

Logic and Modelling

— Miscellaneous —

Jörg Endrullis

VU University Amsterdam

Databases, Queries and Predicate Logic

Queries in Databases: Relational Model

PERSONS		
<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

Queries in Databases: Relational Model

PERSONS		
<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

Relational Model

The table is modelled by a ternary predicate *Person*:

Person(101, Ann, Amsterdam)

Likewise, the predicate holds for every row of the table.

Queries in Databases: Relational Model

PERSONS		
<u>PID</u>	<u>Name</u>	<u>City</u>
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

Relational Model

The table is modelled by a ternary predicate *Person*:

Person(101, Ann, Amsterdam)

Likewise, the predicate holds for every row of the table.

We can formulate **queries using free variables**.

The names of people living in Amsterdam:

Queries in Databases: Relational Model

PERSONS		
<u>PID</u>	<u>Name</u>	<u>City</u>
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

Relational Model

The table is modelled by a ternary predicate *Person*:

$$Person(101, Ann, Amsterdam)$$

Likewise, the predicate holds for every row of the table.

We can formulate **queries using free variables**.

The names of people living in Amsterdam:

$$\{ y \mid \exists i Person(i, y, Amsterdam) \}$$

Queries in Databases: Relational Model

PERSONS		
<u>PID</u>	<u>Name</u>	<u>City</u>
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

Relational Model

The table is modelled by a ternary predicate *Person*:

Person(101, Ann, Amsterdam)

Likewise, the predicate holds for every row of the table.

We can formulate **queries using free variables**.

The names of people living in Amsterdam:

$$\{ y \mid \exists i \text{ Person}(i, y, \text{Amsterdam}) \} = \{ \text{Ann}, \text{Mia} \}$$

Queries in Databases: Tuple Model

Tuple Model

In this model, the **rows are objects** themselves (tuples).

Queries in Databases: Tuple Model

Tuple Model

In this model, the **rows are objects** themselves (tuples).

A unary predicate *Person* tells if a row belongs to table Persons:

$$Person(x)$$

Queries in Databases: Tuple Model

Tuple Model

In this model, the **rows are objects** themselves (tuples).

A unary predicate *Person* tells if a row belongs to table Persons:

$$Person(x)$$

Functions symbols for every column of the table:

Pid(x)

Name(x)

City(x)

Queries in Databases: Tuple Model

Tuple Model

In this model, the **rows are objects** themselves (tuples).

A unary predicate *Person* tells if a row belongs to table Persons:

$$Person(x)$$

Functions symbols for every column of the table:

Pid(x)

Name(x)

City(x)

(project the tuple/row to one of its arguments $\langle y_1, \dots, y_n \rangle \mapsto y_i$)

Queries in Databases: Tuple Model

Tuple Model

In this model, the **rows are objects** themselves (tuples).

A unary predicate *Person* tells if a row belongs to table Persons:

$$Person(x)$$

Functions symbols for every column of the table:

Pid(x)

Name(x)

City(x)

(project the tuple/row to one of its arguments $\langle y_1, \dots, y_n \rangle \mapsto y_i$)

We can formulate **queries using the function symbols**.

The names of people living in Amsterdam:

Queries in Databases: Tuple Model

Tuple Model

In this model, the **rows are objects** themselves (tuples).

A unary predicate *Person* tells if a row belongs to table Persons:

$$Person(x)$$

Functions symbols for every column of the table:

$$Pid(x)$$

$$Name(x)$$

$$City(x)$$

(project the tuple/row to one of its arguments $\langle y_1, \dots, y_n \rangle \mapsto y_i$)

We can formulate **queries using the function symbols**.

The names of people living in Amsterdam:

$$\{ y \mid \exists x (Person(x) \wedge Name(x) = y \wedge City(x) = Amsterdam) \}$$

Queries in Databases: Tuple Model

Tuple Model

In this model, the **rows are objects** themselves (tuples).

A unary predicate *Person* tells if a row belongs to table Persons:

$$Person(x)$$

Functions symbols for every column of the table:

$$Pid(x)$$

$$Name(x)$$

$$City(x)$$

(project the tuple/row to one of its arguments $\langle y_1, \dots, y_n \rangle \mapsto y_i$)

We can formulate **queries using the function symbols**.

The names of people living in Amsterdam:

$$\{ y \mid \exists x (Person(x) \wedge Name(x) = y \wedge City(x) = Amsterdam) \}$$

or equivalently

$$\{ Name(x) \mid Person(x) \wedge City(x) = Amsterdam \}$$

Queries in Databases

Query: the names of people living in Amsterdam.

Relational Model

$$\{ y \mid \exists i \text{ Person}(i, y, \text{Amsterdam}) \}$$

Tuple Model

$$\{ \text{Name}(x) \mid \text{Person}(x) \wedge \text{City}(x) = \text{Amsterdam} \}$$

Queries in Databases

Query: the names of people living in Amsterdam.

Relational Model

$$\{ y \mid \exists i \text{ Person}(i, y, \text{Amsterdam}) \}$$

Tuple Model

$$\{ \text{Name}(x) \mid \text{Person}(x) \wedge \text{City}(x) = \text{Amsterdam} \}$$

SQL Query

```
SELECT Name
FROM Person
WHERE City = 'Amsterdam'
```


Multiple Tables

PERSONS

<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

CHATS

<u>CID</u>	P1	P2	Date
1	102	101	01.04.14
2	103	102	05.04.14
3	104	101	09.05.14
4	102	104	29.11.14

Query: PIDs of all people that have chatted with everyone in the persons table (except themselves).

Multiple Tables

PERSONS

<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

CHATS

<u>CID</u>	P1	P2	Date
1	102	101	01.04.14
2	103	102	05.04.14
3	104	101	09.05.14
4	102	104	29.11.14

Query: PIDs of all people that have chatted with everyone in the persons table (except themselves).

Relational Model

Multiple Tables

PERSONS

<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

CHATS

<u>CID</u>	P1	P2	Date
1	102	101	01.04.14
2	103	102	05.04.14
3	104	101	09.05.14
4	102	104	29.11.14

Query: PIDs of all people that have chatted with everyone in the persons table (except themselves).

Relational Model

{ y |

}

Multiple Tables

PERSONS

<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

CHATS

<u>CID</u>	P1	P2	Date
1	102	101	01.04.14
2	103	102	05.04.14
3	104	101	09.05.14
4	102	104	29.11.14

Query: PIDs of all people that have chatted with everyone in the persons table (except themselves).

Relational Model

$\{ y \mid \exists y_2, y_3 \text{ Person}(y, y_2, y_3)$

}

Multiple Tables

PERSONS

<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

CHATS

<u>CID</u>	P1	P2	Date
1	102	101	01.04.14
2	103	102	05.04.14
3	104	101	09.05.14
4	102	104	29.11.14

Query: PIDs of all people that have chatted with everyone in the persons table (except themselves).

Relational Model

$$\{ y \mid \exists y_2, y_3 \text{ Person}(y, y_2, y_3) \\ \wedge \forall x (\\ \rightarrow \hspace{15em}) \}$$

Multiple Tables

PERSONS

<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

CHATS

<u>CID</u>	P1	P2	Date
1	102	101	01.04.14
2	103	102	05.04.14
3	104	101	09.05.14
4	102	104	29.11.14

Query: PIDs of all people that have chatted with everyone in the persons table (except themselves).

Relational Model

$$\{ y \mid \exists y_2, y_3 \text{ Person}(y, y_2, y_3) \\ \wedge \forall x (x \neq y \wedge \exists x_2, x_3 \text{ Person}(x, x_2, x_3) \\ \rightarrow) \}$$

Multiple Tables

PERSONS

<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

CHATS

<u>CID</u>	P1	P2	Date
1	102	101	01.04.14
2	103	102	05.04.14
3	104	101	09.05.14
4	102	104	29.11.14

Query: PIDs of all people that have chatted with everyone in the persons table (except themselves).

Relational Model

$$\{ y \mid \exists y_2, y_3 \text{ Person}(y, y_2, y_3) \\ \wedge \forall x (x \neq y \wedge \exists x_2, x_3 \text{ Person}(x, x_2, x_3) \\ \rightarrow \exists c_1, c_2 (\text{Chat}(c_1, y, x, c_2) \vee \text{Chat}(c_1, x, y, c_2))) \}$$

Multiple Tables

PERSONS

<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

CHATS

<u>CID</u>	P1	P2	Date
1	102	101	01.04.14
2	103	102	05.04.14
3	104	101	09.05.14
4	102	104	29.11.14

Query: PIDs of all people that have chatted with everyone in the persons table (except themselves).

Tuple Model

Multiple Tables

PERSONS

<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

CHATS

<u>CID</u>	P1	P2	Date
1	102	101	01.04.14
2	103	102	05.04.14
3	104	101	09.05.14
4	102	104	29.11.14

Query: PIDs of all people that have chatted with everyone in the persons table (except themselves).

Tuple Model

$\{ \text{PID}(y) \mid \text{Person}(y) \wedge \forall x (\text{Person}(x) \wedge x \neq y$

\rightarrow

$) \}$

Multiple Tables

PERSONS

<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

CHATS

<u>CID</u>	P1	P2	Date
1	102	101	01.04.14
2	103	102	05.04.14
3	104	101	09.05.14
4	102	104	29.11.14

Query: PIDs of all people that have chatted with everyone in the persons table (except themselves).

Tuple Model

$$\{ \text{PID}(y) \mid \text{Person}(y) \wedge \forall x (\text{Person}(x) \wedge x \neq y$$
$$\rightarrow \exists c (\text{Chat}(c) \wedge$$

)) }

Multiple Tables

PERSONS

<u>PID</u>	Name	City
101	Ann	Amsterdam
102	Joe	Utrecht
103	Rick	Leipzig
104	Mia	Amsterdam

CHATS

<u>CID</u>	P1	P2	Date
1	102	101	01.04.14
2	103	102	05.04.14
3	104	101	09.05.14
4	102	104	29.11.14

Query: PIDs of all people that have chatted with everyone in the persons table (except themselves).

Tuple Model

$$\{ \text{PID}(y) \mid \text{Person}(y) \wedge \forall x (\text{Person}(x) \wedge x \neq y$$
$$\rightarrow \exists c (\text{Chat}(c) \wedge$$
$$((\text{P1}(c) = \text{PID}(x) \wedge \text{P2}(c) = \text{PID}(y))$$
$$\vee (\text{P1}(c) = \text{PID}(y) \wedge \text{P2}(c) = \text{PID}(x))))) \}$$

Multiple Tables

Query: PIDs of all people that have chatted with everyone in the persons table (except themselves).

SQL Query

```
SELECT Y.PID
FROM Person as Y
WHERE NOT EXISTS (
  SELECT *
  FROM Person as X
  WHERE Y.PID != X.PID
    AND NOT EXISTS (
      SELECT *
      FROM Chats as C
      WHERE ( C.P1 = Y.PID AND C.P2 = X.PID )
        OR ( C.P1 = X.PID AND C.P2 = Y.PID )
    )
)
```

There is no FORALL in SQL. Recall $\forall x \phi(x) \equiv \neg \exists x \neg \phi(x)$!

Relations as Query Results

Query: PID and Name of everybody who has chatted.

Relations as Query Results

Query: PID and Name of everybody who has chatted.

Relational Model

$\{ \langle x, y \rangle \mid$

$\}$

Relations as Query Results

Query: PID and Name of everybody who has chatted.

Relational Model

$\{ \langle x, y \rangle \mid \exists z \text{ Person}(x, y, z) \}$

}

Relations as Query Results

Query: PID and Name of everybody who has chatted.

Relational Model

$$\{ \langle x, y \rangle \mid \exists z \text{ Person}(x, y, z) \\ \wedge \exists c_1, c_2, c_3 (\text{Chat}(c_1, x, c_2, c_3) \vee \text{Chat}(c_1, c_2, x, c_3)) \}$$

Relations as Query Results

Query: PID and Name of everybody who has chatted.

Relational Model

$$\{ \langle x, y \rangle \mid \exists z \text{ Person}(x, y, z) \\ \wedge \exists c_1, c_2, c_3 (\text{Chat}(c_1, x, c_2, c_3) \vee \text{Chat}(c_1, c_2, x, c_3)) \}$$

Tuple Model

$$\{ \langle \text{PID}(p), \text{Name}(p) \rangle \mid \text{Person}(p) \}$$

Relations as Query Results

Query: PID and Name of everybody who has chatted.

Relational Model

$$\{ \langle x, y \rangle \mid \exists z \text{ Person}(x, y, z) \\ \wedge \exists c_1, c_2, c_3 (\text{Chat}(c_1, x, c_2, c_3) \vee \text{Chat}(c_1, c_2, x, c_3)) \}$$

Tuple Model

$$\{ \langle \text{PID}(p), \text{Name}(p) \rangle \mid \text{Person}(p) \\ \wedge \exists c (\text{Chat}(c) \wedge (\text{P1}(c) = \text{PID}(p) \vee \text{P2}(c) = \text{PID}(p))) \}$$

Relations as Query Results

Query: PID and Name of everybody who has chatted.

Relational Model

$$\{ \langle x, y \rangle \mid \exists z \text{ Person}(x, y, z) \\ \wedge \exists c_1, c_2, c_3 (\text{Chat}(c_1, x, c_2, c_3) \vee \text{Chat}(c_1, c_2, x, c_3)) \}$$

Tuple Model

$$\{ \langle \text{PID}(p), \text{Name}(p) \rangle \mid \text{Person}(p) \\ \wedge \exists c (\text{Chat}(c) \wedge (\text{P1}(c) = \text{PID}(p) \vee \text{P2}(c) = \text{PID}(p))) \}$$

SQL Query

```
SELECT P.PID, P.Name FROM Person as P
WHERE EXISTS (
  SELECT * FROM Chats as C
  WHERE C.P1 = P.PID OR C.P2 = P.PID
)
```

Exercises

Exercises

Exercises

Explain the meaning of \models and \vdash for predicate logic.

Exercises

Explain the meaning of \models and \vdash for predicate logic.

$\phi_1, \dots, \phi_n \models \psi$ means that all models \mathcal{M} and environments ℓ that make ϕ_1, \dots, ϕ_n true, also make ψ true.

Exercises

Explain the meaning of \models and \vdash for predicate logic.

$\phi_1, \dots, \phi_n \models \psi$ means that all models \mathcal{M} and environments ℓ that make ϕ_1, \dots, ϕ_n true, also make ψ true.

$\phi_1, \dots, \phi_n \vdash \psi$ means ψ is derivable using natural deduction starting from premises ϕ_1, \dots, ϕ_n .

Exercises

Explain the meaning of \models and \vdash for predicate logic.

$\phi_1, \dots, \phi_n \models \psi$ means that all models \mathcal{M} and environments ℓ that make ϕ_1, \dots, ϕ_n true, also make ψ true.

$\phi_1, \dots, \phi_n \vdash \psi$ means ψ is derivable using natural deduction starting from premises ϕ_1, \dots, ϕ_n .

Explain soundness/correctness and completeness.

Exercises

Explain the meaning of \models and \vdash for predicate logic.

$\phi_1, \dots, \phi_n \models \psi$ means that all models \mathcal{M} and environments ℓ that make ϕ_1, \dots, ϕ_n true, also make ψ true.

$\phi_1, \dots, \phi_n \vdash \psi$ means ψ is derivable using natural deduction starting from premises ϕ_1, \dots, ϕ_n .

Explain soundness/correctness and completeness.

Soundness means that everything derivable by natural deduction is also semantically entailed:

$$\Gamma \vdash \phi \implies \Gamma \models \phi$$

Exercises

Explain the meaning of \models and \vdash for predicate logic.

$\phi_1, \dots, \phi_n \models \psi$ means that all models \mathcal{M} and environments ℓ that make ϕ_1, \dots, ϕ_n true, also make ψ true.

$\phi_1, \dots, \phi_n \vdash \psi$ means ψ is derivable using natural deduction starting from premises ϕ_1, \dots, ϕ_n .

Explain soundness/correctness and completeness.

Soundness means that everything derivable by natural deduction is also semantically entailed:

$$\Gamma \vdash \phi \implies \Gamma \models \phi$$

Completeness means that the derivation rules are strong enough to derive everything that is semantically entailed:

$$\Gamma \models \phi \implies \Gamma \vdash \phi$$

Exercises

Assume you want to disprove

$$\phi \vdash \psi$$

How can the soundness or completeness theorem help?

Exercises

Assume you want to disprove

$$\phi \vdash \psi$$

How can the soundness or completeness theorem help?

To show that there is no possible proof might be difficult.

Exercises

Assume you want to disprove

$$\phi \vdash \psi$$

How can the soundness or completeness theorem help?

To show that there is no possible proof might be difficult.

It is easier to give a counter-model.

Exercises

Assume you want to disprove

$$\phi \vdash \psi$$

How can the soundness or completeness theorem help?

To show that there is no possible proof might be difficult.

It is easier to give a counter-model.

That is, a model \mathcal{M} and environment ℓ such that

$$\mathcal{M} \models_{\ell} \phi \quad \text{and} \quad \mathcal{M} \not\models_{\ell} \psi$$

Exercises

Assume you want to disprove

$$\phi \vdash \psi$$

How can the soundness or completeness theorem help?

To show that there is no possible proof might be difficult.

It is easier to give a counter-model.

That is, a model \mathcal{M} and environment ℓ such that

$$\mathcal{M} \models_{\ell} \phi \quad \text{and} \quad \mathcal{M} \not\models_{\ell} \psi$$

Then we know that $\phi \not\vdash \psi$.

Exercises

Assume you want to disprove

$$\phi \vdash \psi$$

How can the soundness or completeness theorem help?

To show that there is no possible proof might be difficult.

It is easier to give a counter-model.

That is, a model \mathcal{M} and environment ℓ such that

$$\mathcal{M} \models_{\ell} \phi \quad \text{and} \quad \mathcal{M} \not\models_{\ell} \psi$$

Then we know that $\phi \not\vdash \psi$.

By the soundness we have

$$\phi \vdash \psi \implies \phi \models \psi$$

Exercises

Assume you want to disprove

$$\phi \vdash \psi$$

How can the soundness or completeness theorem help?

To show that there is no possible proof might be difficult.

It is easier to give a counter-model.

That is, a model \mathcal{M} and environment ℓ such that

$$\mathcal{M} \models_{\ell} \phi \quad \text{and} \quad \mathcal{M} \not\models_{\ell} \psi$$

Then we know that $\phi \not\vdash \psi$.

By the soundness we have

$$\phi \vdash \psi \implies \phi \models \psi$$

Hence we conclude $\phi \not\vdash \psi$.

Exercises

Meaning of the symbols:

a : Alice

$L(x, y)$: x likes y

Translate the following sentences into predicate logic:

Everybody who likes someone, also likes Alice.

Exercises

Meaning of the symbols:

a : Alice

$L(x, y)$: x likes y

Translate the following sentences into predicate logic:

Everybody who likes someone, also likes Alice.

$$\forall x (\exists y L(x, y) \rightarrow L(x, a))$$

Exercises

Meaning of the symbols:

a : Alice

$L(x, y)$: x likes y

Translate the following sentences into predicate logic:

Everybody who likes someone, also likes Alice.

$$\forall x (\exists y L(x, y) \rightarrow L(x, a))$$

The brackets are important. A typical **mistake** in the exam:

$$\forall x \exists y (L(x, y) \rightarrow L(x, a))$$

Exercises

Meaning of the symbols:

a : Alice

$L(x, y)$: x likes y

Translate the following sentences into predicate logic:

Everybody who likes someone, also likes Alice.

$$\forall x (\exists y L(x, y) \rightarrow L(x, a))$$

The brackets are important. A typical **mistake** in the exam:

$$\forall x \exists y (L(x, y) \rightarrow L(x, a))$$

What does this formula mean?

Exercises

Meaning of the symbols:

a : Alice

$L(x, y)$: x likes y

Translate the following sentences into predicate logic:

Everybody likes at most 2 *other* people.

Exercises

Meaning of the symbols:

a : Alice

$L(x, y)$: x likes y

Translate the following sentences into predicate logic:

Everybody likes at most 2 *other* people.

Note that **at most** means that it can be 0, 1 or 2 !

Exercises

Meaning of the symbols:

a : Alice

$L(x, y)$: x likes y

Translate the following sentences into predicate logic:

Everybody likes at most 2 *other* people.

Note that **at most** means that it can be 0, 1 or 2 !

$$\forall x \exists y_1 \exists y_2 \forall z (L(x, z) \wedge z \neq x \rightarrow z = y_1 \vee z = y_2)$$

Exercises

Meaning of the symbols:

a : Alice

$L(x, y)$: x likes y

Translate the following sentences into predicate logic:

Everybody likes at most 2 *other* people.

Note that **at most** means that it can be 0, 1 or 2 !

$$\forall x \exists y_1 \exists y_2 \forall z (L(x, z) \wedge z \neq x \rightarrow z = y_1 \vee z = y_2)$$

or, equivalently

$$\begin{aligned} \forall x \forall y_1 \forall y_2 \forall y_3 (& L(x, y_1) \wedge L(x, y_2) \wedge L(x, y_3) \\ & \wedge x \neq y_1 \wedge x \neq y_2 \wedge x \neq y_3 \\ & \rightarrow y_1 = y_2 \vee y_2 = y_3 \vee y_1 = y_3) \end{aligned}$$

Examples

Give a counter-model for:

$$\forall x \exists y R(x, y),$$

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$$

$$\models \exists x R(x, x)$$

Examples

Give a counter-model for:

$$\forall x \exists y R(x, y),$$

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$$

$$\models \exists x R(x, x)$$

What do the premises it mean?

Examples

Give a counter-model for:

$$\forall x \exists y R(x, y),$$

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$$

$$\models \exists x R(x, x)$$

What do the premises it mean?

(a) Every object has a successor.

Examples

Give a counter-model for:

$$\forall x \exists y R(x, y),$$

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$$

$$\models \exists x R(x, x)$$

What do the premises it mean?

- (a) Every object has a successor.
- (b) The successor-relation is transitive.

Examples

Give a counter-model for:

$$\forall x \exists y R(x, y),$$

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$$

$$\models \exists x R(x, x)$$

What do the premises it mean?

(a) Every object has a successor.

(b) The successor-relation is transitive.

Hence any n -step successor is an immediate successor.

Examples

Give a counter-model for:

$$\forall x \exists y R(x, y),$$

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$$

$$\models \exists x R(x, x)$$

What do the premises it mean?

- (a) Every object has a successor.
- (b) The successor-relation is transitive.

Hence any n -step successor is an immediate successor.

What does the conclusion mean?

Examples

Give a counter-model for:

$$\forall x \exists y R(x, y),$$

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$$

$$\models \exists x R(x, x)$$

What do the premises it mean?

- (a) Every object has a successor.
- (b) The successor-relation is transitive.

Hence any n -step successor is an immediate successor.

What does the conclusion mean?

- ▶ There is an object that is its own successor.

Examples

Give a counter-model for:

$$\forall x \exists y R(x, y),$$

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$$

$$\models \exists x R(x, x)$$

What do the premises it mean?

- (a) Every object has a successor.
- (b) The successor-relation is transitive.

Hence any n -step successor is an immediate successor.

What does the conclusion mean?

- ▶ There is an object that is its own successor.

Can there be finite counter-models?

Examples

Give a counter-model for:

$$\forall x \exists y R(x, y),$$

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$$

$$\models \exists x R(x, x)$$

What do the premises it mean?

- (a) Every object has a successor.
- (b) The successor-relation is transitive.

Hence any n -step successor is an immediate successor.

What does the conclusion mean?

- ▶ There is an object that is its own successor.

Can there be finite counter-models?

No, because by (a) there would be cycles, and by (b) every element on a cycle would be its own successor.

Examples

Give a counter-model for:

$$\forall x \exists y R(x, y),$$

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$$

$$\models \exists x R(x, x)$$

So, what could be an infinite counter-model?

Examples

Give a counter-model for:

$$\forall x \exists y R(x, y),$$

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$$

$$\models \exists x R(x, x)$$

So, what could be an infinite counter-model?

A counter-model:

- ▶ $A = \mathbb{N}$ (the universe is the set of natural numbers)
- ▶ $R = <$

Question Hour

Question Hour

Success with the Exam

Success with the Exam