Mathematical Logic - 2016

Exercises: First Order Logics (FOL)

Originally by Alessandro Agostini and Fausto Giunchiglia
Modified by Fausto Giunchiglia, Rui Zhang, Vincenzo Maltese and Mattia Fumagalli

DPLL

DPLL Procedure: Main Steps

1. It identifies all literal in the input proposition P

$B \wedge \neg C \wedge (B \vee \neg A \vee C) \wedge (\neg B \vee D)$

2. It assigns a truth-value to each variable to satisfy them

$B \wedge \neg C \wedge (B \vee \neg A \vee C) \wedge (\neg B \vee D)$ v(B) = T; v(C) = F

3. It simplifies P by removing all clauses in P which become true under the truth-assignments at step 2 and all literals in P that become false from the remaining clauses (this may generate <u>empty clauses</u>)

D

4. It recursively checks if the simplified proposition obtained in step 3 is satisfiable; if this is the case then P is satisfiable, otherwise the same recursive checking is done assuming the opposite truth value (*).

D YES, it is satisfiable for v(D) = T. NOTE: v(A) can be T/F

DPLL algorithm

- **Input**: a proposition P in CNF
- **Output**: true if "P satisfiable" or false if "P unsatisfiable"

boolean function DPLL(P) {

    if consistent(P) **then return true**;

    if hasEmptyClause(P) **then return false**;

    foreach unit clause C **in** P **do**

        P = unit-propagate(C, P);

    foreach pure-literal L **in** P **do**

        P = pure-literal-assign(L, P);

    L = choose-literal(P);

    return DPLL($P \wedge L$) OR DPLL($P \wedge \neg L$);

}

DPLL algorithm

- **Input**: a proposition P in CNF
- **Output**: true if "P satisfiable" or false if "P unsatisfiable"

boolean function DPLL(P) {

  if consistent(P) **then return true**;

  if hasEmptyClause(P) **then return false**;

  foreach unit clause C **in** P **do**

    P = unit-propagate(C, P);

  foreach pure-literal L **in** P **do**

    P = pure-literal-assign(L, P);

  L = choose-literal(P);

  return DPLL($P \wedge L$) OR DPLL($P \wedge \neg L$);

}

It tests the formula P for consistency, namely it does not contain contradictions (e.g. $A \wedge \neg A$) and all clauses are unit clauses.

DPLL algorithm

- **Input**: a proposition P in CNF
- **Output**: true if "P satisfiable" or false if "P unsatisfiable"

```
boolean function DPLL(P) {
    if consistent(P) then return true;
    if hasEmptyClause(P) then return false;
    foreach unit clause C in P do
        P = unit-propagate(C, P);
    foreach pure-literal L in P do
        P = pure-literal-assign(L, P);
    L = choose-literal(P);
    return DPLL(P ∧ L) OR DPLL(P ∧ ¬L);
}
```

An empty clause does not contain literals.

It can be due to previous iterations of the algorithm where some simplifications has been done.
If any of them exists then P is unsatisfiable.

DPLL algorithm

- **Input**: a proposition P in CNF
- **Output**: true if "P satisfiable" or false if "P unsatisfiable"

boolean function DPLL(P) {

  if consistent(P) **then return true**;

  if hasEmptyClause(P) **then return false**;

  foreach unit clause C **in** P **do**

    P = unit-propagate(C, P);

  foreach pure-literal L **in** P **do**

    P = pure-literal-assign(L, P);

  L = choose-literal(P);

  return DPLL($P \wedge L$) OR DPLL($P \wedge \neg L$);

}

(a) It assigns the right truth value to each literal (true for positives and false for negatives). (b) It simplifies P by removing all clauses in P which become true under the truth-assignment and all literals in P that become false from the remaining clauses.

DPLL algorithm

- **Input**: a proposition P in CNF
- **Output**: true if "P satisfiable" or false if "P unsatisfiable"

boolean function DPLL(P) {

  if consistent(P) **then return true**;

  if hasEmptyClause(P) **then return false**;

  foreach unit clause C **in** P **do**

    P = unit-propagate(C, P);

  foreach pure-literal L **in** P **do**

    P = pure-literal-assign(L, P);

  L = choose-literal(P);

  return DPLL(P $\wedge$ L) OR DPLL(P $\wedge$ $\neg$L);

}

For all literals which appear pure in the formula (i.e. with only one polarity) assign the corresponding value:

- true if positive literal
- false if negative

Not all DPLL versions perform this step.

DPLL algorithm

- **Input**: a proposition P in CNF
- **Output**: true if "P satisfiable" or false if "P unsatisfiable"

boolean function DPLL(P) {

  if consistent(P) **then return true**;

  if hasEmptyClause(P) **then return false**;

  foreach unit clause C **in** P **do**

    P = unit-propagate(C, P);

  foreach pure-literal L **in** P **do**

    P = pure-literal-assign(L, P);

  L = choose-literal(P);

  return DPLL(P $\wedge$ L) OR DPLL(P $\wedge$ $\neg$L);

}

The splitting rule:

Select a variable whose value is not assigned yet.

Recursively call DPLL for the cases in which the literal is true or false.

DPLL Procedure: Example 1

$P = A \wedge (A \vee \neg A) \wedge B$

- There are still variables and clauses to analyze, go ahead
- P does not contain empty clauses, go ahead
- It assigns the right truth-value to A and B: $v(A) = T$, $v(B) = T$
- It simplifies P by removing all clauses in P which become true under $v(A) = T$ and $v(B) = T$

  This causes the removal of all the clauses in P
- It simplifies P by removing all literals in the clauses of P that become false from the remaining clauses: nothing to remove
- It assigns values to pure literals. nothing to assign
- All variables are assigned: it returns true

DPLL Procedure: Example 2

$P = C \wedge (A \vee \neg A) \wedge B$

- There are still variables and clauses to analyze, go ahead
- P does not contain empty clauses, go ahead
- It assigns the right truth-value to C and B: $v(C) = T$, $v(B) = T$
- It simplifies P by removing all clauses in P which become true under $v(C) = T$ and $v(B) = T$.

  P is then simplified to $(A \vee \neg A)$
- It simplifies P by removing all literals in the clauses of P that become false from the remaining clauses: nothing to remove
- It assigns values to pure literals: nothing to assign
- It selects A and applies the splitting rule by calling DPLL on
  - $A \wedge (A \vee \neg A)$ AND $\neg A \wedge (A \vee \neg A)$

  which are both true (the first call is enough). It returns true

DPLL Procedure: Example 3

$P = A \wedge \neg B \wedge (\neg A \vee B)$

- There are still variables and clauses to analyze, go ahead
- P does not contain empty clauses, go ahead
- It assigns the right truth-value to A and B

  $\nu(A) = T, \nu(B) = F$
- It simplifies P by removing all clauses in P which become true under $\nu(A) = T$ and $\nu(B) = F$.

  P is simplified to $(\neg A \vee B)$
- It simplifies P by removing all literals in the clauses of P that become false from the remaining clauses: the last clause becomes empty
- It assigns values to pure literals: nothing to assign
- All variables are assigned but there is an empty clause: it returns false

FOL

The need for greater expressive power

- We need FOL for a greater expressive power. In FOL we have:
  - constants/individuals (e.g. 2)
  - variables (e.g. x)
  - Unary predicates (e.g. Man)
  - N-ary predicates (eg. Near)
  - functions (e.g. Sum, Exp)
  - quantifiers ($\forall$, $\exists$)
  - equality symbol = (optional)

Alphabet of symbols in FOL

- Variables $x_1, x_2, \ldots, y, z$
- Constants $a_1, a_2, \ldots, b, c$
- Predicate symbols $A^1_1, A^1_2, \ldots, A^n_m$
- Function symbols $f^1_1, f^1_2, \ldots, f^n_m$
- Logical symbols $\wedge, \vee, \neg, \supset, \forall, \exists$
- Auxiliary symbols ( )

- Indexes on top are used to denote the number of arguments, called arity, in predicates and functions.
- Indexes on the bottom are used to disambiguate between symbols having the same name.
- Predicates of arity =1 correspond to properties or concepts

Write in FOL the following NL sentences

- "Einstein is a scientist"

  $Scientist(einstein)$

- "There is a monkey"

  $\exists x\ Monkey(x)$

- "There exists a dog which is black"

  $\exists x\ (Dog(x) \wedge Black(x))$

- "All persons have a name"

  $\forall x\ (Person(x) \supset \exists y\ Name(x, y))$

Write in FOL the following NL sentences

- "The sum of two odd numbers is even"

  $\forall x \ \forall y \ (Odd(x) \land Odd(y) \supset Even(Sum(x,y)))$

- "A father is a male person having at least one child"

  $\forall x \ (Father(x) \supset Person(x) \land Male(x) \land \exists y \ hasChilden(x, y))$

- "There is exactly one dog"

  $\exists x \ Dog(x) \land \forall x \ \forall y \ (Dog(x) \land Dog(y) \supset x = y)$

- "There are at least two dogs"

  $\exists x \ \exists y \ (Dog(x) \land Dog(y) \land \neg(x = y))$

The use of FOL in mathematics

- Express in FOL the fact that every natural number x multiplied by 1 returns x (identity):

$$\forall x \, (\mathrm{Natural}(x) \supset (\mathrm{Mult}(x, 1) = x) \,)$$

- Express in FOL the fact that the multiplication of two natural numbers is commutative:

$$\forall x \, \forall y \, (\mathrm{Natural}(x) \wedge \mathrm{Natural}(y) \supset (\mathrm{Mult}(x, y) = \mathrm{Mult}(y, x)) \,)$$

The use of FOL in mathematics

- FOL has being introduced to express mathematical properties

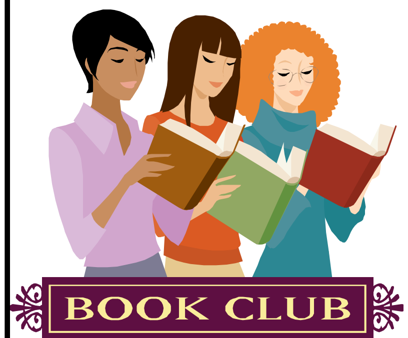
- The set of axioms describing the properties of equality between natural numbers (by Peano):

  Axioms about equality

1. $\forall x_1 (x_1 = x_1)$ reflexivity
2. $\forall x_1 \forall x_2 (x_1 = x_2 \supset x_2 = x_1)$ symmetricity
3. $\forall x_1 \forall x_2 \forall x_3 (x_1 = x_2 \wedge x_2 = x_3 \supset x_1 = x_3)$ transitivity
4. $\forall x_1 \forall x_2 (x_1 = x_2 \supset S(x_1) = S(x_2))$ successor

NOTE: Other axioms can be given for the properties of the successor, the addition (+) and the multiplication (x).

Modeling the club of married problem



There are exactly three people in the club, Tom, Sue and Mary. Tom and Sue are married. If a member of the club is married, their spouse is also in the club. Mary is not married.

L = {tom, sue, mary, Club, Married}

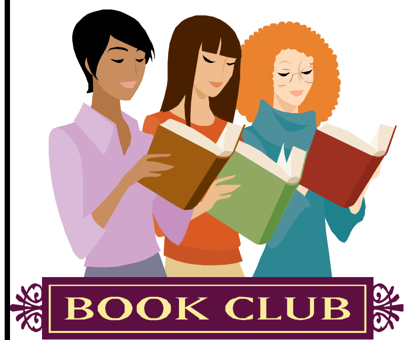
$\text{Club}(tom) \wedge \text{Club}(sue) \wedge \text{Club}(mary) \wedge \forall x\, (\text{Club}(x) \supset (x = tom \vee x = sue \vee x = mary))$

$\text{Married}(tom, sue)$

$\forall x\, \forall y\, ((\text{Club}(x) \wedge \text{Married}(x, y)) \supset \text{Club}(y))$

$\neg\, \exists x\, \text{Married}(mary, x)$

Modeling the club of married problem (II)



There are exactly three people in the club, Tom, Sue and Mary. Tom and Sue are married. If a member of the club is married, their spouse is also in the club.

Add enough common sense FOL statements (e.g. everyone has at most one spouse, nobody can be married to himself or herself, Tom, Sue and Mary are different people) to make it entail that Mary is not married in FOL.

L = {tom, sue, mary, Club, Married}

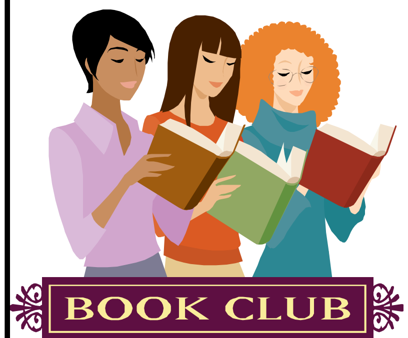
S1: $Club(tom) \wedge Club(sue) \wedge Club(mary) \wedge \forall x\ (Club(x) \supset (x = tom \vee x = sue \vee x = mary))$

S2: $Married(tom, sue)$

S3: $\forall x\ \forall y\ ((Club(x) \wedge Married(x, y)) \supset Club(y))$

S4: $\neg \exists x\ Married(mary, x)$

Modeling the club of married problem (III)



There are exactly three people in the club, Tom, Sue and Mary. Tom and Sue are married. If a member of the club is married, their spouse is also in the club.

Add enough common sense FOL statements (e.g. everyone has at most one spouse, nobody can be married to himself or herself, Tom, Sue and Mary are different people) to make it entail that Mary is not married in FOL.

We need to add the following:

S5:	$\forall x\ \forall y\ \forall z\ ((Married(x, y) \wedge Married(x, z) \supset y = z)$	at most one wife
S6:	$\neg\ \exists x\ Married(x, x)$	nobody is married with himself/herself
S7:	$\neg\ (tom=sue) \wedge \neg\ (tom=mary) \wedge \neg\ (mary=sue)$	unique name assumption

Interpretation

FOL interpretation for a language L

A first order interpretation for the language $L = \langle c_1, c_2, \ldots, f_1, f_2, \ldots, R_1, R_2, \ldots \rangle$ is a pair $\langle \Delta, \mathcal{I} \rangle$ where

- Δ is a non empty set called interpretation domain
- $\mathcal{I}$ is is a function, called interpretation function
  - $\mathcal{I}(c_i) \in \Delta$ (elements of the domain)
  - $\mathcal{I}(f_i) : \Delta^n \longrightarrow \Delta$ (n-ary function on the domain)
  - $\mathcal{I}(P_i) \subseteq \Delta^n$ (n-ary relation on the domain)

where n is the arity of f_i and P_i.

Example of interpretation

Example (Of interpretation)

Symbols

Constants: *alice*, *bob*, *carol*, *robert*
Function: *mother-of* (with arity equal to 1)
Predicate: *friends* (with arity equal to 2)

Domain $\Delta = \{1, 2, 3, 4, \ldots\}$

Interpretation $\mathcal{I}(alice) = 1$, $\mathcal{I}(bob) = 2$, $\mathcal{I}(carol) = 3$, $\mathcal{I}(robert) = 2$

$$\mathcal{I}(mother\text{-}of) = M \quad \begin{array}{l} M(1) = 3 \\ M(2) = 1 \\ M(3) = 4 \\ M(n) = n + 1 \text{ for } n \geq 4 \end{array}$$

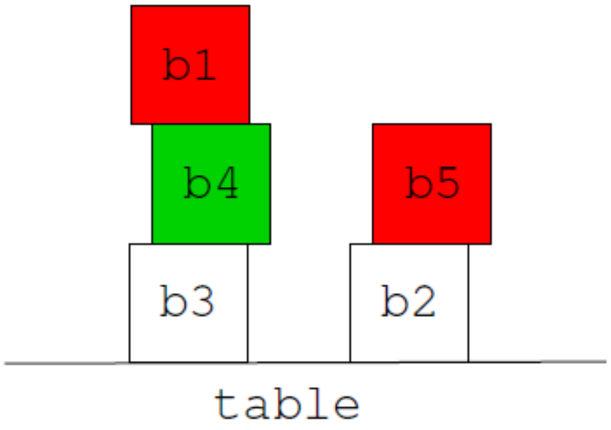
$$\mathcal{I}(friends) = F = \left\{ \begin{array}{lll} \langle 1, 2 \rangle, & \langle 2, 1 \rangle, & \langle 3, 4 \rangle, \\ \langle 4, 3 \rangle, & \langle 4, 2 \rangle, & \langle 2, 4 \rangle, \\ \langle 4, 1 \rangle, & \langle 1, 4 \rangle, & \langle 4, 4 \rangle \end{array} \right\}$$

Modeling "blocks world"

Non Logical symbols

Constants: A, B, C, D, E, F;

Predicates: On^2, $Above^2$, $Free^1$, Red^1, $Green^1$.



Interpretation $\mathcal{I}_1$

- $\mathcal{I}_1(A) = b_1$, $\mathcal{I}_1(B) = b_2$, $\mathcal{I}_1(C) = b_3$, $\mathcal{I}_1(D) = b_4$, $\mathcal{I}_1(E) = b_5$, $\mathcal{I}_1(F) = table$
- $\mathcal{I}_1(On) = \{\langle b_1, b_4\rangle, \langle b_4, b_3\rangle, \langle b_3, table\rangle, \langle b_5, b_2\rangle, \langle b_2, table\rangle\}$
- $\mathcal{I}_1(Above) = \{\langle b_1, b_4\rangle, \langle b_1, b_3\rangle, \langle b_1, table\rangle, \langle b_4, b_3\rangle, \langle b_4, table\rangle, \langle b_3, table\rangle, \langle b_5, b_2\rangle, \langle b_5, table\rangle, \langle b_2, table\rangle\}$
- $\mathcal{I}_1(Free) = \{\langle b_1\rangle, \langle b_5\rangle\}$, $\mathcal{I}_1(Green) = \{\langle b_4\rangle\}$, $\mathcal{I}_1(Red) = \{\langle b_1\rangle, \langle b_5\rangle\}$

Interpretation of terms

Definition (Assignment)

An assignment a is a function from the set of variables to Δ.

$a[x/d]$ denotes the assignment that conincides with a on all the variables but x, which is associated to d.

Interpretation of terms

The interpretation of a term t w.r.t. the assignment a, in symbols $\mathcal{I}(t)[a]$ is recursively defined as follows:

$$
\begin{array}{rcl}
\mathcal{I}(x_i)[a] & = & a(x_i) \\
\mathcal{I}(c_i)[a] & = & \mathcal{I}(c_i) \\
\mathcal{I}(f(t_1, \ldots, t_n))[a] & = & \mathcal{I}(f)(\mathcal{I}(t_1)[a], \ldots, \mathcal{I}(t_n)[a])
\end{array}
$$

Interpretation of terms (example of [a])

A

$\mathcal{I}$(tom)[a] = $\mathcal{I}$(tom) = Tom

$\mathcal{I}$(sue)[a] = $\mathcal{I}$(sue) = Sue

B

$\mathcal{I}$(x) [a] = a(x) = tom

$\mathcal{I}$(y) [a] = a(y) = sue

$\mathcal{I}$(z) [a] = a(z) = 3

C

$\mathcal{I}$(sum(z, 5))[a] = $\mathcal{I}$(sum) $\mathcal{I}$($\mathcal{I}$(z)[a], $\mathcal{I}$(5)[a]) = sum(a(z), 5) = sum(3,5) = 8

Satisfiability of a formula

Definition (Satisfiability of a formula w.r.t. an assignment)

An interpretation $\mathcal{I}$ satisfies a formula ϕ w.r.t. the assignment a according to the following rules:

$$
\begin{array}{rcl}
\mathcal{I} \models t_1 = t_2[a] & \text{iff} & \mathcal{I}(t_1)[a] = \mathcal{I}(t_2)[a] \\
\mathcal{I} \models P(t_1, \dots, t_n)[a] & \text{iff} & \langle \mathcal{I}(t_1)[a], \dots, \mathcal{I}(t_n)[a] \rangle \in \mathcal{I}(P) \\
\mathcal{I} \models \phi \wedge \psi[a] & \text{iff} & \mathcal{I} \models \phi[a] \text{ and } \mathcal{I} \models \psi[a] \\
\mathcal{I} \models \phi \vee \psi[a] & \text{iff} & \mathcal{I} \models \phi[a] \text{ or } \mathcal{I} \models \psi[a] \\
\mathcal{I} \models \phi \supset \psi[a] & \text{iff} & \mathcal{I} \not\models \phi[a] \text{ or } \mathcal{I} \models \psi[a] \\
\mathcal{I} \models \neg\phi[a] & \text{iff} & \mathcal{I} \not\models \phi[a] \\
\mathcal{I} \models \phi \equiv \psi[a] & \text{iff} & \mathcal{I} \models \phi[a] \text{ iff } \mathcal{I} \models \psi[a] \\
\mathcal{I} \models \exists x \phi[a] & \text{iff} & \text{there is a } d \in \Delta \text{ such that} \mathcal{I} \models \phi[a[x/d]] \\
\mathcal{I} \models \forall x \phi[a] & \text{iff} & \text{for all } d \in \Delta, \mathcal{I} \models \phi[a[x/d]]
\end{array}
$$

Analogy with Databases

EMPLOYEE			
NAME	**GENDER**	**CITY**	**SALARY**
Mary	Female	Rome	2200
Paul	Male	Florence	1800
George	Male	Naples	1700
Leon	Male	London	2500
Luc	Male	Rome	1800
Lucy	Female	Rome	1700

DEPARTMENT	
EMPLOYEE	**NAME**
Mary	Administration
Paul	Marketing
George	Customer Care
Leon	Production
Luc	Production
Lucy	Production

1. provide a First Order formula which retrieves the name and the city of all the employees earning more than 1750 and working at the Production department,
2. provide the possible assignments making the formula true.

Solution. $\exists y \exists w (Employee(x, y, z, w) \wedge Department(x, Production) \wedge (w > 1750))$ with assignments (Leon, London) and (Luc, Rome)

Decide whether the following formula are satisfied by $\mathcal{I}_1$

1. *"A is above C, D is above F and on E."*
   $\phi_1 : Above(A, C) \wedge Above(E, F) \wedge On(D, E)$ (NO, "On")
2. *"A is green while C is not."*
   $\phi_2 : Green(A) \wedge \neg Green(C)$ (NO)
3. *"Everything is on something."*
   $\phi_3 : \forall x \exists y.On(x, y)$ (NO, "table")
4. *"Everything that is free has nothing on it."*
   $\phi_4 : \forall x.(Free(x) \rightarrow \neg\exists y.On(y, x))$ (YES)
5. *"Everything that is green is free."*
   $\phi_5 : \forall x.(Green(x) \rightarrow Free(x))$ (NO)
6. *"There is something that is red and is not free."*
   $\phi_6 : \exists x.(Red(x) \wedge \neg Free(x))$ (NO)
7. *"Everything that is not green and is above B, is red."*
   $\phi_7 : \forall x.(\neg Green(x) \wedge Above(x, B) \rightarrow Red(x))$ (YES)

FOL Tableaux

. . . for propositional connectives

α rules

$$\frac{\phi \wedge \psi}{\begin{array}{c}\phi\\ \psi\end{array}} \qquad \frac{\neg(\phi \vee \psi)}{\begin{array}{c}\neg\phi\\ \neg\psi\end{array}} \qquad \frac{\neg\neg\phi}{\phi} \qquad \frac{\neg(\phi \supset \psi)}{\begin{array}{c}\phi\\ \neg\psi\end{array}}$$

β rules

$$\frac{\phi \vee \psi}{\phi \mid \psi} \qquad \frac{\phi \supset \psi}{\neg\phi \mid \psi} \qquad \frac{\neg(\phi \wedge \psi)}{\neg\phi \mid \neg\psi} \qquad \frac{\phi \equiv \psi}{\begin{array}{c|c}\phi & \neg\phi\\ \psi & \neg\psi\end{array}} \qquad \frac{\neg(\phi \equiv \psi)}{\begin{array}{c|c}\phi & \neg\phi\\ \neg\psi & \psi\end{array}}$$

γ rules

$$\frac{\forall x.\phi(x)}{\phi(t)} \qquad \frac{\neg\exists x.\phi(x)}{\neg\phi(t)}$$

Where t is a term free for x in ϕ

δ rules

$$\frac{\neg\forall x.\phi(x)}{\neg\phi(c)} \qquad \frac{\exists x.\phi(x)}{\phi(c)}$$

where c is a new constant not previously appearing in the tableaux

FOL Tableaux

Check via tableaux the validity/satisability of the formula:

- $\phi = \forall xy(P(x) \supset Q(y)) \supset (\exists xP(x) \supset \forall yQ(y))$

$$\neg(\forall xy(P(x) \supset Q(y)) \supset (\exists xP(x) \supset \forall yQ(y)))$$

$$(\forall xy(P(x) \supset Q(y))$$
$$\neg(\exists xP(x) \supset \forall yQ(y))$$

$$\exists xP(x)$$
$$\neg\forall yQ(y)$$

$$P(a)$$
$$\neg Q(b)$$

$$P(a) \supset Q(b)$$

$\neg P(a)$ $Q(b)$

X   X

MODAL LOGIC

Introduction

- We want to model situations like this one:
  1. "Fausto is always happy" circumstances"
  2. "Fausto is happy under certain
- In PL/ClassL we could have: HappyFausto
- In modal logic we have:
  1. $\Box$ HappyFausto
  2. $\Diamond$ HappyFausto

As we will see, this is captured through the notion of "possible words" and of "accessibility relation"

Syntax

- We extend PL with two logical modal operators:

  □ (box) and ◊ (diamond)

  □P : "Box P" or "necessarily P" or "P is necessary true"

  ◊P : "Diamond P" or "possibly P" or "P is possible"

  Note that we define □P = ¬◊¬P, i.e. □ is a primitive symbol

- The grammar is extended as follows:

  <Atomic Formula> ::= A | B | ... | P | Q | ... | ⊥ | ⊤ |

  <wff> ::= <Atomic Formula> | ¬<wff> | <wff> ∧ <wff> | <wff> ∨ <wff> |
  <wff> → <wff> | <wff> ↔ <wff> | □ <wff> | ◊ <wff>

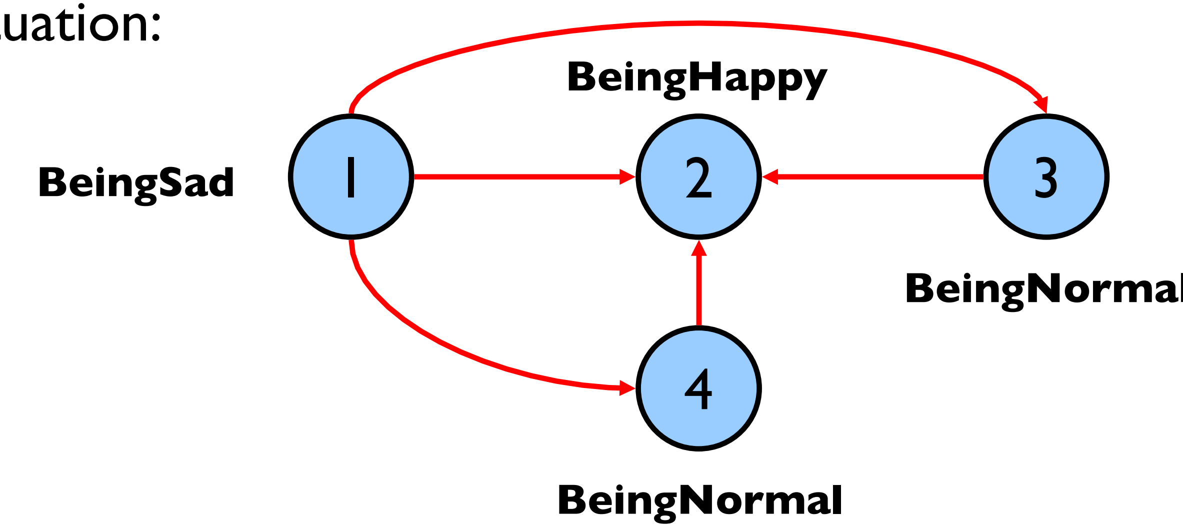
Semantics: Kripke Model

- A Kripke Model is a triple M = <W, R, I> where:
  - W is a non empty set of worlds
  - $R \subseteq W \times W$ is a binary relation called the accessibility relation
  - I is an interpretation function $I: L \rightarrow pow(W)$ such that to each proposition P we associate a set of possible worlds I(P) in which P holds

- Each $w \in W$ is said to be a world, point, state, event, situation, class … according to the problem we model
- In a Kripke model, <W, R> is called frame and is a relational structure.

Semantics: Kripke Model

- Consider the following situation:



- M = <W, R, I>

  W = {1, 2, 3, 4}

  R = {<1, 2>, <1, 3>, <1, 4>, <3, 2>, <4, 2>}

  I(BeingHappy) = {2}   I(BeingSad) = {1}   I(BeingNormal) = {3, 4}

Satisfiability of modal formulas

Truth is relative to a world, so we define that relation of $\vDash$ between a world in a model and a formula (NOTE: wRw' can be read as "w' is accessible from w via R")

$M, w \vDash p$ iff $w \in I(p)$

$M, w \vDash \varphi \wedge \psi$ iff $M, w \vDash \varphi$ and $M, w \vDash \psi$

$M, w \vDash \varphi \vee \psi$ iff $M, w \vDash \varphi$ or $M, w \vDash \psi$

$M, w \vDash \varphi \supset \psi$ iff $M, w \vDash \varphi \Rightarrow$ *implies* $M, w \vDash \psi$

$M, w \vDash \varphi \equiv \psi$ iff $M, w \vDash \varphi$ iff $M, w \vDash \psi$

$M, w \vDash \neg\varphi$ iff not $M, w \vDash \varphi$

$M, w \vDash \Box\varphi$ iff for all w' s.t. wRw', $M, w' \vDash \varphi$

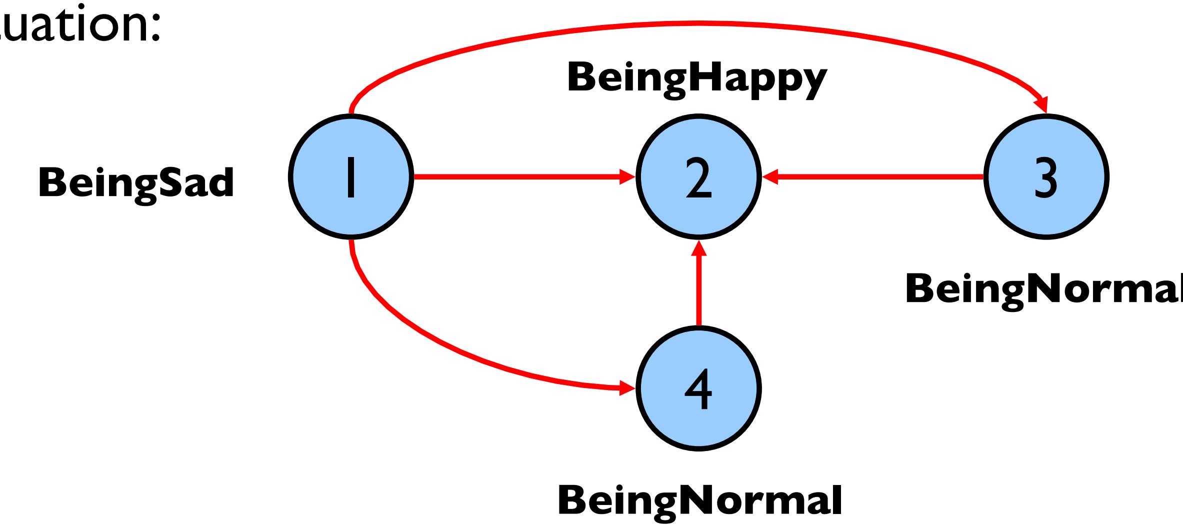
$M, w \vDash \Diamond\varphi$ iff there is a w' s.t. wRw' and $M, w' \vDash \varphi$

φ is globally satisfied in a model M, in symbols, $M \vDash \varphi$ if

$M, w \vDash \varphi$ for all $w \in W$

Semantics: Kripke Model

- Consider the following situation:



- M = <W, R, I>

  W = {1, 2, 3, 4}

  R = {<1, 2>, <1, 3>, <1, 4>, <3, 2>, <4, 2>}

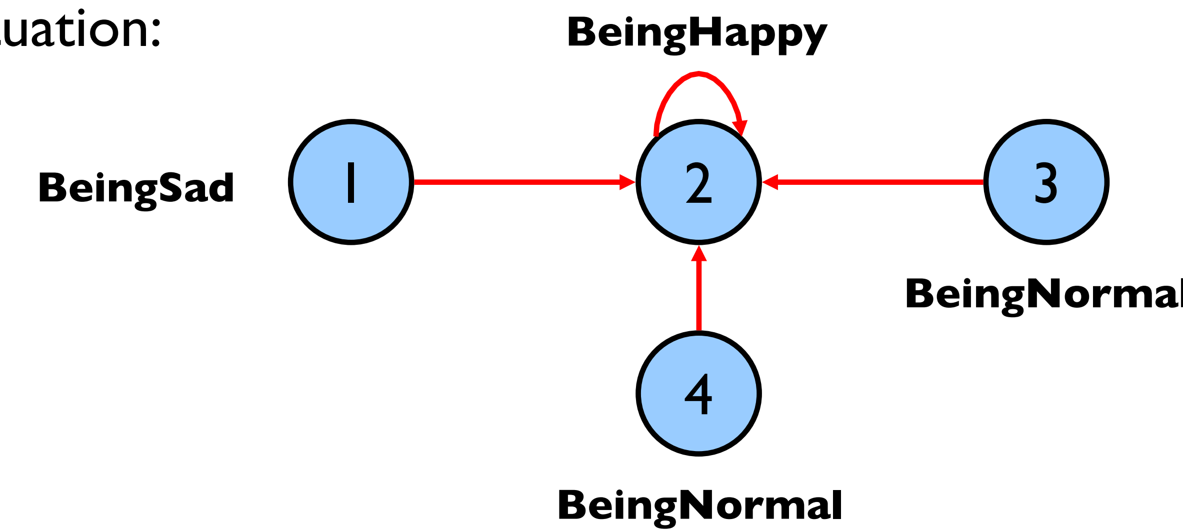
  I(BeingHappy) = {2} I(BeingSad) = {1} I(BeingNeutral) = {3, 4}

  M, 2 ⊨ BeingHappy M, 2 ⊨ ¬BeingSad

  M, 4 ⊨ □BeingHappy M, 1 ⊨ ◊BeingHappy M, 1 ⊨ ¬◊BeingSad

Satisfiability

- Consider the following situation:



- M = <W, R, I>

  W = {1, 2, 3, 4}

  R = {<1, 2>, <2, 2>, <3, 2>, <4, 2>}

  I(BeingHappy) = {2}   I(BeingSad) = {1}   I(BeingNormal) = {3, 4}

  Does □BeingHappy is satisifable?

  M, w ⊨ □BeingHappy for all w ∈ W, therefore □BeingHappy is satisfiable in M.

Validity relation on frames

A formula φ is valid in a world w of a frame F, in symbols
$F, w \vDash \varphi$ iff

$$M, w \vDash \varphi \text{ for all } I \text{ with } M = \langle F, I \rangle$$

A formula φ is valid in a frame F, in symbols $F \vDash \varphi$ iff

$$F, w \vDash \varphi \text{ for all } w \in W$$

If C is a class of frames, then a formula φ is valid in the class of frames C, in symbols $\vDash_C \varphi$ iff

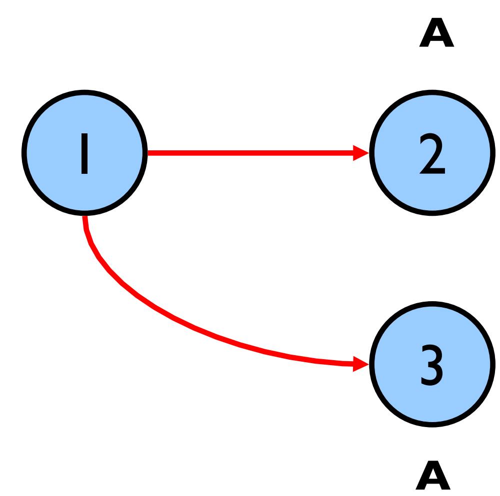
$F \vDash \varphi$ for all $F \in$ C

A formula φ is valid, in symbols $\vDash \varphi$ iff

$$F \vDash \varphi \text{ for all models frames } F$$

Validity

- Prove that P: $\Box A \rightarrow \Diamond A$ is valid



- In all models M = <W, R, I>,

(1) $\Box A$ means that for every $w \in W$ such that wRw' then $M, w' \vDash A$

(2) $\Diamond A$ means that for some $w \in W$ such that wRw' then $M, w' \vDash A$

It is clear that if (1) then (2) in the example

(as we will see this is valid in <u>serial frames</u>)

Kinds of frames

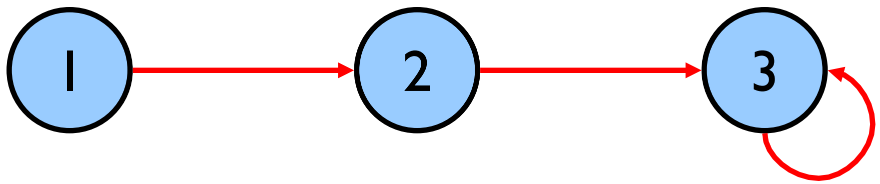
- Given the frame F = <W, R>, the relation R is said to be:

  - Serial iff for every $w \in W$, there exists $w' \in W$ s.t. wRw'
  - Reflexive iff for every $w \in W$, wRw
  - Symmetric iff for every $w, w' \in W$, if wRw' then $w'Rw$
  - Transitive iff for every $w, w', w'' \in W$, if wRw' and $w'Rw''$ then wRw''

- We call a frame <W, R> serial, reflexive, symmetric or transitive according to the properties of the relation R

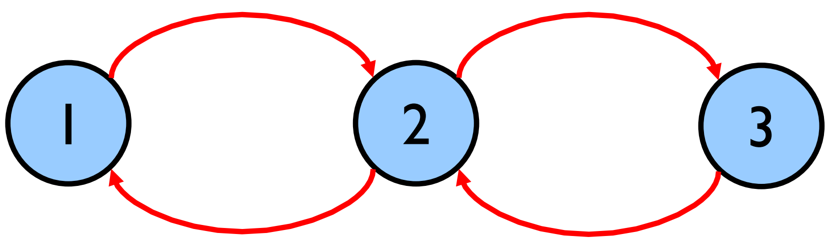
Kinds of frames

- Serial: for every w ∈ W, there exists w' ∈ W s.t. wRw'

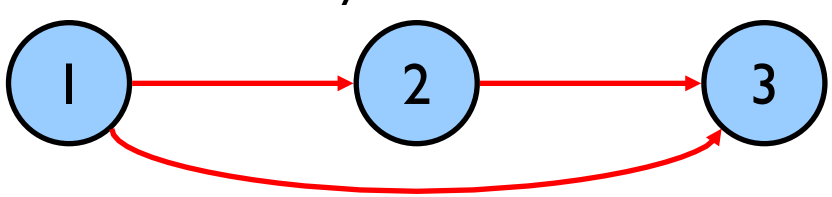


- Reflexive: for every w ∈ W, wRw

- Symmetric: for every w, w' ∈ W, if wRw' then w'Rw



- Transitive: for every w, w', w'' ∈ W, if wRw' and w'Rw'' then wRw''



Modal logic exercise

Given the Kripke model M = <W, R, I> with:

W = {1, 2, 3}   R = {<1, 2>, <2, 1>, <1, 3>, <3, 3>}   I(A) = {1, 2} and I(B) = {2, 3}

- **Say whether the frame <W, R> is serial, reflexive, symmetric or transitive.**
  - It is serial.
- **Is M, 1 $\models \Diamond(A \wedge B)$? Provide a proof for your response.**
  - Yes, because $A \wedge B$ is true in 2 and 2 is accessible from 1.
- **Is $\Box A$ satisfiable in M? Provide a proof for your response.**
  - We should have that M, w $\models \Box A$ for all worlds w. This means that for all worlds w there is a w' such that wRw' and M, w' $\models$ A.
  - For w = 1 we have 1R3 and M, 3 $\models \neg A$. Therefore the response is NO.