Chapter 12

Propositional Logic of Finite Domains

A googolplex is much larger than a googol, but is still finite, as the inventor of the name was quick to point out. It was first suggested that a googolplex should be 1, followed by writing zeros until you got tired. This is a description of what would happen if one actually tried to write a googolplex, but different people get tired at different times and it would never do to have Carnera a better mathematician than Dr Einstein, simply because he had more endurance. The googolplex then, is a specific finite number, with so many zeros after the 1 that the number of zeros is a googol. A googolplex is much bigger than a googol, much bigger even than a googol times a googol. A googol times a googol would be 1 with 200 zeros, whereas a googolplex is 1 with a googol of zeros. You will get some idea of the size of this very large but finite number from the fact that there would not be enough room to write it, if you went to the farthest star, touring all the nebulae and putting down zeros every inch of the way.
Edward Kasner, James R. Newman. Mathematics and the Imagination. Penguin. 1940

Contents

In this chapter we introduce *propositional logic of finite domains* (PLFD). This logic is convenient for specifying properties of systems whose state can be described by values of variables, so that every variable has a finite set of possible values. Propositional logic can be regarded as a special case of PLFD in which every variable ranges over the set of boolean values.

In Section 12.1 we define the syntax and semantics of PLFD. The only difference between the syntax of PLFD and propositional logic is the definition of atomic formulas. To show the similarity of PLFD and propositional logic, in Section 12.2 we give model-preserving translations between formulas of PLFD and propositional logic. Using this translation one can, in principle, work with PLFD using propositional logic. For example, to find a model of a PLFD formula, one can translate it into a propositional formula and search for models of the translated formula using any propositional logic method, such as DPLL. However, one can also modify these methods to work directly with PLFD. We illustrate this by defining a tableau system for PLFD in Section 12.3.

12.1 Syntax and Semantics

12.1.1 Motivation

Consider a microwave oven (I will actually use the one in my kitchen as an example). It has several *modes of operation*, such as micro power, convection, grill, combination, defrost etc., a *door* that can open or closed, a *temperature display*, and a *timer*. It has several buttons that can be used to control the operation of the microwave. It may be turned on or off. More complex models may include more advanced features.

If you are a designer of such a system as this microwave oven, you should write a program controlling its functioning. Your controlling program should satisfy some important conditions, for example, if the door is open, then the mode should be idle. These conditions may be related to *safety* of the device but also to other properties. To express these conditions, we can use logic.

If we try to use propositional logic for modeling the microwave, we will soon find out that it is not very natural. Consider, for example, the mode of operation. It has several possible values (grill, idle etc.), so we should be able to express properties such as that the mode is idle. If we introduce a boolean variable $idle$ to express that the mode is idle and a boolean variable $grill$ to express that the mode is grill, then these variables will turn out not to be independent: indeed, in our model $idle$ and $grill$ cannot be true at the same time. But propositional logic has interpretations in which both $idle$ and $grill$ are true, so we have to add axioms like $\neg idle \vee \neg grill$ to model the microwave adequately.

It seems more natural to have a logic in which we can express properties of the microwave more directly than by introducing many boolean variables to express every possible value for the mode. Of course we do not want to have exactly the microwave logic, we

variable	domain of values
mode	$\{idle, micro, grill, defrost\}$
door	$\{open, closed\}$
content	$\{none, burger, pizza, cabbage\}$
user	$\{nobody, student, veggie, mcdonald\}$
temperature	$\{0,\ 150,\ 160,\ 170,\ 180,\ 190,\ 200,\ 210,\ 220,\ 230,\ 240,\ 250\}$

Figure 12.1: Variables and Domains

want to be able to describe formally various devices, from very simple ones like a vending machine, to very complex ones like a Mars rover. What is common in such devices is that their state can be characterized by values of a set of *variables*. For example, mode can be considered a variable and $idle$ a possible value for this variable. Another variable for the microwave example is door whose possible values are $open$ and $closed$.

Propositional logic of finite domains, or simply PLFD, is a logic in which atomic formulas express properties of the form "a variable x has a value v", written in the form $\mathsf{x} = v$. For example, we can write $\mathsf{door} = open \rightarrow \mathsf{mode} = idle$ to express the safety policy that the microwave should not be functioning when the door is open.

The set of possible variables and values depend on what kind of property we would like to express. For example, if we are mostly interested in how a microwave is functioning in a departmental tea room, we may introduce variables denoting users, queue, and content of the microwave.

In any case, before expressing properties of the microwave, we should fix a collection of variables describing it and possible values for each variable. For example, Figure 12.1 shows a set of variables representing both the microwave, its users and the content.

12.1.2 Syntax

Propositional logic of finite domains, or simply *PLFD* is, in fact, a *family of logics*. Each instance of this family uses a finite set $\mathcal{X}$ of *variables* and a finite set of *domains*. Each domain is a finite non-empty set. Elements of domains are called *values*. Each variable $x \in \mathcal{X}$ has an associated domain, denoted by $dom(x)$. We will say that the variable x *ranges over* the domain $dom(x)$, or that $dom(x)$ is the *domain for* x.

For example, we can regard the table of Figure 12.1 as representing an instance of PLFD. The variables of this logic are shown in the left column of the table, while the domain for each variable is shown in the right column.

DEFINITION 12.1 (Formula) *Formulas* are defined inductively as follows.

(1) If x is a variable and $v \in dom(x)$ is a value in the domain for x, then $x = v$ is a formula, also called *atomic formula*, or simply *atom*.

(2) Other formulas are build from atomic formulas as in propositional logic, see Definition 3.1, using the connectives $\top$, $\bot$, $\wedge$, $\vee$, $\neg$, $\rightarrow$, and $\leftrightarrow$. ❑

For example, assuming variables and domains as in Figure 12.1, the following is a formula of PLFD:

$$\mathsf{mode} = \mathit{grill} \rightarrow \mathsf{door} = \mathit{closed} \wedge \neg\mathsf{temperature} = 0 \wedge \neg\mathsf{user} = \mathit{nobody}.$$

The expression $\mathsf{user} = \mathit{none}$ is *not* a formula, since none is not a value in the domain for user. When we would like to be specific about which instance of PLFD we are dealing with, we will write $PLFD(\mathcal{X}, dom)$ to refer to the instance with the set of variables $\mathcal{X}$ and domain function dom.

12.1.3 Semantics

The semantics of PLFD is defined similar to that of propositional logic. We define a suitable notion of interpretation and the meaning of atomic formulas in an interpretation. The meaning of arbitrary formulas can then be defined from the meaning of atomic ones in exactly the same way as in propositional logic.

DEFINITION 12.2 (Interpretation, Truth) An *interpretation* for a set of variables X is a mapping I from X to the set of values such that for all $x \in X$ we have $I(x) \in dom(x)$.

We will now extend interpretations to mappings from formulas to boolean values. The definition is by induction.

(1) $I(x = v) = 1$ if and only if $I(x) = v$.

(2) If F is not atomic, then $I(F)$ is defined as in Definition 3.2 for propositional formulas.

The definitions of *truth*, *models*, *validity*, *satisfiability*, and *equivalence* are defined exactly as in propositional logic. ❑

For example, given variables and their domains as in Figure 12.1, the following is an interpretation:

$$\{\mathsf{mode} \mapsto \mathit{micro}, \mathsf{door} \mapsto \mathit{open}, \mathsf{content} \mapsto \mathit{burger}, \\ \mathsf{user} \mapsto \mathit{veggie}, \mathsf{temperature} \mapsto 0\}.$$

The formula $\neg\mathsf{mode} = \mathit{idle} \rightarrow \mathsf{door} = \mathit{closed}$ is false in this interpretation.

Note that the definitions of the syntax and semantics for PLFD are *parametrized* by the domain function dom. That is, different domain functions give us different instances of PLFD. Some of the properties of PLFD essentially depend on the concrete instance. For example, the formula $\mathsf{door} = \mathit{open} \vee \mathsf{door} = \mathit{closed}$ is valid in any logic in which $dom(\mathsf{door}) = \{\mathit{open}, \mathit{closed}\}$ but not valid if the domain for door contains more than two values.

12.2 PLFD and Propositional Logic

In this section show that propositional logic is, in fact, an instance of PLFD. We also give a model-preserving translation of PLFD into propositional logic.

12.2.1 Propositional Logic as an Instance of PLFD

Consider an instance of PLFD in which every variable p_i ranges over the domain $\{0, 1\}$ of boolean values. Note that propositional logic and this instance of PLFD have the same set of interpretations: interpretations in both logics map variables to boolean values. In this section we will give a *model-preserving translation* of propositional logic to PLFD. This translation preserves models in the following sense: it translates any formula F of propositional logic into a formula F' of PLFD such that for every interpretation I, we have $I \models F$ in propositional logic if and only if $I \models F'$ in PLFD.

This translation will be defined as a *mapping* $plfd$ from formulas of propositional logic to formulas in this instance of PLFD as follows. For every propositional formula F, the formula $plfd(F)$ is obtained from F by replacing every propositional atomic formula p by the PLFD atomic formula $p = 1$. For example, the formula $plfd(p \rightarrow \neg q)$ is $(p = 1) \rightarrow \neg(q = 1)$

THEOREM 12.3 *Formulas F and $plfd(F)$ have the same models.*

PROOF. We prove it in the case when F is an atomic formula p, the proof for arbitrary formulas carries over by straightforward induction. Let I be a mapping from variables to boolean values. In propositional logic we have $I \models p$ if and only if $I(p) = 1$. But in PLFD we have $I \models (p = 1)$ if and only if $I(p) = 1$. Therefore, $I \models p$ in propositional logic if and only if $I \models (p = 1)$ in PLFD. ❑

12.2.2 Translation of PLFD to Propositional Logic

If we take an arbitrary instance of PLFD which has a non-boolean variable, then the set of interpretations for this instance is different from the set of interpretations for propositional logic with the same set of variables. In this case there is no model-preserving translation between the two logics. Nonetheless, one can define a translation that preserves a suitable mapping between models and use this translation for finding models and checking satisfiability and other properties of PLFD formulas.

The idea of translation is to introduce boolean variables denoting atomic formulas $x = v$. To this end, for every variable x and value $v \in dom(x)$ we use a boolean variable x_v. After that, given a PLFD formula F we can replace every atom $x = v$ in F by the propositional atom x_v, obtaining a propositional formula F'. This transformation does not directly preserve satisfiability. For example, for the microwave example the formula $\mathsf{mode} = idle \wedge \mathsf{mode} = micro$ is unsatisfiable, while the corresponding propositional formula $\mathsf{mode}_{idle} \wedge \mathsf{mode}_{micro}$ is satisfiable. The problem is that the propositional formula has

a model $\{\mathsf{mode}_{idle} \mapsto 1, \mathsf{mode}_{micro} \mapsto 1\}$ that does not correspond to any PLFD interpretation: indeed, in every such interpretation the variable mode has exactly one value.

To provide a meaningful translation, one has to add additional propositional formulas which prevent unintended interpretations from becoming models. These formulas axiomatize properties of domains for variables, for example one can add the formula $\neg\mathsf{mode}_{idle} \vee \neg\mathsf{mode}_{micro}$ to exclude models in which both mode_{idle} and mode_{micro} are true.

DEFINITION 12.4 (Domain Axiom) Let x be a variable and $dom(x) = \{v_1, \ldots, v_n\}$. Then the *domain axiom* for x is the propositional formula

$$(x_{v_1} \vee \ldots \vee x_{v_n}) \wedge \bigwedge_{i<j} (\neg x_{v_i} \vee \neg x_{v_j}). \qquad \text{❑}$$

For instance, in the microwave example the domain axiom for the variable mode is

$$\begin{array}{l}(\mathsf{mode}_{idle} \vee \mathsf{mode}_{micro} \vee \mathsf{mode}_{grill} \vee \mathsf{mode}_{defrost}) \wedge \\ (\neg\mathsf{mode}_{idle} \vee \neg\mathsf{mode}_{micro}) \wedge \\ (\neg\mathsf{mode}_{idle} \vee \neg\mathsf{mode}_{grill}) \wedge \\ (\neg\mathsf{mode}_{idle} \vee \neg\mathsf{mode}_{defrost}) \wedge \\ (\neg\mathsf{mode}_{micro} \vee \neg\mathsf{mode}_{grill}) \wedge \\ (\neg\mathsf{mode}_{micro} \vee \neg\mathsf{mode}_{defrost}) \wedge \\ (\neg\mathsf{mode}_{grill} \vee \neg\mathsf{mode}_{defrost}).\end{array}$$

The domain axiom for the variable temperature is considerably longer and contains 144 occurrences of variables. The following lemma shows that the domain axiom indeed eliminates the unintended models.

LEMMA 12.5 *Let $dom(x) = \{v_1, \ldots, v_n\}$. Let F be the domain axiom for x and I be an interpretation defined on the set of boolean variables $X = \{x_{v_1}, \ldots, x_{v_n}\}$. Then $I \models F$ if and only if exactly one of the boolean variables in X is satisfied in I.*

PROOF. It is easy to see that the models of $x_{v_1} \vee \ldots \vee x_{v_n}$ are the interpretations satisfying *at least* one variable in X. The models of $\neg x_{v_i} \vee \neg x_{v_j}$ are the interpretations satisfying *at most* one variable among $x_{v_i}, \neg x_{v_j}$. Therefore, the models of the domain axiom are the interpretations satisfying *exactly* one variable in X. ❑

Now we will define formally in what respect an interpretation for PLFD corresponds to a model of the domain axioms for this instance of PLFD. To this end we introduce a mapping from interpretations for PLFD to propositional interpretations, denoted by $prop$. Given an interpretation I for PLFD, define $prop(I)$ as follows:

$$prop(I)(x_v) = 1 \stackrel{\text{def}}{=} I(x) = v.$$

In other words, x_v is true in $prop(I)$ if and only if the value of x in I is v. We also define a mapping from formulas of PLFD to propositional formulas, also denoted by $prop$, as

follows. Let F be a formula of PLFD with variables $x_1, \ldots, x_n$ and $F_1, \ldots, F_n$ be the domain axioms for $x_1, \ldots, x_n$. Let a propositional formula F' be obtained from F by replacing every atom of the form $x = v$ by the propositional atom x_v. Then $prop(F)$ is defined to be the formula $F_1 \wedge \ldots \wedge F_n \wedge F'$.

THEOREM 12.6 *Let F be a formula of PLFD. Then*

(*1*) *For every interpretation I, $I \models F$ if and only if $prop(I) \models prop(F)$.*

(*2*) *Each model of $prop(F)$ has the form $prop(I)$ for some interpretation I of PLFD.*

PROOF. The first property is proved by induction on F. When F is an atomic formula $x = v$, the claim is immediate by the definition of $prop(I)$. For non-atomic formulas the induction arguments are straightforward.

Let us prove the second property. Let I' be a model of $prop(F)$. Then it is also a model of the domain axioms for all variables of the logic. Define an interpretation I of PLFD as follows. For every variable x, let $I(x) \stackrel{\text{def}}{=} v$, where v is such that such that $I' \models x_v$. By Lemma 12.5, for every variable x the interpretation I' satisfies exactly one atom of the form x_v, so I is well-defined. It is not hard to argue that $prop(I) = I'$. ❑

The translations from propositional logic to PLFD and back allow to shift forth and back between the two logics. PLFD is convenient for *specifying* or *modeling* systems, while propositional logic is simpler for defining reasoning algorithms. Therefore, in the future we will sometimes use PLFD for modeling systems but switch to propositional logic when it comes to reasoning about their properties.

12.3 A Tableau System for PLFD

We are interested in checking satisfiability, validity, and equivalence for PLFD formulas. This can be done by defining a suitable proof system for PLFD. In this section we will define a tableau system for this logic which can obtained by extending the tableau proof system for propositional logic to PLFD. To define the system, we will first introduce a different kind of atomic formulas.

Let x be a variable and $D \subseteq dom(x)$. Then we consider expressions $x \in D$ as atomic formulas with the following straightforward semantics: for an interpretation I we have $I \models x \in D$ if and only $I(x) \in D$. It is not hard to argue that any formula $x = v$ is equivalent to the formula $x \in \{v\}$, and any formula $x \in \{v_1, \ldots, v_n\}$ is equivalent to $x = v_1 \vee \ldots \vee x = v_n$, so the logics with atoms of the form $x = a$ and atoms of the form $x \in D$ have the same expressive power.

The *tableau system for PLFD* operates on signed formulas and uses the same branch expansion rules as in the case of propositional logic, see Figure 9.2 on page 120. In addition to these rules, it also uses several rules specific for the treatment of atomic formulas $x \in D$. Before defining the new rules, let us use the following convention: a signed formula $(x \in$

$$\begin{array}{rcl} x \notin D & \rightsquigarrow & x \in dom(x) \setminus D \\ x \in D_1, x \in D_2 & \rightsquigarrow & x \in D_1 \cap D_2 \end{array}$$

Figure 12.2: Additional tableau rules for PLFD

$D) = 1$ will simply be denoted by $x \in D$, likewise, a signed formula $(x \in D) = 0$ will be denoted by $x \notin D$.

The additional tableau rules are given in Figure 12.2. Note that the second rule requires two formulas $x \in D_1$ and $x \in D_2$ to be on the same branch.

A branch in a tableau is called *closed* if it contains a signed formula of one of the following kinds: $\top = 0$, $\bot = 1$, or $x \in \{\}$.

EXAMPLE 12.7 Let us show that the following formula F is valid in the PLFD with variables and domains as in Figure 12.1:

$$\begin{array}{l} ((\mathsf{user} = mcdonald \rightarrow \mathsf{content} = none \vee \mathsf{content} = burger) \wedge \\ (\mathsf{user} = veggie \rightarrow \mathsf{content} = none \vee \mathsf{content} = cabbage) \wedge \\ (\mathsf{user} = nobody \rightarrow \mathsf{content} = none)) \rightarrow \\ (\mathsf{content} = pizza \rightarrow \mathsf{user} = student). \end{array}$$

To this end we show that the signed formula $F = 0$ is unsatisfiable. To apply the tableau algorithm for this signed formula we replace all atoms of the form $x = v$ by atoms $x \in \{v\}$ and then try to build a tableau for it. A part of a semantic tableau for this signed formula is shown in Figure 12.3. All branches of the full tableau are closed, hence the signed formula $F = 0$ is unsatisfiable, so F is valid. ❑

THEOREM 12.8 (Soundness and Completeness) *Let T be a tableau resulting from a terminated game for a signed formula γ. Then γ is satisfiable if and only if all branches in T are closed.*

PROOF. We will use the arguments used in the proof of Theorem 9.17. Namely, we prove that every tableau inference is invertible and that a tableau to which no rule is applicable is satisfiable if and only if it is non-empty.

Invertibility of the tableau rules used in propositional logic is established in Lemma 9.16, so it is enough to establish invertibility of the rules of Figure 12.2 and the branch closure rules. Invertibility of all of these rules is obvious, for example, it is obvious that $I \vDash x \in D_1$ and $I \vDash x \in D_2$ if and only if $I \vDash x \in D_1 \cap D_2$.

It remains to prove that a non-empty tableau to which no rule is applicable is satisfiable. Take any such tableau and consider any branch B in it. It is not hard to argue that the branch has the form

$$\{x_1 \in D_1, \ldots, x_n \in D_n\},$$

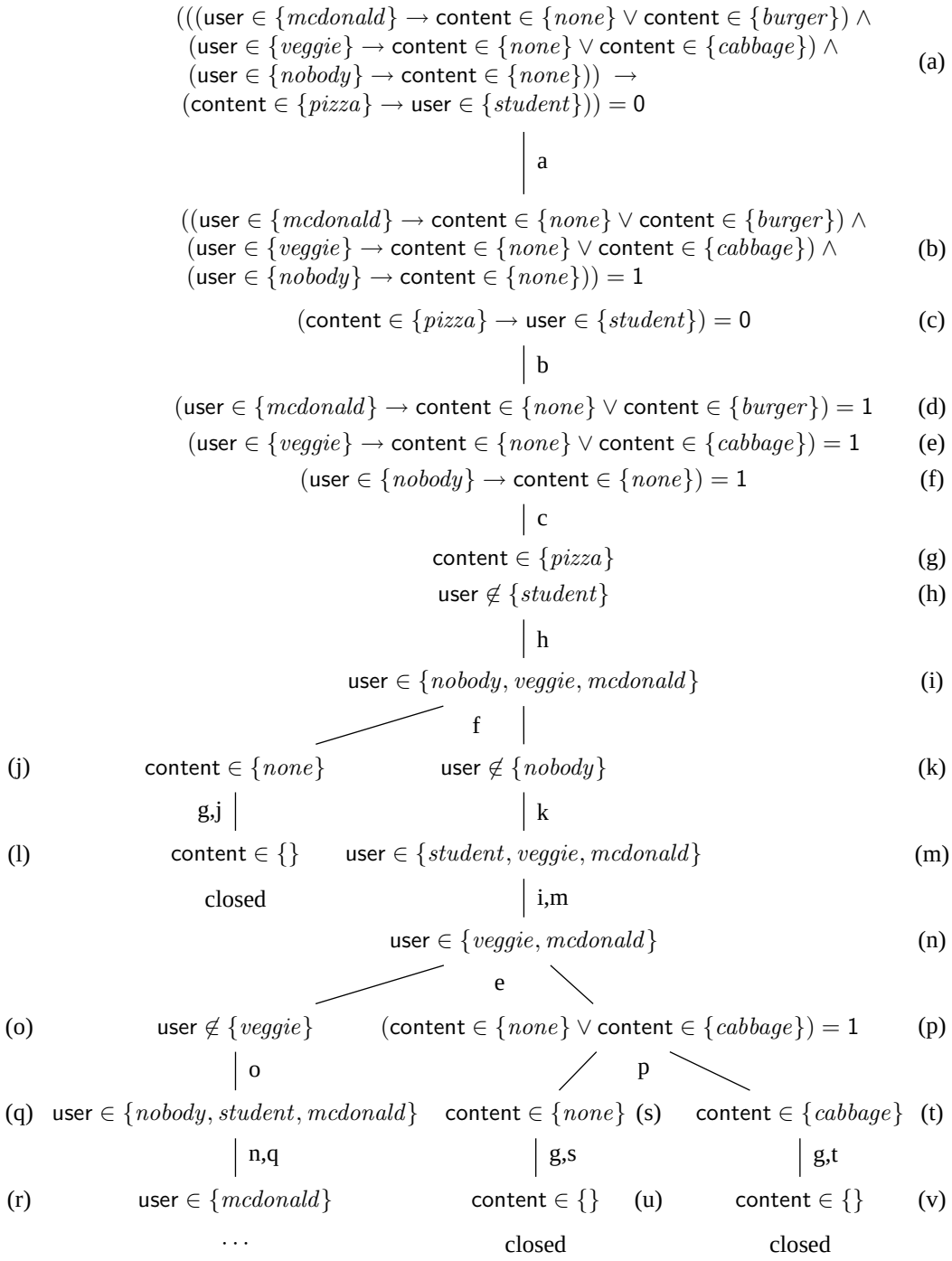

Figure 12.3: Part of a semantic tableau for the formula of Example 12.7.

where $x_1, \ldots, x_n$ are distinct variables and $D_1, \ldots, D_n$ are non-empty sets, moreover $D_i \in dom(x_i)$ for all i. Since all D_i are non-empty, there exists an interpretation I such that $I(x_i) = v_i$ for all i. It is not hard to argue that I is a model of this branch. ❑

The proof of this theorem shows how one can extract models from an open branch of a tableau. For every variable x, if the branch contains a signed formula $x \in D$ to which no rule has been applied, then we can assign to x any value in D. If the branch contains no such formula, then we can assign to x any value in its domain. We will illustrate this in Example 12.9 below.

12.4 Using Boolean Variables in PLFD

When some variable x is boolean, that is, has the domain $\{0, 1\}$ using the PLFD atomic formulas $x = 0$ and $x = 1$ for it may lead to unnecessarily complex formulas. To use boolean variables in PLFD one can adopt the following syntactic convention: if x is a boolean variable, we will use the propositional logic syntax for this variable. Namely, will use x as an atomic formula and write x instead of $x = 1$ and $\neg x$ instead of $x = 0$. For instance, if we change the variable door in the microwave example by a boolean variable door_open with an obvious meaning, then instead of

$$\mathsf{mode} = \mathit{grill} \rightarrow \mathsf{door\_open} = \mathit{false} \wedge \neg \mathsf{temperature} = \mathit{0} \wedge \neg \mathsf{user} = \mathit{nobody}.$$

we will write

$$\mathsf{mode} = \mathit{grill} \rightarrow \neg \mathsf{door\_open} \wedge \neg \mathsf{temperature} = \mathit{0} \wedge \neg \mathsf{user} = \mathit{nobody}.$$

This convention is especially convenient when there are many boolean variables and only few non-boolean ones, as it is often the case in applications. We can also modify the tableau system for PLFD to use boolean variables in the same way as they are used in the tableau system for propositional logic. That is, we use atomic formulas $x \in D$ for every non-boolean variable x and atomic formulas x for every boolean variable x. A tableau branch is closed in one of the following three cases:

(1) it contains the signed formula $\top = 0$ or $\bot = 1$;

(2) it contains the signed formula $x \in \{\}$ for a non-boolean variable x;

(3) it contains signed formulas $x = 0$ and $x = 1$ for a boolean variable x.

Let us demonstrate such a mixed tableau by a small but practical example.

EXAMPLE 12.9 There have been a lot of discussion whether a recent war was justified. A textbook on computational logic is hardly an appropriate place for expressing an opinion on this complex issue. However, we can investigate the arguments used for justification from a logical viewpoint. The question we ask is whether it is possible to start a war against a country that is not guilty. We will use the following variables:

(1) war: one can start a war;

(2) guilty: the country is guilty;

(3) has: the country has weapons of mass destruction.

As far as the author could understand, the following arguments were used for the justification. First, if a country has weapons of mass destruction, then it is guilty. Second, to start a war against the country one has to have a very good reason, possessing weapons of mass destruction is such a reason. This gives us two formulas:

$$\begin{aligned} &\mathsf{has} \rightarrow \mathsf{guilty}, \\ &\mathsf{war} \rightarrow \mathsf{has}. \end{aligned}$$

If we would like to check whether, under the above assumptions, it is possible that a war started against a country that is not guilty, then the answer in "no". Indeed, the set of formulas

$$\begin{aligned} &\mathsf{has} \rightarrow \mathsf{guilty}, \\ &\mathsf{war} \rightarrow \mathsf{has}, \\ &\mathsf{war}, \\ &\neg\mathsf{guilty} \end{aligned}$$

is unsatisfiable.

Let us consider a slightly different situation, when the domain for the variable has consists of the values yes, no, and a third value, for example, $suspected$. We can reformulate our assumptions in the following way

$$\begin{aligned} &\mathsf{has} = yes \rightarrow \mathsf{guilty}, \\ &\mathsf{war} \rightarrow \neg(\mathsf{has} = no). \end{aligned}$$

In other words, if a country has weapons of mass destruction, then it is guilty, and to start a war we have to be sure that it is impossible that the country has no weapons of mass destruction.

If we ask now whether it is a possible that a war started against a country that is not guilty, then the answer is "yes". Indeed, the set of formulas

$$\begin{aligned} &\mathsf{has} = yes \rightarrow \mathsf{guilty}, \\ &\mathsf{war} \rightarrow \neg(\mathsf{has} = no), \\ &\mathsf{war}, \\ &\neg\mathsf{guilty} \end{aligned}$$

is satisfiable.

To find a model of this set, we build a semantic tableaux, see Figure 12.4. This tableau has an open branch, which gives us the only model of this set of formulas:

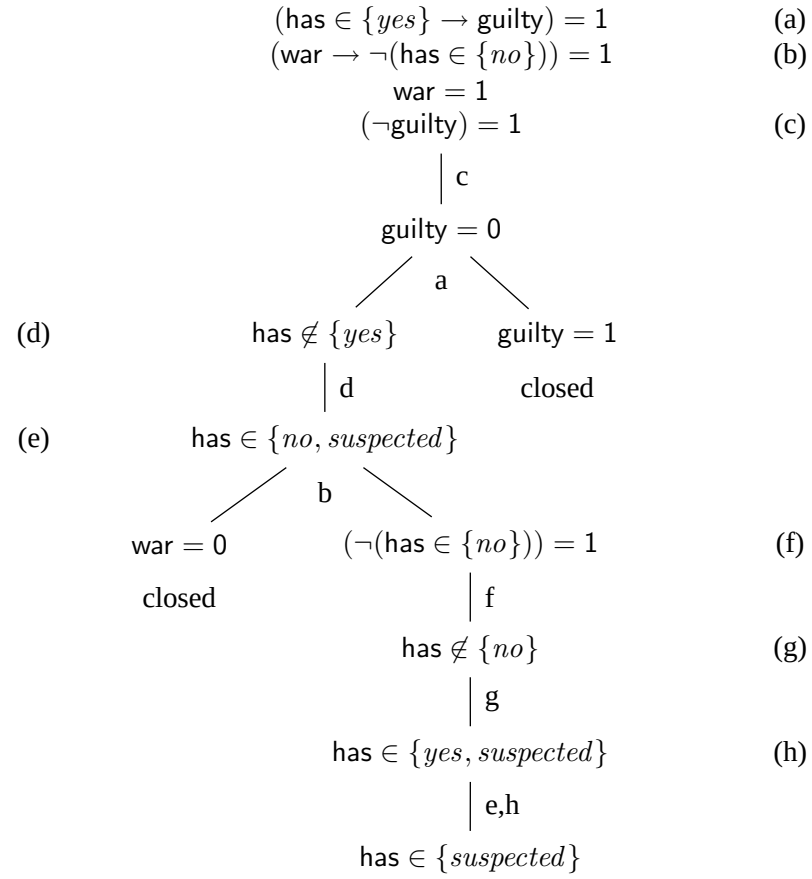

Figure 12.4: A semantic tableau for PLFD with boolean variables

$$\{\mathsf{war} \mapsto 1, \mathsf{guilty} \mapsto 0, \mathsf{has} \mapsto \mathit{suspected}\},$$

that is, it is possible that the war started, the country is not guilty but suspected to have weapons of mass destruction. ❑

Exercises

EXERCISE 12.1 Complete the tableau of Figure 12.3. ❑

EXERCISE 12.2 Write down the domain axiom for the variable user in the microwave example. ❑

EXERCISE 12.3 Is the domain axiom a formula in CNF or not? ❑

EXERCISE 12.4 What is the size (measured as the number of occurrences of variables) of the domain axiom for a variable whose domain contains 1000 values? ❑

EXERCISE 12.5 Take the domain axiom for a variable whose domain contains 1000 values and transform into CNF using the standard CNF transformation. What is the number of clauses in the resulting CNF? ❑

EXERCISE 12.6 Let m, n be positive integers and $m < n$. Express in propositional logic the following property: exactly m variables among $x_1, \ldots, x_n$ are true. ❑

EXERCISE 12.7 Show, using the tableau method, that the formula

$$\neg((\neg\mathsf{content} = burger \wedge \neg\mathsf{content} = pizza) \rightarrow \mathsf{content} = cabbage \vee \mathsf{content} = none)$$

is unsatisfiable in the logic for the microwave example. ❑

EXERCISE 12.8 Find, using the tableau method, a model of the formula

$$\neg((\neg\mathsf{content} = burger \wedge \neg\mathsf{content} = pizza) \rightarrow \mathsf{content} = none).$$ ❑

EXERCISE 12.9 What is the number of different models of the formula $\neg\mathsf{content} = none$ in the logic for the microwave example? ❑

EXERCISE 12.10 Consider a formula $x \in D$ used in semantic tableaux.

(1) For which D is $x \in D$ a tautology?

(2) For which D is $x \in D$ unsatisfiable? ❑

EXERCISE 12.11 Transform the propositional formula $p_1 \rightarrow (p_2 \wedge p_3)$ into PLFD. ❑

EXERCISE 12.12 Transform each of the following formulas of PLFD for the microwave example into propositional logic:

$$\neg\mathsf{mode} = defrost;$$
$$\mathsf{user} = nobody \rightarrow \mathsf{mode} = idle.$$ ❑

EXERCISE 12.13 Let x be a variable with the domain $\{u, v, w\}$ and p be a boolean variable. Transform the following formula of PLFD into a propositional formula:

$$\neg x = v \rightarrow x = u \wedge p = 0.$$ ❑

EXERCISE 12.14 Show that the formulas $\mathsf{door} = closed \rightarrow \mathsf{mode} = micro$ and $\neg\mathsf{mode} = idle \rightarrow \neg\mathsf{door} = closed$ are not equivalent in the PLFD for the microwave example by finding an interpretation in which they have different truth values. ❑

EXERCISE 12.15 Let x be a variable with the domain $\{a, b, c\}$ and p be a boolean variable. Show, using the tableau method, that the following formula is unsatisfiable.

$$\neg((p \rightarrow \neg x = a) \rightarrow x = b \vee x = c \vee \neg p).$$ ❑

EXERCISE 12.16 Take Exercise 12.15 but assume that the domain for x is $\{a, b, c, d\}$. Find, using the tableau method, a model of the formula of that exercise. ❑