

Cse352
ARTIFICIAL INTELLIGENCE

Short REVIEW for FINAL

Professor Anita Wasilewska
Computer Science Department
Stony Brook University

Part 1: PREDICATE LOGIC CONCEPTUALIZATION

- Translations from Natural Language
- **BE CAREFUL!**
- **YOU MUST ALWAYS DO DIRECT TRANSLATION**
- **Never translate some logically EQUIVALENT FORM like in this case (via de Morgan Laws)**
- **“All houses are not red”**

PREDICATE LOGIC CONCEPTUALIZATION

- Translations from Natural Language
- Translate: **“All houses are not red”**
- 1. Domain: $X \neq \phi$
- 2. Predicates: $A(x)$ – x is a house $B(x)$ – x is red
- 3. Functions: (none)
- 4. Connectives: \neg - **“not”**
- 5. Quantifiers: $\forall_{A(x)}$ – **“All houses” (restricted)**
- 6. RESTRICTED FORMULA: $\forall_{A(x)} \neg B(x)$
- 7. LOGIC FORMULA: $\forall_x (A(x) \Rightarrow \neg B(x))$

PART 1: PREDICATE LOGIC CONCEPTUALIZATION

- Translations from Natural Language
- Translate: **“No house is red”**
- 1. Domain: $X \neq \phi$
- 2. Predicates: $A(x)$ – x is a House $B(x)$ – x is red
- 3. Functions: (none)
- 4. Connectives: \neg - **“not”**
- 5. Quantifiers: $\exists_{A(x)}$ – **“some houses”** (restricted)
- 6. RESTRICTED FORMULA: $\neg \exists_{A(x)} B(x)$
- 7. LOGIC FORMULA: $\neg \exists x (A(x) \wedge B(x))$

Part 2: Propositional Resolution

GOAL: Use Resolution to prove/ disapprove $\models A$

PROCEDURE

Step 1: Write $\neg A$ and transform $\neg A$ into set of clauses $CL_{\{\neg A\}}$ using Transformation rules

Step 2: Consider $CL_{\{\neg A\}}$ and look at if you can get a deduction of $\{\}$ from $CL_{\{\neg A\}}$

ANSWER

1. $CL_{\{\neg A\}} \vdash_R \{\}$ — Yes, $\models A$
2. $CL_{\{\neg A\}} \nvdash \{\}$ (i.e. you never get $\{\}$) — No, not $\models A$

Rules of transformation

- **Rules of transformation** of a formula A into a logically equivalent set of clauses CL_A
- **Rule (U): $(A \cup B)$ + Information**

What “Information” mean?

Example: $a, b, (a \cup \neg(a \Rightarrow b)), \neg c$

$a, b, a, \neg(a \Rightarrow b), \neg c$

$a, b, \neg c$ is Information

Rule (U) : $I, (A \cup B), J$

I, A, B, J

I, J --- Information around

Implication Rule (\Rightarrow)

• I, (A \Rightarrow B), J

(A \Rightarrow B)

I, \neg A, B, J

\neg A, B

Example: a, (a \cup b), (a \Rightarrow \neg a), (a \wedge b), c

(\Rightarrow)

a, (a \cup b), \neg a, \neg a, (a \wedge b), c

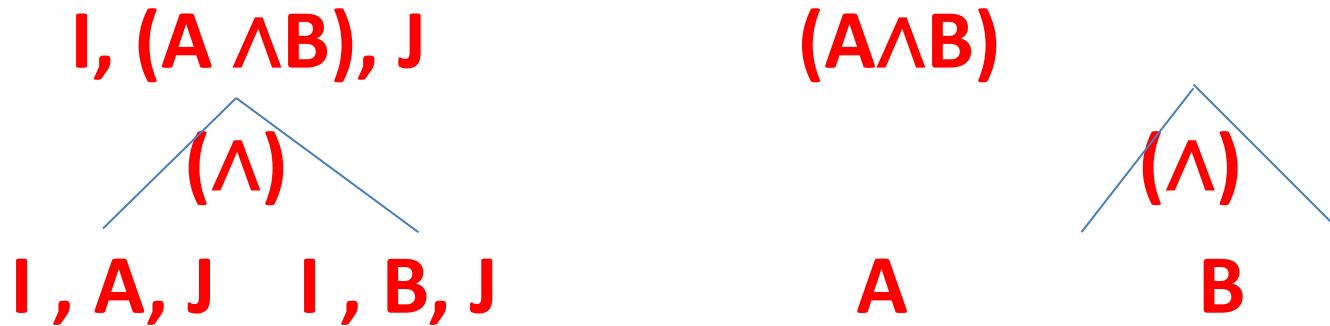
(\cup)

a, a, b, \neg a, \neg a, (a \wedge b), c

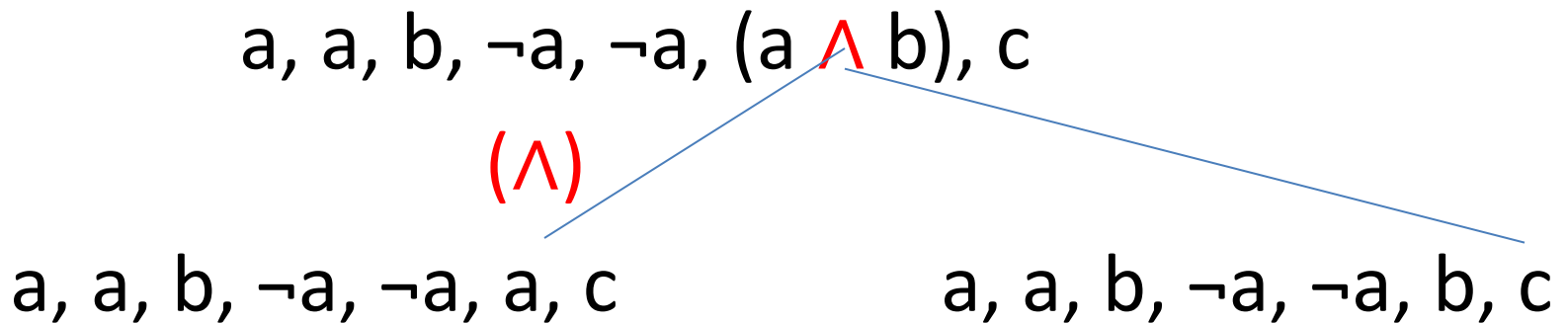
next step?

we need (\wedge) Rule!

Conjunction Rule (\wedge)



Example:



STOP when get **only literals**

Form clauses out of the **leaves**

Set of Clauses

Procedure: Leaves – to – Clauses

1. make **SETS** out of each leaf;
each leaf becomes a **clause C**

2. make a set of clauses **CL** as a **set of all clauses C** obtained in 1.

Leaf 1: $\{a, a, b, \neg a, \neg a, a, c\} = \{a, b, \neg a, c\}$

Leaf 2: $\{a, a, b, \neg a, \neg a, b, c\} = \{a, b, \neg a, c\}$

- Observe that we end-up with only **one set** of clauses
- $\mathbf{CL} = \{\text{Leaf 1, Leaf 2}\} = \{ \{a, b, \neg a, c\} \}$

Negation of Implication Rule ($\neg \Rightarrow$)

$I, \neg (A \Rightarrow B), I$

$(\neg \Rightarrow)$

$I, A, I \quad I, \neg B, I$

$\neg (A \Rightarrow B)$

$(\neg \Rightarrow)$

$A \quad \neg B$

Example:

$a, b, a, \neg (a \Rightarrow b), \neg c$

$(\neg \Rightarrow)$

$a, b, a, a, \neg c$

$a, b, a, \neg b, \neg c$

Stop – when only literals :

Form clauses out of $a, b, a, a, \neg c$ and

$a, b, a, \neg b, \neg c$

Clauses

- Leaf1: $a, b, a, a, \neg c$ makes clause $\{a, b, \neg c\}$
- Leaf 2: $a, b, a, \neg b, \neg c$ makes clause $\{a, b, \neg b, c\}$
- $CL = \{\{a, b, \neg c\}, \{a, b, \neg b, c\}\}$
- CL is set of clauses corresponding to $a, b, a, \neg (a \Rightarrow b), \neg c$

Negation of Conjunction Rule ($\neg\wedge$)

I, $\neg(A \wedge B)$, J

($\neg\wedge$)

I, $\neg A$, $\neg B$, J

$\neg(A \wedge B)$

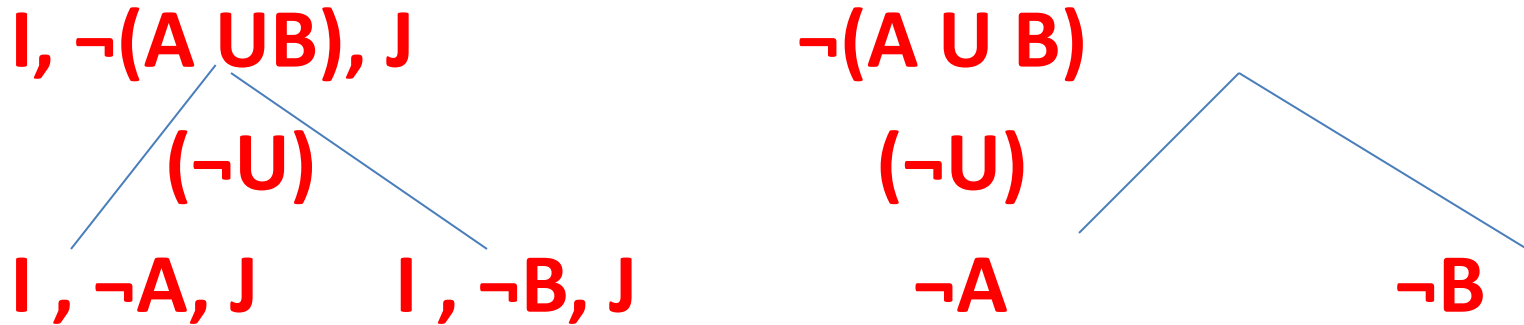
($\neg\wedge$)

$\neg A$, $\neg B$

Corresponds to DeMorgan Law

$$\neg(A \wedge B) \equiv (\neg A \cup \neg B)$$

Negation of Disjunction Rule ($\neg U$)



- Corresponds to DeMorgan Law:

$$\neg(A \cup B) \equiv (\neg A \wedge \neg B)$$

Negation of Negation Rule ($\neg\neg$)



Corresponds to

$$\neg\neg(A) \equiv A$$

Transformation Rules :

$$(\wedge), (\cup), (=>), (\neg\wedge), (\neg\cup), (\neg=>)$$

Transformation Rules Shorthand Form

$(A \cup B)$ (U)

A, B

$(A \wedge B)$ (\wedge)

A B

$(A \Rightarrow B)$ (\Rightarrow)

$\neg A, B$

$\neg\neg A$ ($\neg\neg$)

A

$\neg(A \cup B)$ ($\neg U$)

$\neg A$

$\neg B$

$\neg(A \wedge B)$ ($\neg\wedge$)

$\neg A, \neg B$

$\neg(A \Rightarrow B)$ ($\neg\Rightarrow$)

A

$\neg B$

+ Keep all Information

End when all leaves are literals

ARGUMENTS (rules of inference)

- From (premises) A_1, \dots, A_n we conclude B

$$\frac{A_1, \dots, A_n}{B}$$

Definition:

Argument $\frac{A_1, \dots, A_n}{B}$ is **VALID** iff

$$\models ((A_1 \wedge \dots \wedge A_n) \Rightarrow B)$$

ARGUMENTS

- Otherwise

Argument is **NOT VALID**

Valid Arguments \equiv Tautologically Valid

A_1, \dots, A_n, C

are formulas of **Propositional** or **Predicate**
Language

Validity of Arguments

Remember: $\models A$ iff $\models \neg A$

Tautology (always true), **Contradiction** (always false)

This means that if we want to **decide** $\models A$ we **decide** $\models \neg A$
and use **Resolution** for that

STEPS

Step 1: Negate A ; i.e. take $\neg A$ and **find** the set of clauses
corresponding to $\neg A$ i.e. **find** $CL_{\{\neg A\}}$

Step 2: Use **Completeness of Resolution**

$\models A$ iff $CL_{\{\neg A\}} \vdash_R \{\}$ i.e.

1. Look for a **deduction** of $\{\}$
2. if **YES** – we have $\models A$
3. If there is **no deduction** of $\{\}$ we have: $\models A$

Exercise

- **Prove By Propositional Resolution**

- $\models (\neg(a \Rightarrow b) \Rightarrow (a \wedge \neg b))$

Remember: $\models A$ iff $\models \neg A$ + use **Resolution**

Steps

Step 1: Find set of clauses corresponding to $\neg A$

i.e. $CL_{\{\neg A\}}$

Step 2: Find deduction of $\{\}$ from $CL_{\{\neg A\}}$

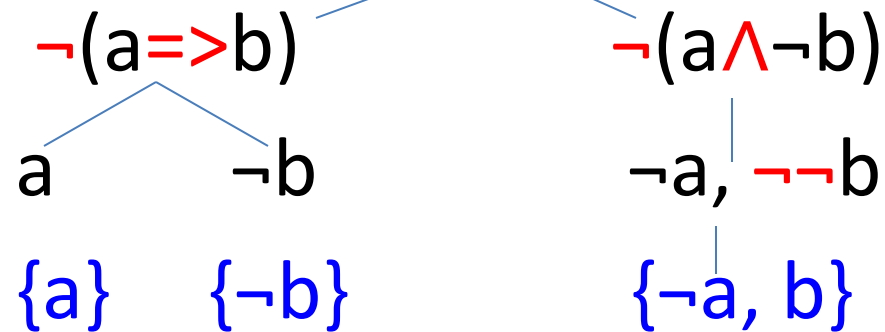
i.e. show that $CL_{\{\neg A\}} \vdash_R \{\}$

DO IT!

Exercise Solution

- **Step 1:** Negate A and find the set of clauses for $\neg A$
i.e. $\mathbf{CL}_{\{\neg A\}}$

- $\neg(\neg(a \Rightarrow b) \Rightarrow (a \wedge \neg b))$



$$\mathbf{CL}_{\{\neg A\}} = \{\{a\}, \{\neg b\}, \{\neg a, b\}\}$$

$\{b\}$

Step 2: Check if $\mathbf{CL}_{\{\neg A\}} \vdash_R \{\}$ – **YES!**

$\{\}$

Remark: $\models A$ iff there is **no** deduction of $\{\}$ from $\mathbf{CL}_{\{\neg A\}}$

Back To Arguments

- Use resolution to show that from A_1, \dots, A_n we can deduce B

“We can” deduce B from A_1, \dots, A_n means **validity** of argument $\frac{A_1, \dots, A_n}{B}$

B

iff by definition

$$\models (A_1 \wedge \dots \wedge A_n \Rightarrow B)$$

We have to use **Resolution** to prove that this is a **Tautology**

Arguments

$\models (A_1 \wedge \dots \wedge A_n \Rightarrow B)$ iff

$\models \neg (A_1 \wedge \dots \wedge A_n \Rightarrow B)$ iff

$\models (A_1 \wedge \dots \wedge A_n \wedge \neg B)$

- **Step 1:** we transform $(A_1 \wedge \dots \wedge A_n \wedge \neg B)$ to clauses

- Take A_1, \dots, A_n and find $CL_{A_1}, \dots, CL_{A_n}$

and also find $CL_{\neg B}$

and form

$CL_{A_1} \cup \dots \cup CL_{A_n} \cup CL_{\neg B} = CL$

Step 2: examine whether $CL \vdash_R \{\}$

Remember

- Argument A_1, \dots, A_n is **valid** iff

B

$$CL_{A_1} \cup \dots \cup CL_{A_n} \cup CL_{\neg B} \not\vdash_R \{ \}$$



Argument is **not valid**

iff **never** $CL_{A_1} \cup \dots \cup CL_{A_n} \cup CL_{\neg B} \vdash_R \{ \}$

We have some **Resolution Strategies** that allow us to cut down **number of cases** to consider

Part 3: Classification Learning Process

- **Classification process** operate in three stages:
 - Stage 1:** build the **basic patterns** structure
-training
 - Stage 2:** optimize **parameter settings**;
can use (N:N) re-substitution
- parameter tuning
 - Stage 3:** use **test data** to compute
predictive accuracy/error rate

Classifier, Model Terminology

- Books use the words “classifier” and “model” interchangeably
- Sometimes “classifier” means Stage 1 basic classifier model (rules, patterns) ready for **testing**
- Sometimes “classifiers” means classifiers models (rules, patterns) obtained **by training - testing** methods (like **k-fold cross validation**, repeated **holdout**, etc..). i.e. are the results of **Stages 1- 3**

Classifier, Model Terminology

- In some cases the term “**learned models**”
- or “**base classifiers**” are used for results of
- **Stages 1-3**

- It happens when the method is presented how to **combine** them in a way that would the best to return a **class prediction** for unknown records, i.e. to **build the final**
- **CLASSIFIER**

TESTING

Define a holdout procedure

- **Holdout procedure**
is a method of splitting original data into training and test data sets

TESTING

Describe shortly the two main methods of **predictive accuracy** evaluations

- (1) **k-fold cross-validation** ($N - N/k ; N/k$)
- (2) **Leave-one-out** ($N - 1 ; 1$)

TESTING

(1) **k-fold cross-validation** ($N - N/k ; N/k$)

First step:

split data into k **disjoint** subsets

$D_1, \dots, D_k,$

of equal size, called **folds**

Second step:

use each subset in turn for **testing**, the remainder for **training**

Training and **testing** is performed k times

TESTING

- (2) **Leave-one-out** (**$N-1$; 1**)

Leave-one-out is a particular form of **cross-validation**

- We set **number** of **folds** to number of **training instances**, i.e. we put **$k = N$** for **N instances**
- We **repeat** the **training – testing** cycle **N times**

Correctly and Not Correctly Classified

- A **test data record is correctly classified** if and only if the following conditions hold:
 - (1) we **can classify** the record, i.e. there is a **pattern** or a **rule** such that its **LEFT** side **matches** the record,
 - (2) **classification determined by the pattern** or the **rule is correct**, i.e. the **RIGHT** side of the rule **matches** the value of the record's **class attribute**

OTHERWISE

- **the record is not correctly classified**
- **Words used:**
 - **not correctly = incorrectly = misclassified**
 - **Validation data = Test data** or a subset of Test Data

Re-substitution Error Rate

- **Re-substitution error rate** is obtained from **training data**
- **Training Data Error: uncertainty of the rules**
- **The error rate is not always 0%**, but usually (and hopefully) very low!
- **Re-substitution error rate** indicates only how **good (bad)** are our **results** (rules, patterns, NN) on the **TRAINING data**
- It expresses some **knowledge about the algorithm** used

Re-substitution Error Rate

- **Re-substitution error rate** is usually used as the **performance measure**:

The **training error rate** reflects **imprecision** of the training results

The lower training error rate the better

In the case of **rules** it is called **rules accuracy**

Predictive Accuracy

Predictive accuracy reflects how **good** are the **training results** with respect to the **test data**

The higher predictive accuracy **the better**

(N:N) re-substitution **does not** compute **predictive accuracy**

- Re-substitution error rate = **training data error rate**

Validation Data

- Proper **classification process** uses three sets of data:
- **training data, validation data and test data**
- Validation data is **not used** for **parameter tuning**
- **Training data** is NOT validation data
- Validation data is the **test data**, or a subset
- of the **test data**
- The **Test data can not** be used for the **parameter tuning!**

Classifier, Model Terminology

- When a book talks about **comparison of classifiers**, “**classifier**” means comparison of classifiers models (rules, patterns) obtained by **train- test methods** i.e. means comparison results of **Stages 1- 3**
- These **comparison methods** or other methods are called “**model selection**”
- Their goal is to **choose** the best one to be
- **THE CLASSIFIER-**
- the final product that would the best **classify unknown records**