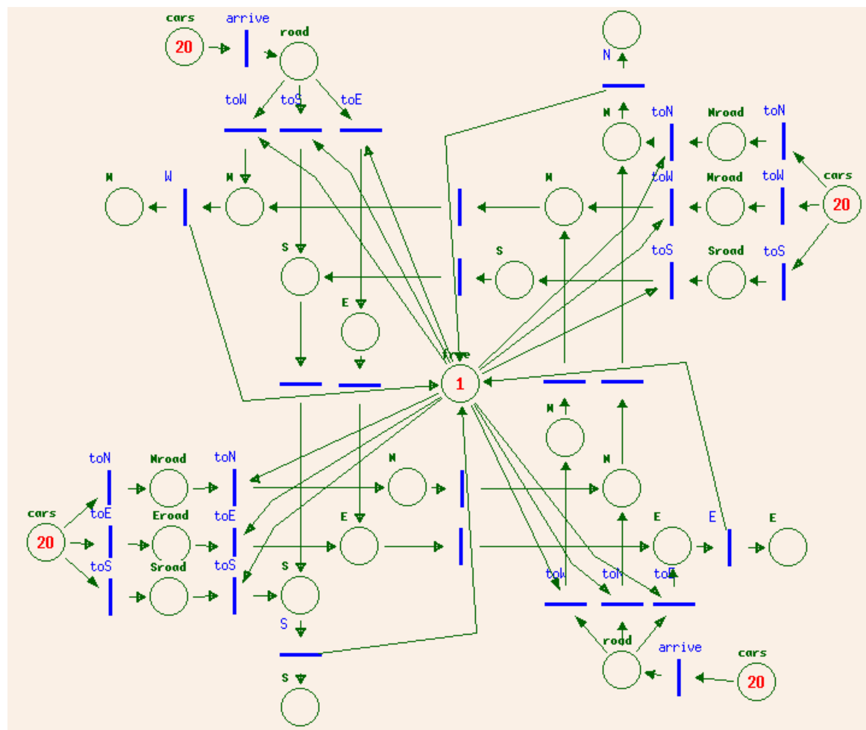
1. Modelling formalism *syntax* and *semantics*.
   The **Causal Block Diagram** formalisms.



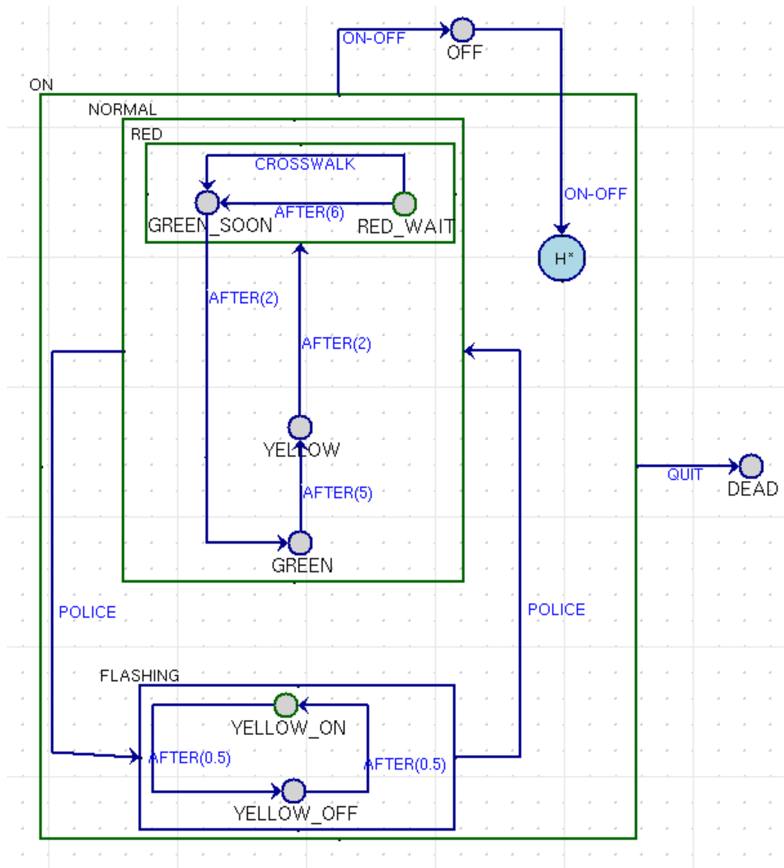
2. Untimed Discrete Event Formalisms:

   (a) (non)Deterministic **State Automata**.

   (b) Adding Concurrency and Synchronisation: **Petri Nets** (*e.g.*, specifying network protocols).

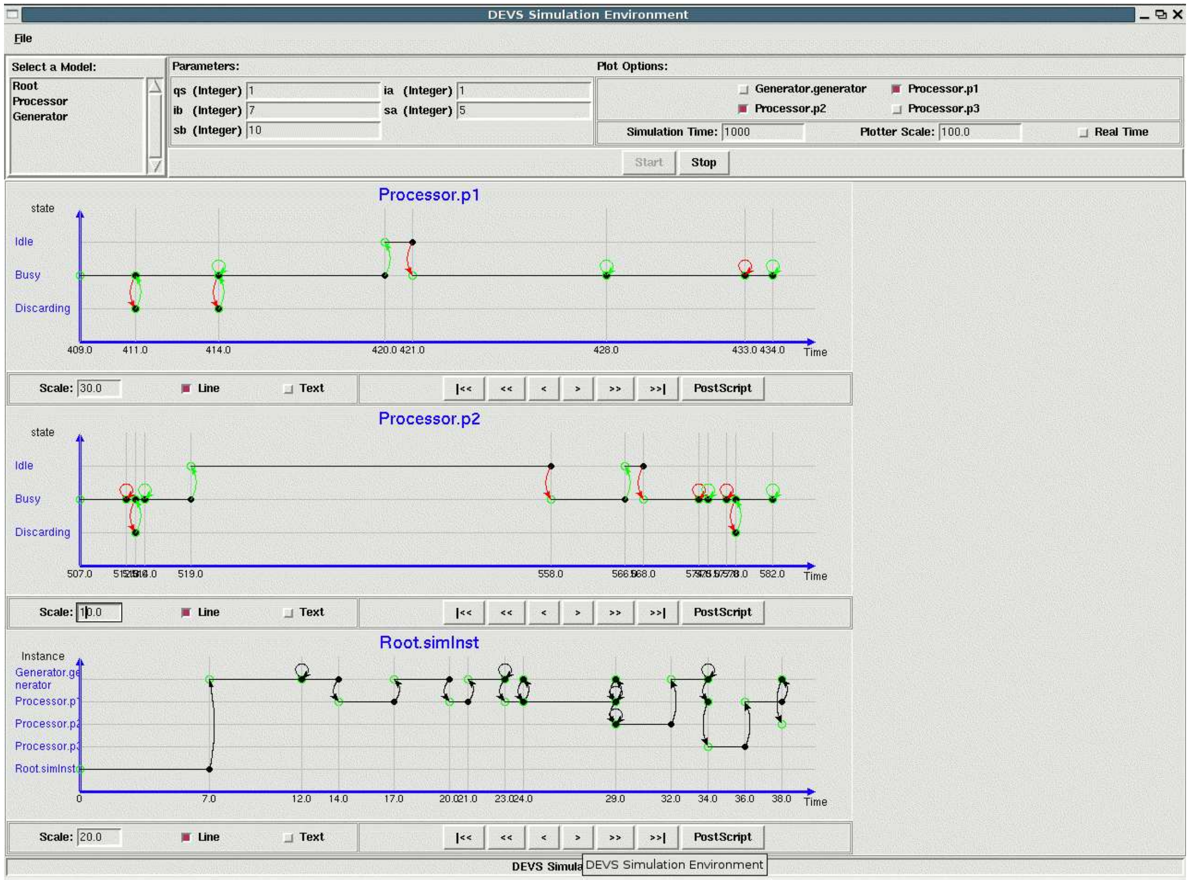


(c) Adding Hierarchy and Orthogonality: **Statecharts** (*e.g.*, UML, specifying reactive software).



(d) (Adding Space: Cellular Automata).

3. Timed Discrete Event Formalisms:

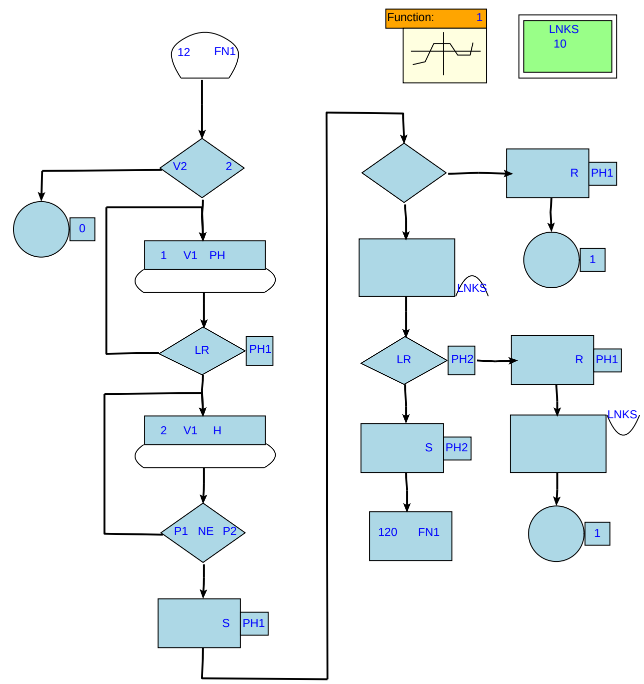


(a) **Event Scheduling**.

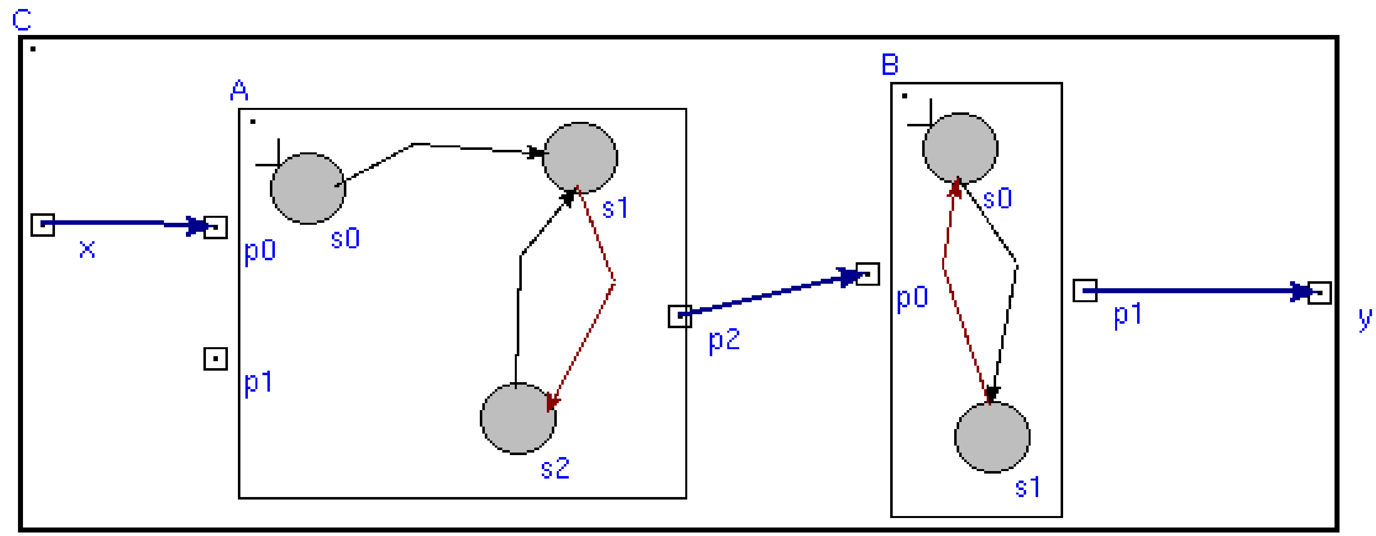
(b) **Activity Scanning**.

(c) **Three Phase Approach**.

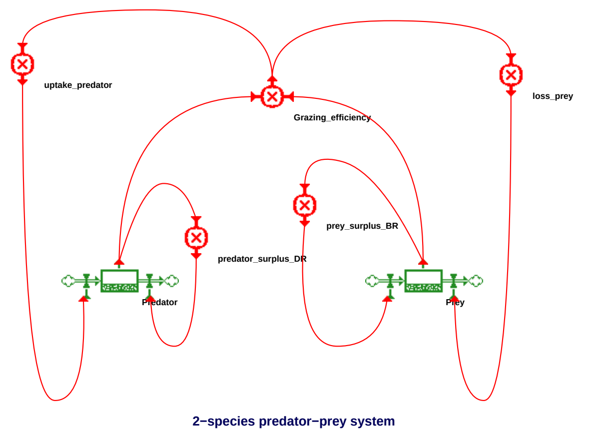
(d) **Process Interaction** for queueing systems (**GPSS**).

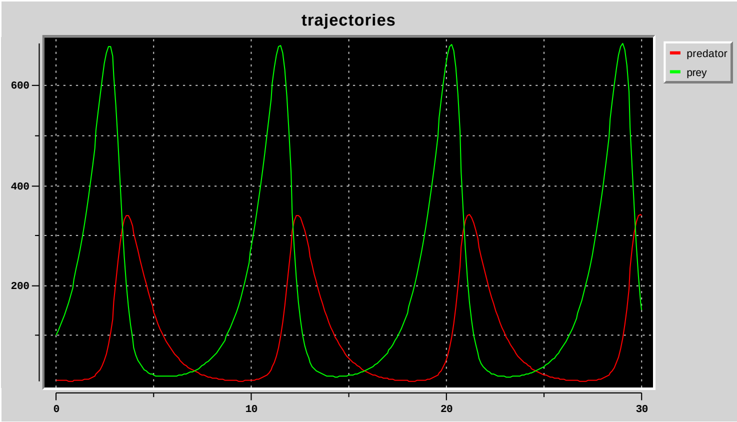


(e) **DEVS** as a rigourous basis for hierarchical modelling.



4. Deterministic Simulation of Stochastic Processes:
   (a) Pseudo Random Number Generation.
   (b) Gathering Statistics (performance metrics).
5. Animation
6. Continuous-time Formalisms:
   (a) **Ordinary Differential Equations**, Algebraic Equations, Differential Algebraic Equations.
   (b) CSSLs: sorting and algebraic loop detection.
   (c) **Forrester System Dynamics**, **Population Dynamics**.





(d) Object-oriented Physical Systems Modelling:
non-causal modelling, **Modelica** (`www.modelica.org`).

(e) Object-oriented Physical Systems Modelling: Bond Graphs.

7. Putting it all together (theory):
Hierarchy of System Specifications, Systems Theory.

8. **Hybrid** (continuous-discrete) modelling and simulation.

Do we live in a Simulation?

Questions?

Hierarchy of System Specification of Structure and Behaviour

- Basis of System Specification:
  sets theory, time base, segments and trajectories
- Hierarchy of System Specification (**causal**, **deterministic**)
  1. I/O Observation Frame
  2. I/O Observation Relation
  3. I/O Function Observation
  4. I/O System
- Multicomponent Specifications
- Non-causal models

ref: Waymore, Zeigler, Klir, . . .

Set Theory

Properties:

$$\{1, 2, \ldots, 9\}$$

$$\{a, b, \ldots, z\}$$

$$\mathbb{N}, \mathbb{N}^+, \mathbb{N}^+_\infty$$

$$\mathbb{R}, \mathbb{R}^+, \mathbb{R}^+_\infty$$

$$EV = \{ARRIVAL, DEPARTURE\}$$

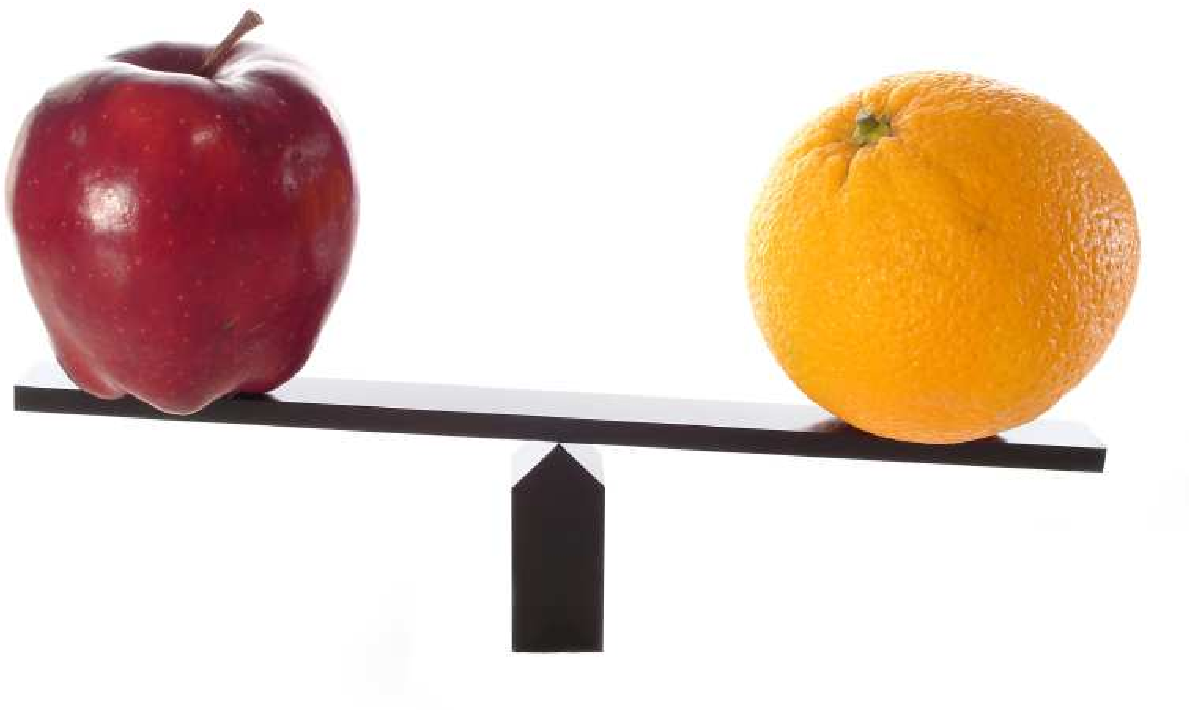
$$EV^\phi = EV \cup \{\phi\}$$

Structuring:

$$A \times B = \{(a, b) | a \in A, b \in B\}$$

$$G = (E, V), V \subseteq E \times E$$

Comparing things



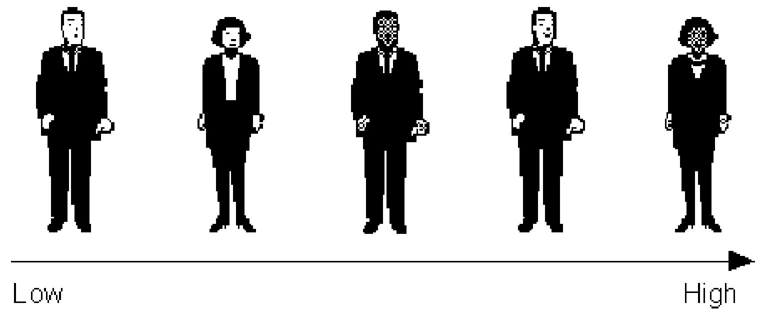
Nominal Scale: e.g., gender

A scale that assigns a *category label* to an individual.
Establishes no explicit ordering on the category labels.

Only a notion of *equivalence* "$=$" is defined with properties:

1. Reflexivity: $x = x \vee x \neq x$.
2. Symmetry of equivalence: $x = y \Leftrightarrow y = x$.
3. Transitivity: $x = y \wedge y = z \rightarrow x = z$.

Ordinal Scale: e.g., degree of happiness



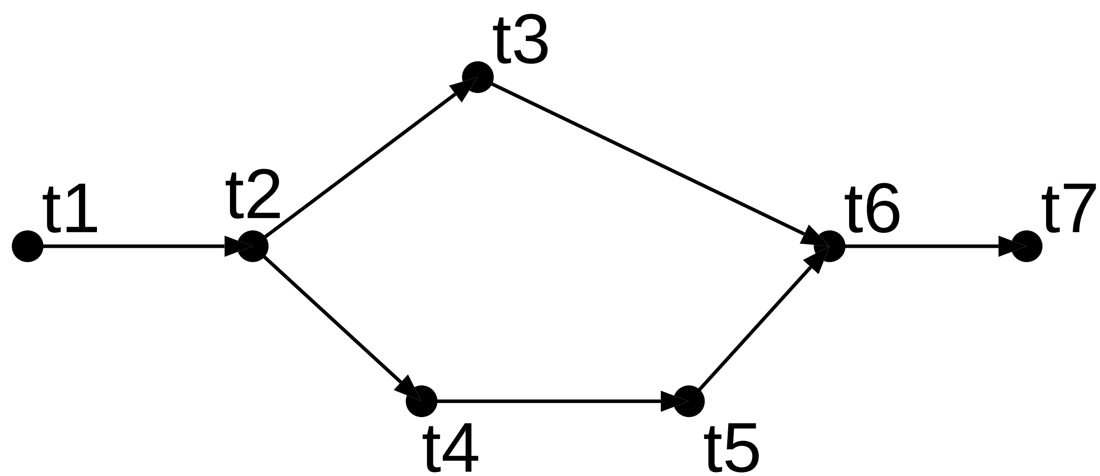
A scale in which data can be *ranked*, but in which no arithmetic transformations are meaningful. It is meaningless to talk about difference (distance).

In addition to equivalence, a notion of *order* $<$ is defined with properties:

1. Symmetry of equivalence: $x = y \Leftrightarrow y = x$.
2. Asymmetry of order: $x < y \rightarrow y \not< x$.
3. Irreflexivity: $x \not< x$.
4. Transitivity: $x < y \wedge y < z \rightarrow x < z$.

Partial ordering

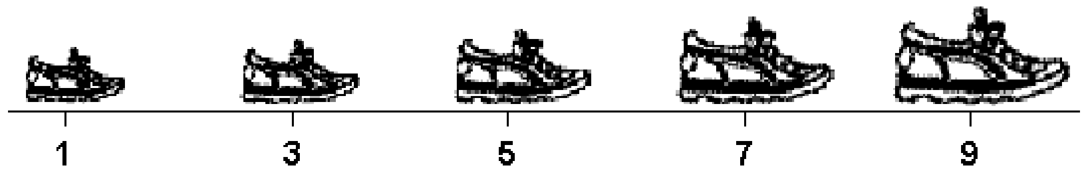
The ordering may be *partial* (some data items cannot be compared).



The ordering may be *total* (all data items can be compared).

$$\forall x, y \in X : x < y \lor y < x \lor x = y$$

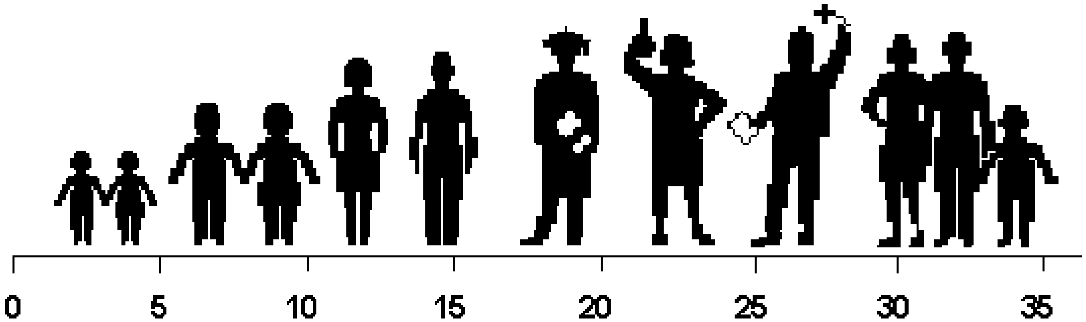
Interval Scale: e.g., Shoe Size



A scale where *distances* between data are meaningful. On interval measurement scales, one unit on the scale represents the *same magnitude* of the characteristic being measured across the whole range of the scale. Interval scales do not have a "true" zero point, however, and therefore it is not possible to make statements about how many times higher one value is than another.

In addition to equivalence and order, a notion of *interval* is defined. The choice of a zero point is arbitrary.

Ratio Scale: e.g., age



Both *intervals* between values and *ratios* of values are meaningful. A meaningful *zero* point is known. "A is twice as old as B".

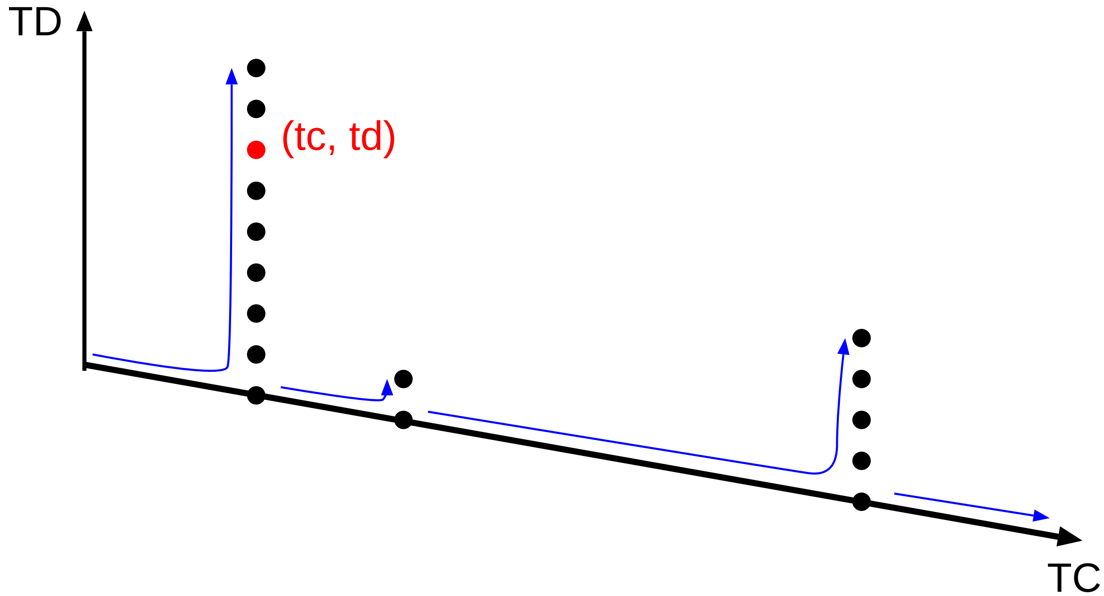
Time Base

- Simulation of **Dynamic** Systems: irreversible passage of *time*.

- Time Base T:
  - $\{NOW\}$ (instantaneous)
  - $\mathbb{R}$: *continuous-time*
  - $\mathbb{N}$ or isomorphic: *discrete-time*
- Ordering:
  - Ordinal Scale (possibly partial ordering)
  - Interval Scale
  - Ratio Scale

Time Bases for hybrid system models

Time Bases for hybrid system models

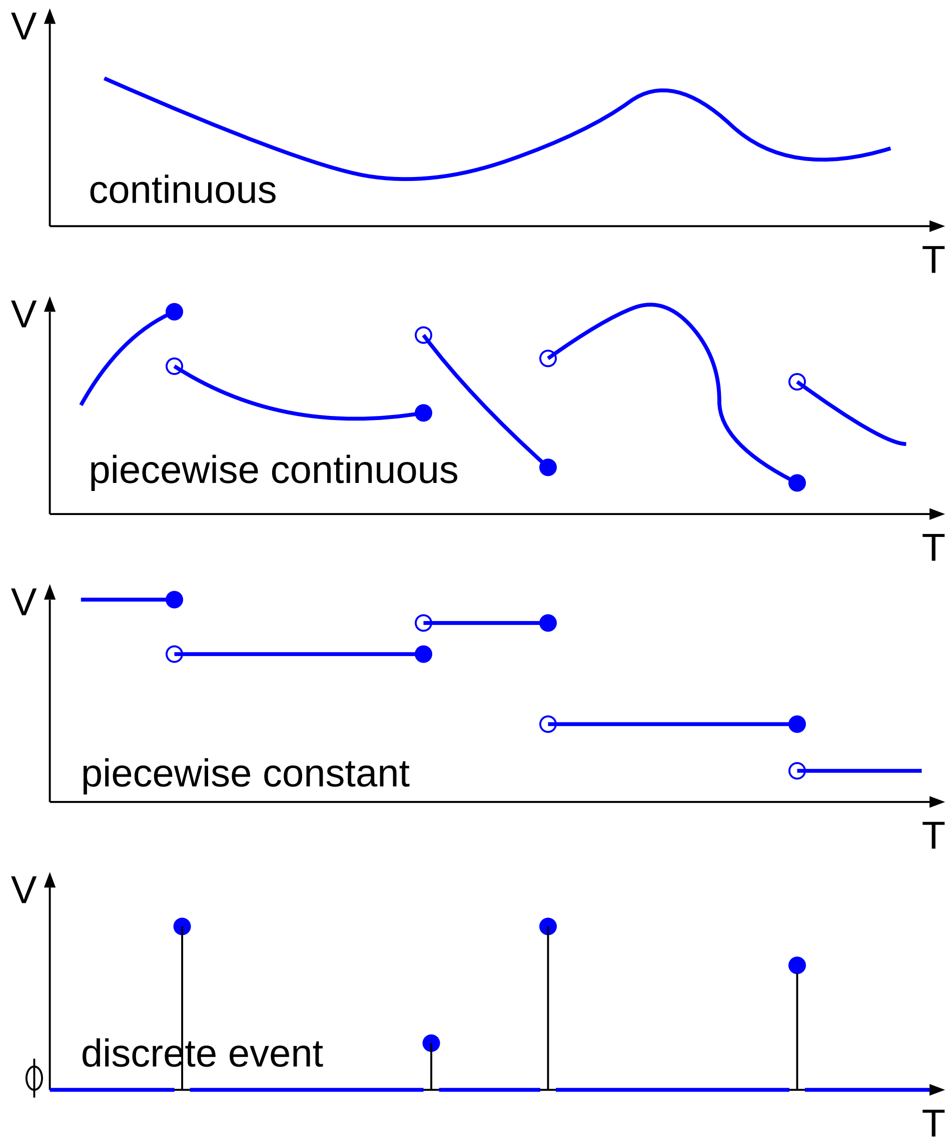


"nested time" for nested experiments.

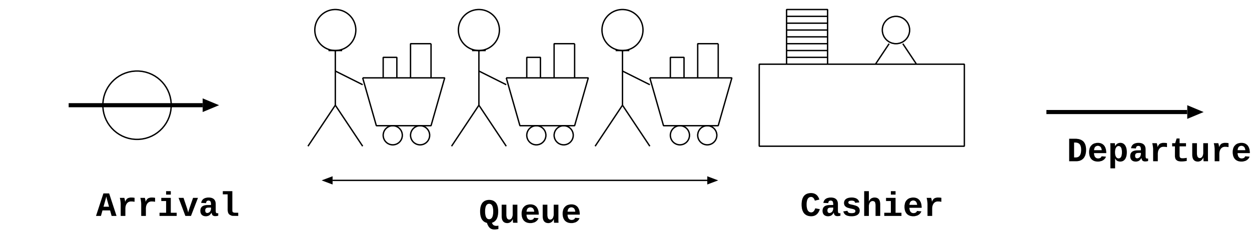
Behaviour $\equiv$ Evolution over Time

- With time base, describe *evolution over time*
- Time function, **trajectory**, signal: $f : T \rightarrow V$
- Restriction to $T' \subseteq T$
  $f|T' : T' \rightarrow V$, $\forall t \in T' : f|T'(t) = f(t)$
  - Past of f: $f|T_{t\rangle}$
  - Future of f: $f|T_{\langle t}$
- Restriction to an interval: **segment**
  $\omega : \langle t_1, t_2 \rangle \rightarrow V$

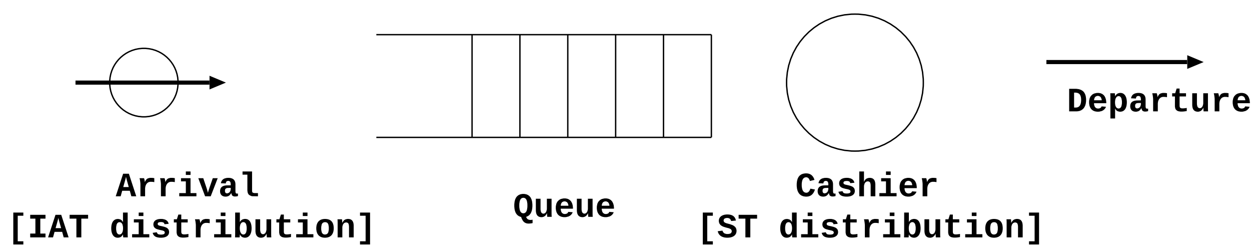
Types of Segments



Cashier-Queue System



Physical View



Abstract View

Trajectories