CS:4350 Logic in Computer Science

Propositional Logic of Finite Domains

Cesare Tinelli

Spring 2021

The University of Iowa

Credits

These slides are largely based on slides originally developed by **Andrei Voronkov** at the University of Manchester. Adapted by permission.

Outline

Propositional Logic of Finite Domains

- Logic and modeling
- State-changing systems
- PLFD
- PLFD and propositional logic
- Tableau system for PLFD

Logic and Modeling

Satisfiability-checking in propositional logic has many applications

Unfortunately, there is a gap between real-life problems and their representation in propositional logic

Many application domains have special modeling languages for describing problems

because propositional logic is not convenient for modeling

However, in many cases, problems expressed in these languages can then be translated to propositional logic

Logic and Modeling

Satisfiability-checking in propositional logic has many applications

Unfortunately, there is a gap between real-life problems and their representation in propositional logic

Many application domains have special modeling languages for describing problems

because propositional logic is not convenient for modeling

However, in many cases, problems expressed in these languages can then be translated to propositional logic

Logic and Modeling

Satisfiability-checking in propositional logic has many applications

Unfortunately, there is a gap between real-life problems and their representation in propositional logic

Many application domains have special modeling languages for describing problems
because propositional logic is not convenient for modeling

However, in many cases, problems expressed in these languages can then be translated to propositional logic

Logic and Modeling

Satisfiability-checking in propositional logic has many applications

Unfortunately, there is a gap between real-life problems and their representation in propositional logic

Many application domains have special modeling languages for describing problems

because propositional logic is not convenient for modeling

However, in many cases, problems expressed in these languages can then be translated to propositional logic

Logic and Modeling

Satisfiability-checking in propositional logic has many applications

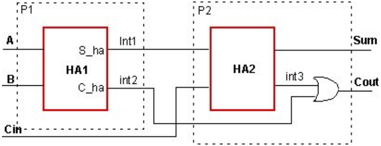
Unfortunately, there is a gap between real-life problems and their representation in propositional logic

Many application domains have special modeling languages for describing problems

because propositional logic is not convenient for modeling

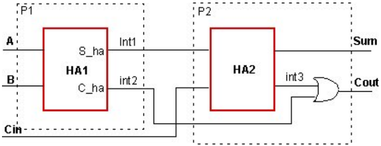
However, in many cases, problems expressed in these languages can then be translated to propositional logic

Circuit Design



Circuit: propositional logic

Circuit Design




```
library ieee;
use ieee.std_logic_1164.all;
entity FULL_ADDER is
  port (A, B, Cin :  in std_logic;
    Sum, Cout :  out std_logic);
end FULL_ADDER;
architecture BEHAV_FA of FULL_ADDER is
signal int1, int2, int3:  std_logic;
begin
P1:  process (A, B)
  begin
    int1<= A xor B;
    int2<= A and B;
  end process;
P2:  process (int1, int2, Cin)
  begin
    Sum <= int1 xor Cin;
    int3 <= int1 and Cin;
    Cout <= int2 or int3;
  end process;
end BEHAV_FA;
```

Circuit: propositional logic

Design: high-level description (VHDL)

Scheduling

All Second Year Timetable 2009-2010 — Level 2

Printable Timetable	Monday	Tuesday	Wednesday	Thursday	Friday
08:00	-	-	-	-	-
09:00	MATH20701 CRAW TH.1	COMP20051 1.1	GCOMP20340[B] G23 GCOMP20340[A] IT407 HCOMP20411[A] G23 BMAN20890 MBS EAST B8	FCOMP20411[A] G23 FCOMP20081[B] G23 HCOMP20010 UNIX BMAN10621 ROSCOE 1.007	ICOMP20340[B] UNIX ICOMP20340[A] IT407 FCOMP20051[A w3+] G23 GCOMP20411[A] UNIX HCOMP20081[B] G23
10:00	BMAN20880† SIMON B (B.41) COMP20340 1.1 MATH20701 Mans Coop G20	BMAN21061 CRAW TH.1 GCOMP20010 G23 FCOMP20241[w3+] Toot 1 BMAN10621 1.1	GCOMP20340[B] G23 GCOMP20340[A] IT407 HCOMP20411[A] G23	BMAN10621 ROSCOE 1.007 FCOMP20411[A] G23 FCOMP20081[B] G23 HCOMP20010 UNIX	MATH20701 RENO C016 ICOMP20340[B] UNIX ICOMP20340[A] IT407 FCOMP20051[A w3+] G23 GCOMP20411[A] UNIX HCOMP20081[B] G23
11:00	BMAN20871 MBS EAST B8 MATH29631 SACKVILLE F047 MATH10141 SIMON 3	BMAN21061 CRAW TH.1 GCOMP20010 G23 FCOMP20241[w3+] Toot 1 BMAN10621 1.1	COMP20081[A] 1.1 F+ICOMP20081[B] G23 MATH29631 RENO G002 BMAN10621 ROSCOE 1.008	GCOMP20051[A w3+] G23 ICOMP20010 UNIX EEEN20027 RENO C009 MATH20111 TURING G.107	HCOMP20340[B] UNIX HCOMP20340[A] IT407 ICOMP20081[B] G23 ICOMP20411[A] G23 FCOMP20241 LF15 MATH10141 RENO C016
12:00	BMAN21061 ROSCOE 1.008 EEEN20019 RENO C002 MATH20411 SCH BLACKETT	COMP-PASS LF15 MATH20411 TURING G.107	G+HCOMP20081[B] G23 MATH10141 RENO C016 MATH20701 SCH MOS	MATH20111 TURING G.207 GCOMP20051[A w3+] G23 ICOMP20010 UNIX	MATH20201 UNI PL B HCOMP20340[B] UNIX HCOMP20340[A] IT407 ICOMP20081[B] G23 ICOMP20411[A] G23
13:00	FCOMP20340[A] IT407 FCOMP20340[B] UNIX GCOMP20081[B] G23 ICOMP20051[A w3+] G23 MATH20411 TURING G.107	COMP20411 1.1	-	COMP20141 1.1 MATH20701 TURING G.107	EEEN20019 SSB A16
14:00	BMAN20880 SIMON 3 (3.40) EEEN20019 RENO C009 MATH20111 TURING G.207 FCOMP20340[A] IT407 FCOMP20340[B] UNIX GCOMP20081[B] G23 ICOMP20051[A w3+] G23	EEEN-LAB ? COMP20411 1.1	-	BMAN21061 CRAW TH.2 MATH20201 ROSC A	COMP20141 1.1 EEEN20019 SSB A16
15:00	HCOMP20051[A w3+] G23 FCOMP20010 UNIX BMAN20880 SIMON 3 (3.40)	2nd Yr Tutorial GCOMP20241[w3+] Toot 1 EEEN-LAB ?	-	COMP20051 1.1	COMP20010 1.1 MATH29631 SACKVILLE G037
16:00	MATH20201 RENO C016 HCOMP20051[A w3+] G23 FCOMP20010 UNIX	CARS20021 UNI PL B MATH20411 SCH BLACKETT GCOMP20241[w3+] Toot 1 EEEN-LAB ?	-	COMP20081 1.1 BMAN20890 CRAW TH.2 2nd Yr Tutorial	EEEN20027 RENO C009 MATH20111 ZOCHONIS TH.B (G.7)
17:00	-	CARS20021 UNI PL B	-	BMAN20890 CRAW TH.2	-
Notes	† BMAN20880 weeks 8,9 & 10				

Constraints on Solutions

Registration Week Timetables

Year 1

- All First Years
- All Single Hons (+CBA/IC) A+W+X+Y+Z
- All Single Hons (-CBA/IC) W+X+Y+Z
- Group A - (CBA + IC)
- Group B - (CSwBM: C+D)
- Group C - (CSwBM)
- Group D - (CSwBM)
- Group E - (CSE)
- Group M - (CM)
- Group W - (CS,SE,DC,AI)
- Group X - (CS,SE,DC,AI)
- Group Y - (CS,SE,DC,AI)
- Group Z - (CS,SE,DC,AI)
- Lab grouping A+Z
- Lab grouping C+X
- Lab grouping D+E+Y
- Lab grouping D+Y
- Lab grouping M+W
- Service Units
- Taking BMAN courseunits A+B

Year 2

- All Second Year
- Joint Hons (CM)
- Joint Hons (CSE)
- Joint Hons (CSwBM)
- Lab Group F
- Lab Group G
- Lab Group H
- Lab Group I
- Single Hons (CBA)
- Single Hons (CS, SE, DC, AI)

Year 3

- All Former SoI
- All Third Years
- Joint Hons (CM)
- Joint Hons (CSwBM)
- Single Hons (CBA)
- Single Hons (Computer Science)
- Single Hons (Internet Computing)
- Single Hons (Software Engineering - Informatics)

Room Timetables

UG Teaching Rooms

G33	24 seats
Advisory	? seats
LF5	9 seats
LF6	9 seats
LF15	70 seats
LF17	27 seats
IT406	24 seats
IT407	100 seats

PG Teaching Rooms

2.19	100 seats
2.15	40 seats

UG Labs

Toot 1	40 seats
Toot 0	28 seats
Collab 2	4 Pods seats
Collab 1	8 Pods seats
PEVELab	? seats
G23	65 seats
3rdLab	61 seats
UNIX	70 seats

[All labs]

Meeting Rooms

1.20	? seats
2.33	15 seats
Atlas 1	28 seats
Atlas 2	22 seats
IT401	24 seats
Mercury	24 seats

1. Rooms should have enough seats
2. Instructors cannot teach two courses at the same time
3. Prof. Nightowl cannot teach at 9am
4. …

State-changing systems

Our main interest from now on is modeling *state-changing systems*

We assume a discrete notion of time, with each time corresponding to a *step* taken by the system

Informally	
At each step, the system is in a particular state	
The system state changes over time There are actions (controlled or not) that change the state	

State-changing systems

Our main interest from now on is modeling *state-changing systems*

We assume a discrete notion of time, with each time corresponding to a *step* taken by the system

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

State-changing systems

Our main interest from now on is modeling *state-changing systems*

We assume a discrete notion of time, with each time corresponding to a *step* taken by the system

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

Computational systems are state-changing systems

Reactive systems:
systems that maintain an ongoing interaction with their environment rather than produce some final value upon termination

Examples: air traffic control system, controllers in mechanical devices (microwaves, traffic lights, trains, planes, . . .)

Concurrent systems
Systems executing simultaneously, and potentially interacting with each other.

Examples: operating systems, networks, . . .

Computational systems are state-changing systems

Reactive systems:
systems that maintain an ongoing interaction with their environment rather than produce some final value upon termination

Examples: air traffic control system, controllers in mechanical devices (microwaves, traffic lights, trains, planes, ...)

Concurrent systems
Systems executing simultaneously, and potentially interacting with each other.

Examples: operating systems, networks, ...

Reasoning about state-changing systems

1. Build a formal model the state-changing system which describes, in particular, its temporal behavior or some abstraction of it

2. Use a logic to specify and verify properties of the system

Reasoning about state-changing systems

1. Build a formal model the state-changing system which describes, in particular, its temporal behavior or some abstraction of it

2. Use a logic to specify and verify properties of the system

Propositional Logic of Finite Domains (PLFD)

Our first step to modeling state-changing systems:

introduce a logic for expressing state variables and their values

PLFD is a family of logics

Each instance of PLFD is characterized by

- a set X of variables
- a set V of values
- a mapping dom from X to subsets of V, such that
  for every $x \in X$, $dom(x)$ is a non-empty finite set, the *domain for x*

Propositional Logic of Finite Domains (PLFD)

Our first step to modeling state-changing systems:

introduce a logic for expressing state variables and their values

PLFD is a family of logics

Each instance of PLFD is characterized by

- a set X of variables
- a set V of values
- a mapping *dom* from X to subsets of V, such that
  for every $x \in X$, $dom(x)$ is a non-empty finite set, the *domain for x*

Propositional Logic of Finite Domains (PLFD)

Our first step to modeling state-changing systems:

introduce a logic for expressing state variables and their values

PLFD is a family of logics

Each instance of PLFD is characterized by

- a set X of variables
- a set V of values
- a mapping dom from X to subsets of V, such that
  for every $x \in X$, $dom(x)$ is a non-empty finite set, the *domain for* x

Syntax of PLFD

Formulas:

- For all $x \in X$ and $v \in dom(x)$, the equality $x = v$ is a formula, also called *atomic formula*, or simply *atom*
- Other formulas are built from atomic formulas as in propositional logic, using the connectives $\top$, $\bot$, $\wedge$, $\vee$, $\neg$, $\rightarrow$, and $\leftrightarrow$

Syntax of PLFD

Formulas:

- For all $x \in X$ and $v \in dom(x)$, the equality $x = v$ is a formula, also called *atomic formula*, or simply *atom*
- Other formulas are built from atomic formulas as in propositional logic, using the connectives $\top$, $\bot$, $\wedge$, $\vee$, $\neg$, $\rightarrow$, and $\leftrightarrow$

Semantics

Consider a set X of variables and a set V of values for them

Interpretation: a mapping $\mathcal{I} : X \to V$ such that $\mathcal{I}(x) \in dom(x)$ for all $x \in X$

We extend interpretations to mappings from formulas to Boolean values as follows

1. $\mathcal{I}(x = v) = 1$ iff $\mathcal{I}(x) = v$
2. If formula is not atomic, then as for propositional formulas

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

Semantics

Consider a set X of variables and a set V of values for them

Interpretation: a mapping $\mathcal{I} : X \to V$ such that $\mathcal{I}(x) \in dom(x)$ for all $x \in X$

We extend interpretations to mappings from formulas to Boolean values as follows

1. $\mathcal{I}(x = v) = 1$ iff $\mathcal{I}(x) = v$
2. If formula is not atomic, then as for propositional formulas

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

Semantics

Consider a set X of variables and a set V of values for them

Interpretation: a mapping $\mathcal{I} : X \to V$ such that $\mathcal{I}(x) \in dom(x)$ for all $x \in X$

We extend interpretations to mappings from formulas to Boolean values as follows

1. $\mathcal{I}(x = v) = 1$ iff $\mathcal{I}(x) = v$
2. If formula is not atomic, then as for propositional formulas

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

Example

If $dom(x) = \{ a, b, c \}$, then the following is a formula which is valid:

$$\neg x = a \rightarrow x = b \vee x = c$$

In contrast, if $dom(x) = \{ a, b, c, d \}$, then the formula above is *not* valid as it is falsified by $\mathcal{I} = \{ x \mapsto d \}$:

$$\{ x \mapsto d \} \not\models \neg x = a \rightarrow x = b \vee x = c$$

Example

If $dom(x) = \{ a, b, c \}$, then the following is a formula which is valid:

$$\neg x = a \rightarrow x = b \vee x = c$$

In contrast, if $dom(x) = \{ a, b, c, d \}$, then the formula above is *not* valid
as it is falsified by $\mathcal{I} = \{ x \mapsto d \}$:

$$\{ x \mapsto d \} \not\models \neg x = a \rightarrow x = b \vee x = c$$

Example

If $dom(x) = \{ a, b, c \}$, then the following is a formula which is valid:

$$\neg x = a \rightarrow x = b \vee x = c$$

In contrast, if $dom(x) = \{ a, b, c, d \}$, then the formula above is *not* valid as it is falsified by $\mathcal{I} = \{ x \mapsto d \}$:

$$\{ x \mapsto d \} \not\models \neg x = a \rightarrow x = b \vee x = c$$

Example: microwave

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

$$\text{mode} = \textit{grill} \rightarrow \text{door} = \textit{closed} \wedge \neg(\text{temperature} = \textit{0}) \wedge \neg(\text{user} = \textit{nobody})$$

Propositional Logic as PLFD

Consider propositional variables as variables over the domain $\{ 0, 1 \}$
Instead of atoms p use $p = 1$

One can also use $p = 0$ for $\neg p$, since $(p = 0) \equiv \neg(p = 1)$

This transformation preserves models. For example, the models of

$$p \wedge q \to \neg r$$

are exactly the models of

$$p = 1 \wedge q = 1 \to r = 0$$

Propositional Logic as PLFD

Consider propositional variables as variables over the domain $\{0, 1\}$
Instead of atoms p use $p = 1$

One can also use $p = 0$ for $\neg p$, since $(p = 0) \equiv \neg(p = 1)$

This transformation preserves models. For example, the models of

$$p \wedge q \rightarrow \neg r$$

are exactly the models of

$$p = 1 \wedge q = 1 \rightarrow r = 0$$

Propositional variables in PLFD

We say that p is a boolean variable if $dom(p) = \{0, 1\}$

In instances of PLFD with both boolean and non-boolean, we will use boolean variables as in propositional logic:

- p instead of $p = 1$
- $\neg p$ instead of $p = 0$

Translation of PLFD into Propositional Logic

1. Introduce a propositional variable x_v for each variable x and value $v \in dom(x)$
2. Replace every atom $x = v$ by x_v
3. Add domain axiom for each variable x:

$$(x_{v_1} \vee \cdots \vee x_{v_n}) \wedge \bigwedge_{i<j} (\neg x_{v_i} \vee \neg x_{v_j})$$

   where $dom(x) = \{ v_1, \ldots, v_n \}$

Example

To check satisfiability of the formula

$$\neg(x = b \lor x = c)$$

where $dom(x) = \{ a, b, c \}$, we have to check satisfiability of the formula

$$\underbrace{(x_a \lor x_b \lor x_c) \land (\neg x_a \lor \neg x_b) \land (\neg x_a \lor \neg x_c) \land (\neg x_b \lor \neg x_c)}_{\text{domain axiom}} \land \neg(x_b \lor x_c)$$

Example

To check satisfiability of the formula

$$\neg(x = b \vee x = c)$$

where $dom(x) = \{ a, b, c \}$, we have to check satisfiability of the formula

$$\underbrace{(x_a \vee x_b \vee x_c) \wedge (\neg x_a \vee \neg x_b) \wedge (\neg x_a \vee \neg x_c) \wedge (\neg x_b \vee \neg x_c)}_{\text{domain axiom}} \wedge \neg(x_b \vee x_c)$$

Domain axiom for mode in microwave:

$$\begin{aligned}
&(\text{mode}_{idle} \vee \text{mode}_{micro} \vee \text{mode}_{grill} \vee \text{mode}_{defrost}) \;\wedge \\
&(\neg\text{mode}_{idle} \vee \neg\text{mode}_{micro}) \;\wedge \\
&(\neg\text{mode}_{idle} \vee \neg\text{mode}_{grill}) \;\wedge \\
&(\neg\text{mode}_{idle} \vee \neg\text{mode}_{defrost}) \;\wedge \\
&(\neg\text{mode}_{micro} \vee \neg\text{mode}_{grill}) \;\wedge \\
&(\neg\text{mode}_{micro} \vee \neg\text{mode}_{defrost}) \;\wedge \\
&(\neg\text{mode}_{grill} \vee \neg\text{mode}_{defrost})
\end{aligned}$$

Preservation of models

Suppose that $\mathcal{I}$ is a propositional model of all the domain axioms

Define a PLFD interpretation $\mathcal{I}'$ as follows:

$$\mathcal{I}'(x) = v \stackrel{\text{def}}{=} \mathcal{I} \models x_v$$

Theorem 1

Let F' be a PLFD formula and F be obtained by translating F' to propositional logic. If $\mathcal{I} \models F$, then $\mathcal{I}' \models F'$.

Tableau System for PLFD

- Use signed formulas
- Use new kind of atomic formula: $x \in \{v_1, \ldots, v_n\}$ equivalent to $x = v_1 \vee \cdots \vee x = v_n$ (also using $x \in \{v\}$ instead of $x = v$)
- Abbreviations: instead of $(x \in D)^1$ write $x \in D$, instead of $(x \in D)^0$ write $x \notin D$
- Tableau rules for PL + new tableau rules:

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

- A branch is closed if it contains $\top^0$, $\bot^1$, or $x \in \{\}$

Example

$$x \notin D \quad \rightsquigarrow \quad x \in dom(x) \setminus D$$
$$x \in D_1,\ x \in D_2 \quad \rightsquigarrow \quad x \in D_1 \cap D_2$$

Let's prove the validity of

$$F = \begin{array}{l} ((\text{user} \in \{nobody\} \rightarrow \text{content} \in \{none\}) \ \wedge \\ \ (\text{user} \in \{prof\} \rightarrow \text{content} \in \{none, cabbage\}) \ \wedge \\ \ (\text{user} \in \{staff\} \rightarrow \text{content} \in \{none, burger\}) \\ ) \ \rightarrow (\text{content} \in \{pizza\} \rightarrow \text{user} \in \{student\}) \end{array}$$

by deriving a closed tableaux from F^0

Example

$$x \notin D \quad \rightsquigarrow \quad x \in dom(x) \setminus D$$
$$x \in D_1,\ x \in D_2 \quad \rightsquigarrow \quad x \in D_1 \cap D_2$$

$$(((\text{user} \in \{\mathit{nobody}\} \to \text{content} \in \{\mathit{none}\}) \land (\text{user} \in \{\mathit{prof}\} \to \text{content} \in \{\mathit{none}, \mathit{cabbage}\}) \land (\text{user} \in \{\mathit{staff}\} \to \text{content} \in \{\mathit{none}, \mathit{burger}\})) \to (\text{content} \in \{\mathit{pizza}\} \to \text{user} \in \{\mathit{student}\}))^0$$

|

$$((\text{user} \in \{\mathit{nobody}\} \to \text{content} \in \{\mathit{none}\}) \land (\text{user} \in \{\mathit{prof}\} \to \text{content} \in \{\mathit{none}, \mathit{cabbage}\}) \land (\text{user} \in \{\mathit{staff}\} \to \text{content} \in \{\mathit{none}, \mathit{burger}\}))^1$$
$$(\text{content} \in \{\mathit{pizza}\} \to \text{user} \in \{\mathit{student}\})^0$$

|

$$(\text{user} \in \{\mathit{nobody}\} \to \text{content} \in \{\mathit{none}\})^1$$
$$(\text{user} \in \{\mathit{prof}\} \to \text{content} \in \{\mathit{none}, \mathit{cabbage}\})^1$$
$$(\text{user} \in \{\mathit{staff}\} \to \text{content} \in \{\mathit{none}, \mathit{burger}\})^1$$

|

$$\text{content} \in \{\mathit{pizza}\}$$
$$\text{user} \notin \{\mathit{student}\}$$

|

$$\text{user} \in \{\mathit{nobody}, \mathit{prof}, \mathit{staff}\}$$

Example, continued

$$x \notin D \quad \rightsquigarrow \quad x \in dom(x) \setminus D$$
$$x \in D_1,\ x \in D_2 \quad \rightsquigarrow \quad x \in D_1 \cap D_2$$