# *State-changing systems*

| Informally | Formally |
| --- | --- |
| At each time moment, the system is in a particular state. | This state can be characterized by values of some variables, called the state variables. |
| The system state is changing in time. There are actions (controlled or not) that change the state. | Actions change values of some state variables. |

## *Examples*

- Reactive systems. These are the systems that interact with their environment.
- Concurrent systems. These are the systems which consist of a set of components functioning together. Usually, functioning of these components can be described independently, but they communicate through shared variables or some kind of communication channels, for example queues.

# *Reasoning about state-changing systems*

- Build a formal model of this state-changing system which describes, in particular, functioning of the system, or some abstraction thereof.
- Using a logic to specify and verify properties of the system.

# *Vending machine example*

Consider an example state-changing system: a vending machine which dispenses drinks in a university department. The machine has several components, including at least the following: a storage space for storing and preparing drinks, a box for dispensing drinks and a coin slot for putting coins. When the machine is operating, it goes through several states depending on the behaviour of the current customer.

# *Vending machine example*

Each action undertaken by the customer or by the machine itself may change the state of the machine. For example, when the customer inserts a coin in the coin slot, the amount of money collected in the slot changes.

Actions which may change the state of the system are called transitions.

# *Modelling state-changing systems*

To build a formal model of a particular state-changing system, we should define

- What are the state variables.
- What are the possible values of the state variables.
- What are the transitions and how they change the values of the state variables.

A state can be identified with a function from variables to values, e.g. with an interpretation in a PLFD.

## *Transition systems*

A transition system is a tuple $\mathbb{S} = (\mathcal{X}, D, dom, In, T)$, where

- $\mathcal{X}$ is a finite set of state variables.
- $D$ is a non-empty set, called the domain. Elements of $D$ are called values.
- $dom$ is a mapping from $\mathcal{X}$ to the set of non-empty subsets of $D$. For each state variable $v \in \mathcal{X}$, the set $dom(v)$ is called the domain for $v$.
- $In$ is a set of states, called initial states.
- $T$ is a set of transitions.

# *State and Transition*

A state of a transition system $\mathbb{S}$ is a function $s : \mathcal{X} \to D$ such that for every $x \in \mathcal{X}$ we have $s(x) \in dom(x)$.
So states correspond to interpretations.

A transition is a set of pairs of states.

## *Transitions*

- A transition $t$ is applicable to a state $s$ if there exists a state $s'$ such that $(s, s') \in t$.
- A transition $t$ is deterministic if for every state $s$ there exists at most one state $s'$ such that $(s, s') \in t$.
- The transition relation of $S$, denoted by $Tr_S$, is the set of pairs of states $\bigcup_{t \in T} t$, i.e., it is the union of all transitions in the system.
- A transition system $S$ is said to be finite-state if its domain $D$ is finite, and infinite-state otherwise. We will consider only finite-state transition systems!

## *Vending Machine*

- The vending machine contains a drink storage, a coin slot, and a drink dispenser. The drink storage stores drinks of two kinds: beer and coffee.
- The coin slot can accommodate up to three coins.
- The drink dispenser can store at most one drink. If it contains a drink, this drink should be removed before the next one can be dispensed.
- A can of beer costs two coins. A cup of coffee costs one coin.
- There are two kinds of customers: students and professors. Students drink only beer, professors drink only coffee.
- From time to time the drink storage can be recharged.

# *Formalization: Variables and Domains*

| variable | domain | explanation |
|---|---|---|
| st_coffee | $\{0, 1\}$ | drink storage contains coffee |
| st_beer | $\{0, 1\}$ | drink storage contains beer |
| disp | $\{\mathit{none}, \mathit{beer}, \mathit{coffee}\}$ | content of drink dispenser |
| coins | $\{\mathit{0}, \mathit{1}, \mathit{2}, \mathit{3}\}$ | number of coins in the slot |
| customer | $\{\mathit{none}, \mathit{student}, \mathit{prof}\}$ | customer |

# *Transitions*

- *Recharge* which results in the drink storage having both beer and coffee.
- *Customer_arrives*, after which a customer appears at the machine.
- *Customer_leaves*, after which the customer leaves.
- *Coin_insert*, when the customer inserts a coin in the machine.
- *Dispense_beer*, when the customer presses the button to get a can of beer.
- *Dispense_coffee*, when the customer presses the button to get a cup of coffee.
- *Take_drink*, when the customer removes a drink from the dispenser.

## *Symbolic Representation of Sets of States*

State explosion problem: The number of states is the number of all possible interpretations, (in a given PLFD)!

So the number of states is exponential in the number of variables!

(Systems with more that 300 variables could not be explicitly stored in any computer).

# *Symbolic Representation of Sets of States*

State explosion problem: The number of states is the number of all possible interpretations, (in a given PLFD)!

So the number of states is exponential in the number of variables!

(Systems with more that 300 variables could not be explicitly stored in any computer).

Solution: Symbolic representation of transition systems.

Let $\mathbb{S} = (\mathcal{X}, D, dom, I, T)$ be a finite-state transition system.

- Then $dom$ defines an instance of PLFD $\mathcal{L}(\mathbb{S})$.
- Every state of $\mathbb{S}$ is an interpretation for $\mathcal{L}(\mathbb{S})$ and vice versa.
- Therefore, every formula $F$ of $\mathcal{L}(\mathbb{S})$ defines a set of states:

$$\{s \mid s \models F\}.$$

## *Example*

Let us represent the set of states in which the machine is ready to dispense a drink. In every such state, a drink should be available, the drink dispenser empty, and the coin slot contains enough coins. This can be expressed by:

$$(\mathsf{st\_coffee} \vee \mathsf{st\_beer}) \wedge \mathsf{disp} = \mathit{none} \wedge$$
$$((\mathsf{coins} = \mathit{1} \wedge \mathsf{st\_coffee}) \vee \mathsf{coins} = \mathit{2} \vee \mathsf{coins} = \mathit{3}).$$

# *Symbolic Representation of Transitions*

- In addition to the set of propositional variables $\mathcal{X} = \{x_1, \ldots, x_n\}$, introduce a set of next state variables $\mathcal{X}' = \{x'_1, \ldots, x'_n\}$.
- Pairs of states as interpretations. For every variable $x \in \mathcal{X}$ define

$$
\begin{array}{rcl}
(s, s')(x) & \stackrel{\text{def}}{=} & s(x); \\
(s, s')(x') & \stackrel{\text{def}}{=} & s'(x).
\end{array}
$$

- Symbolic representation. Formula $F$ of variables $\mathcal{X} \cup \mathcal{X}'$ represents a transition $t$ if $t = \{(s, s') \mid (s, s') \models F\}$.

# *Example*

The transition *Recharge*:

$$\mathsf{customer} = none \wedge \mathsf{st\_coffee}' \wedge \mathsf{st\_beer}'.$$

# *Example*

The transition *Recharge*:

$$\mathsf{customer} = none \wedge \mathsf{st\_coffee}' \wedge \mathsf{st\_beer}'.$$

But this formula includes describes a very strange transition after which, for example

- coins may appear in and disappear from the slot;
- drinks may appear in and disappear from the dispenser.
- ...

# *Frame problem*

One has to express explicitly, maybe for a large number of state variables, that the values of these variables do not change after a transition. For example,

$$
\begin{aligned}
&(\mathsf{coins} = \mathit{0} \leftrightarrow \mathsf{coins}' = \mathit{0}) \wedge \\
&(\mathsf{coins} = \mathit{1} \leftrightarrow \mathsf{coins}' = \mathit{1}) \wedge \\
&(\mathsf{coins} = \mathit{2} \leftrightarrow \mathsf{coins}' = \mathit{2}) \wedge \\
&(\mathsf{coins} = \mathit{3} \leftrightarrow \mathsf{coins}' = \mathit{3}).
\end{aligned}
$$

This frame problem arises in artificial intelligence, knowledge representation, and reasoning about actions.

# *Notation for the frame formula*

Abbreviations (we assume $dom(x) = dom(y)$):

$$
\begin{array}{lcl}
x \neq v & \stackrel{\text{def}}{=} & \neg(x = v) \\
x = y & \stackrel{\text{def}}{=} & \bigwedge_{v \in dom(x)} (x = v \leftrightarrow y = v).
\end{array}
$$

Let $\mathbb{S}$ be a transition system and $\{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ be a set of state variables of $\mathcal{L}(\mathbb{S})$. Define

$$
only(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \bigwedge_{y \in \mathcal{X} \setminus \{x_1, \ldots, x_n\}} y = y'.
$$

This formula expresses that $x_1, \ldots, x_n$ are the only variables whose values can be changed by the transition.

# *Preconditions and postconditions*

When we represent a transition symbolically using a formula $F$ of variables $\mathcal{X} \cup \mathcal{X}'$, the formula $F$ is usually represented as the conjunction $F_1 \wedge F_2$ of two formulas:

- $F_1$ expresses some conditions on the variables $\mathcal{X}$ which are necessary to execute the transition (precondition);
- $F_2$ expresses some conditions relating variables in $\mathcal{X}$ to those in $\mathcal{X}'$, i.e., conditions which show how the values of the variables after the transition relate to their values before the transition (postcondition).

# *Transitions*

$$
\begin{array}{rcl}
\mathit{Recharge} & \stackrel{\text{def}}{=} & \mathsf{customer} = \mathit{none} \wedge \mathsf{st\_coffee}' \wedge \mathsf{st\_beer}' \wedge \\
 & & \mathit{only}(\mathsf{st\_coffee}, \mathsf{st\_beer}). \\
\mathit{Customer\_arrives} & \stackrel{\text{def}}{=} & \mathsf{customer} = \mathit{none} \wedge \mathsf{customer}' \neq \mathit{none} \wedge \\
 & & \mathit{only}(\mathsf{customer}) \\
\mathit{Customer\_leaves} & \stackrel{\text{def}}{=} & \mathsf{customer} \neq \mathit{none} \wedge \mathsf{customer}' = \mathit{none} \wedge \\
 & & \mathit{only}(\mathsf{customer}). \\
\mathit{Coin\_insert} & \stackrel{\text{def}}{=} & \mathsf{customer} \neq \mathit{none} \wedge \mathsf{coins} \neq \mathit{3} \wedge \\
 & & (\mathsf{coins} = \mathit{0} \rightarrow \mathsf{coins}' = \mathit{1}) \wedge (\mathsf{coins} = \mathit{1} \rightarrow \mathsf{coins}' = \mathit{2}) \wedge \\
 & & (\mathsf{coins} = \mathit{2} \rightarrow \mathsf{coins}' = \mathit{3}) \wedge \\
 & & \mathit{only}(\mathsf{coins}).
\end{array}
$$

# *Transitions*

$$
\begin{aligned}
\mathit{Dispense\_beer} \;\stackrel{\text{def}}{=}\;& \mathsf{customer} = \mathit{student} \land \mathsf{st\_beer} \land \\
& \mathsf{disp} = \mathit{none} \land (\mathsf{coins} = \mathit{2} \lor \mathsf{coins} = \mathit{3}) \land \\
& \mathsf{disp}' = \mathit{beer} \land \\
& (\mathsf{coins} = \mathit{2} \rightarrow \mathsf{coins}' = \mathit{0}) \land (\mathsf{coins} = \mathit{3} \rightarrow \mathsf{coins}' = \mathit{1}) \land \\
& \mathit{only}(\mathsf{st\_beer}, \mathsf{disp}, \mathsf{coins}).
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Dispense\_coffee} \;\stackrel{\text{def}}{=}\;& \mathsf{customer} = \mathit{prof} \land \mathsf{st\_coffee} \land \\
& \mathsf{disp} = \mathit{none} \land \mathsf{coins} \neq \mathit{0} \land \\
& \mathsf{disp}' = \mathit{coffee} \land \\
& (\mathsf{coins} = \mathit{1} \rightarrow \mathsf{coins}' = \mathit{0}) \land (\mathsf{coins} = \mathit{2} \rightarrow \mathsf{coins}' = \mathit{1}) \land \\
& (\mathsf{coins} = \mathit{3} \rightarrow \mathsf{coins}' = \mathit{2}) \land \\
& \mathit{only}(\mathsf{st\_coffee}, \mathsf{disp}, \mathsf{coins}).
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Take\_drink} \;\stackrel{\text{def}}{=}\;& \mathsf{customer} \neq \mathit{none} \land \mathsf{disp} \neq \mathit{none} \land \\
& \mathsf{disp}' = \mathit{none} \land \\
& \mathit{only}(\mathsf{disp}).
\end{aligned}
$$

# *Summary (Transition Systems)*

A transition system is a tuple $\mathbb{S} = (\mathcal{X}, D, dom, In, T)$.

With a transition system $\mathbb{S}$ we associate a PLFD $\mathcal{L}(\mathbb{S})$.

The set of all states of $\mathbb{S}$ is the set of all interpretations in $\mathcal{L}(\mathbb{S})$.

A transition is a set of pairs of sets.

Symbolic Representation:

- a set of sets $\{s \mid s \models F\}$.
- a transition $\{(s, s') \mid (s, s') \models F'\}$,
  where $F'$ is over variables $\mathcal{X} \cup \mathcal{X}'$.

## *Temporal properties of transition systems*

We have used PLFD for describing properties of states and transitions.

PLFD can not express temporal properties — properties about evolution of the system over time:

- There is no state in which a professor and a student are both customers.
- Professors never drink beer.
- The machine cannot dispense drinks forever without recharging.
- If a student is a customer then at some future state a professor will be a customer and vice versa.
- The coin slot should be not empty between any two states in which different drinks are dispensed.

# *Kripke Structures*

Assume an instance of PLFD with the set of variables $\mathcal{X}$. Denote the set of all interpretations for this instance of PLFD by $\mathbb{I}$.

A Kripke structure is a tuple $K = (S, In, T, L)$, where

- $S$ is a finite non-empty set, called the set of states of $K$.
- $In \subseteq S$ is a non-empty set of states, called the set of initial states of $M$.
- $T \subseteq S \times S$ is a set of pairs of states, called the transition relation of $K$.
- $L : S \to \mathbb{I}$ is a function, called the labelling function of $K$.

# *State Transition Graph*

State Transition Graph of a Kripke structure $K$:

- The nodes are the states of $K$.
- The edges are elements of the transition relation: there is an edge from a state $s$ to a state $s'$ if and only if $(s, s') \in T$.

# *Example*

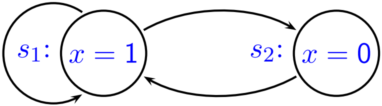
Consider:

a PLFD variables: $\{x\}$, domain $dom\{x\} = \{0, 1\}$

a Kripke structure $K$ over $\{x\}$:

- two states $s_1, s_2$, where $s_1$ is the initial,
- transition relation $T$ is $\{(s_1, s_1), (s_1, s_2), (s_2, s_1)\}$,
- labelling function maps $s_1$ into $\{x \mapsto 1\}$ and $s_2$ into $\{x \mapsto 0\}$.

State Transition Graph:




We will assume that for all nodes there is an outgoing edge.

# *Transition Systems as Kripke Structures*

Let $\mathbb{S}$ be a finite-state transition system. It can be made into a Kripke structure $K = (S, In, T, L)$ as follows.

- The set of variables of $K$ is the set of variables $\mathcal{X}$ of $\mathbb{S}$.
- The set of states $S$ is the set of states of $\mathbb{S}$.
- The set of initial states $In$ is the set of initial states of $\mathbb{S}$.
- The transition relation $T$ is the transition relation of $\mathbb{S}$.
- The labelling function $L$ is defined as follows: for every state $s$ and state variable $x$ we have

$$L(s) \stackrel{\text{def}}{=} s, \text{ that is, } L(s)(x) = s(x).$$

## *Example (see enclosed page!)*

Consider a simplified model of the vending machine: at most two coins, one kind of customer, one kind of drink. The set $\mathcal{X}$ contains the following boolean variables

- boolean variable storage: the storage is non-empty.
- boolean variable dispenser: the dispenser is non-empty.
- variable coins with the domain $\{0, 1, 2\}$ denoting the number of coins in the slot.
- boolean variable customer: a customer is present.

# *Computation Tree*

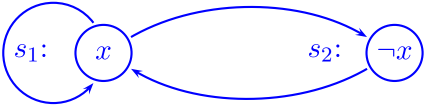
Let $K = (S, In, T, L)$ be a Kripke structure over a set of variables $\mathcal{X}$ and $s \in S$ be a state. The computation tree for $K$ starting at $s$ is the following infinite tree.

- The nodes of the tree are labelled by states in $S$.
- The root of the tree is labelled by $s$.
- For every node $s'$ in the tree, its children are exactly such nodes $s'' \in S$ that $(s', s'') \in T$.

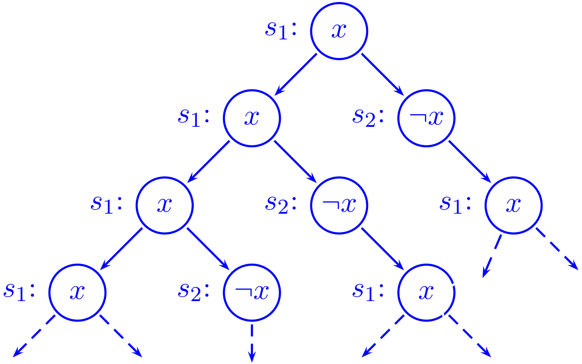
A computation path for $K$: any branch $s_0, s_1, \ldots$ in the tree.

# *Computation Tree*

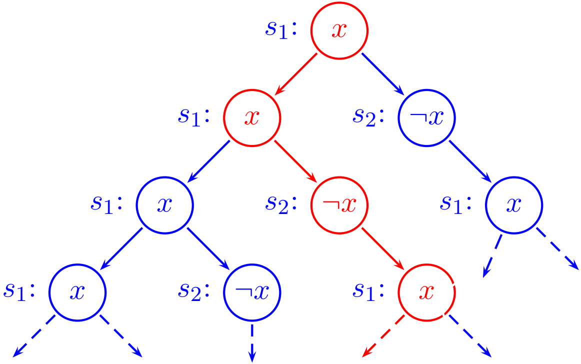
State Transition Graph:




Computation Tree:



# *Computation Path*



## *Properties*

- Computation paths for a Kripke structure are exactly all branches in the computation trees for this Kripke structure.
- Let $n$ be a node in a computation tree $C$ for $K$ labelled by $s'$. Then the subtree of $C$ rooted at $s'$ is the computation tree for $K$ starting at $s'$. In other words, every subtree of a computation tree rooted at some node is itself a computation tree.
- For every Kripke structure $K$ and state $s$ there exists a unique computation tree for $K$ starting at $s$, up to the order of children.

# *Linear Temporal Logic* LTL

Linear Temporal Logic is a logic for reasoning about properties of computation paths.

This logic can express temporal properties of computation paths such as:

- There is no state (on the path) at which a professor and a student are both customers.
- The machine does not dispense drinks forever without recharging.
- If a student is a customer then at some future state a professor will be a customer and vice versa.
- The coin slot should be not empty between any two states in which different drinks are dispensed.

# *Syntax of LTL*

Syntax: extending PLFD with temporal operators.

PLFD part:

- Atomic formulas: $x = v$
- Propositional connectives: $\top$, $\bot$, $\land$, $\lor$, $\neg$, $\rightarrow$ and $\leftrightarrow$.

Temporal part: If $F$ is a formula, then

- $\bigcirc F$ — "Next: $F$ holds at the next state",
- $\Box F$ — "Always: $F$ holds at all future states",
- $\Diamond F$ — "Eventually: $F$ holds at some future state",
- $F \,\mathbf{U}\, G$ — "Until: $F$ holds until $G$ holds",

Example: $\Box(\mathsf{customer} = \mathit{student} \rightarrow \Diamond \mathsf{disp} = \mathit{beer})$

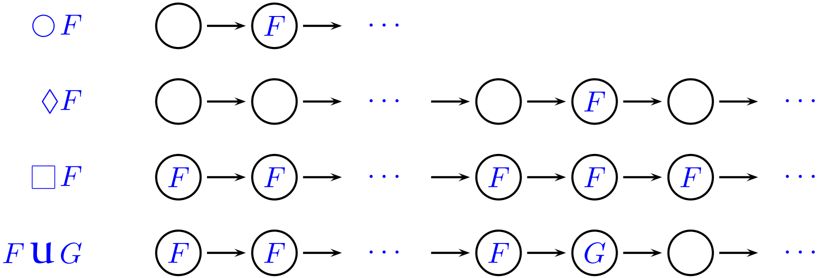
## *Syntax of LTL, Priorities of Operators*

| Operator Connective | Name | Priority |
|---|---|---|
| $\bigcirc$ | Next | 5 |
| $\Box$ | Box | 5 |
| $\Diamond$ | Diamond | 5 |
| $\neg$ | ... | 5 |
| $\mathbf{U}$ | Until | 4 |
| $\wedge, \vee$ | ... | 3 |
| $\rightarrow$ | ... | 2 |
| $\leftrightarrow$ | ... | 1 |

Example: $\Box\neg\Diamond A \, \mathbf{U} \bigcirc B \rightarrow \Diamond A \vee \Box B$ is the same as

$$((\Box\neg\Diamond A)\,\mathbf{U}\,(\bigcirc B)) \rightarrow ((\Diamond A) \vee (\Box B))$$

# *Semantics*



# *Semantics (I)*

Let $\pi = s_0, s_1, s_2 \ldots$ be a sequence of states and $F$ be an LTL formula. We define the notion $F$ is true on $\pi$, denoted by $\pi \models F$, by induction on $F$ as follows. For all $i = 0, 1, \ldots$ denote by $\pi_i$ the sequence of states $s_i, s_{i+1}, s_{i+2} \ldots$ (note that $\pi_0 = \pi$).

- $\pi \models \top$ and $\pi \not\models \bot$.
- $\pi \models x = v$ if $s_0 \models x = v$.
- $\pi \models F_1 \wedge \ldots \wedge F_n$ if for all $j = 1, \ldots, n$ we have $\pi \models F_j$;
- $\pi \models F_1 \vee \ldots \vee F_n$ if for some $j = 1, \ldots, n$ we have $\pi \models F_j$.
- $\pi \models \neg F$ if $\pi \not\models F$.
- $\pi \models F \rightarrow G$ if either $\pi \not\models F$ or $\pi \models G$;
- $\pi \models F \leftrightarrow G$ if either both $\pi \not\models F$ and $\pi \not\models G$ or both $\pi \models F$ and $\pi \models G$.

## *Semantics (I)*

- $\pi \models \bigcirc F$ if $\pi_1 \models F$;
- $\pi \models \Diamond F$ if for some $i = 0, 1, \ldots$ we have $\pi_i \models F$;
- $\pi \models \Box F$ if for all $i = 0, 1, \ldots$ we have $\pi_i \models F$.
- $\pi \models F \, \mathbf{U} \, G$ if for some $k = 0, 1, \ldots$ we have $\pi_k \models G$ and
$$\pi_0 \models F, \ldots, \pi_{k-1} \models F.$$

Two LTL formulas $F$ and $G$ are called equivalent, denoted $F \equiv G$, if for every path $\pi$ we have $\pi \models F$ if and only if $\pi \models G$.

## *Expressing Some Properties*

A path property is defined by a set of paths.

A temporal formula is expressing a property if the set of paths that satisfy the formula is exactly the set of paths which define the property.

Examples: The property "formula $A$ holds at all states" is defined by a formula

- $A$ holds at some state is defined by
- $A$ never holds at two consecutive states
- starting from some state $A \to \neg B$ always holds
- $A$ holds only a finite number of times
- from some state $\neg A$ always holds and until then $B$ holds

# *Expressing Some Properties*

A path property is defined by a set of paths.

A temporal formula is expressing a property if the set of paths that satisfy the formula is exactly the set of paths which define the property.

Examples: The property "formula $A$ holds at all states" is defined by a formula $\Box A$.

- $A$ holds at some state is defined by
- $A$ never holds at two consecutive states
- starting from some state $A \to \neg B$ always holds
- $A$ holds only a finite number of times
- from some state $\neg A$ always holds and until then $B$ holds

# *Expressing Some Properties*

A path property is defined by a set of paths.

A temporal formula is expressing a property if the set of paths that satisfy the formula is exactly the set of paths which define the property.

Examples: The property "formula $A$ holds at all states" is defined by a formula $\Box A$.

- $A$ holds at some state is defined by $\Diamond A$
- $A$ never holds at two consecutive states
- starting from some state $A \to \neg B$ always holds
- $A$ holds only a finite number of times
- from some state $\neg A$ always holds and until then $B$ holds

# *Expressing Some Properties*

A path property is defined by a set of paths.

A temporal formula is expressing a property if the set of paths that satisfy the formula is exactly the set of paths which define the property.

Examples: The property "formula $A$ holds at all states" is defined by a formula $\Box A$.

- $A$ holds at some state is defined by $\Diamond A$
- $A$ never holds at two consecutive states $\Box(A \rightarrow \bigcirc\neg A)$
- starting from some state $A \rightarrow \neg B$ always holds
- $A$ holds only a finite number of times
- from some state $\neg A$ always holds and until then $B$ holds

# *Expressing Some Properties*

A path property is defined by a set of paths.

A temporal formula is expressing a property if the set of paths that satisfy the formula is exactly the set of paths which define the property.

Examples: The property "formula $A$ holds at all states" is defined by a formula $\Box A$.

- $A$ holds at some state is defined by $\Diamond A$
- $A$ never holds at two consecutive states $\Box(A \to \bigcirc \neg A)$
- starting from some state $A \to \neg B$ always holds $\Diamond \Box(A \to \neg B)$
- $A$ holds only a finite number of times
- from some state $\neg A$ always holds and until then $B$ holds

## *Expressing Some Properties*

A path property is defined by a set of paths.

A temporal formula is expressing a property if the set of paths that satisfy the formula is exactly the set of paths which define the property.

Examples: The property "formula $A$ holds at all states" is defined by a formula $\Box A$.

- $A$ holds at some state is defined by $\Diamond A$
- $A$ never holds at two consecutive states $\Box(A \rightarrow \bigcirc \neg A)$
- starting from some state $A \rightarrow \neg B$ always holds $\Diamond \Box (A \rightarrow \neg B)$
- $A$ holds only a finite number of times $\Diamond \Box \neg A$
- from some state $\neg A$ always holds and until then $B$ holds

## *Expressing Some Properties*

A path property is defined by a set of paths.

A temporal formula is expressing a property if the set of paths that satisfy the formula is exactly the set of paths which define the property.

Examples: The property "formula $A$ holds at all states" is defined by a formula $\Box A$.

- $A$ holds at some state is defined by $\Diamond A$
- $A$ never holds at two consecutive states $\Box(A \to \bigcirc \neg A)$
- starting from some state $A \to \neg B$ always holds $\Diamond \Box (A \to \neg B)$
- $A$ holds only a finite number of times $\Diamond \Box \neg A$
- from some state $\neg A$ always holds and until then $B$ holds $B \, \mathbf{U} \, \Box \neg A$

# *Some useful properties*

- Reachability and safety properties.

  Let unsafe describe states which are unsafe.

  Then $\Box\neg\mathsf{unsafe}$ express a safety requirement.

  Ex: $\Box\neg(\mathsf{disp} = \mathsf{beer} \wedge \mathsf{customer} = \mathsf{prof})$

- Mutual exclusion. Two processes are not in the critical section.

  Ex: $\Box\neg(\mathsf{critical}_1 \wedge \mathsf{critical}_2)$

- Fairness. Ex: $\Box(\mathsf{customer} = \mathsf{student} \rightarrow \Diamond\mathsf{customer} = \mathsf{prof})$

- Responsiveness: every request will be eventually acknowledged

  Ex: $\Box(\mathsf{request} \rightarrow (\mathsf{request}\,\mathbf{U}\,\mathsf{ack}))$

- Alternation. Ex: $A \wedge \Box(A \leftrightarrow \neg\bigcirc A)$

  $$A \rightarrow \neg A \rightarrow A \rightarrow \neg A \rightarrow A \rightarrow \neg A \rightarrow A \rightarrow \cdots$$

## Some Equivalences

Two LTL formulas $F$ and $G$ are called equivalent, denoted $F \equiv G$, if for every path $\pi$ we have $\pi \models F$ if and only if $\pi \models G$.

Negation:

$$
\begin{array}{rcl}
\neg \bigcirc A & \equiv & \bigcirc \neg A \\
\neg \Diamond A & \equiv & \Box \neg A \\
\neg \Box A & \equiv & \Diamond \neg A \\
\neg (A \,\mathbf{U}\, B) & \equiv & (A \wedge \neg B) \,\mathbf{U}\, (\neg A \wedge \neg B) \vee \Box \neg B
\end{array}
$$

Expressing operators through $\mathbf{U}$.

$$
\begin{array}{rcl}
\Diamond A & \equiv & \\
\Box A & \equiv &
\end{array}
$$

LTL without $\Box, \Diamond$ has the same expressive power as LTL.

## *Some Equivalences*

Two LTL formulas $F$ and $G$ are called equivalent, denoted $F \equiv G$, if for every path $\pi$ we have $\pi \models F$ if and only if $\pi \models G$.

Negation:

$$
\begin{array}{rcl}
\neg \bigcirc A & \equiv & \bigcirc \neg A \\
\neg \Diamond A & \equiv & \Box \neg A \\
\neg \Box A & \equiv & \Diamond \neg A \\
\neg (A \,\mathbf{U}\, B) & \equiv & (A \land \neg B) \,\mathbf{U}\, (\neg A \land \neg B) \lor \Box \neg B
\end{array}
$$

Expressing operators through $\mathbf{U}$.

$$
\begin{array}{rcl}
\Diamond A & \equiv & \top \,\mathbf{U}\, A \\
\Box A & \equiv &
\end{array}
$$

LTL without $\Box, \Diamond$ has the same expressive power as LTL.

## *Some Equivalences*

Two LTL formulas $F$ and $G$ are called equivalent, denoted $F \equiv G$, if for every path $\pi$ we have $\pi \models F$ if and only if $\pi \models G$.

Negation:

$$
\begin{array}{rcl}
\neg \bigcirc A & \equiv & \bigcirc \neg A \\
\neg \Diamond A & \equiv & \Box \neg A \\
\neg \Box A & \equiv & \Diamond \neg A \\
\neg (A \, \mathbf{U} \, B) & \equiv & (A \wedge \neg B) \, \mathbf{U} \, (\neg A \wedge \neg B) \vee \Box \neg B
\end{array}
$$

Expressing operators through $\mathbf{U}$.

$$
\begin{array}{rcl}
\Diamond A & \equiv & \top \, \mathbf{U} \, A \\
\Box A & \equiv & \neg (\top \, \mathbf{U} \, \neg A)
\end{array}
$$

LTL without $\Box, \Diamond$ has the same expressive power as LTL.

# *How to Show that Two Formulas are not Equivalent?*

Find a path that satisfies one of the formulas but not the other.

For example to show that $\square(F \vee G) \not\equiv \square F \vee \square G$ consider:

$$F \longrightarrow G \longrightarrow F \longrightarrow G \longrightarrow \cdots$$

# *LTL and Kripke structures*

For an LTL formula $F$ we can consider at least two kinds of properties of a Kripke structure $K$:

- does $F$ hold on some computation path for $K$ from an initial state?
- does $F$ hold on all computation paths for $K$ from an initial state?

# *Putting it All Together*

When we design a system, we would like to be sure that it will satisfy all requirements, such as safety.

- We can formally represent transition systems (the symbolic representation);
- We can express the desired properties of the systems in temporal logic.

What is missing?

# *Model Checking*

Model checking problem:

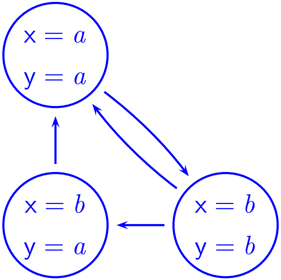
Given

- A (symbolic) representation of a transition system;
- A temporal formula $F$,

check if every (some) computation of the system satisfies this formula, preferably in a fully automatic way.

# *Example*

Consider a Kripke structure with the following transition graph:



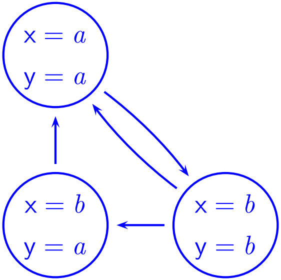

The initial state is the top one.

Are the following formulas true on all path (from the initial state)?

- $\Box(x = a \leftrightarrow y = a)$
- $\Box(x = b \rightarrow \Diamond y = a)$
- $\Box\Diamond y = b$
- $\Box\Diamond y \neq b$

# *Example*

Consider a Kripke structure with the following transition graph:



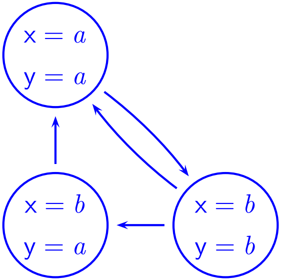

The initial state is the top one.

Are the following formulas true on all path (from the initial state)?

| | |
|---|---|
| $\Box(x = a \leftrightarrow y = a)$ No | $\Box\Diamond y = b$ |
| $\Box(x = b \rightarrow \Diamond y = a)$ | $\Box\Diamond y \neq b$ |

# *Example*

Consider a Kripke structure with the following transition graph:



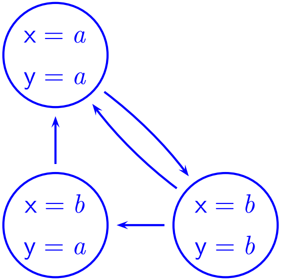

The initial state is the top one.

Are the following formulas true on all path (from the initial state)?

| | | | |
|---|---|---|---|
| $\Box(x = a \leftrightarrow y = a)$ | No | $\Box\Diamond y = b$ | |
| $\Box(x = b \rightarrow \Diamond y = a)$ | Yes | $\Box\Diamond y \neq b$ | |

# *Example*

Consider a Kripke structure with the following transition graph:



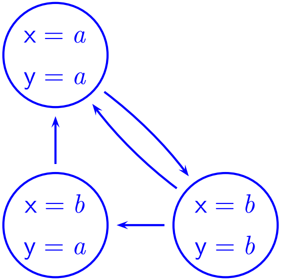

The initial state is the top one.

Are the following formulas true on all path (from the initial state)?

| | | | |
|---|---|---|---|
| $\Box(x = a \leftrightarrow y = a)$ | No | $\Box\Diamond y = b$ | Yes |
| $\Box(x = b \rightarrow \Diamond y = a)$ | Yes | $\Box\Diamond y \neq b$ | |

# *Example*

Consider a Kripke structure with the following transition graph:




The initial state is the top one.

Are the following formulas true on all path (from the initial state)?

| | | | |
|---|---|---|---|
| $\Box(x = a \leftrightarrow y = a)$ | No | $\Box\Diamond y = b$ | Yes |
| $\Box(x = b \rightarrow \Diamond y = a)$ | Yes | $\Box\Diamond y \neq b$ | Yes |

# *Symbolic Model Checking for Reachability*

A reachability property is expressed by a formula

$$\Diamond F,$$

where $F$ is a propositional formula.

Usually $F$ is a PLFD formula which represents a set of unsafe states.

We want to check that these unsafe states are not reachable in any computation of our Kripke structure (transition system).

# *Reachability*

Fix a Kripke structure $K$ with the transition relation $T$. We write $s_0 \to s_1$ for $(s_0, s_1) \in T$ (that is, there is a transition from $s_0$ to $s_1$.

- A state $s$ is reachable in $n$ steps from a state $s_0$ if there exists a sequence of states $s_1, \ldots, s_n$ such that $s_n = s$ and

$$s_0 \to s_1 \to \ldots \to s_n.$$

- A state $s$ is reachable from a state $s_0$ if $s$ is reachable from $s_0$ in $n \geq 0$ steps.

# *Reachability*

Fix a Kripke structure $K$ with the transition relation $T$. We write $s_0 \to s_1$ for $(s_0, s_1) \in T$ (that is, there is a transition from $s_0$ to $s_1$.

- A state $s$ is reachable in $n$ steps from a state $s_0$ if there exists a sequence of states $s_1, \ldots, s_n$ such that $s_n = s$ and

$$s_0 \to s_1 \to \ldots \to s_n.$$

- A state $s$ is reachable from a state $s_0$ if $s$ is reachable from $s_0$ in $n \geq 0$ steps.

**Theorem.** Let $F$ be a propositional formula. The formula $\Diamond F$ holds on some computation path if and only if there exists an initial state $s_0$ and a state $s$ such that $s \models F$ and $s$ is reachable from $s_0$.

# *Reformulation of reachability*

Given

- PLFD formula representing a set of initial states $In$;
- PLFD formula representing a set of unsafe states $Unsafe$;
- PLFD formula representing $Tr$ the transition relation of the transition system $\mathbb{S}$,

is any unsafe state reachable from an initial state in $\mathbb{S}$?

# $k$-Step Reachability

Using this observation, we can define a sequence of formulas $reach_{\leq k}$ for reachability in $\leq k$ states:

$$\begin{aligned} reach_{\leq 0}(\bar{x}) &\stackrel{\text{def}}{=} In(\bar{x}) \\ reach_{\leq k+1}(\bar{x}) &\stackrel{\text{def}}{=} reach_{\leq k}(\bar{x}) \vee \exists \bar{y}(reach_{\leq k}(\bar{y}) \wedge Tr(\bar{y}, \bar{x})) \end{aligned}$$

Lemma. The formula $reach_{\leq k}(V)$ represents reachability in $\leq k$ steps in the following sense. For every state $s$, we have

$$\{s | s \models reach_{\leq k}(V)\} = reach\_set_{\leq k}$$

is the set of all reachable states in $\leq k$ steps.

# The set of Reachable States

$$\begin{aligned} reach\_set_{\leq 0}(\bar{x}) \quad &\subset reach\_set_{\leq 1}(\bar{x}) \subset reach\_set_{\leq 2}(\bar{x}) \ldots \\ &\subset reach\_set_{\leq k}(\bar{x}) = reach\_set_{\leq k+1}(\bar{x}) = reach\_set \end{aligned}$$

Since the number of all states is finite we will always reach a point $k$ after which we cannot reach more states.

At this point we obtain the set of all reachable states: $reach\_set$.

# The set of Reachable States

$$\begin{aligned} reach\_set_{\leq 0}(\bar{x}) \quad &\subset reach\_set_{\leq 1}(\bar{x}) \subset reach\_set_{\leq 2}(\bar{x}) \ldots \\ &\subset reach\_set_{\leq k}(\bar{x}) = reach\_set_{\leq k+1}(\bar{x}) = reach\_set \end{aligned}$$

Since the number of all states is finite we will always reach a point $k$ after which we cannot reach more states.
At this point we obtain the set of all reachable states: $reach\_set$.

Finally we need to check that there is no reachable unsafe states, i.e.

$$\exists \bar{x}(reach(\bar{x}) \wedge Unsafe(\bar{x}))$$

is false.

# *Efficient Representation of Reachability*

Efficient representation of this symbolic computation:

- use OBDDs to represent $reach\_set_{\leq i}(\bar{x})$
- for this we need to use quantifier elimination algorithm for OBDDs
- to check that $reach\_set_{\leq k}(\bar{x}) = reach\_set_{\leq k+1}(\bar{x}) = reach\_set$ we need equivalence check on OBDDs

# *Summary LTL/Model Checking*

Linear Time Logic (LTL) is used to describe temporal properties of computation paths of Kripke structures.

LTL Extends PLFD with temporal operators $\Box, \Diamond, \mathbf{U}$.

Model checking problem: Given

- A (symbolic) representation of a transition system;
- A temporal formula $F$,

check if every (some) computation of the system satisfies $F$.

Algorithm for Symbolic Model Checking of safety properties.

- use OBDDs to represent the set of reachable states in $\leq k$ steps
- we use algorithms for constructing OBDDs, quantifier elimination and equivalence checking.