Transition Systems
Computational Models for Complex Systems

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa
http://pages.di.unipi.it/milazzo
milazzo@di.unipi.it

Laurea Magistrale in Informatica
A.Y. 2019/2020
Introduction

Assume we are interested in studying whether a system can reach a bad state.

We have seen that

- ODEs allow us to study the “average” behavior of a system.
- Stochastic simulation provide us with a number of different possible behaviors (as many as the simulations we run).

How can we guarantee that the systems will never reach the bad state?

- Simulations are not exhaustive.
Introduction

For example, assume we want to avoid $X$ to reach value 0

We may run 1000 simulations and conclude that this never happens...
Introduction

For example, assume we want to avoid $X$ to reach value 0

... but maybe this happens very rarely (e.g. once every $10^6$ times) and simply it didn’t happen in the 1000 simulations!
Introduction

From the 1000 simulations we can conclude that the probability that $X$ reaches 0 is very small (roughly, smaller than $1/1000$), but they cannot guarantee that this will never happen.

In some cases we would like to explore all possible systems behaviors

- in order to verify behavioral properties
- e.g. in the study of safety-critical systems

This requires a new way of modeling the system behavior:

- Transition Systems
Roadmap

Where are we going?
Roadmap

Where are we going?
Roadmap

Where are we going?
Roadmap

Where are we going?
Roadmap

Where are we going?

MODELLING LANGUAGES

MODELS OF BEHAVIOR

ANALYSIS METHODS

CUEm

Petri Nets

CCS

MSR

... D

ODEs

TRANSMITION SYSTEMS

Numerical Simulation

Stochastic Simulation

Model Checking

Agents

Prog. Lang.

Discrete Events Simulation
Transition Systems

**Definition:** Transition System (TS)

A Transition System is a pair \((S, \rightarrow)\) where

- \(S\) is a set of states and
- \(\rightarrow \subseteq S \times S\) is the transition relation

Given \(s, s' \in S\), \((s, s') \in \rightarrow\) is usually denoted as \(s \rightarrow s'\). Moreover, usually \(s \nrightarrow\) denotes that there exists no \(s' \in S\) such that \(s \rightarrow s'\).

So, a TS is essentially a graph...

A TS aims ad **modeling the behavior** of a system. Note that:

- the set of states can be **infinite** (but it is typically assumed to be recursively enumerable)
- transitions describe system state changes
- a state may have more than one outgoing transitions \((s \rightarrow s'\) and \(s \rightarrow s''\)) capturing **non-deterministic behaviors**
Transition Systems

Non-deterministic does not mean random!

The fact that $s_1$ may evolve either into $s_2$ or into $s_3$ does not mean that there is a random choice between the two possibilities.

The choice could depend on:

- a timer (e.g. “enter the PIN in 30 sec. or your card will be eaten”)
- a scheduler (e.g. if the two transitions correspond to the progresses of two different concurrent processes)
- a probabilistic choice

In general, non-determinism is an abstraction of a choice criterion that we simply do not want to model...
Transition Systems

Example: soccer match

Diagram showing the possible outcomes of a soccer match from 0-0 to 2-2, with final scores indicated in red.
Traces

In a TS \((S, \rightarrow)\), often one state \(s_0 \in S\) is chosen as initial state.

A possibile behavior of the system starting from the initial state \(s_0\) corresponds to a trace of the TS.

**Definition: Trace**

A trace \(t\) of a Transition System \((S, \rightarrow)\) with initial state \(s_0\), is a (possibly infinite) sequence of states \(t = s_0, s_1, s_2, \ldots\) such that for each \(s_{i+1}\) with \(i \in \mathbb{N}\) in \(t\) it holds \(s_i \rightarrow s_{i+1}\).

A few notes:

- \(s_0\) is the minimal trace
- a trace \(t\) is maximal if either
  - \(t\) is infinite
  - \(t = s_0, s_1, \ldots, s_n\) and \(s_n \not\rightarrow\)
Traces

In the soccer example (with initial state (0-0)):

- (0-0),(1-0),(1-1) is a trace
- (0-0),(1-0),(1-1),(1-1 final) is a maximal trace
- (0-0),(1-0),(2-0),(3-0),... is a maximal trace
Reachability

Traces allow us to define a notion of reachability of states.

**Definition: Reachability**

A state $s$ of a Transition System $(S, \rightarrow)$ with initial state $s_0$ is reachable (from the initial state) if there exist $s_1, \ldots, s_n \in S$ such that $s_0, s_1, \ldots, s_n, s$ is a trace.

Very often, reachability of a particular (good or bad) state is the property one wants to verify on a TS:

- for example through a Breadth-First Search (BFS), with on-the-fly generation of states (if the graph is huge or infinite)
Kripke Structures

A particular class of Transition Systems are Kripke Structures

In a Kripke Structure, states are characterized by a set of atomic propositions that can be either true or false

**Definition: Kripke Structure**

Given a (finite) set of atomic propositions $AP$, a Kripke Structure $K$ is a Transition System $(S, \rightarrow)$ where $S = \mathcal{P}(AP)$.

Some notes:

- $\mathcal{P}(AP)$ denotes the powerset of $AP$.
- The interpretation is that an atomic proposition $a$ is contained in a state if and only if it is true in that state.
Kripke Structures

An example: Microwave Oven
Transition Systems over a set of variables

Very often the definition of the state of a transition system is based on a set of variables describing the features of the state.

**Definition:** Transition System over a set of variables

Given a set of variables $X = \{X_1, \ldots, X_n\}$ and a set of domains $\{D_1, \ldots, D_n\}$ s.t. $D_i$ is the domain of $X_i$, a Transition System over $X$ is a Transition System $(S, \rightarrow)$ with $S = D_1 \times \cdots \times D_n$.

Some notes:

- Each domain $D_i$ should be a recursively enumerable set of values (integers, naturals, rationals, ...) or, better, a finite set of values (bounded integers, bounded rationals, enums, booleans, ...).
- The number of variables impacts significantly on the number of states of the TS (combinatorial explosion).
Transition Systems over a set of variables

Example: soccer match
- Variables: team1 (natural), team2 (natural), final (boolean)
Transition Systems over a set of variables

Example: server with a queue
- if the server is busy, requests are enqueue
- the size of the queue is $N$
- Variables: $\text{busy}$ (boolean), $q$ (bounded natural)

Example: Kripke structures
- Kripke structures are a particular case of TS over a set of (boolean) variables
Specifying Transition Systems

Transition Systems over a set of variables can be specified by giving a set of transition rules (or if-then rules) having the following form:

\[
guard \rightarrow \text{update}
\]

where

- **guard** is a conjunction of conditions on the state variables, each having the form \(X_i \text{ op } Exp\) with \text{ op} a comparison operator.
- **update** is a conjunction of assignments to state variables, each having the form \(X'_i = Exp\), with \(X'_i\) denoting the new value of \(X_i\).
Specifying Transition Systems

\[ \text{guard} \rightarrow \text{update} \]

The idea is that the transition relation with contain a transition between each pair of states \( s_1, s_2 \) such that:

- \( s_1 \) satisfies the guard
- \( s_2 \) can be obtained by applying to \( s_1 \) the assignments described in update
Specifying Transition Systems

Example: soccer match

- Variables: team1 (natural), team2 (natural), final (boolean)

Specification:

- final=false → team1’=team1+1
- final=false → team2’=team2+1
- final=false → final’=true
Specifying Transition Systems

Example: server with a queue

- if the server is busy, requests are enqueue
- the size of the queue is $N$
- Variables: busy (boolean), $q$ (bounded natural)

Specification:

$$\begin{align*}
\text{busy}=\text{false} & \rightarrow \text{busy}'=\text{true} \\
\text{busy}=\text{true} \land q<N & \rightarrow q'=q+1 \\
q>0 & \rightarrow q'=q-1 \\
q=0 & \land \text{busy}=\text{true} \rightarrow \text{busy}'=\text{false}
\end{align*}$$
Labeled Transition Systems

Labeled Transition Systems are an extended version of Transition Systems in which transitions are enriched with labels.

**Definition: Labeled Transition System (LTS)**

A Labeled Transition System (LTS) is a triple \((S, L, \rightarrow)\) where:

- \(S\) is a set of states,
- \(L\) is a set of labels, and
- \(\rightarrow \subseteq S \times L \times S\) is a labeled transition relation.

Given \(s, s' \in S\) and \(\ell \in L\), \((s, \ell, s') \in \rightarrow\) is usually denoted \(s \xrightarrow{\ell} s'\).

Moreover, usually \(s \nrightarrow \ell\) denotes that there exists no \(s' \in S\) such that \(s \xrightarrow{\ell} s'\).
LTSs and Concurrent Interactive Systems

LTSs are usually well suited to model the behavior of CONCURRENT INTERACTIVE SYSTEMS in a COMPOSITIONAL WAY

- Concurrent interactive systems are systems consisting of a number of independent components which may perform some actions synchronizing with each other or with the environment.
- Compositional modeling consist in inferring the model of the behavior of the system from the models of the behaviors of the components.
- The idea:
  1. Specify the LTS of each component
  2. Combine the LTSs by taking synchronizations into account
The role of transition labels:

- transition labels describe the action performed by the system (or component) during the transition
- Label $\tau$ describes an internal action
  - performed in isolation, without synchronizing with any other component
- Other labels, $a, b, c, \ldots$ describe potential actions the system (or component) could perform by interacting with some other component
Two approaches to synchronization:

- **Binary synchronization**: non-$\tau$ actions are split into two sets denoted $\{a, b, c, \ldots\}$ and $\{\overline{a}, \overline{b}, \overline{c}, \ldots\}$.
  
  ▶ a transition with label $a$ has to be performed together with a transition with label $\overline{a}$ (same symbol, but with overline) of another component

- **Global synchronization**: non-$\tau$ actions are synchronized among all system components
  
  ▶ all components having a transition with label $a$ must perform such a transition together

The synchronization of a number of transitions result in a new $\tau$ transition.
Example: compositional modeling of a coffee machine

LTSs modeling the behaviors of a coffee machine (left) and a user (right)
Example: compositional modeling of a coffee machine

This is the LTS that describes the behaviour of the system consisting of a machine and a user:
Example: compositional modeling of a coffee machine

How to obtain the LTS of the whole system from the LTSs of the components (machine and user)?

First: consider all possible states by combining states of the components
Example: compositional modeling of a coffee machine

How to obtain the LTS of the whole system from the LTSs of the components (machine and user)?

Second: ”copy” transitions of the two components
Example: compositional modeling of a coffee machine

How to obtain the LTS of the whole system from the LTSs of the components (machine and user)?

Third: add \( \tau \) transitions to states in which different components can perform co-actions. The result is the LTS of the whole system.
Example: compositional modeling of a coffee machine

How to obtain the LTS of the whole system from the LTSs of the components (machine and user)?

Fourth: if all systems components have been considered, remove the non-$\tau$ transitions (and $\tau$ transitions unreachable from the initial state)
Example: compositional modeling of a coffee machine

Let’s try with a "wrong" user (only one coin)

We obtain:

The system reaches a **deadlock** with the machine waiting for a second coin
Example: compositional modeling of a coffee machine

Let’s try with two users competing for the machine

![Diagram of coffee machine model]
Example: compositional modeling of a coffee machine

First, we "merge" the machine with the first user:
Example: compositional modeling of a coffee machine

Then we "merge" also the second user and remove non-\( \tau \) transitions:

Depending on the order of interactions, either both users are served or the system reaches a deadlock state (non deterministic behavior)
Lesson Learnt

Transition Systems are a way to model the behavior/dynamics of a system in an exhaustive way

- all the possible behaviors are taken into account
- problem state explosion problem

Non-determinism is the most abstract description of alternative behaviors

- it is simple, but misses important quantitative informations on the system behavior such as probabilities and rates
  - We will see probabilistic/stochastic transition systems

Reachability of states is the often the property of interest

- but sometimes we may be interested in properties which deal with a sequence of states, such as behavioral patterns (steady states, oscillations, ...)
  - We will see model checking