INST0072 Lecture 3: Propositional and Predicate Logic

1. Recap From Lecture 1
2. Soundness and Completeness
3. Is This Inference Rule Sound?
4. Resolution and Conjunctive Normal Form
5. Literals, Clauses and Conjunctive Normal Form (CNF)
6. Examples of CNF Formulas and Theories
7. Which of These Formulas Are In CNF?
8. Converting Formulas to CNF
9. The General Resolution Rule
10. The Empty Clause and Other Special Cases
11. The Factorisation Rule and the Resoultion Calculus
12. Proof by Contradiction Using Resolution
13. An Example Proof Using Resolution
14. An Example Resolution Proof / continued
15. Horn Clauses and Backward Chaining
16. Predicate Logic
17. Mathematical Symbols and Equality
18. Atomic Formulas, Literals, Formulas and Sentences
19. Example Sentences (mostly from Ertel)
20. Some Practice with Quantifiers
21. Relations over a Domain of Discourse
22. Unary and Binary Relations
23. Functions Over a Domain of Discourse
24. Signatures, Interpretations and Variable Assignments
25. An Example Interpretation $\mathbb{I}$ and Variable Assignment $\mathbb{V}$
26. Example Interpretations with a Small Domain of Discourse
27. Satisfaction of Quantifer-free Formulas - An Example
28. More Examples of Satisfaction

1. Recap From Lecture 1

- **The Deduction Theorem**
  For any two formulas A and B
  $$A \models B \quad \text{if and only if} \quad \models A \rightarrow B$$

- An inference rule $\frac{P_1, \ldots, P_n}{C}$ is *sound* if the conclusion is always a semantic consequence of the premises.
- An example derivation from $KB = \{a,\ a \rightarrow \neg b,\ c \rightarrow b\}$:

$$\{a,\ a \rightarrow \neg b,\ c \rightarrow b\} \vdash \neg c$$

derivation uses modus ponens followed by modus tollens:

$$\{a,\ a \rightarrow \neg b,\ c \rightarrow b\} \Rightarrow \frac{a,\ a \rightarrow \neg b}{\neg b} \Rightarrow \{a,\ a \rightarrow \neg b,\ c \rightarrow b,\ \neg b\}$$
$$\Rightarrow \frac{c \rightarrow b,\ \neg b}{\neg c} \Rightarrow \{a,\ a \rightarrow \neg b,\ c \rightarrow b,\ \neg b,\ \neg c\}$$

2. Soundness and Completeness

Given a particular propositional calculus (i.e. a particular set of inference rules),

- the calculus is *sound* if, for any set of formulas KB and formula F,
  $$\text{if } KB \vdash F \text{ then } KB \models F$$
- the calculus is *complete* if, for any set of formulas KB and formula F,
  $$\text{if } KB \models F \text{ then } KB \vdash F$$

So *soundness* ensures that we cannot use the calculus to infer a formula that is not a semantic consequence of the knowledge base. *Completeness* ensures that we can use the calculus to infer any formula that is a semantic consequence of the knowledge base.

3. Is This Inference Rule Sound?

- Use the Deduction Theorem, the definition of soundness of an inference rule, and a truth table to show that the following inference rule is, or is not, sound:

$$\frac{A \vee B,\ \ A \to C,\ \ B \to C}{C}$$

4. Resolution and Conjunctive Normal Form

- *Resolution* is a propositional calculus with just two inference rules, the *general resolution rule* and the *factorisation rule*.
- It works on theories and formulas written in *conjunctive normal form* (CNF).
- It is important for A.I. and knowledge representation for three reasons:
  1. it is sound, and complete with respect to proofs by contradiction,
  2. in many cases it can be automated reasonably efficiently (e.g. Prolog uses resolution), and
  3. any propositional formula can be transformed into CNF.

5. Literals, Clauses and Conjunctive Normal Form (CNF)

Assume a signature of propositions $p_1, \ldots, p_n$.

- A *literal* is a proposition or its negation. In other words, all the formulas $p_1, \neg p_1, \ldots, p_n, \neg p_n$ are literals.
- A *clause* is either a literal or a *disjunction* of literals, i.e. it is a formula of the general form $(L_i \vee \ldots \vee L_k)$, where each L is either p_j or $\neg p_j$ for some proposition p_j in the signature.
- A formula is in *conjunctive normal form* (CNF) if it is a clause or a conjunction of clauses, i.e. if it is of the general form $C_1 \wedge \ldots \wedge C_m$, where each C is a clause.
- Similarly, a theory or knowledge base is in CNF if it is a set $\{C_1, \ldots, C_m\}$ of clauses.

6. Examples of CNF Formulas and Theories

For signature $\{red, big, soft, new\}$ the following formulas are in CNF:

- $(red \vee \neg soft \vee big) \wedge (\neg new \vee \neg big) \wedge soft$
- $\neg soft \vee big$
- $soft \wedge big$
- $\neg new$
- $(red \vee big \vee soft \vee new) \wedge (\neg red \vee \neg big \vee \neg soft \vee \neg new)$

For signature $\{touch\_wire, electric\_shock\}$ the following theory (knowledge base) is in CNF:

$$\{\neg touch\_wire \vee electric\_shock,\\ touch\_wire \vee \neg electric\_shock,\\ touch\_wire\}$$

7. Which of These Formulas Are In CNF?

1. $red \vee big \vee soft \vee new$
2. $red \wedge big \wedge soft \wedge new$
3. $soft \wedge \neg soft$
4. $soft \vee \neg soft$
5. $(red \vee big) \wedge (soft \vee new)$
6. $red \vee (big \wedge soft) \vee new$
7. $\neg(red \vee big \vee soft \vee new)$
8. $\neg\neg(red \wedge big \wedge soft \wedge new)$

8. Converting Formulas to CNF

- We can use the equivalences on slide 12 of Lecture 1 to convert any propositional formula to CNF. For example:

$$
\begin{array}{lr}
(big \vee new) \rightarrow (red \wedge soft) & \\
\equiv \neg(big \vee new) \vee (red \wedge soft) & \text{[implication]} \\
\equiv (\neg big \wedge \neg new) \vee (red \wedge soft) & \text{[de Morgan]} \\
\equiv ((\neg big \wedge \neg new) \vee red) \wedge ((\neg big \wedge \neg new) \vee soft) & \text{[distribution]} \\
\equiv (red \vee (\neg big \wedge \neg new)) \wedge (soft \vee (\neg big \wedge \neg new)) & \text{[}\vee\text{ reordering]} \\
\equiv ((red \vee \neg big) \wedge (red \vee \neg new)) \wedge ((soft \vee \neg big) \wedge (soft \vee \neg new)) & \text{[}\vee\text{ distribution]} \\
\equiv (red \vee \neg big) \wedge (red \vee \neg new) \wedge (soft \vee \neg big) \wedge (soft \vee \neg new) & \text{[}\wedge\text{ associativity]}
\end{array}
$$

- Note that this is semantically equivalent to the knowledge base $\{red \leftarrow big, red \leftarrow new, soft \leftarrow big, soft \leftarrow new\}$. (Later, we will see how this corresponds to a Prolog program.)
- We will refer to the conversion of formula A into CNF as $\mathbf{cnf}[A]$, and the equivalent set of clauses as $\mathbf{cnfset}[A]$. e.g:

$$\mathbf{cnf}[(\neg red \vee \neg soft) \rightarrow \neg big] = (red \vee \neg big) \wedge (soft \vee \neg big)$$
$$\mathbf{cnfset}[(\neg red \vee \neg soft) \rightarrow \neg big] = \{(red \vee \neg big), (soft \vee \neg big)\}$$

9. The General Resolution Rule

The *general resolution rule* is an inference rule that works on a pair of clauses as follows. Let $L_1, \ldots, L_i, \ldots L_m, L'_1, \ldots, L'_j, \ldots L'_n$ be literals and p be a proposition. Then both p and $\neg p$ are also literals, so that the two formulas

$$(L_1 \vee \ldots \vee L_i \vee p \vee L_{i+1} \vee \ldots \vee L_m)$$

and

$$(L'_1 \vee \ldots \vee L'_j \vee \neg p \vee L'_{j+1} \vee \ldots \vee L'_n)$$

are both clauses. The general resolution rule says that we can eliminate the p and $\neg p$ and combine the remaining literals into one clause:

$$\frac{(L_1 \vee \ldots \vee L_i \vee p \vee L_{i+1} \vee \ldots \vee L_m), \quad (L'_1 \vee \ldots \vee L'_j \vee \neg p \vee L'_{j+1} \vee \ldots \vee L'_n)}{(L_1 \vee \ldots \vee L_m \vee L'_1 \vee \ldots \vee L'_n)}$$

Note that p and $\neg p$ can be positioned anywhere in the two respective premises (including at the beginning or at the end).

10. The Empty Clause and Other Special Cases

A special case of the general resolution rule is when the premises have no literals other than p and $\neg p$, in which case the conclusion is the *empty clause*:

$$\frac{(p), \quad (\neg p)}{(\,)}$$

Many descriptions of resolution use a special symbol for the empty clause. In this module we will use the symbol $\bot$, so the special case above can be written:

$$\frac{p, \quad \neg p}{\bot}$$

Since $p \wedge \neg p$ is a contradiction, the empty clause $\bot$ is regarded as a contradition as well, and so can only take the truth value *false* (f).

Note that, since $a \rightarrow b$ is equivalent to $\neg a \vee b$, when applied to single propositions (e.g. a and b) both modus ponens ($\frac{a, \ \neg a \vee b}{b}$) and modus tolens ($\frac{\neg a \vee b, \ \neg b}{\neg a}$) are also special cases of resolution.

11. The Factorisation Rule and the Resoultion Calculus

To ensure that the resolution calculus can be effectively applied in all cases, it is necessary to be able to delete repeated literals from clauses, so we include the *factorisation rule*:

$$\frac{(L_1 \vee \ldots \vee L_i \vee L \vee L_{i+1} \vee \ldots \vee L_j \vee L \vee L_{j+1} \vee \ldots \vee L_k)}{(L_1 \vee \ldots \vee L_i \vee L \vee L_{i+1} \vee \ldots \vee L_j \vee L_{j+1} \vee \ldots \vee L_k)}$$

Resolution is the propositional calculus that uses the general resolution and factorisation inference rules to generate new clauses from a CNF knowledge base of clauses. We write

$$KB \vdash_{res} C$$

to signify that it is possible to derive the clause C from the knowledge base KB of clauses using the resolution calculus.

12. Proof by Contradiction Using Resolution

- Recall the principle of *proof by contradiction* from Lecture 1, Slide 15: for a theory KB and formula F, $KB \models F$ if and only if $KB \cup \{\neg F\}$ is unsatisfiable (i.e. a contradiction).
- We can use this principle in the resolution calculus to construct a proof of any formula F that is entailed from a CNF knowledge base KB. We do this by converting the negation of the formula ($\neg F$) to CNF, adding this to the knowledge base, and then proving the contradiction $\bot$.
- The following theorem states that the resolution calculus used in this way is sound and complete:

Resolution Calculus Soundness and Completeness
For any CNF knowledge base KB and formula F

$$KB \models F \quad \text{if and only if} \quad KB \cup \mathbf{cnfset}[\neg F] \vdash_{res} \bot$$

13. An Example Proof Using Resolution

$$KB = \{(\neg big \vee \neg soft \vee new \vee red), (soft \vee \neg big), (\neg red \vee new), (\neg red \vee big)\}$$

Show that $KB \models big \rightarrow new$

- By resolution soundness (see slide 12), it is sufficient to show that

$$KB \cup \mathbf{cnfset}[\neg(big \rightarrow new)] \vdash_{res} \bot$$

- To find $\mathbf{cnf}[\neg(big \rightarrow new)]$:

$\neg(big \rightarrow new)$

$\equiv \neg(\neg big \vee new)$ [implication]

$\equiv \neg\neg big \wedge \neg new$ [de Morgan]

$\equiv big \wedge \neg new$ [¬ cancellation]

- So $\mathbf{cnfset}[\neg(big \rightarrow new)] = \{big,\ \neg new\}$

14. An Example Resolution Proof / continued

$KB \cup \mathbf{cnfset}[\neg(big \rightarrow new)] = \{$

(1) $(\neg big \vee \neg soft \vee new \vee red)$,

(2) $(soft \vee \neg big)$,

(3) $(\neg red \vee new)$,

(4) $(\neg red \vee big)$,

(5) (big),

(6) $(\neg new)$ $\}$

Derivation:

(7)	$(\neg soft \vee new \vee red)$	by (1), (5), resolution
(8)	$(\neg big \vee new \vee red)$	by (7), (2), resolution
(9)	$(\neg big \vee red)$	by (8), (6), resolution
(10)	(red)	by (9), (5), resolution
(11)	(new)	by (10), (3), resolution
(12)	$\bot$	by (11), (6), resolution

15. Horn Clauses and Backward Chaining

- Clauses with at most one positive literal are called *Horn clauses*.
- The Horn clause $(\neg L_1 \vee \ldots \vee \neg L_n \vee L)$ is semantically equivalent to

$$L \leftarrow L_1 \wedge \ldots \wedge L_n$$

  When the clause is re-written this way, L is called the *head* of the clause, and $L_1 \wedge \ldots \wedge L_n$ is called the *body*.
- A Horn clause of the form L (with no negative literals) is called a *fact*.
- A Horn clause of the form $(\neg L_1 \vee \ldots \vee \neg L_n)$ (with no positive literal) can be re-written as $\neg(L_1 \wedge \ldots \wedge L_n)$. When re-written in this form it is called a *constraint*.
- Some types of Prolog programs can be regarded as collections of Horn clauses. More about this in the book by Ertel and in later lectures.

16. Predicate Logic

- Predicate logic is concerned with describing *relationships between objects*, and the ways in which different these relationships are inter-connected.
- A collection of all the objects to which the logic might refer is called a *domain of discourse*.
- Three types of symbols are used to refer to individual objects:
  - *Constants*, such as *rob*, *room_243*, 3.14 or *4th_july_1776*.
  - *Function symbols*, such as *father_of*(_) or *sum_of*(_ , _) that refer to an object in terms of another object or objects.
  - *Variables*, signified by single letters $x, x_1, \ldots, y, y_1, \ldots$ etc, that act as placeholders for object names and can be *quantified* with the symbols $\forall$ ("for all") or $\exists$ ("there exists").
- *Predicate symbols* such as *is_bigger_than*(_ , _) or *is_hungry*(_) take *terms* signifying objects as *arguments*, and (like propositions) can be *true* or *false*, depending on the argument values.

17. Mathematical Symbols and Equality

- Many common mathematical symbols are, logically speaking, either function symbols or predicate symbols.
- When these symbols take two arguments (i.e. are *binary*) they often use *infix* notation. For example:
  - We write $3 + 2$, rather than $+(3, 2)$, for the function $+$.
  - We write $4 > 2$, rather than $>(4, 2)$, for the predicate $>$.
- Other examples of maths functions: $-$, $\times$, $log(\_)$, $sin(\_)$, etc.
- Other examples of maths predicates: $<$, $\leq$, $\geq$, $=$, $\neq$.
- $=$ and $\neq$ are particularly important predicate symbols, also used widely outside of mathematics:
  - $A = B$ means *"A and B refer to the same object in the domain of discourse"*.
  - $A \neq B$ means *"A and B refer to different objects in the domain of discourse"*.

18. Atomic Formulas, Literals, Formulas and Sentences

- An *atomic formula* is a predicate symbol with the arguments filled in with object terms. For example:
  - $is\_bigger\_than(rob, father\_of(rob))$
  - $is\_hungry(father\_of(x))$
- A *literal* is an atomic formula or its negation:
  - $\neg is\_bigger\_than(x, y)$
  - $\neg is\_hungry(father\_of(father\_of(father\_of(x))))$
- A *formula* is literal, or a collection of literals combined with logical connectives (and possibly with quantifiers as well):
  - $is\_bigger\_than(y, rob) \rightarrow \neg is\_hungry(father\_of(y))$
- A *sentence* is formula with all variables in the *scope* of a quantifier (to give them a *meaning* within the formula):
  - $\forall x \forall y.(is\_bigger\_than(x, y) \rightarrow \neg is\_bigger\_than(y, x))$
  - $\forall x \exists y.is\_bigger\_than(y, x)$
  - $\exists y \forall x.is\_bigger\_than(y, x)$

19. Example Sentences (mostly from Ertel)

Sentence	**Intended Meaning**
$\forall x.[frog(x) \rightarrow green(x)]$	All frogs are green.
$\forall x.[(frog(x) \wedge brown(x)) \rightarrow big(x)]$	All brown frogs are big.
$\forall x.likes(x, cake)$	Everyone likes cake.
$\neg\forall x.likes(x, cake)$	Not everyone likes cake.
$\exists x.\neg likes(x, cake)$	Somebody doesn't like cake.
$\neg\exists x.likes(x, cake)$	Nobody likes cake.
$\forall x.\neg likes(x, cake)$	Nobody likes cake.
$\exists x\forall y.likes(y, x)$	There is something that everyone likes.
$\exists x\forall y.likes(x, y)$	Somebody likes everything.
$\forall x\exists y.likes(y, x)$	Everything is liked by someone.
$\forall x\exists y.likes(x, y)$	Everyone likes something.
$\exists x.[baker(x) \wedge \forall y.(customer(y) \rightarrow likes(x, y))]$	There is a baker who likes all of her customers.

20. Some Practice with Quantifiers

Using predicates $is\_bigger\_than(\_\,,\_)$, $is\_insect(\_)$, $is\_bird(\_)$ and $\_ \neq \_$, and appropriately quantified variables, express the following phrases as logical sentences:

- "Birds are bigger than insects."
- "Some birds are bigger than others."
- "Insects are not birds."
- "Some things are neither insects nor birds."

21. Relations over a Domain of Discourse

- *Reminder:* a collection of objects to which a predicate logic might refer is called a *domain of discourse*. Mathematically, a domain of discourse is a *non-empty set*, e.g. $\mathbb{D} = \{\bigcirc, \triangle, \square\}$.
- For any positive whole number n, the *n-ary Cartesian power* of a set S, written S^n or $S \times ... \times S$, is the set of all n-tuples that can be constructed from its members.
- For example, $\mathbb{D}^2 = \mathbb{D} \times \mathbb{D} = \{(\bigcirc, \bigcirc), (\bigcirc, \triangle), (\bigcirc, \square),$
  $(\triangle, \bigcirc), (\triangle, \triangle), (\triangle, \square), (\square, \bigcirc), (\square, \triangle), (\square, \square)\}$.
- A *relation of arity n* over a set S is a subset of S^n (possibly empty, possibly all of S^n, possibly something in between).
- For example, if $\triangle$ and $\square$ are the daughters of $\bigcirc$, then the relation "is father of" of arity 2 over D is $\{(\bigcirc, \triangle), (\bigcirc, \square)\}$, and the relation "is sister of" is $\{(\square, \triangle), (\triangle, \square)\}$.

22. Unary and Binary Relations

- A *unary relation* is a relation of arity 1.
- Question: how many unary relations is it possible to form over the set $\mathbb{D} = \{\bigcirc, \triangle, \square\}$?
- A *binary relation* is a relation of arity 2.
- Question: how many binary relations is it possible to form over the set $\mathbb{D} = \{\bigcirc, \triangle, \square\}$?

23. Functions over a Domain of Discourse

- A *function of arity n* over a set S is a one-to-one or a many-to-one mapping from all of S^n to S.
- For example, if $\mathbb{D} = \{\bigcirc, \triangle, \square\}$, then one such binary function (i.e. function of arity 2) over $\mathbb{D}$ is as follows:

$$\begin{array}{lll} (\bigcirc, \bigcirc) \mapsto \bigcirc & (\bigcirc, \triangle) \mapsto \triangle & (\bigcirc, \square) \mapsto \bigcirc \\ (\triangle, \bigcirc) \mapsto \triangle & (\triangle, \triangle) \mapsto \bigcirc & (\triangle, \square) \mapsto \triangle \\ (\square, \bigcirc) \mapsto \bigcirc & (\square, \triangle) \mapsto \triangle & (\square, \square) \mapsto \bigcirc \end{array}$$

- *Many-to-one* means that different n-tuples of objects may be mapped to the same object, but each n-tuple is mapped to exactly one object.
- Another example: if $\mathbb{D} = \{0, 1\}$, the binary "times" ($\times$) function is:

$$(0,0) \mapsto 0 \qquad (0,1) \mapsto 0 \qquad (1,0) \mapsto 0 \qquad (1,1) \mapsto 1$$

24. Signatures, Interpretations and Variable Assignments

- The *signature* of a predicate logic is a triple $\langle \mathcal{K}, \mathcal{F}, \mathcal{P} \rangle$, where $\mathcal{K}$ is the set of constant symbols in the language, $\mathcal{F}$ is the set of function symbols, and $\mathcal{P}$ is the set of predicate symbols.
- The symbols in $\mathcal{F}$ and $\mathcal{P}$ each have an associated *arity* (number of arguments). "$/n$" written after the symbol means it has arity n (e.g. *likes*$/2$ or *team_leader*$/5$).
- An *interpretation* $\mathbb{I}$ of $\langle \mathcal{K}, \mathcal{F}, \mathcal{P} \rangle$ is a domain of discourse $\mathbb{D}$ together with an assignment (i.e. mapping) of each symbol in $\mathcal{K}$ to an object in $\mathbb{D}$, an assignment of each symbol in $\mathcal{F}$ to a function over $\mathbb{D}$ of the same arity, and an assignment of each symbol in $\mathcal{P}$ to a relation over $\mathbb{D}$ of the same arity.
- For a given domain of discourse $\mathbb{D}$, a *variable assignment* $\mathbb{V}$ is an assignment of each variable in the logic to an object in $\mathbb{D}$.

25. An Example Interpretation $\mathbb{I}$ and Variable Assignment $\mathbb{V}$

- Suppose the signature of a particular predicate logic is $\langle\{mani, nina\}, \{partner\_of/1\}, \{likes/2\}\rangle$, and the logic uses three variable symbols x, y and z.
- To construct an example interpretation $\mathbb{I}$ and example variable assignment $\mathbb{V}$ we will use the example domain of discourse $\mathbb{D} = \{\bigcirc, \triangle, \square, \star\}$.
- Interpretation of constants: $\mathbb{I}(mani) = \square$, $\mathbb{I}(nina) = \triangle$.
- Interpretation of functions:
  $\mathbb{I}(partner\_of) = \langle\ \bigcirc \mapsto \triangle,\ \triangle \mapsto \bigcirc,\ \square \mapsto \star,\ \star \mapsto \square\ \rangle$.
- Interpretation of predicates:
  $\mathbb{I}(likes) = \{\ (\bigcirc, \star),\ (\star, \bigcirc),\ (\triangle, \square),\ (\square, \triangle)\ \}$.
- Assignment of variables: $\mathbb{V}(x) = \bigcirc$, $\mathbb{V}(y) = \triangle$, $\mathbb{V}(z) = \bigcirc$.

26. Example Interpretations with a Small Domain of Discourse

- How many interpretations are possible of the signature

  $$\langle\{mani, nina\}, \{partner\_of/1\}, \{likes/2\}\rangle$$

  using the domain of discourse $\mathbb{D} = \{\star\}$?

27. Satisfaction of Quantifer-free Formulas - An Example

Going back to the example from slide 25:

signature: $\langle \{mani, nina\}, \{partner\_of/1\}, \{likes/2\} \rangle$
$\mathbb{D} = \{\bigcirc, \triangle, \square, \star\}$, interpretation $\mathbb{I}$, variable assignment $\mathbb{V}$
$\mathbb{I}(mani) = \square$, $\mathbb{I}(nina) = \triangle$
$\mathbb{I}(partner\_of) = \langle\ \bigcirc \mapsto \triangle,\ \triangle \mapsto \bigcirc,\ \square \mapsto \star,\ \star \mapsto \square\ \rangle$
$\mathbb{I}(likes) = \{\ (\bigcirc, \star),\ (\star, \bigcirc),\ (\triangle, \square),\ (\square, \triangle)\ \}$
variables x, y and z: $\mathbb{V}(x) = \bigcirc$, $\mathbb{V}(y) = \triangle$, $\mathbb{V}(z) = \bigcirc$

The quantifier-free formula

$$\neg likes(x, partner\_of(x)) \wedge likes(mani, nina)$$

is said to be *satified* by $\mathbb{I}$ and $\mathbb{V}$, because $\mathbb{I}(partner\_of) : \bigcirc \mapsto \triangle$, $(\bigcirc, \triangle) \notin \mathbb{I}(likes)$ and $(\square, \triangle) \in \mathbb{I}(likes)$. So we write:

$$\langle \mathbb{I}, \mathbb{V} \rangle \models \neg likes(x, partner\_of(x)) \wedge likes(mani, nina)$$

28. More Examples of Satisfaction

Given:

signature: $\langle \{mani, nina\}, \{partner\_of/1\}, \{likes/2\} \rangle$
$\mathbb{D} = \{\bigcirc, \triangle, \square, \star\}$, interpretation $\mathbb{I}$, variable assignment $\mathbb{V}$
$\mathbb{I}(mani) = \square$, $\mathbb{I}(nina) = \triangle$
$\mathbb{I}(partner\_of) = \langle\ \bigcirc \mapsto \triangle,\ \triangle \mapsto \bigcirc,\ \square \mapsto \star,\ \star \mapsto \square\ \rangle$
$\mathbb{I}(likes) = \{\ (\bigcirc, \star),\ (\star, \bigcirc),\ (\triangle, \square),\ (\square, \triangle)\ \}$
variables x, y and z: $\mathbb{V}(x) = \bigcirc$, $\mathbb{V}(y) = \triangle$, $\mathbb{V}(z) = \bigcirc$

which of the following statements are correct?

- $\langle \mathbb{I}, \mathbb{V} \rangle \models likes(y, partner\_of(partner\_of(x))) \rightarrow likes(mani, x)$
- $\langle \mathbb{I}, \mathbb{V} \rangle \models likes(partner\_of(nina), partner\_of(mani))$
- $\langle \mathbb{I}, \mathbb{V} \rangle \models \neg likes(y, partner\_of(partner\_of(mani)))$