

Chapter 13

Transition Systems and Temporal Properties

A core belief of Buddhism, as in Hinduism, is the belief that actions in this life have consequences for the next life.
The Rev. Myung Shin Ham, pastor.

Contents

13.1 State-Changing Systems: Informal Discussion	192
13.2 Example: a Vending Machine	193
13.2.1 State Variables and States	194
13.2.2 Transitions	195
13.3 Transition Systems	196
13.4 Symbolic Representation of Transition Systems	198
13.4.1 Representing Sets of States	198
13.4.2 Representing Transitions	199
13.4.3 Frame Problem	201
13.5 Temporal Properties of Transition Systems	204
Exercises	205

In this chapter we discuss state-changing systems and their formalization as *transition systems*. State-changing systems are the systems functioning in time and changing state under the influence of various actions. Transition systems are a specific formalization of state-changing systems in which a system is represented by a set of variables and a state consists of an assignment of values to these variables.

We begin this chapter with an informal discussion of state-changing systems and the problem of their formalization in Section 13.1. Then in Section 13.2 we consider a concrete example of a state-changing system: a vending machine. We illustrate the key concepts

of state, state variable, and transition. A formal analogue of a state-changing system is a *transition system*. This notion is formally defined in Section 13.3, together with a notion of *finite-state transition system*. In Section 13.4 we discuss how one can represent transition systems using logic. It turns out that propositional logic of finite domains is ideally suited for representing sets of states of finite-state transition systems. By using a technique of introducing *next-state variables* one can also represent transitions in this logic. Finally, in Section 13.5 we discuss *temporal properties* that are not expressible in PLFD. Studying temporal properties will be our main motivation for the following chapters.

13.1 State-Changing Systems: Informal Discussion

The logics we introduced so far: propositional logic, PLFD and the logic of quantified boolean formulas, can be used to describe a static application world. By “static” we mean that the world does not change in time, or that we are only interested in one particular state of the world. There are numerous applications where we have to reason about systems whose state changes in time. Typical examples are operating systems, networks, protocols, hardware, autonomous devices, and many others.

For example, when we reason about the safety of cryptographic protocols, we have to discuss possible sequences of actions undertaken by an intruder. Each action changes the state of the system that uses the protocol. When we reason about functioning of hardware, we also discuss possible sequences of state changes in hardware, for example, the states through which a processor goes when executing a sequence of instructions.

A *state-changing system* is usually characterized by the following properties.

- (1) At each particular time moment, the system is in a particular *state*.
- (2) The state of the system may change, normally under the influence of some kind of *action*. Actions can be carried out by the system itself but can also be out of its control, for example, performed by the system environment.

Our aim is to build a mathematical model of such systems. Such a model is usually based on the following abstractions of the notions of state and action.

- (1) We introduce *variables* to characterize some parameters of the system (for example, the temperature) and assume that the state is identified by fixing the values of the variables at the state. These variables are called the *state variables* of this system.
- (2) When we formalize an *action*, we try to determine what are the possible changes in the state induced by this action, which amounts to finding out how it changes the values of state variables.

The word *transition* is normally used instead of action, hence the name “transition systems”.

The following examples of state-changing systems typically arise in various applications.

- (1) *Reactive systems*. These are the systems that interact with their environment. A vending machine can be regarded as a reactive system, in which the environment is identified by customers and their requests.
- (2) *Concurrent systems*. These are the systems that consist of a set of components functioning together. Usually, functioning of each of these components can be described independently, but they also communicate through shared variables or some kind of *communication channels*, for example queues.

Formal reasoning about a state-changing system usually follows two stages.

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

The aim of this chapter is building formal models of state-changing systems. The following chapters will deal with logics for reasoning about such systems.

13.2 Example: a Vending Machine

In this section we consider a vending machine as an example of a state-changing system and informally discuss issues related to building a formal model of this system.

We assume that the vending machine dispenses drinks in a university department. It 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 inserting coins. When the machine is operating, it comes through several stages depending on the behavior of a customer. Each action undertaken by the customer or by the machine itself may change the state of the system. For example, when the customer inserts a coin in the coin slot, the amount of money in it changes. We would like to include the customer in our formalization too, so in fact the system we are describing consists of both the machine and the customer.

Let us try to build a formal model of the vending machine. In order to do this, we have to identify

- (1) what are the state variables;
- (2) what are the possible values of the state variables;
- (3) what are the transitions and how they change the values of the state variables.

We will consider these issues in the following sections.

13.2.1 State Variables and States

When building a formal model of a state-changing system, we can use different *abstractions* of the real system, depending on the level of details and properties we are interested in. For example, concerning the money slot, we can use the following information:

what is the amount of money in the slot? (13.1)

To represent this information, we can introduce an integer-valued variable representing the amount of money. Let us denote this variable by *money*. Since the variable *money* ranges over an infinite set of possible values, the number of possible states of the system becomes infinite, unless we impose a restriction on the maximal amount of money in the slot. If we impose such a restriction, the variable *money* will range over a finite set of integers $\{0, 1, \dots, m\}$.

This abstraction is very fine and may contain more information than we need. Depending on our aims, we may only be interested in knowing

is the slot empty or not,
is money sufficient to buy a drink, and
should any change be given? (13.2)

In this case we can use the following finite domain of possible values for the variable *money*: $\{none, not_enough, just_enough, too_much\}$.

But it is also quite possible that we are not interested in functioning of the change-giving component of the machine, and are only interested in the following

is money enough to buy a drink? (13.3)

In this case we obtain a coarser abstraction and treat *money* as a boolean-valued variable: *money* has the boolean value 1 if and only if the slot contains enough money to buy a drink.

The variable *money* describes only one component of the vending machine. In general we may be interested in having more variables describing other components. Then for each variable we have to define the set of possible values, called the *domain* for this variable. This observation makes us think of a possible analogy with propositional logic of finite domains. Indeed, in order to identify an instance of this logic, we also had to identify a set of variables and a domain for each variable of this set.

As soon as we have identified all state variables and their possible values, we can define the state of the system as a collection of the values of all variables. More precisely, it can be defined as a mapping from the set of variables to the set of values such that every variable is mapped into a value in its domain. When we define the state in this way, we effectively *identify* the state with the mapping of variables to values. Going back to our analogy with propositional logic of finite domains, a state is simply an interpretation of this logic. By identifying the state with a mapping, we create an *abstraction* of the real system. For example, in our abstraction of the vending machine we are not interested in such properties as the color or make of the machine.

13.2.2 Transitions

Let us now discuss how one can formalize transitions in a state-changing system. A transition brings a system from one state to another. Consider, for example, the transition caused by pressing a button to dispense a drink. If there is sufficient money in the coin slot, the relevant drink available, and the dispenser is empty, then this transition can be applied. It can affect nearly every component of the system, apart from the user. It changes the amount of money in the coin slot, the amount of drinks stored in the storage, and the content of the drink dispenser.

The question is how one can formalize the notion of transition in general. A transition changes the state of the system, so one particular way of viewing a transition is by defining it as a pair (s, s') , where s is the state of the system before the transition, and s' is the state after. Such a definition is not convenient for us for several reasons explained below.

Suppose that the coin slot in our formal model of the vending machine is represented by an integer (restricted or not) variable *money*. The effect of the “pressing button” transition described above on the value of *money* is the following: if the price p of the drink is less than or equal to the value of *money*, then the value of *money* decreases by p . Otherwise, the value of *money* remains the same. Note that this transition is applicable to *many* different states of the system. Indeed, the only requirement to the state is that the value of *money* is greater than or equal to p , so there may be quite a range of admissible values for *money*. In addition, there are no restrictions on the content of the storage or drink dispenser.

It follows that this transition is not a pair of states, but rather a function on states. If we identify a state with the value m of *money*, then the function is defined as follows:

$$drink_dispensing(m) = \begin{cases} m - p, & \text{if } m \geq p; \\ m, & \text{if } m < p. \end{cases}$$

However, even formalizing a transition as a function on states may be insufficient. There are at least two reasons for this

First, sometimes we may wish to say that a particular transition is not applicable to a certain state. For example, we could wish to consider *drink_dispensing* undefined when $m < p$. Then defining transition as a total function would not be appropriate.

Second, some transitions may be inherently non-deterministic. For example, suppose that we want to formalize an erroneous behavior of the vending machine, when it can by mistake swallow some amount of money in the slot. If the value of *money* was $m > 0$ before the transition, then after the transition it may become any number between 0 and $m - 1$. Therefore, in this case a transition is not even a partial function.

In this book we will take the most general definition of a transition as a set of pairs of states. For example, the money-swallowing transition can be represented by the set of pairs

$$\{(m, m') \mid m > 0, 0 \leq m' < m\}.$$

This implies that for a state s there may be more than one state s' such that s can be changed into s' by the transition. In this case we will say that the transition is *non-deterministic*.

13.3 Transition Systems

Based on the discussion of the previous section, we can now define a formalization of state-changing systems as *transition systems*.

DEFINITION 13.1 (Transition System) A *transition system* is a tuple $\mathbb{S} = (\mathcal{X}, D, dom, In, T)$, where

- (1) \mathcal{X} is a finite set of *state variables*.
- (2) D is a non-empty set, called the *domain*. Elements of D are called *values*.
- (3) dom is a mapping from \mathcal{X} to the set of non-empty subsets of D . For each state variable $x \in \mathcal{X}$, the set $dom(x)$ is called the *domain for x* .

The meaning of In and T will be defined below after we have defined the notions of state and transition.

A *state* of a transition system \mathbb{S} is a function $s : \mathcal{X} \rightarrow D$ such that for every $x \in \mathcal{X}$ we have $s(x) \in dom(x)$. A *transition* is a set of pairs of states. Concerning In and T , we have the following.

- (4) In is a set of states, called the *initial states* of \mathbb{S} .
- (5) T is a finite set of transitions.

A transition t is said to be *applicable* to a state s if there exists a state s' such that $(s, s') \in t$. A transition t is said to be *deterministic* if for every state s there exists at most one state s' such that $(s, s') \in t$, and *non-deterministic* otherwise.

The *transition relation* of \mathbb{S} , denoted by $Tr_{\mathbb{S}}$, 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 \mathbb{S} is said to be *finite-state* if its set of states is finite, and *infinite-state* otherwise. \square

Let t be a transition. It is easy to see that t is non-deterministic if and only if there exist states s, s'_1, s'_2 such that $s'_1 \neq s'_2$, $(s, s'_1) \in t$ and $(s, s'_2) \in t$.

Let \mathbb{S} be a transition system. It is not hard to argue that \mathbb{S} is a finite-state system if and only if for every state variable x of \mathbb{S} the domain for this variable is finite. Note that according to our definition, the states are only those mappings s which map x into a value in $dom(x)$. Therefore, if we replace D by $\bigcup_{x \in \mathcal{X}} dom(x)$, i.e., use only those values that belong to a domain for at least one variable, then the set of states of the system does not change. Therefore, we can assume that $D = \bigcup_{x \in \mathcal{X}} dom(x)$. Under this assumption, finite-state systems are exactly those whose domain D is finite. In this book we will only study finite-state systems.

Let us try to formalize a vending machine as a finite-state transition system. First, we describe the state variables and the effect of all transitions in a semi-formal way.

- (1) 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. We are only interested in whether a particular kind of drink is currently being stored or not, but not interested in the amount of it.
- (2) The coin slot can accommodate up to 3 coins.
- (3) 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.
- (4) A can of beer costs two coins. A cup of coffee costs one coin.
- (5) There are two kinds of customers: students and professors. Students only drink beer, professors only drink coffee.
- (6) From time to time the drink storage can be recharged. After the recharge both beer and coffee become available.

Our formalization will use the following state variables.

- (1) Boolean state variables *st_coffee* and *st_beer* signaling whether the corresponding drink is currently being stored in the drink storage.
- (2) A state variable *disp* with the domain *none, beer, coffee* whose value is the current content of the drink dispenser.
- (3) A state variable *coins* denoting the current number of coins in the slot. Its possible values are *0, 1, 2, 3*.
- (4) A state variable *customer* denoting the current customer. The domain for this variable contains three values *student, prof, none*.

One can have an idea of introducing a single variable for representing the content of the drink storage. Then this variable would have four values. However, using such a variable is hardly a good idea, since presence of beer in the storage is independent of the presence of coffee, so it is better to have two independent variables. The variables and their domains are summarized in Figure 13.1. It is not hard to argue that the system has 144 different states.

To define the transition system, we should also define the sets of initial states and transitions. We assume that there is one initial state, in which the drink storage, the drink dispenser, and the coin slot are empty, and there is no customer. As for the transitions, we assume the following ones:

- (1) *Recharge*, which results in the drink storage having both beer and coffee;
- (2) *Customer_arrives*, after which a customer appears at the machine;

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

Figure 13.1: State variables and their domains for the vending machine example.

- (3) *Customer_leaves*, after which the customer leaves;
- (4) *Coin_insert*, when the customer inserts a coin into the coin slot;
- (5) *Dispense_beer*, when the customer presses the button to get a can of beer;
- (6) *Dispense_coffee*, when the customer presses the button to get a cup of coffee;
- (7) *Take_drink*, when the customer removes a drink from the dispenser.

We can describe every transition by defining states to which it is applicable and its effect on the values of all variables. For example, *Take_drink* is applicable to the states where the dispenser is non-empty and there is a customer, i.e., the value of *disp* is different from *none* and the value of *customer* is different from *none*. The only effect of this transition is setting the value of *disp* to *none*.

A representation of a transition by an explicit enumeration of all pairs of states in it is not very convenient and may be practically impossible because of the large number of such pairs. What we need next is a convenient formalism for specifying sets of states and transitions.

13.4 Symbolic Representation of Transition Systems

In this section we introduce a so-called *symbolic representation* of transition systems. A symbolic representation uses formulas in some logic to describe sets of states and transitions.

13.4.1 Representing Sets of States

Note that the set of variables of a finite-state system together with the mapping *dom* identifies an instance of propositional logic of finite domains, moreover the states of the system are exactly the interpretations of this instance. Every formula of this logic can be considered as representing a set of states, namely the set of states in which this formula is true.

DEFINITION 13.2 (Logic $\mathcal{L}(\mathbb{S})$ and Symbolic Representation of Sets of States) Let \mathbb{S} be a transition system. Denote by $\mathcal{L}(\mathbb{S})$ the following instance of propositional logic of finite domains. The set of variables of $\mathcal{L}(\mathbb{S})$ is the set of variables \mathcal{X} of \mathbb{S} and the domain mapping of $\mathcal{L}(\mathbb{S})$ is the domain mapping dom of \mathbb{S} . We say that a formula F of $\mathcal{L}(\mathbb{S})$ *symbolically represents* a set of states S if

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

Instead of *symbolically represents* we will usually simply say *represents*. \square

For example, the formula $coins = 2$ represents the set of states in which the variable $coins$ has the value 2, that is, the set of states in which the coin slot contains exactly two coins.

Let us try to represent some sets of states for the vending machine example. We will use a convenient abbreviation $x \neq v$ instead of $\neg(x = v)$, where x is a state variable and v is a value. As discussed above, the transition *Take_drink* is applicable to the states in which the dispenser is non-empty and a customer is present. This can be expressed by the formula

$$disp \neq none \wedge customer \neq none.$$

Let us try to represent the states in which the machine is ready to dispense a drink (independently of whether the right kind of customer is present). In every such state, the drink should be available, the drink dispenser empty, and the coin slot contain enough coins. This can be expressed by the following formula

$$\begin{aligned} & (st_coffee \vee st_beer) \wedge disp = none \wedge \\ & ((coins = 1 \wedge st_coffee) \vee coins = 2 \vee coins = 3). \end{aligned}$$

Alternatively, one can write an equivalent formula

$$\begin{aligned} & (st_coffee \vee st_beer) \wedge disp = none \wedge \\ & coins \neq 0 \wedge (coins = 1 \rightarrow st_coffee). \end{aligned}$$

13.4.2 Representing Transitions

A transition is a set of pairs of states. Therefore, to represent a transition, we have to be able to represent properties of two consecutive states. The standard technique for doing so is to use two collections of variables: each one refers to one of the states. The first is usually called the *current state*. The second is called the *next state*. To denote variables referring to the next state, we use copies of the variables in \mathcal{X} , obtained by suffixing them with “'”. So for every state variable $x \in \mathcal{X}$ we will use x' to refer to this variable in the next state.

For every transition system $S = (\mathcal{X}, D, dom, In, T)$ denote by \mathcal{X}' the following set of variables

$$\mathcal{X}' \stackrel{\text{def}}{=} \{x' \mid x \in \mathcal{X}\}.$$

Let us introduce an instance $\mathcal{L}'(\mathbb{S})$ of PLFD which will be used for expressing properties of both the current and the next state. The set of variables of $\mathcal{L}'(\mathbb{S})$ is $\mathcal{X} \cup \mathcal{X}'$, the domain of a variable $x \in \mathcal{X}$ is $dom(x)$, the domain of a variable $x' \in \mathcal{X}'$ is also $dom(x)$.

Every pair of states (s, s') can be naturally made into an interpretation of $\mathcal{L}'(\mathbb{S})$ as follows. For all variables $x \in \mathcal{X}$ and $x' \in \mathcal{X}'$ we define

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

It is not hard to argue that a pair of states (s, s') considered as an interpretation of $\mathcal{L}'(\mathbb{S})$ satisfies the following property for all variables $x \in \mathcal{X}$, $x' \in \mathcal{X}'$ and values $v \in dom(x)$:

$$\begin{aligned} (s, s') \models (x = v) &\Leftrightarrow s(x) = v; \\ (s, s') \models (x' = v) &\Leftrightarrow s'(x) = v. \end{aligned}$$

Essentially, this means that the variables in \mathcal{X} refer to the current state s , while the variables in \mathcal{X}' refer to the next state s' .

DEFINITION 13.3 (Symbolic Representation of Transitions) We say that a formula F of $\mathcal{L}'(\mathbb{S})$ *symbolically represents* a transition t if

$$t = \{(s, s') \mid (s, s') \models F\}.$$

Instead of *symbolically represents* we will usually simply say *represents*. □

Let us try to represent symbolically the vending machine transitions described on page 197.

The simplest transition is *Recharge* which results in the machine having both coffee and beer in the drink storage. We assume that this transition is applicable when there are no customers waiting to be served. The definition of the symbolic representation may suggest that this transition could be expressed by the formula

$$\text{customer} = \text{none} \wedge \text{st_coffee}' \wedge \text{st_beer}' ,$$

but it is not so. For example, this formula has models (s, s') such that $s \models \text{coins} = 0$ and $s' \models \text{coins} = 3$, which is not what we intended: in real life it is unreasonable to expect coins to magically appear in the coin slot as a result of a recharge. To solve this problem, we have to assert that the value of coins is not changed by the transition. In fact, we have to assert similar statements for all variables apart from `st_coffee` and `st_beer`. To express such properties in a succinct way, we introduce an abbreviation. Let x, y be two variables of PLFD such that $dom(x) = dom(y)$. Then we denote by $x = y$ the formula

$$\bigwedge_{v \in dom(x)} (x = v \leftrightarrow y = v).$$

For example `coins = coins'` denotes the formula

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

A correct symbolic representation of the transition *Recharge* can be given by the formula

$$\begin{aligned}
& \text{customer} = \text{none} \wedge \text{st_coffee}' \wedge \text{st_beer}' \wedge \\
& \text{customer} = \text{customer}' \wedge \text{disp} = \text{disp}' \wedge \text{coins} = \text{coins}'.
\end{aligned} \tag{13.4}$$

Note that the transition relation of \mathbb{S} is a set of pairs of states, and can therefore be considered as a transition. The following theorem shows how a symbolic representation of the transition relation of \mathbb{S} can be obtained from symbolic representations of each transition in \mathbb{S} .

THEOREM 13.4 *Let t_1, \dots, t_n be all transitions of \mathbb{S} and formulas F_1, \dots, F_n symbolically represent t_1, \dots, t_n , respectively. Then the formula $F_1 \vee \dots \vee F_n$ symbolically represents the transition relation of \mathbb{S} .*

PROOF. Take any pair of states (s, s') and note the following equivalences:

$$\begin{aligned}
& (s, s') \in Tr_{\mathbb{S}} \Leftrightarrow \\
& \quad \text{(by the definition of } Tr_{\mathbb{S}}) \\
& (s, s') \in t_i \text{ for some } i \Leftrightarrow \\
& \quad \text{(since } F_i \text{ symbolically represents } t_i) \\
& (s, s') \models F_i \text{ for some } i \Leftrightarrow \\
& (s, s') \models F_1 \vee \dots \vee F_n. \quad \square
\end{aligned}$$

Now we can give the main definition of this chapter.

DEFINITION 13.5 (Symbolic Representation of Transition Systems) Let $\mathbb{S} = (\mathcal{X}, D, dom, In, T)$ be a transition system. We say that a pair of formulas F, F' of logics $\mathcal{L}(\mathbb{S}), \mathcal{L}'(\mathbb{S})$ symbolically represents \mathbb{S} if F symbolically represents the set of initial states In and F' symbolically represents the transition relation $Tr_{\mathbb{S}}$ of \mathbb{S} . \square

13.4.3 Frame Problem

For systems with many state variables a symbolic representation of a transition usually contains a large number of formulas $x = x'$ asserting that the value of a state variable x after the transition is equal to its value before the transition. These formulas are known as *frame formulas*. For example, the frame formula for the state variable *coins* is $\text{coin} = \text{coins}'$.

The problem of expressing that the values of some state variables do not change after a transition or an action is well-known in artificial intelligence, knowledge representation, and reasoning about actions. It is called the *frame problem*. The frame problem causes the symbolic representation of transitions grow to an unmanageable size. To cope with this

problem, we will introduce an abbreviation for a formula which expresses that the values of some variables do not change.

Let \mathbb{S} be a transition system and $Y \subseteq \mathcal{X}$ be a set of variables of $\mathcal{L}(\mathbb{S})$. Define the formula $only(Y)$ by

$$only(Y) \stackrel{\text{def}}{=} \bigwedge_{x \in \mathcal{X} \setminus Y} x = x'.$$

In other words, $only(Y)$ means that the variables in Y are the only ones that may change. Instead of $only(\{x_1, \dots, x_n\})$ we will write $only(x_1, \dots, x_n)$. Using the new notation, formula (13.4) which gives us the symbolic representation of the *Recharge* transition, can be written in a more concise way as

$$\text{customer} = \text{none} \wedge \text{st_coffee}' \wedge \text{st_beer}' \wedge only(\text{st_coffee}, \text{st_beer}, \text{coins}). \quad (13.5)$$

Note that $only(Y)$ does not mean that the variables in Y *must* change their values after the transition. For example, formula (13.5) does not imply $\neg(\text{st_coffee} \leftrightarrow \text{st_coffee}')$.

Using the frame notation, we can represent all transitions of the vending machine system as shown in Figure 13.2.

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:

- (1) F_1 expresses some conditions on the variables \mathcal{X} which are necessary to execute the transition;
- (2) 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.

The formula F_1 is sometimes called the *precondition* of the transition, and F_2 called the *postcondition* of it. For example, in the formula that represents the transition *Take_drink* in Figure 13.2, the precondition is

$$\text{customer} \neq \text{none} \wedge \text{disp} \neq \text{none},$$

while the postcondition is

$$\text{disp}' = \text{none} \wedge only(\text{disp}).$$

Note that the notions of precondition and postcondition are rather informal: indeed, they refer not to the transition itself, but to its particular representation by a formula.

Observe the following property of the symbolic presentation of transitions of Figure 13.2. In each of these transitions apart from *Dispense_beer* and *Dispense_coffee* the variables

$$\begin{array}{l}
\textit{Recharge} \stackrel{\text{def}}{=} \text{customer} = \textit{none} \wedge \text{st_coffee}' \wedge \text{st_beer}' \wedge \\
\textit{Customer_arrives} \stackrel{\text{def}}{=} \text{customer} = \textit{none} \wedge \text{customer}' \neq \textit{none} \wedge \\
\textit{Customer_leaves} \stackrel{\text{def}}{=} \text{customer} \neq \textit{none} \wedge \text{customer}' = \textit{none} \wedge \\
\textit{Coin_insert} \stackrel{\text{def}}{=} \text{customer} \neq \textit{none} \wedge \text{coins} \neq 3 \wedge \\
\textit{Dispense_beer} \stackrel{\text{def}}{=} \text{customer} = \textit{student} \wedge \text{st_beer} \wedge \\
\textit{Dispense_coffee} \stackrel{\text{def}}{=} \text{customer} = \textit{prof} \wedge \text{st_coffee} \wedge \\
\textit{Take_drink} \stackrel{\text{def}}{=} \text{customer} \neq \textit{none} \wedge \text{disp} \neq \textit{none} \wedge
\end{array}$$

$$\begin{array}{l}
\text{only}(\text{st_coffee}, \text{st_beer}). \\
\text{only}(\text{customer}) \\
\text{only}(\text{customer}). \\
(\text{coins} = 0 \rightarrow \text{coins}' = 1) \wedge (\text{coins} = 1 \rightarrow \text{coins}' = 2) \wedge \\
(\text{coins} = 2 \rightarrow \text{coins}' = 3) \wedge \\
\text{only}(\text{coins}). \\
\text{disp} = \textit{none} \wedge (\text{coins} = 2 \vee \text{coins} = 3) \wedge \\
\text{disp}' = \textit{beer} \wedge \\
(\text{coins} = 2 \rightarrow \text{coins}' = 0) \wedge (\text{coins} = 3 \rightarrow \text{coins}' = 1) \wedge \\
\text{only}(\text{st_beer}, \text{disp}, \text{coins}). \\
\text{disp} = \textit{none} \wedge \text{coins} \neq 0 \wedge \\
\text{disp}' = \textit{coffee} \wedge \\
(\text{coins} = 1 \rightarrow \text{coins}' = 0) \wedge (\text{coins} = 2 \rightarrow \text{coins}' = 1) \wedge \\
(\text{coins} = 3 \rightarrow \text{coins}' = 2) \wedge \\
\text{only}(\text{st_coffee}, \text{disp}, \text{coins}). \\
\text{disp}' = \textit{none} \wedge \\
\text{only}(\text{disp}).
\end{array}$$

Figure 13.2: Symbolic representation of transitions of the vending machine transition system

shown in the frame formula are exactly those variables x such that x' occurs in the transition. For this reason, symbolic representations of transitions in many model-checkers do not use the frame formulas at all. Instead, they use a convention that, whenever a variable x' does not occur in the formula which represents the transition, it is assumed that the formula implicitly contains $x = x'$. If we accept this convention, then the frame formula can be removed from all transitions but *Dispense_beer* and *Dispense_coffee*. We have to be careful about transitions in which a variable x can change its value to an arbitrary value in its domain. For such transitions we can add to the transition formula any valid formula containing x' , for example, $x' = x'$. For example, under this convention the transition *Dispense_beer* can be represented by

$$\begin{aligned}
\text{Dispense_beer} &\stackrel{\text{def}}{=} \text{customer} = \text{student} \wedge \text{st_beer} \wedge \\
&\text{disp} = \text{none} \wedge (\text{coins} = 2 \vee \text{coins} = 3) \wedge \\
&\text{disp}' = \text{beer} \wedge \\
&(\text{coins} = 2 \rightarrow \text{coins}' = 0) \wedge (\text{coins} = 3 \rightarrow \text{coins}' = 1) \wedge \\
&(\text{st_beer}' \leftrightarrow \text{st_beer}').
\end{aligned}$$

13.5 Temporal Properties of Transition Systems

We will call properties of transition systems referring to their functioning in time *temporal properties*. We are interested in expressing and verifying temporal properties of transition systems. Consider some typical examples of properties one might be interested in for the vending machine example (we do not claim that all these properties hold for this system).

- (1) The machine always contains some drink.
- (2) Students sometimes drink coffee.
- (3) The machine cannot dispense drinks forever without recharging.

The first of these properties refers to a single state, and for a particular state can be expressed by $\text{st_beer} \vee \text{st_coffee}$. However, we have no formalism that can express that this property holds for *all states* through which the machine goes.

As for the second property, we have no variable which expresses who drinks what, but we can represent this property by referring to two consecutive states. We can say that a student drinks coffee, if there exists a future state s in which the formula

$$\text{customer} = \text{student} \wedge \text{disp} = \text{coffee}$$

holds, and a state s' *immediately after* s in which $\text{disp} = \text{none}$ holds. To represent properties of two consecutive states, we can use formulas of variables $\mathcal{X} \cup \mathcal{X}'$. But again, we have no way to express that two consecutive states with this property will occur *in some future state*.

The third property is even more complicated. It refers neither to a particular state in future nor to two consecutive states. Essentially, it means that the system cannot execute an *infinite number of transitions* *Dispense_beer* and *Dispense_coffee* without a *Recharge* in between.

The logic we introduced in this section allows one to specify sets of states and transitions, and therefore specify transition systems in a symbolic form. However, it does not allow us to specify complex temporal properties of the system or reason about them. To reason about temporal properties of transition systems, we need a more expressive logic, which can express properties like “in some future state”, “in all future states”, “forever”, or even “infinitely often”. Such a logic will be introduced in the next chapter.

Exercises

EXERCISE 13.1 This exercise is about the symbolic representation of set of states for the vending machine system.

- (1) Represent the set of initial states.
- (2) We know that students only drink beer while professors only drink coffee. Represent the set of states in which there is a drink in the dispenser but it does not suit the current customer.
- (3) Represent the set of states in which there is only one drink available.

EXERCISE 13.2 Represent symbolically the following transitions for the vending machine system.

- (1) The money-swallowing transition which can remove any amount of coins from the coin slot.
- (2) A student leaves taking all drinks from the dispenser and all money.
- (3) Change-giving transition (it should remove *all* money from the coin slot).

EXERCISE 13.3 Explain why the logics $\mathcal{L}(\mathbb{S})$ and $\mathcal{L}'(\mathbb{S})$ are not expressive enough for reasoning about infinite-state systems. Hint: consider an integer-valued variable x and try to express the property that x is even.

EXERCISE 13.4 Let \mathbb{S} be a transition system with the state variables v_1, \dots, v_n . Define the *idle transition* as the transition $\{(s, s) \mid s \text{ is a state}\}$, i.e., the transition which is defined on all states, but does not change the value of any variable. Define a symbolic representation of the idle transition.

EXERCISE 13.5 For any transition system \mathbb{S} , write a symbolic representation of the greatest transition: the transition which can change the value of any state variable w to any value in $\text{dom}(w)$.

EXERCISE 13.6 Change symbolic representations transitions of Figure 13.2 as follows.

- (1) Change the transition *Customer_leaves* so that the customer cannot leave without a drink.
- (2) Change the transition *Customer_leaves* so that a student always removes all money from the coin slot.
- (3) Change the transition *Take_drink* so that the customer may leave after taking a drink.

EXERCISE 13.7 Which of the transitions of Figure 13.2 are non-deterministic?

EXERCISE 13.8 Two variables x, y range over the same domain. Represent the transition which swaps the values of x and y .

EXERCISE 13.9 The variable x range over the domain $\{1, 2, 3\}$. Represent the transition which strictly increases the value of x .

EXERCISE 13.10 The state variables of a system are x_1, \dots, x_n . Represent the transition that changes the value of *every* state variable of the system. Is this transition deterministic?

EXERCISE 13.11 What is the number of all possible transitions for the vending machine system?

EXERCISE 13.12 Find a symbolic representation of a transition not applicable to any state.