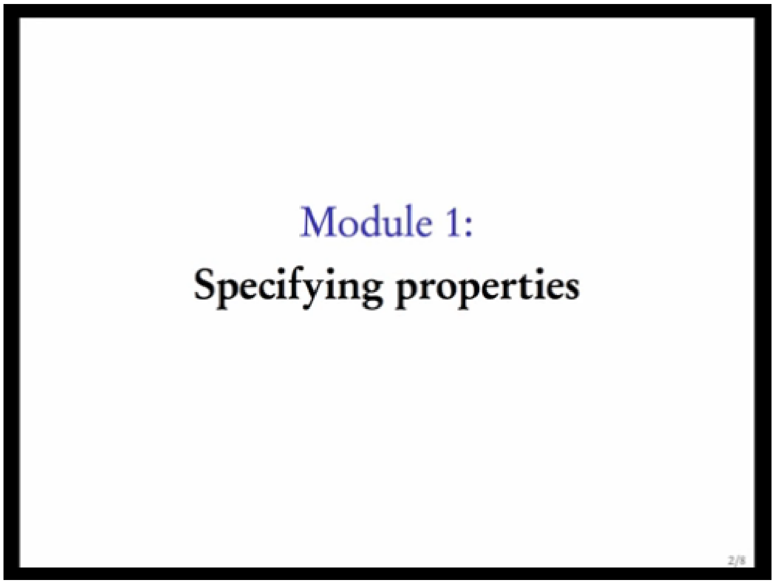
Model Checking

Prof. B. Srivathsan
Department of Computer Science and Engineering
Indian Institute of Technology – Madras

Lecture - 23
Specifying Properties

Welcome to unit 5 of this course. In the last unit we saw what are called regular properties. In this unit we will be looking at what are called omega regular properties. This symbol stands for the Greek letter omega.

(Refer Slide Time: 00:22)



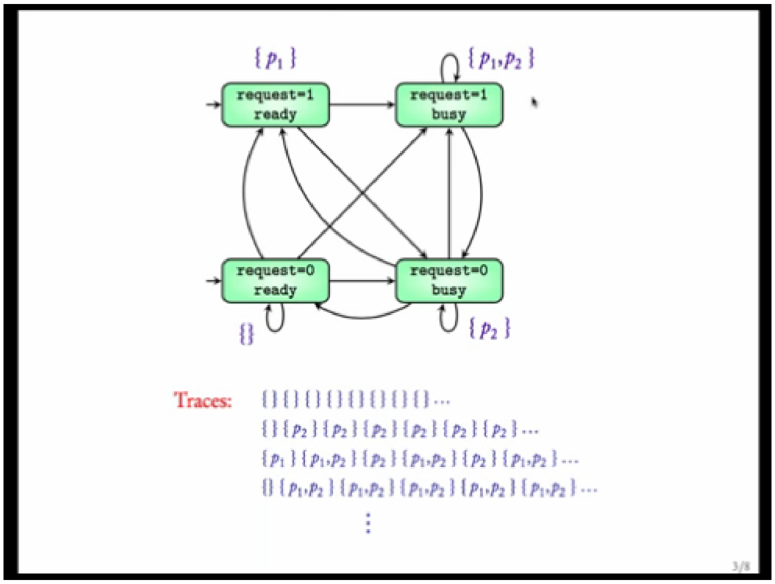
Module 1:

Specifying properties

2/8

In this module I will do a short recap of the tools to describe properties. I will conclude with an overview of what we would be seeing in this unit.

(Refer Slide Time: 00:38)



Traces: $\{\}\{\}\{\}\{\}\{\}\{\}\{\}\{\}\cdots$

$\{\}\{p_2\}\{p_2\}\{p_2\}\{p_2\}\{p_2\}\{p_2\}\cdots$

$\{p_1\}\{p_1, p_2\}\{p_2\}\{p_1, p_2\}\{p_2\}\{p_1, p_2\}\cdots$

$\{\}\{p_1, p_2\}\{p_1, p_2\}\{p_1, p_2\}\{p_1, p_2\}\{p_1, p_2\}\cdots$

$\vdots$

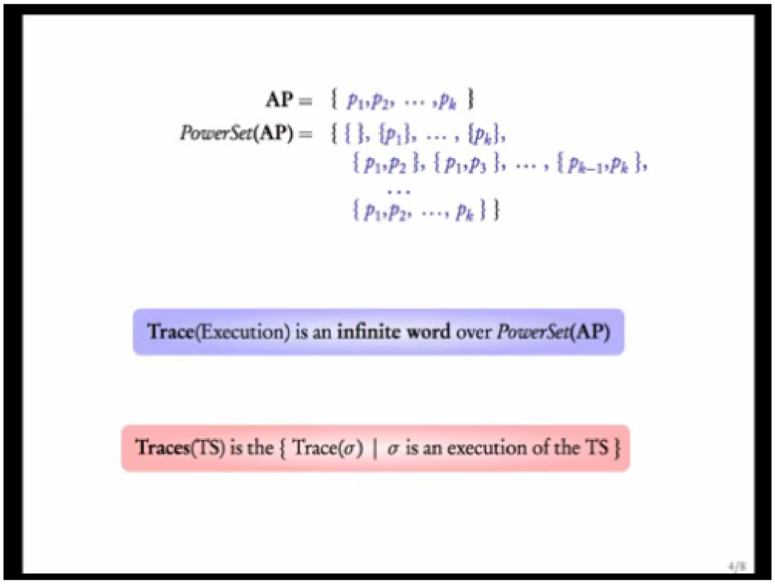
We have been looking at code as transition systems so this is the transition system. Additionally in order to be talking about properties of this transition system we define what are called atomic propositions. Here there are 2 atomic propositions p1 and p2 once we have them we labeled a state with the sets of atomic propositions that are true in that state.

For example here p1 is true, here p1 and p2 are true, here p2 is only true and here none of them is true. Once we have this model with us we can talk about what are called traces. Lets us look at executions of this transition system for example looping this is an initial state looping in this state forever is 1 possible execution of the transition system.

Take that execution and look at the label of each state in that execution that will give us what we called a trace. So this trace corresponds to the execution which just stays in this state. What about this? This corresponds to the execution that starts from this initial state goes to this state and keeps looping here forever.

What about this one? It starts here, goes here then takes this transition to come here and then goes back here and keeps doing this loop and this is trace corresponding to this execution and so on. So given a transition system like this along with the labels we can talk about traces.

(Refer Slide Time: 02:40)



$$\mathbf{AP} = \{\ p_1, p_2,\ \cdots, p_k\ \}$$

$$\begin{aligned} \mathit{PowerSet}(\mathbf{AP}) = \{\ &\{\ \},\ \{p_1\},\ \cdots,\ \{p_k\}, \\ &\{\ p_1, p_2\ \},\ \{\ p_1, p_3\ \},\ \cdots,\ \{\ p_{k-1}, p_k\ \}, \\ &\cdots \\ &\{\ p_1, p_2,\ \cdots, p_k\ \}\ \} \end{aligned}$$

Trace(Execution) is an **infinite word** over *PowerSet*(**AP**)

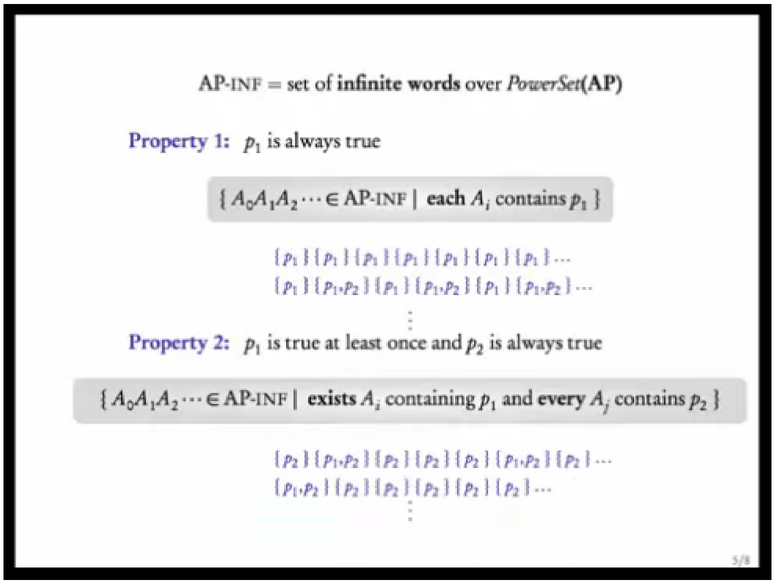
Traces(TS) is the { Trace(σ) | σ is an execution of the TS }

4/8

Here is more formal explanation let AP be the set of atomic proposition given this set AP we looked at the PowerSet of AP. So the PowerSet is the standard think that you know, it looks at all subsets of AP starting with the empty, singletons and then p1, p2, p3 and so on till the set consisting of all of them this is the standard PowerSet. The trace of an execution is an infinite word here I have emphasized the infinite word over PowerSet AP.

So each trace is an infinite word over the alphabet PowerSet AP. So, if you consider this to be a finite alphabet each trace is a word over that alphabet and given a transition system we can talk about traces of a transition system. We look at each execution and then take the corresponding trace and the set of such cases is denoted as traces of TS. This is something which we have seen already the point that we need to note is that this is our alphabet and these are all words over the alphabet.

(Refer Slide Time: 04:09)



AP-INF = set of **infinite words** over *PowerSet*(AP)

Property 1: p_1 is always true

$\{ A_0 A_1 A_2 \cdots \in$ AP-INF | **each** A_i contains p_1 }

$\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\cdots$
$\{p_1\}\{p_1,p_2\}\{p_1\}\{p_1,p_2\}\{p_1\}\{p_1,p_2\}\cdots$
$\vdots$

Property 2: p_1 is true at least once and p_2 is always true

$\{ A_0 A_1 A_2 \cdots \in$ AP-INF | **exists** A_i containing p_1 and **every** A_j contains p_2 }

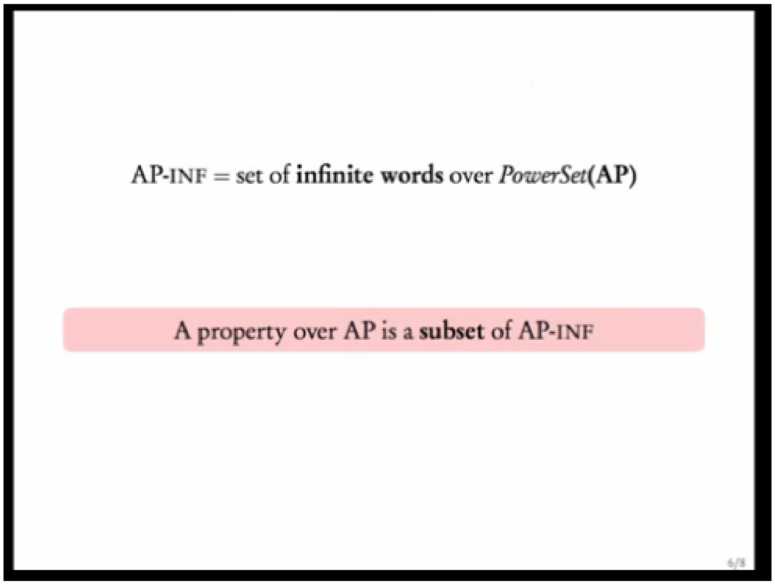
$\{p_2\}\{p_1,p_2\}\{p_2\}\{p_2\}\{p_2\}\{p_1,p_2\}\{p_2\}\cdots$
$\{p_1,p_2\}\{p_2\}\{p_2\}\{p_2\}\{p_2\}\{p_2\}\cdots$
$\vdots$

5/8

Now, let us come to property so far we have been looking only at the model. Considering this to be a alphabet we denote AP into be the set of possible infinite words. Now, what is a property? For example the property which says p1 should always be true will give us the set of all infinite words where in each letter p1 should be true.

There could be other things true as well but p1 should necessarily be true in each letter. Let us look at another example p1 is true at least once and p2 is always true. So this will give us another set of words where p2 is always true and then p1 should be true somewhere.

(Refer Slide Time: 04:59)



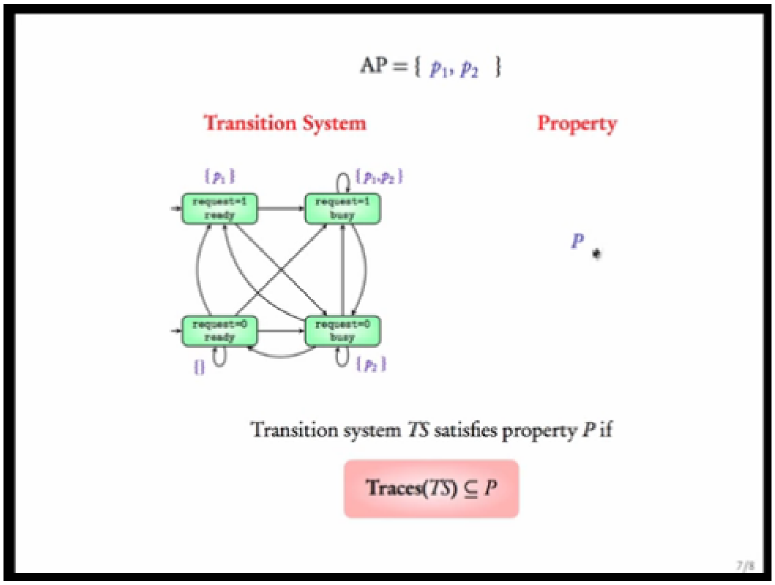
AP-INF = set of **infinite words** over *PowerSet*(AP)

A property over AP is a **subset** of AP-INF

6/8

The point to note is that a property over a set of atomic propositions is a subset of AP-INF. What is the AP-INF? It is the set of all words over this alphabet and a property is just a set of such words.

(Refer Slide Time: 05:19)



$\text{AP} = \{ p_1, p_2 \}$

Transition System

Property

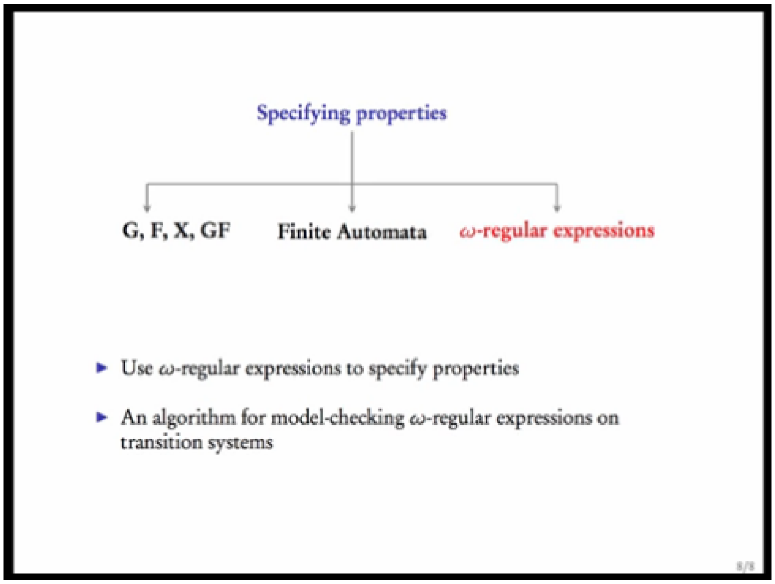
Transition system *TS* satisfies property *P* if

$\text{Traces}(TS) \subseteq P$

7/8

Here is where we are given atomic propositions we have this transition system and then somehow we described a property. And we say that the transition system satisfies the property if all its traces are contained in this set. Now, the question is how do we describe the set?

(Refer Slide Time: 05:47)



Specifying properties

G, F, X, GF

Finite Automata

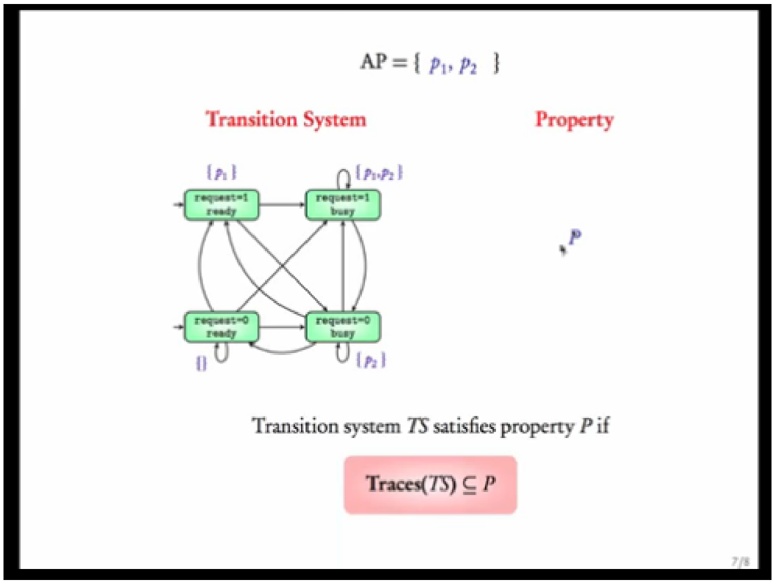
ω-regular expressions

- Use ω-regular expressions to specify properties
- An algorithm for model-checking ω-regular expressions on transition systems

8/8

We have seen ways of describing the set using the operators G, F, X, GF and in the last unit we have been seen what are called finite automata and finite automata can help us specify safety properties. And how will do that? They describe the set of bad prefixes and the property is just the words which avoid the bad prefixes. In this unit we will be looking at something more general called omega regular expressions.

(Refer Slide Time: 06:28)



AP = { p_1, p_2 }

Transition System

Property

P

Transition system TS satisfies property P if

Traces(TS) $\subseteq P$

We will be using only omega regular expressions to specify properties like this. There are 2 goals the first thing we would use omega regular expressions to specify properties. So in the next module I will define what omega regular expressions are and I will give a lot of examples to make it clear.

Once we do that what we want is an algorithm to check this. Suppose I give an omega regular expression and this transition system how does one check or what is an algorithm that checks if the set of all traces is contained inside this omega regular expression. Note that the set of all traces is infinite and the set of all words belonging to this property is also infinite could potentially be infinite.

However, the descriptions are finite using these finite descriptions we want to give an algorithm that checks if all traces of this transition system satisfied the property given by this omega regular expression. In this unit we would not be able to complete the second point. However, we would give some necessary mathematical objects that will help us describe an algorithm in the next unit. This is the broad over view of what we have seen and what we will be seen in this unit you can now jump to the next module.