On Under-Determined Dynamical Systems

Oded Maler

CNRS - VERIMAG
Grenoble, France

EMSOFT 2011

The ABC of Model Based Design

- To build complex systems other than by trial and error you need **models**
- Regardless of the language or tool used to build a model, at the end there is some kind of **dynamical system**
- A mathematical entity that generates **behaviors** which are progression of states and events in time
- Sometimes you can reason about such systems analytically
- But typically you **simulate** the model on the computer and generate behaviors
- If the model is related to **reality** you will learn something from the simulation about the **actual** behavior of the system which is, after all, the goal

The Message of this Talk

- **Under-determined** dynamical systems: systems where not all the details have been filled out
- Systems that need additional information in order to produce a **simulation** trace
- This information is taken from some **uncertainty space** (or **ignorance space**)
- We make distinction between **static** (punctual) and **dynamic** under-determination
- Simulation, testing, formal verification, monte-carlo, parameter-space exploration are all different ways to take this uncertainty into account

Outline

- Dynamical systems: **continuous**, discrete, hybrid and **timed**
- Static under-determination: initial states and parameters
- Sensitivity-based exploration of parameter space
- Dynamic under-determination: ongoing influence of the external environment (external = outside the model)
- Handling dynamic under-determination: test coverage and reachability computation for continuous systems
- Two slides on timed systems

Dynamical Systems in General

- The following abstract features of dynamical systems are common to both **continuous** and **discrete** systems:
- **State variables** whose set of **valuations** determine the **state space**
- A **time domain** along which these values evolve
- A **dynamic law** which says **how** state variables evolve over time, possibly under the influence of **external** factors
- System **behaviors** are **progressions** of **states** in **time**
- Having such a model, knowing an initial state $x[0]$ one can **predict**, to some extent, the value of $x[t]$

Classical Dynamical Systems

- State variables: **real numbers** (location, velocity, energy, voltage, concentration)
- Time domain: the **real time axis** $\mathbb{R}$ or a discretization of it
- Dynamic law: **differential equations**

  $$\dot{x} = f(x, u)$$

  or their **discrete-time** approximations

  $$x[t+1] = f(x[t], u[t])$$

- Behaviors: **trajectories** in the continuous state space
- What you would construct using tools like Matlab Simulink, Modelica, etc.

Discrete-Event Dynamical Systems (Automata)

- An **abstract discrete state space**, state variables need **not** have a numerical meaning
- A **logical time domain** defined by the **events** (order but not metric)
- Dynamics defined by **transition rules**: input event **a** takes the system from state **s** to state **s**′
- Behaviors are **sequences** of **states** and/or **events**
- **Composition** of large systems from small ones using: different modes of **interaction**: synchronous/asynchronous, state-based/event-based
- What you will build using tools like Raphsody or Stateflow (or even C programs or digital HDL)

Timed and Hybrid Systems

- Mixing discrete and continuous dynamics
- **Hybrid automata**: automata with a different continuous dynamics in each state
- Transitions = mode switchings (valves, thermostats, gears)
- **Timed systems**: an intermediate level of abstraction
- Timed Behaviors = discrete events embedded in metric time, Boolean signals, Gantt charts
- Used implicitly by **everybody** doing real-time, scheduling, embedded, planning in professional **and** real life
- Formally: **timed automata** (automata with clock variables)

Dynamical Models

- A dynamical system model generates behaviors (runs, trajectories, executions ...)
- A **trace**:
$$x[0], x[1], x[2], \dots$$
- What does a simulator need to produce such a trace?
- For **deterministic** systems the dynamic rule is a function $f : X \to X$
- The rule allows the simulator to proceed from one state to another
$$x[i+1] = f(x[i])$$
- You just have to **fix** the initial state $x[0]$

Static/Punctual Under-Determination

- Some systems may have a **unique** initial state (reboot)
- Otherwise, to produce a trace you need to fix $x[0]$
- Without this information, the system is **under-determined** and **cannot** generate a trace
- It has an **empty slot** that needs to be filled by some **point** in $x \in X_0 \subseteq \mathbb{R}^n$, the set of all possible initial states
- Hence we call it **punctual** under-determination

Reminder: Models and Reality

- Whenever our models are supposed to represent something non-trivial they are just **approximations**
- This is evident for anybody working in modeling concrete **physical** systems
- It is less so for those working on the functionality of **digital hardware** or **software**
- There you have strong **deterministic** abstractions (logical gates, program instructions)
- A common way to pack our ignorance in a compact way is to introduce **parameters** ranging in some **parameter space**

Examples:

- **Biochemical reactions** in cells following the **mass action** law
- Many parameters related to the affinity between molecules
- Cannot be deduced from first principles, only measured by isolated experiments under different conditions
- **Voltage level** modeling and simulation of circuits
- A lot of variability in transistor characteristics depending on production batch, place in the chip, temperature, etc.
- **Timing performance analysis** of a new application (task graph) on a new multi-core architecture
- Precise execution times of tasks are not known before the application is written and the architecture is built

Parameterize Dynamical Systems

- The dynamics f becomes a **template** with some empty slots to be filled by parameter values
- Taken from some parameter space $P \subseteq \mathbb{R}^m$
- Each p instantiates f into a concrete function f_p that can be used to produce traces
- Parameters like initial states are instances of **punctual** under-determination: you choose them only once when starting the simulation
- In fact, you can add the parameters as **static** state variables, replacing (X, f) by (X', f'):

$$X' = X \times P \qquad f'(x, p) = (f_p(x), p)$$

- As if at time zero the system decides which dynamics to follow

So What?

- So you have a model which is under-determined, or equivalently an **infinite** number of models
- For simulation you **need** to determine, to make a choice to pick a point p in the parameter space
- The simulation shows you something about **one** possible behavior of the system, or a behavior of **one** possible system
- But another choice of parameter values could have produced a completely different behavior
- Ho do you live with that?

Possible Attitudes

- The answer depends on many factors
- One is the **responsibility** of the modeler/simulator
- What are the consequences of not taking under-determination seriously
- Is there a penalty for jumping into conclusions based on one or few simulations?
- Another factor is the mathematical and real natures of the system you are dealing with
- And as usual, it may depend on culture, background and tradition in the industrial or academic community

Non Responsibility: a Caricature

- Suppose you are a scientist not engineer, say biologist
- You conduct experiments and observe traces
- You propose a model and **tune** the parameters until you obtain a trace similar to the one observed experimentally
- These are **nominal** values of the parameters
- Then you can publish a paper about your model
- Except for picky reviewers there are no real consequences for neglecting under-determination
- The situation is different if some engineering is involved (pharmacokinetics, synthetic biology)
- Or if you want others to **compose** their models with yours

Justified Nominal Value

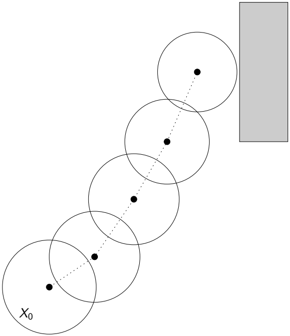
- You can get away with using a nominal value if your system is very **continuous** and **well-behaving**
- Points in the neighborhood of p generate **similar** traces
- There are also mathematical techniques (bifurcation diagrams, etc.) that can tell you sometimes what happens when you change parameters
- This smoothness is easily broken by mode switching
- Another justification for ignoring parameter variability:
- When the system is adaptive anyway to deviations from nominal behavior (control, feedback)

Taking Under-Determination More Seriously: Sampling

- One can **sample** the parameter space with or without probabilistic assumptions
- Make a grid in the parameter space (exponential in the number of parameters)
- Or pick parameter values at random according to some distribution
- In the sequel I illustrate a technique (due to **A. Donze**) for adaptive search in the parameter space
- Sensitivity information from the numerical simulator tells you where to **refine** the coverage
- Arbitrary dimensionality of the state space, but no miracles against the dimensionality of the parameter space

Sensitivity-based Exploration I

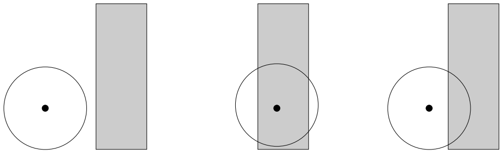
- We want to prove **all** trajectories from X_0 do not reach a bad set of states
- Take $x_0 \in X_0$ and build a ball B_0 around it that covers X_0



- Simulate from x_0 and generate a sequence of balls $B_0, B_1, \ldots$
- B_i **contains** all points reachable from B_0 in i steps

Sensitivity-based Exploration II

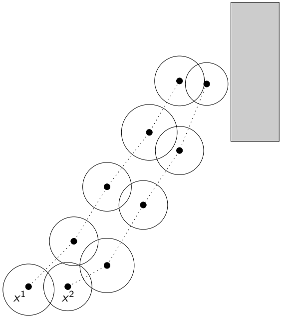
- After k steps, three things may happen:



- 1. No ball intersects bad set and the system is **safe** (over-approximation)
- 2. The concrete trajectory intersects the bad set and the system is **unsafe**
- 3. Ball B_k intersects the bad set but we do not know if it is a real or spurious behavior

Sensitivity-based Exploration III

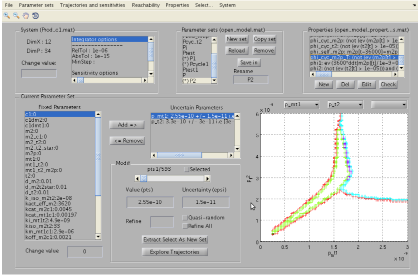
- In the latter case we refine the coverage and repeat the process for two **smaller** balls

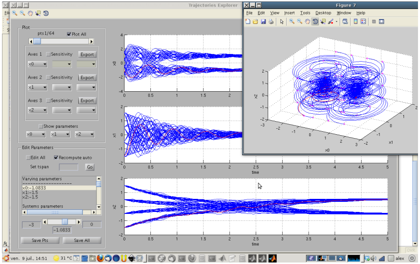


- Can prove correctness using a **finite** number of simulations, focusing on the interesting values
- Can approximate the boundary between parameter values that yield some qualitative behaviors and values that do not

The Breach Toolboox

- Parameter-space exploration for arbitrary continuous dynamical systems relative to **quantitative temporal properties**
- Applied to embedded control systems, analog circuits, biochemical reactions
- Available for download





Dynamic Under-Determination

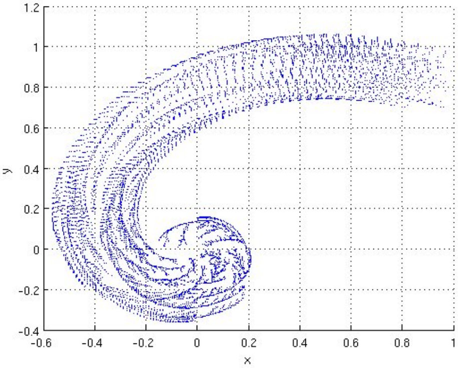
- The system is modeled as **open**, exposed to external disturbances
- Dynamics of the form

$$x[i+1] = f(x[i], v[i])$$

- The natural way to represent the influence of other unmodeled subsystems and the external environment
- Under-determination becomes dynamic: to produce a trace you need to give the value of v at every step in time, a signal/sequence $v[1], \ldots, v[k]$
- A priory a much larger space to sample from: dimension mk compared to m for static
- One can use a nominal value: constant, step, periodic signal, random noise, etc.

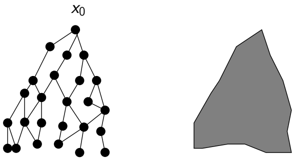
Taking Under-Determination More Seriously: Sampling

- A method due to **T. Dang**:
- Use ideas from robotic motion planning (RRT) to generate inputs that yield a good **coverage** of the reachable state space
- Applied to analog circuits



Taking Under-Determination More Seriously: Verification

- Paranoid **worst-case** formal verification attitude:
- If we say something about the system it should be provably true for **all** choices of p, $x[0]$ and $v[1], \dots, v[k]$
- Instead of doing a simple simulation you do **set-based** simulation, computing **tubes of trajectories** covering everything
- Breadt-first rather than depth-first exploration

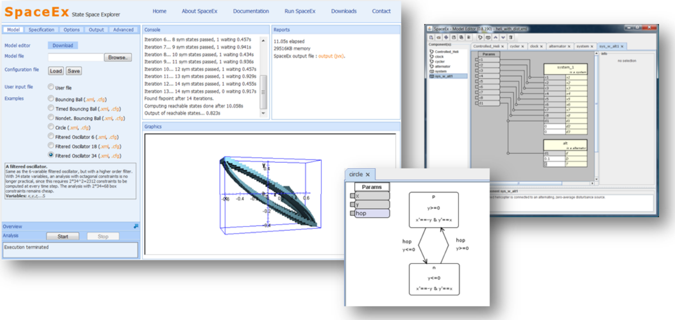


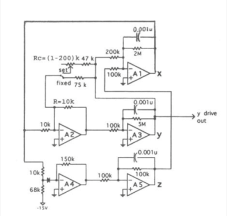
- Advantages: works also for hybrid (switched) systems
- Limitations: manipulates geometric objects in high dimension

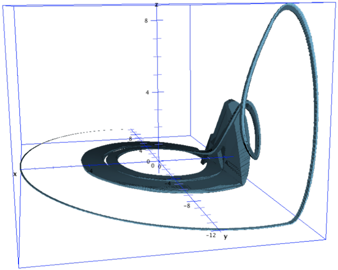
State of the Art

- Linear and piecewise-linear dynamics ~ 200 variables using algorithms of **C. Le Guernic and A. Girard**
- The technique is explained in the proceedings article
- Nonlinear dynamics with $10 - 20$ variables - an ongoing research activity
- Implemented into the **SpaceEx** tool developed under the direction of **G. Frehse**
- Available on `http://spaceex.imag.fr` with web interface, model editor, visualization and more
- Waiting for more beta testers

The State-Space Explorer (SpaceEx)







Timed Dynamical Systems

- Processes that take some time to conclude after having started:
  - Propagation delay between *send* and *receive*
  - Execution time of a program
  - Duration of a step in a manufacturing process
- Mathematically they are simple timed automata:

$$\boxed{\overline{p}} \xrightarrow[]{\textit{start}\; x := 0} \boxed{p} \xrightarrow[]{\phi(x)\; \textit{end}} \boxed{\underline{p}}$$

- A waiting state $\overline{p}$; a *start* transition which resets a clock x to measure time elapsed in active state p
- An *end* transition guarded by a temporal condition $\phi(x)$
- Condition ϕ can be $x = d$ (deterministic) or $x \in [a, b]$ (non-deterministic)

Handling Timed Under Determination

- We want to analyze the behavior of a complex network of such under-determined timed components
- The product of the duration intervals associated with each process form the **duration space**
- We can choose a nominal value for each duration, simulate and see what happens
- We can try to compute for all possible values (verification of timed automata a-la UPPAAL, IF)
- We can **sample** under some probabilistic assumptions
- We can even try to compute expected behavior in a piecewise-analytic manner

The Message of this Talk

- **Under-determined** dynamical systems: systems where not all the details have been filled out
- Systems that need additional information in order to produce a **simulation** trace
- This information is taken from some **uncertainty space**
- We make distinction between **static** (punctual) and **dynamic** under-determination
- Simulation, testing, formal verification, monte-carlo, parameter-space exploration are all different ways to take this uncertainty into account