FORMAL METHODS

LECTURE II: MODELING SYSTEMS

Alessandro Artale

Faculty of Computer Science – Free University of Bolzano
Room 2.03

artale@inf.unibz.it  http://www.inf.unibz.it/~artale/

Some material (text, figures) displayed in these slides is courtesy of: M. Benerecetti, A. Cimatti, M. Fisher, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani.
Summary of Lecture II

- Types of Systems.
- Modeling Systems as Kripke Models.
- Languages for Describing Kripke Models.
- Properties of Systems.
We describe here Concurrent Reactive systems.

- **Reactive Systems**: Systems that interact with their environment and usually do not terminate (e.g. communication protocols, hardware circuits).

- **Concurrent Systems** consist of a set of components that execute together.

We distinguish two types of Concurrent Systems:

1. **Asynchronous or Interleaved Systems**. Only one component makes a step at a time;
2. **Synchronous Systems**. All components make a step at the same time.
We need to construct a *Formal Specification* of the system which abstract from irrelevant details.

- **State**: Snapshot of the system that captures the values of the variables at a particular point in time.
- **System Transition**: How the state of the system evolves as the result of some action.
- **Computation**: Infinite sequence of states along the different transitions.
Summary

Types of Systems.
Modeling Systems as Kripke Models.
Languages for Describing Kripke Models.
Properties of Systems.
Kripke Structures are transition diagrams that represent the dynamic behavior of a reactive system.

Kripke Structures consist of a set of states, a set of transitions between states, and a set of properties labeling each state.

A path in a Kripke structure represents a computation of the system.
Formally, a Kripke model \( \langle S, I, R, AP, L \rangle \) consists of:

- a set of states \( S \);
- a set of initial states \( I \subseteq S \);
- a set of transitions \( R \subseteq S \times S \);
- a set of atomic propositions \( AP \);
- a labeling function \( L : S \rightarrow 2^{AP} \).

A path in a Kripke model \( M \) from a state \( s_0 \) is an infinite sequence of states

\[
\pi = s_0, s_1, s_2, \ldots
\]

such that \((s_i, s_{i+1}) \in R\), for all \( i \geq 0 \).
Example: Kripke model for mutual exclusion

- We model two **concurrent asynchronous processes** sharing a resource ensuring they do not access it at the same time.

- Each process has **critical sections** in its code and only one process can be in its critical section at a time.

- We want to find a **protocol** for mutual exclusion which, for example, guarantee the following properties:
  - **Safety**: Only one process is in its critical section at a time.
  - **Liveness**: Whenever any process requests to enter its critical section it will *eventually* be permitted to do so.
  - **Non-Blocking**: A process can always request to enter its critical section.
Each process can be in its non-critical state (N), or trying to enter its critical state (T), or in its critical state (C). The variable `turn` considers the first process that went into its trying state.
Composing Kripke Models

Complex Kripke Models are typically obtained by composition of smaller ones.

Components can be combined via:

- **synchronous** composition
- **asynchronous** composition.
Synchronous Composition

- Components evolve in parallel.
- At each time instant, every component performs a transition.

Typical example: sequential hardware circuits.
Asynchronous Composition

- Interleaving of evolution of components.
- At each time instant, one component is selected to perform a transition.

Typical example: communication protocols.
Summary

- Types of Systems.
- Modeling Systems as Kripke Models.
- Languages for Describing Kripke Models.
- Properties of Systems.
Tipically a Kripke model is not given explicitly, rather it is usually presented in a structured language (e.g., NuSMV, SDL, PROMELA, StateCharts, VHDL, ...)
Each component is presented by specifying:

- A set of system variables
- Initial values for state variables
- Instructions
The correspondence between a description language and the Kripke Model is the following:

1. **States**: all possible assignments for system variables;
2. **Initial States**: Initial values for system variables;
3. **Transitions**: Instructions;
4. **Atomic Propositions**: Propositions associated to the values of the system variables;
5. **Labeling**: Set of atomic propositions true at a state.
The NuSMV (New Symbolic Model Verifier) model-checking system is an Open Source product (nusmv.irst.itc.it).

NuSMV programs consist of:

- Type declarations of the system variables;
- Assignments that define the valid initial states (e.g., \texttt{init(b0) := 0}).
- Assignments that define the transition relation (e.g., \texttt{next(b0) := !b0}).
MODULE main
VAR
  b0 : boolean;
  b1 : boolean;
  reset : boolean;
  out : 0..3;

ASSIGN
  init(b0) := 0;
  next(b0) := case
    reset = 1: 0;
    reset = 0: !b0;
  esac;

  init(b1) := 0;
  next(b1) := case
    reset: 0;
    1 : ((!b0 & b1) | (b0 & !b1));
  esac;

  out := b0 + 2*b1;
Summary

- Types of Systems.
- Modeling Systems as Kripke Models.
- Languages for Describing Kripke Models.
- Properties of Systems.
Nothing Bad Ever Happens.

- Deadlock: two processes waiting for input from each other, the system is unable to perform a transition.
- No reachable state satisfies a “bad” condition, e.g. never two processes in critical section at the same time

It is expressed by a temporal formula saying that “it’s never the case that $p$”.

\[ \neg \exists x \ (p(x)) \\Rightarrow \neg \exists y \ (p(y)) \]
Liveness Properties

Something Desirable Will Eventually Happen.

- Whenever a subroutine takes control, it will always return it (sooner or later).

It is expressed by a temporal formula saying that “at each state it will be the case that $p$”.

Can be refuted by infinite behaviour (represented as a loop)
Summary of Lecture II

- Types of Systems.
- Modeling Systems as Kripke Models.
- Languages for Describing Kripke Models.
- Properties of Systems.