CS:4350 Logic in Computer Science

Propositional Logic of Finite Domains

Cesare Tinelli

Spring 2022

Credits

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

Outline

Logic and Modeling

Satisfiability-checking in propositional logic has many applications

Unfortunately, there is an abstract gap between real-life problems and their propositional logic representation which is too low-level

Propositional logic is not convenient for modeling problems

Many application domains have specialized modeling languages for describing problems at a level of abstraction closer to that of natural language

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 an abstract gap between real-life problems and their propositional logic representation which is too low-level

Propositional logic is not convenient for modeling problems

Many application domains have specialized modeling languages for describing problems at a level of abstraction closer to that of natural language

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 an abstract gap between real-life problems and their propositional logic representation which is too low-level

Propositional logic is not convenient for modeling problems

Many application domains have specialized modeling languages for describing problems at a level of abstraction closer to that of natural language

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 an abstract gap between real-life problems and their propositional logic representation which is too low-level

Propositional logic is not convenient for modeling problems

Many application domains have specialized modeling languages for describing problems at a level of abstraction closer to that of natural language

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

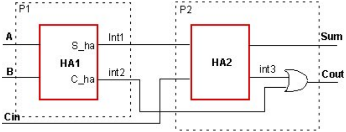
Unfortunately, there is an abstract gap between real-life problems and their propositional logic representation which is too low-level

Propositional logic is not convenient for modeling problems

Many application domains have specialized modeling languages for describing problems at a level of abstraction closer to that of natural language

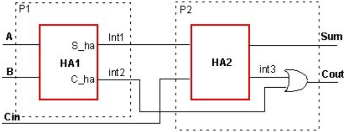
However, in many cases, problems expressed in these languages can then be translated to propositional logic

Circuit Design



Circuit: propositional logic

Circuit Design

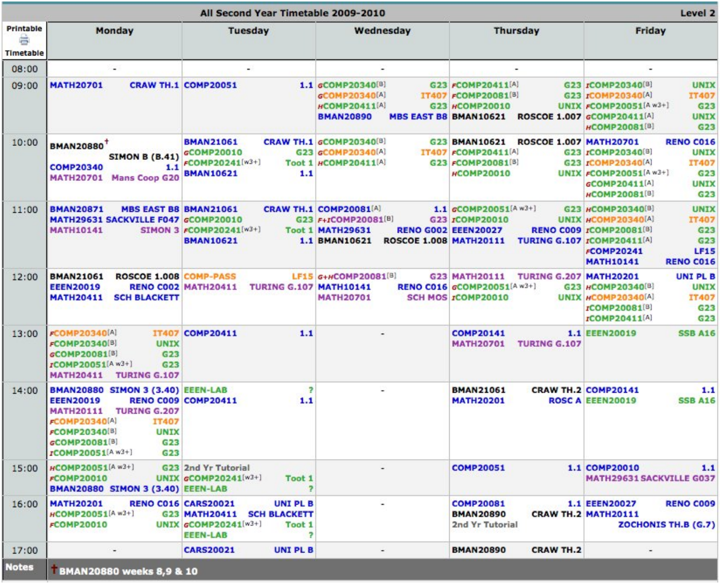


```
library ieee;
use ieee.std_logic_1164.all;
entity FULL_ADDER is
  port (A, B, Cin :  in std_logic;
    Sum, Cout :  out std_logic);
end FULL_ADDER;
architecture BEHAV_FA of FULL_ADDER is
signal int1, int2, int3:  std_logic;
begin
P1:  process (A, B)
  begin
    int1<= A xor B;
    int2<= A and B;
  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 language (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	*G*COMP20340[B] G23 *G*COMP20340[A] IT407 *H*COMP20411[A] G23 BMAN20890 MBS EAST B8	*F*COMP20411[A] G23 *F*COMP20081[B] G23 *H*COMP20010 UNIX BMAN10621 ROSCOE 1.007	*I*COMP20340[B] UNIX *I*COMP20340[A] IT407 *F*COMP20051[A w3+] G23 *G*COMP20411[A] UNIX *H*COMP20081[B] G23
10:00	BMAN20880† SIMON B (B.41) COMP20340 1.1 MATH20701 Mans Coop G20	BMAN21061 CRAW TH.1 *G*COMP20010 G23 *F*COMP20241[w3+] Toot 1 BMAN10621 1.1	*G*COMP20340[B] G23 *G*COMP20340[A] IT407 *H*COMP20411[A] G23	BMAN10621 ROSCOE 1.007 *F*COMP20411[A] G23 *F*COMP20081[B] G23 *H*COMP20010 UNIX	MATH20701 RENO C016 *I*COMP20340[B] UNIX *I*COMP20340[A] IT407 *F*COMP20051[A w3+] G23 *G*COMP20411[A] UNIX *H*COMP20081[B] G23
11:00	BMAN20871 MBS EAST B8 MATH29631 SACKVILLE F047 MATH10141 SIMON 3	BMAN21061 CRAW TH.1 *G*COMP20010 G23 *F*COMP20241[w3+] Toot 1 BMAN10621 1.1	COMP20081[A] 1.1 *F+I*COMP20081[B] G23 MATH29631 RENO G002 BMAN10621 ROSCOE 1.008	*G*COMP20051[A w3+] G23 *I*COMP20010 UNIX EEEN20027 RENO C009 MATH20111 TURING G.107	*H*COMP20340[B] UNIX *H*COMP20340[A] IT407 *I*COMP20081[B] G23 *I*COMP20411[A] G23 *F*COMP20241 LF15 MATH10141 RENO C016
12:00	BMAN21061 ROSCOE 1.008 EEEN20019 RENO C002 MATH20411 SCH BLACKETT	COMP-PASS LF15 MATH20411 TURING G.107	*G+H*COMP20081[B] G23 MATH10141 RENO C016 MATH20701 SCH MOS	MATH20111 TURING G.207 *G*COMP20051[A w3+] G23 *I*COMP20010 UNIX	MATH20201 UNI PL B *H*COMP20340[B] UNIX *H*COMP20340[A] IT407 *I*COMP20081[B] G23 *I*COMP20411[A] G23
13:00	*F*COMP20340[A] IT407 *F*COMP20340[B] UNIX *G*COMP20081[B] G23 *I*COMP20051[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 *F*COMP20340[A] IT407 *F*COMP20340[B] UNIX *G*COMP20081[B] G23 *I*COMP20051[A w3+] G23	EEEN-LAB ? COMP20411 1.1	-	BMAN21061 CRAW TH.2 MATH20201 ROSC A	COMP20141 1.1 EEEN20019 SSB A16
15:00	*H*COMP20051[A w3+] G23 *F*COMP20010 UNIX BMAN20880 SIMON 3 (3.40)	2nd Yr Tutorial *G*COMP20241[w3+] Toot 1 EEEN-LAB ?	-	COMP20051 1.1	COMP20010 1.1 MATH29631 SACKVILLE G037
16:00	MATH20201 RENO C016 *H*COMP20051[A w3+] G23 *F*COMP20010 UNIX	CARS20021 UNI PL B MATH20411 SCH BLACKETT *G*COMP20241[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

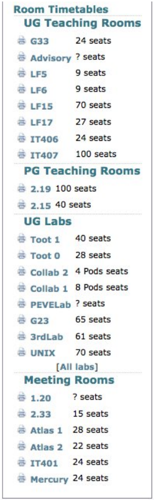
- All First Years
- All Single Hons (+CBA/IC) A+W+X+Y+Z
- All Single Hons (-CBA/IC) W+X+Y+Z
- Group A - (CBA + IC)
- Group B - (CSwBM: C+D)
- Group C - (CSwBM)
- Group D - (CSwBM)
- Group E - (CSE)
- Group M - (CM)
- Group W - (CS,SE,DC,AI)
- Group X - (CS,SE,DC,AI)
- Group Y - (CS,SE,DC,AI)
- Group 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	States can be characterized by values of a set of *state variables*.
The system state changes over time There are actions (controlled or not) that change the state	Actions change values of some of the 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	States can be characterized by values of a set of *state variables.*
The system state changes over time There are actions (controlled or not) that change the state	Actions change values of some of the state variables

Computational systems are state-changing systems

Reactive systems

Systems maintaining an ongoing interaction with their environment, as opposed to producing 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 maintaining an ongoing interaction with their environment, as opposed to producing 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 of 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 of 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:

(1) 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:

(1) 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:

(1) 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 an *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 an *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 an *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 $\leftrightarrow$

Note: $\neg x = v$ is parsed as $\neg(x = v)$ whereas $x_1 \wedge x_2 = v$ or $x = v_1 \vee v_1$, for instance, are not well-formed

Syntax of PLFD

Formulas:

- For all $x \in X$ and $v \in dom(x)$, the equality $x = v$ is an *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 $\leftrightarrow$

Note: $\neg x = v$ is parsed as $\neg(x = v)$ whereas $x_1 \wedge x_2 = v$ or $x = v_1 \vee v_1$, for instance, are not well-formed

Notation: We will often write $x \neq v$ as an abbreviation of $\neg x = v$

Semantics

Fix 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 $x \in X$

Interpretations extend to mappings from formulas to Boolean values as follows

1. $\mathcal{I}(x = v) = 1$ iff $\mathcal{I}(x) = v$
2. $\mathcal{I}(F)$ is as for propositional formulas if F is not atomic

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

Semantics

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

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

Interpretations extend to mappings from formulas to Boolean values as follows

1. $\mathcal{I}(x = v) = 1$ iff $\mathcal{I}(x) = v$
2. $\mathcal{I}(F)$ is as for propositional formulas if F is not atomic

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

Semantics

Fix 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 $x \in X$

Interpretations extend to mappings from formulas to Boolean values as follows

1. $\mathcal{I}(x = v) = 1$ iff $\mathcal{I}(x) = v$
2. $\mathcal{I}(F)$ is as for propositional formulas if F is not atomic

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 this is a formula which is also valid:

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

In contrast, if $dom(x) = \{ a, b, c, d \}$, then the formula above is not valid

Example

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

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

In contrast, if $dom(x) = \{ a, b, c, d \}$, then the formula above is not valid

Example

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

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

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

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

Example: microwave

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

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

Propositional Logic as a sublogic of PLFD

Turn propositional variables into variables over the domain $\{ 0, 1 \}$

Instead of atoms p use $p = 1$

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

This transformation preserves models. For example, the models of

$$p \land q \to \neg r$$

are exactly the models of

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

Propositional Logic as a sublogic of PLFD

Turn propositional variables into variables over the domain $\{ 0, 1 \}$

Instead of atoms p use $p = 1$

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

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 variable, we will write boolean literals as in propositional logic:

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

Translation of PLFD into Propositional Logic

While we can embed PL into PLFD we can also translate PLFD to PL!

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

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

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

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

Example

To check satisfiability of the formula

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

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

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

Domain axiom for mode in microwave:

$$\begin{aligned}
&(\text{mode}_{idle} \lor \text{mode}_{micro} \lor \text{mode}_{grill} \lor \text{mode}_{defrost}) \land \\
&(\neg\text{mode}_{idle} \lor \neg\text{mode}_{micro}) \land \\
&(\neg\text{mode}_{idle} \lor \neg\text{mode}_{grill}) \land \\
&(\neg\text{mode}_{idle} \lor \neg\text{mode}_{defrost}) \land \\
&(\neg\text{mode}_{micro} \lor \neg\text{mode}_{grill}) \land \\
&(\neg\text{mode}_{micro} \lor \neg\text{mode}_{defrost}) \land \\
&(\neg\text{mode}_{grill} \lor \neg\text{mode}_{defrost})
\end{aligned}$$

Preservation of models

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

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

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

Theorem 1

Let F' be a PLFD formula and let F be the translation of F' to propositional logic. If $\mathcal{I} \models F$, then $\mathcal{I}' \models F'$.

A 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 use $x \in \{v\}$ 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:

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

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

Example 1

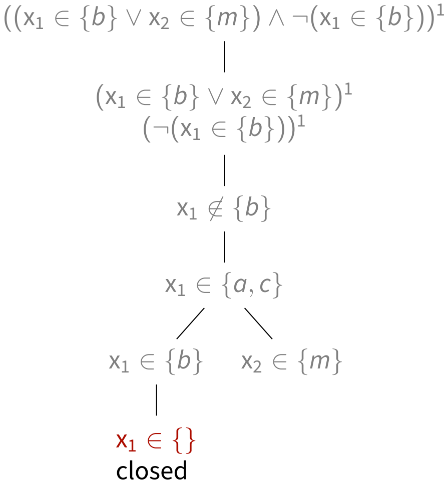
$$\begin{array}{rcl} x \notin D & \rightsquigarrow & x \in dom(x) \setminus D \\ x \in D_1,\, x \in D_2 & \rightsquigarrow & x \in D_1 \cap D_2 \end{array}$$

$$dom(x_1) = \{a, b, c\}$$

$$dom(x_2) = \{s, m, l\}$$

Example 1

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



$((\mathrm{x}_1 \in \{b\} \vee \mathrm{x}_2 \in \{m\}) \wedge \neg(\mathrm{x}_1 \in \{b\}))^1$

$(\mathrm{x}_1 \in \{b\} \vee \mathrm{x}_2 \in \{m\})^1$

$(\neg(\mathrm{x}_1 \in \{b\}))^1$

$\mathrm{x}_1 \notin \{b\}$

$\mathrm{x}_1 \in \{a, c\}$

$\mathrm{x}_1 \in \{b\}$

$\mathrm{x}_2 \in \{m\}$

$\mathrm{x}_1 \in \{\}$

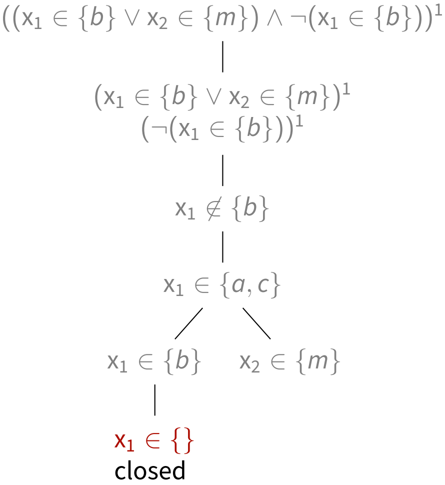
closed

$$dom(x_1) = \{a, b, c\}$$

$$dom(x_2) = \{s, m, l\}$$

Example 1

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



$((x_1 \in \{b\} \vee x_2 \in \{m\}) \wedge \neg(x_1 \in \{b\}))^1$

$(x_1 \in \{b\} \vee x_2 \in \{m\})^1$
$(\neg(x_1 \in \{b\}))^1$

$x_1 \notin \{b\}$

$x_1 \in \{a, c\}$

$x_1 \in \{b\}$ $\quad$ $x_2 \in \{m\}$

$x_1 \in \{\}$
closed

$$dom(x_1) = \{a, b, c\}$$

$$dom(x_2) = \{s, m, l\}$$

Models:

1. $\{x_1 \mapsto a,\ x_2 \mapsto m\}$
2. $\{x_1 \mapsto c,\ x_2 \mapsto m\}$

Example 2

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

Let’s prove the validity of

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

by deriving a closed tableaux from F^0

Example 2

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

$$\begin{gathered} (((\text{user} \in \{\mathit{nobody}\} \rightarrow \text{content} \in \{\mathit{none}\}) \;\wedge\; (\text{user} \in \{\mathit{prof}\} \rightarrow \text{content} \in \{\mathit{none}, \mathit{soup}\}) \;\wedge \\ (\text{user} \in \{\mathit{staff}\} \rightarrow \text{content} \in \{\mathit{none}, \mathit{burger}\})) \;\rightarrow (\text{content} \in \{\mathit{pizza}\} \rightarrow \text{user} \in \{\mathit{student}\}))^0 \end{gathered}$$

$$|$$

$$\begin{gathered} ((\text{user} \in \{\mathit{nobody}\} \rightarrow \text{content} \in \{\mathit{none}\}) \;\wedge\; (\text{user} \in \{\mathit{prof}\} \rightarrow \text{content} \in \{\mathit{none}, \mathit{soup}\}) \;\wedge \\ (\text{user} \in \{\mathit{staff}\} \rightarrow \text{content} \in \{\mathit{none}, \mathit{burger}\}))^1 \\ (\text{content} \in \{\mathit{pizza}\} \rightarrow \text{user} \in \{\mathit{student}\})^0 \end{gathered}$$

$$|$$

$$\begin{gathered} (\text{user} \in \{\mathit{nobody}\} \rightarrow \text{content} \in \{\mathit{none}\})^1 \\ (\text{user} \in \{\mathit{prof}\} \rightarrow \text{content} \in \{\mathit{none}, \mathit{soup}\})^1 \\ (\text{user} \in \{\mathit{staff}\} \rightarrow \text{content} \in \{\mathit{none}, \mathit{burger}\})^1 \end{gathered}$$

$$|$$

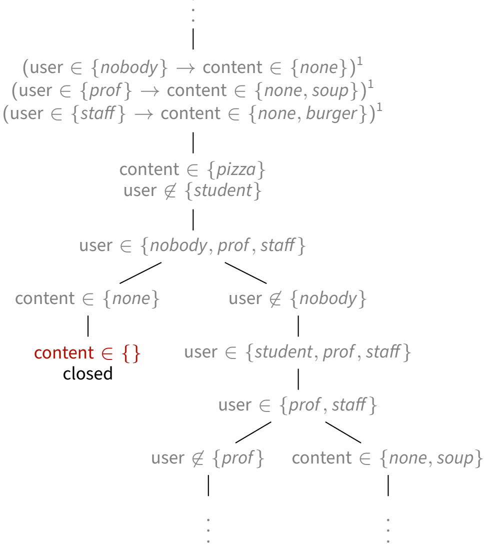
$$\begin{gathered} \text{content} \in \{\mathit{pizza}\} \\ \text{user} \notin \{\mathit{student}\} \end{gathered}$$

$$|$$

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

Example 2, continued

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



$\vdots$

$(\text{user} \in \{nobody\} \rightarrow \text{content} \in \{none\})^1$
$(\text{user} \in \{prof\} \rightarrow \text{content} \in \{none, soup\})^1$
$(\text{user} \in \{staff\} \rightarrow \text{content} \in \{none, burger\})^1$

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

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

$\text{content} \in \{none\}$

$\text{content} \in \{\}$
closed

$\text{user} \notin \{nobody\}$

$\text{user} \in \{student, prof, staff\}$

$\text{user} \in \{prof, staff\}$

$\text{user} \notin \{prof\}$

$\vdots$

$\text{content} \in \{none, soup\}$

$\vdots$

Natural Deduction for PLFD

- Use again atomic formulas of the form $x \in \{v_1, \ldots, v_n\}$ equivalent to $x = v_1 \vee \cdots \vee x = v_n$

- Use natural deduction rules for PL + new rules:

$$\frac{x \notin D}{x \in dom(x) \setminus D} \notin e \qquad \frac{x \in \{\}}{\bot} \bot i$$

$$\frac{x \in D_1,\ x \in D_2}{x \in D_1 \cap D_2} \cap i \qquad \frac{x \in D_1 \cup D_2}{x \in D_1 \vee x \in D_2} \cup i$$

Example 3

Let’s prove the validity of the judment

$$P_1,\ P_2\ \vdash\ F$$

where

$$\begin{array}{rcl} P_1 & = & \text{user} \in \{\mathit{nobody}, \mathit{prof}\} \to \text{content} \in \{\mathit{none}, \mathit{soup}\} \\ P_2 & = & \text{user} \in \{\mathit{staff}\} \to \text{content} \in \{\mathit{none}, \mathit{burger}\} \\ F & = & \text{content} \in \{\mathit{pizza}\} \to \text{user} \in \{\mathit{student}\} \end{array}$$

by deriving F from premises P_1 and P_2 by natural deduction

Example 3

1 $\text{user} \in \{\textit{nobody}, \textit{prof}\} \rightarrow \text{content} \in \{\textit{none}, \textit{soup}\}$ premise

2 $\text{user} \in \{\textit{staff}\} \rightarrow \text{content} \in \{\textit{none}, \textit{burger}\}$ premise

> 3 $\text{content} \in \{\textit{pizza}\}$ assumption
>
> > 4 $\text{user} \notin \{\textit{student}\}$ assumption
> >
> > 5 $\text{user} \in \{\textit{nobody}, \textit{prof}, \textit{staff}\}$ $\notin\mathrm{e}$ 4
> >
> > 6 $\text{user} \in \{\textit{nobody}, \textit{prof}\} \vee \text{user} \in \{\textit{staff}\}$ $\cup\mathrm{i}$ 5
> >
> > 7 $\cdots$
> >
> > 8 $\text{content} \in \{\}$ $\vee\mathrm{e}$ 6 – 8
> >
> > 9 $\bot$ $\bot\mathrm{i}$ 8
>
> 10 $\text{user} \in \{\textit{student}\}$ PBC 4 – 9

11 $\text{content} \in \{\textit{pizza}\} \rightarrow \text{user} \in \{\textit{student}\}$ $\rightarrow\mathrm{i}$ 3 – 10