# Formal Methods / Nachum Dershowitz
# Lecture 11
# Automata and their Interaction

Notes by : David Porat

6/6/2000

## 1 Introduction

In this lecture we present the concepts of **interacting automata** and **continuous-time automaton**.

**Interacting Automata :**
We examine a system of interacting automata, and describe the general properties of the system as composed of the properties of its individual components.
We distinguish between the synchronous and asynchronous models.

**Continuous Time Automaton :**
Our principal objective is to adapt basic concepts of automata theory from discrete to continuous(real) time.
An obvious transition from discrete to continuous time is as follows : instead of signals defined over a discrete sequence of time instants, consider signals defined over the non-negative reals.

We recall the notion of a **discrete-time deterministic automaton**.

**Definition 1** *A* ***discrete-time deterministic automaton*** *$M$ is defined by:*
*$X$ - a set of states*
*$U$ - an action alphabet*
*and a map* ***nextstate*** $: X \times U \longrightarrow X$

*The **terminal transition map** $\Psi : X \times \tilde{U} \longrightarrow X$ extends the map **nextstate** from singleton action $u \epsilon U$ to action sequence $\tilde{u} = u_1 \cdots u_l$.*

*The associated **full transition map** $\tilde{\Psi} : X \times \tilde{U} \longrightarrow \tilde{X}$ , when applied to a state $x \epsilon X$ and an action sequence $\tilde{u}$, returns the state-sequence of length $l+1$, that starts with $x$ and leads to the terminal state $x' = \Psi(x, \tilde{u})$*

## 2 Interacting Automata

We define an automata $M$ as the interaction of automata $M_i$ . We show how to define the **nextstate**, **terminal transition** and **full transition** maps of $M$, as a composition of its components $M_i$ .

### 2.1 Synchrony

In the synchronous model, execution is partitioned into rounds, and in each round, every automaton $M_i$ receives an action as input and computes its $nextstate_i$.

In this model, the output of the composed automata, depends on the output of *every* individual automata.

Consider $M_i$ with state-space $X_i \times X_0$ and action-space $U_i \times U_0$, where $X_0$ is a shared register and $U_0$ a shared port.

**Definition 2** *$M$ is the **Synchronous composition** of automata $M_1$ and $M_2$. We write $M = M_1 \times M_2$ . The state-space of $M$ is $X_1 \times X_0 \times X_2$ , the action-space is $U_1 \times U_0 \times U_2$.*

*The transitions are defined as follows :*

$$M(x_1x_0x_2, u_1u_0u_2, x_1'x_0'x_2') \overset{\text{def}}{=} M_1(x_1x_0, u_1u_0, x_1'x_0') \;\&\; M_2(x_2x_0, u_2u_0, x_2'x_0')$$

*We define the **nextstate** map as follows :*

$nextstate(x_1x_0x_2, u_1u_0u_2) = x_1'x_0'x_2'$ *iff*

$nextstate_1(x_1x_0, u_1u_0) = x_1'x_0'$ *&* $nextstate_2(x_2x_0, u_2u_0) = x_2'x_0'$

We cannot extend the **nextstate map** to a **terminal transition map** of the form

$\Psi(x_1x_0x_2, \tilde{u}_1\tilde{u}_0\tilde{u}_2) = x_1'x_0'x_2'$ iff

$\Psi_1(x_1x_0, \tilde{u}_1\tilde{u}_0) = x_1'x_0'$ & $\Psi_2(x_2x_0, \tilde{u}_2\tilde{u}_0) = x_2'x_0'$

The reason is that for given $\tilde{u}_1\tilde{u}_0\tilde{u}_2$ the $M_i$ are required only to reach the same terminal value $x_0'$, which would guarantee that $\Psi(x_1x_0x_2, \tilde{u}_1\tilde{u}_0\tilde{u}_2)$ is defined. However, we require also definedness for all prefixes of $\tilde{u}_1\tilde{u}_0\tilde{u}_2$ .

We can solve this problem by using full transitions instead of terminal ones.

**Definition 3** *Full transition map*
$\tilde{\Psi}(x_1x_0x_2, \tilde{u}_1\tilde{u}_0\tilde{u}_2) = \tilde{x_1}\tilde{x_0}\tilde{x_2}$ *iff*
$\tilde{\Psi}_1(x_1x_0, \tilde{u}_1\tilde{u}_0) = \tilde{x}_1\tilde{x}_0$ *&* $\tilde{\Psi}_2(x_2x_0, \tilde{u}_2\tilde{u}_0) = \tilde{x}_2\tilde{x}_0$

We define terminal transitions in the case when the components $M_i$ don't share registers.

**Definition 4** *Terminal transition map*
$\Psi(x_1x_2, \tilde{u}_1\tilde{u}_0\tilde{u}_2) = x'_1x'_2$ *iff*
$\Psi_1(x_1, \tilde{u}_1\tilde{u}_0) = x'_1$ *&* $\Psi_2(x_2, \tilde{u}_2\tilde{u}_0) = x'_2$

## 2.2 Asynchrony

In the asynchronous model,there is no fixed upper bound on the time it takes for an automaton to compute its *nextstate*.

Unlike the synchronous model, we cannot rely on the output of every individual automata on every round. in this case we require that *at least* one individual automata computes its output, for the composed automata to do so.

Consider $M_i$ with state-space $X_i \times X_0$ and action-space $U_i \oplus U_0$.

**Definition 5** *M is the* ***Asynchronous composition*** *of automata* $M_1$ *and* $M_2$*.We write* $M = M_1 \| M_2$ *. The state-space of* $M$ *is* $X_1 \times X_0 \times X_2$ *, the action-space is* $U_1 \bigoplus U_0 \bigoplus U_2$*.*

*We define the* ***nextstate*** *map as follows :*
$nextstate(x_1x_0x_2, u_i) = x'_1x'_0x'_2$
*holds in one of the following cases :*
$nextstate_1(x_1x_0, u_i) = x'_1x'_0$ *&* $x'_2 = x_2$ *if* $i = 1$
$nextstate_2(x_2x_0, u_i) = x'_2x'_0$ *&* $x'_1 = x_1$ *if* $i = 2$
$nextstate_1(x_1x_0, u_i) = x'_1x'_0$ *&* $nextstate_2(x_2x_0, u_i) = x'_2x'_0$ *if* $i = 0$

As in the synchronous case, we cannot easily extend the **nextstate** map to **terminal transition** and **full transition** maps. We deal with this problem by assuming either that there are no shared ports or no shared registers:

**Definition 6** *(No shared ports).*

*Assume $M_i$ with state-space $X_i \times X_0$ and action-space $U_i$. Then $M$ has state-space $X_1 \times X_0 \times X_2$, action-space $U_1 \oplus U_2$.*

$\Psi(x_1 x_0 x_2, \tilde{u}_i) = x'_1 x'_0 x'_2$

*holds in one of the following cases :*

$\Psi_1(x_1 x_0, \tilde{u}_i) = x'_1 x'_0$ *&* $x'_2 = x_2$ *if* $i = 1$

$\Psi_2(x_2 x_0, \tilde{u}_i) = x'_2 x'_0$ *&* $x'_1 = x_1$ *if* $i = 2$

**Definition 7** *(No shared registers).*

*Assume $M_i$ with state-space $X_i$ and action-space $U_i \oplus U_0$.*

$\Psi(x_1 x_2, \tilde{u}_i) = x'_1 x'_2$

*holds in one of the following cases :*

$\Psi_1(x_1, \tilde{u}_i) = x'_1$ *&* $x'_2 = x_2$ *if* $i = 1$

$\Psi_2(x_2, \tilde{u}_i) = x'_2$ *&* $x'_1 = x_1$ *if* $i = 2$

$\Psi_1(x_1, \tilde{u}_i) = x'_1$ *&* $\Psi_2(x_2, \tilde{u}_i) = x'_2$ *if* $i = 0$

## 3 Nets & Webs

In both the synchronous and asynchronous models we faced difficulties in trying to define transition maps for components with shared ports and registers.

We look at communication mechanisms where components are not allowed to share both ports and registers.

**Nets :** Communication, if any, is via shared ports (shared memory); all registers of a component are private.

**Nets :** Communication, if any, is via shared registers (message passing); all ports of a component are private.

The different architectures are :

| | **synchronous** | **asynchronous** |
|---|---|---|
| **private registers** | synchronous net | asynchronous net |
| **private ports** | synchronous web | asynchronous web |

## 4 Petri Nets

We shall first give the basic definition of a **Petri net** and then adapt it to our specific needs.

**Definition 8** *A* ***Petri net*** *is a finite directed graph with two types of nodes, referred to as* ***places*** *and* ***transitions****.*

*Every arrow in a Petri net goes either from a place to a transition or from a transition to a place.*

*Consider a transition* $\boldsymbol{a}$*. Every place* $\boldsymbol{p}$ *such that there is an arrow from* $\boldsymbol{p}$ *to* $\boldsymbol{a}$ *is referred to as an* ***input****. Every place* $\boldsymbol{q}$ *such that there is an arrow from* $\boldsymbol{a}$ *to* $\boldsymbol{q}$ *is referred to as an* ***output****. The same place can be both an input and an output place of* $\boldsymbol{a}$*.*

*A* ***marking*** *of a Petri net is a mapping* $\boldsymbol{m}$ *of the set of places into the set of non-negative integers. We say that there are* $\boldsymbol{k}$ ***tokens*** *in the place* $\boldsymbol{p}$ *if* $\boldsymbol{m(p) = k}$*.*

*We now define the* ***operation*** *of a Petri net. A transition is* ***enabled*** *(at a marking) iff all of its input places have at least one* ***token****. An enabled transition may* ***fire*** *by removing one token from each of its input places and adding one token to each of its output places.*

Our goal is to analyze a Petri net as a system of interacting components. Applying the definition of a **Petri net** to **Nets & Webs**, we design the Petri net **G** to be composed of ports (as circles) and registers (as boxes), instead of places and transitions respectively. As defined before, every arrow in the net goes either from a port to a register, or from a register to a port.

We say that **G** is an **atomic net** if it consists of a single circle with its neighboring boxes. **G** is an **atomic web** if it consists of a single box with its neighboring circles.

Hence if our net G is composed of **k** circles and **m** boxes, we can decompose the net into **k atomic subnets** (each one with a single circle) or into **m atomic subwebs** (each one with a single box).

## 5 Deterministic Continuous-Time Automata

We examine basic concepts of classical automata theory and adapt them from discrete to continuous time.

**Time**

*Discrete Time* : Every natural number represents a time moment. The number zero represents the beginning of time.

We replace discrete time by:

*Continuous Time* : Time is continuous, every time is represented by a non-negative real. The number zero represents the beginning of time.

We wish to expand the discrete-time **terminal transition map** $\Psi : X \times \tilde{U} \longrightarrow X$,to a continuous-time map with $\tilde{u}$ defined as a finite path of $U$ within certain time limits $t$ and $t'$ .

If state $x$ occurs at time $t$, then $\tilde{u}$ with life-time $[t, t+\delta]$ produces state $x'$ at time $t+\delta$. A finite path $\tilde{u}$ is admissible for $x$ iff $\Psi(x, \tilde{u})$ is defined.

An infinite path $\tilde{u}$ is admissible for $x$ if all its finite prefixes are admissible for $x$.

**Axioms 1**

$\Psi(x, \epsilon) = x$

$\Psi(x, \tilde{u}\tilde{v}) = \Psi(\Psi(x, \tilde{u}), \tilde{v})$

Extending the **terminal transition map** to the **full transition map** is straightforward in this case.

**Definition 9** *Full transition map of M.*

$\tilde{\Psi}(x, \tilde{u}) = \tilde{x}$ *iff* $\forall t \epsilon [0, \delta). \Psi(x, \tilde{u}|t) = \tilde{x}(t)$

We want to maintain the following behaviour on the transition from discrete to continuous time.

**Input-Output Behaviour**

*Deterministic Behaviour* : The output signals are completely determined by the input signals.

*Retrospective Behaviour* : The output at a moment $t$ does not depend on the inputs at later time.

F is retrospective if for any $x, y$ and $t$, the following condition holds :

If $x$ and $y$ coincide in the interval $[0, t]$ then $Fx$ and $Fy$ coincide in the interval $[0, t]$.

**or**

*Strong Retrospective Behaviour* : The output at a moment $t$ does not depend on the inputs at moment $t$ and at later time.

F is strongly retrospective if for any $x, y$ and $t$, the following condition holds :

If $x$ and $y$ coincide in the interval $[0, t)$ then $Fx$ and $Fy$ coincide in the interval $[0, t]$.

The **full transition map** $\tilde{\Psi}$ is deterministic and strongly retrospective.