Artificial Intelligence

CS4365 --- Fall 2022

Predicate Logic

Instructor: Yunhui Guo

Propositional logic is a weak language

- Hard to identify “individuals” (e.g., Mary, 3)
- Can’t directly talk about properties of individuals or relations between individuals (e.g., “Bill is tall”)
- Generalizations, patterns, regularities can’t easily be represented (e.g., “all triangles have 3 sides”)

Length of Resolution Proof

Consider Pigeon-Hole (PH) problem: Formula encodes that you cannot place n+1 pigeons in n holes (one per hole).

PH takes exponentially many steps (no matter in what order)!

PH hidden in many practical problems. Makes theorem proving expensive.

Pigeon-Hole Principle

$P_{i,j}$ for Pigeon i in hole j.

$$P_{1,1} \vee P_{1,2} \vee P_{1,3} ... P_{1,n}$$

$$P_{2,1} \vee P_{2,2} \vee P_{2,3} ... P_{2,n}$$

....

$$P_{(n+1),1} \vee P_{(n+1),2} \vee P_{(n+1),3} ... P_{(n+1),n}$$

and ??

Pigeon-Hole Principle

$(\neg P_{1,1} \vee \neg P_{1,2}), (\neg P_{1,1} \vee \neg P_{1,3}), (\neg P_{1,1} \vee \neg P_{1,4}), ...$

$(\neg P_{1,(n-1)} \vee \neg P_{1,n}),$

$(\neg P_{2,1} \vee \neg P_{2,2}) ... (\neg P_{2,(n-1)} \vee \neg P_{2,n})$

etc.

$(\neg P_{1,1} \vee \neg P_{2,1}), (\neg P_{1,1)} \vee \neg P_{3,1}), ...$

$(\neg P_{1,2} \vee \neg P_{2,2}), (\neg P_{1,2} \vee \neg P_{3,2}),$ etc.

Pigeon-Hole Principle

Resolution proof of inconsistency requires at least an exponential number of clauses, no matter in what order you resolve things!

A More Concise Formulation

$\forall x\ \exists y\ (x \in \text{Pigeons})\ (\ y \in \text{Holes})\ IN(x, y)$
$\forall x \forall x' \forall y (IN(x,y)\ \wedge\ IN(x', y)$ ... ??
$\forall x \forall y \forall y' (IN(x,y)\ \wedge\ IN(x, y')$ ... ??

$\text{Pigeons} = \{p_1, p_2, ..., p_{n+1}\}$,
$\text{Holes} = \{h_1, h_2, ..., h_n\}$

We have first-order logic with some set-theory notation.

Predicate Logic

- Predicate logic (or first-order logic) is an extension of propositional logic that permits concisely reasoning about whole classes of entities and relations.

- **Propositional logic** treats simple propositions (sentences) as atomic entities

- In contrast, **predicate logic** distinguishes the subject of a sentence from its predicate.

Predicate Logic

- Plato is a philosopher
- Socrates is a philosopher

- Propositional logic:
    - P: Plato is a philosopher
    - Q: Socrates is a philosopher
- Predicate logic:
    - Variable: a
    - Predicate: “is a philosopher”

Subjects and Predicates

- In the sentence "The dog is sleeping":
  - The phrase "the dog" denotes the subject - the entity/object that the sentence is about
  - The phrase "is sleeping" denotes the predicate - a property that is true of the subject.

- In predicate logic, a predicate is modeled as a function $P(\cdot)$ from **objects** to **propositions**

- $P(x)$ = "x is sleeping" where x is any object

More About Predicates

- Convention: Lowercase variables x, y, z... denote objects/entities; uppercase variables P, Q, R… denote propositional functions (predicates).

- The result of applying a predicate P to an object x is the proposition P(x). But P itself (e.g. P="is sleeping") is not a proposition (not a complete sentence).
    - E.g. if P(x) = “x is primer number”,
    P(3) is the proposition “3 is a prime number.”

Propositional Functions

- Predicate logic generalizes the grammatical notion of a predicate to also include propositional functions of any number of arguments, each of which may take any grammatical role that a noun can take

- E.g. let P(x, y, z) = "x gave y the grade z",

then if x = "Mike", y = "Mary", z = "A", then P(x, y, z) = "Mike gave Mary the grade A".

Symbols

- Constant symbols: objects

  e.g., Richard, John

- Predicate symbol: relations

  e.g., IsPhilosopher(x), Brother(x, y), OnHead(x, y)

- Function symbols: functions

  e.g., LeftLeg(x)

  f(x): the father of x

Term

- A term is a logical expression that refers to an object
  - Constant symbols
  - Function symbols:
    - LeftLeg(John)

Atomic sentences

- Formed from a predicate symbol optionally followed by a parenthesized list of terms
    - Brother (Richard, John).
    - Married(Father (Richard), Mother (John))

Complex sentences

- Connectives: build sentences from atomic sentences, using connectives $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$

  (and / or / not / implies / equivalence (biconditional))

- We can use logical connectives to construct more complex sentences

  e.g., $\neg$Brother (LeftLeg(Richard), John)

  Brother (Richard, John) $\wedge$ Brother (John, Richard)

Semantics

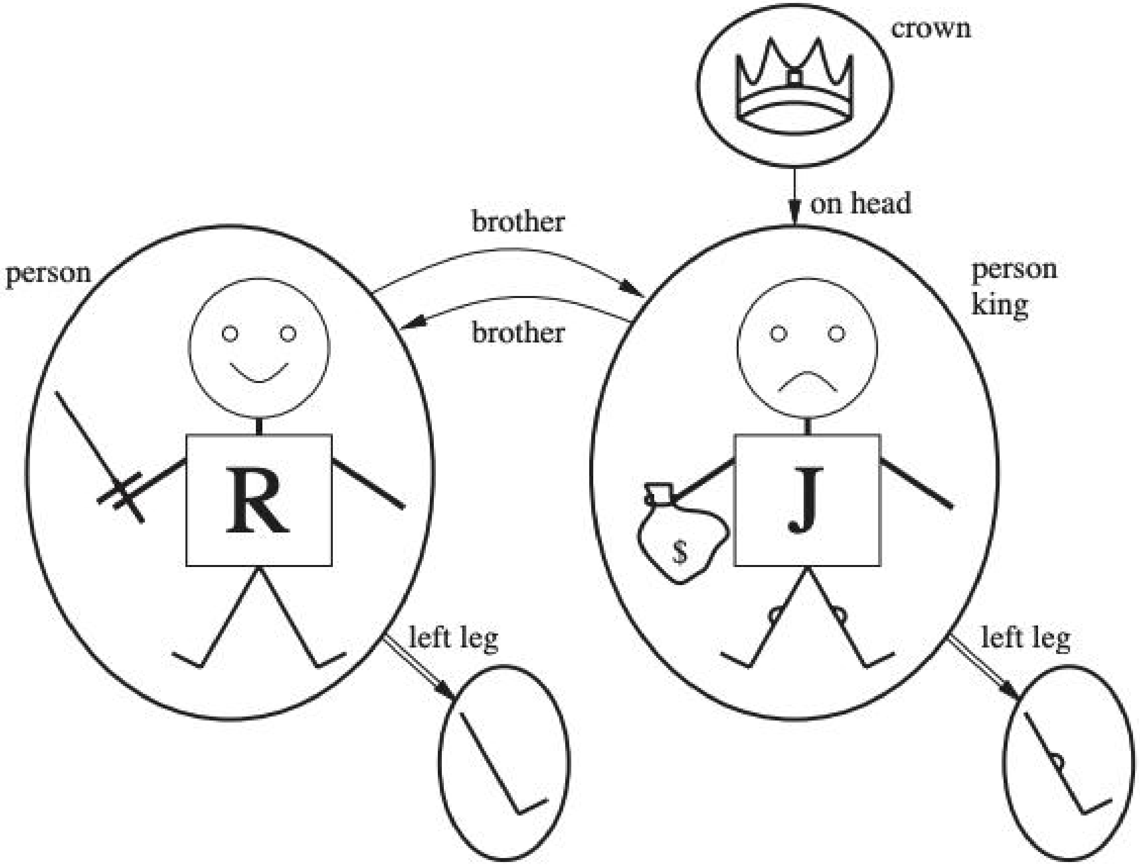
- **Models**: the formal structures that consist possible worlds under consideration
- **Domain**: the set of objects the model contains
- The objects in the model may be related in various ways.

- A relation is just the set of tuples of objects that are related
  - The **brotherhood** relation

{<Richard the Lionheart, King John>, <King John, Richard the Lionheart>}

Model

- A model containing five objects



Interpretations

- Each model includes an interpretation that specifies exactly which objects, relations and functions are referred to by the **constant**, **predicate**, and **function** symbols.

e.g., Richard refers to Richard the Lionheart and John refers to the evil King John.

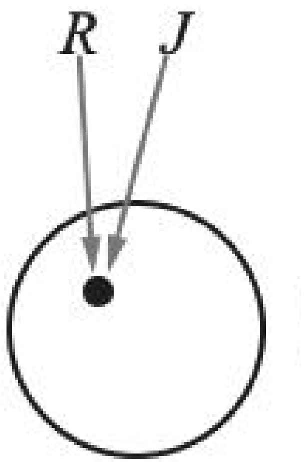
- There are many possible **interpretations:**
  - If there are 5 objects in the model, there are 25 possible interpretation for 2 symbols

Intended Interpretations

- **Richard** refers to **Richard the Lionheart** and **John** refers to the **evil King John**.
- **Brother** refers to the **brotherhood relation**;
- **OnHead** refers to the “**on head**” relation that holds between the crown and King John;
- **Person**, **King**, and **Crown** refer to the sets of objects that are **persons**, **kings**, and **crowns**.
- **LeftLeg** refers to the “**left leg**” function

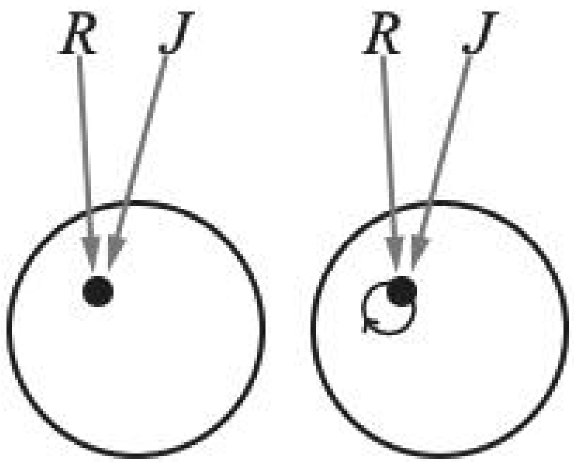
Interpretations

- All possible models: unbounded



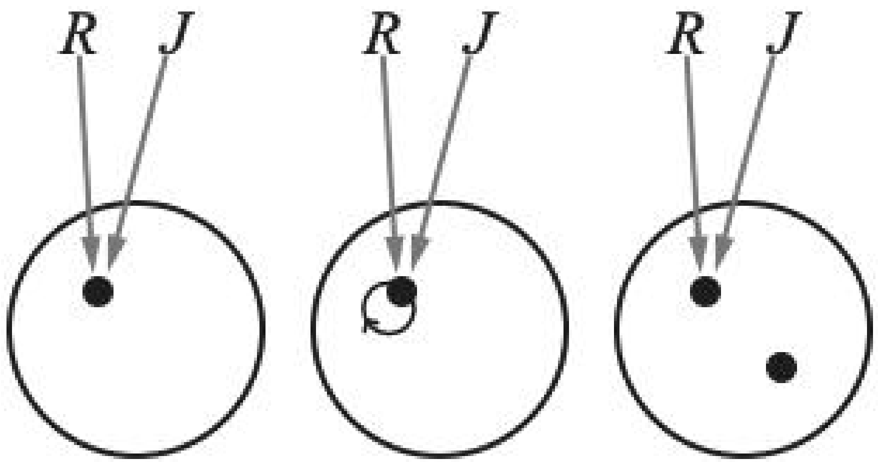
Interpretations

- All possible models: unbounded



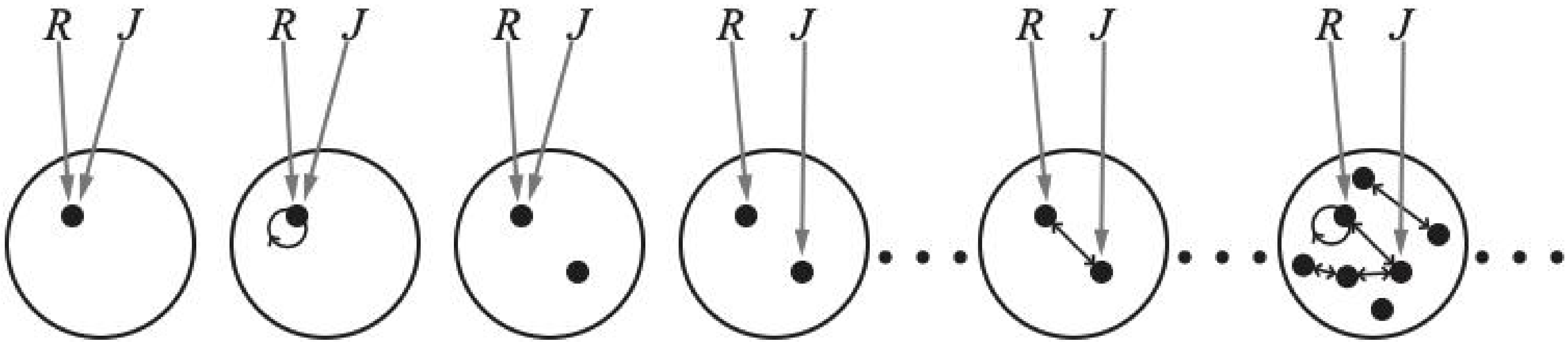
Interpretations

- All possible models: unbounded



Interpretations

- All possible models: unbounded



Summary

- A model for predicate logic:
  - A set of objects
  - An interpretation which maps the constant symbols to objects, predicate symbols to relations on those objects, and function symbols to functions on those objects

Universes of Discourse (U.D.)

- The power of distinguishing objects from predicates is that it lets you state things about many objects at once.

- E.g., let $P(x)$ = “$x+1 > x$”. We can then say, “For all number x, $P(x)$ is true” instead of $(0+1>0) \wedge (1+1>1) \wedge (2+1>2) \wedge \ldots$

- The collection of values that a variable x can take is called x’s universe of discourse.

Quantifier Expressions

- Quantifiers provide a notation that allows us to quantify (count) how many objects in the u.d. satisfy a given predicate.

- "$\forall$" is the FOR ALL or **universal quantifier.**

$\forall x\, P(x)$ means for all x in the u.d., P holds

- "$\exists$" is the EXISTS or **existential quantifier.**

$\exists x\, P(x)$ means there exists an x in the u.d. (that is, 1 or more) such that P(x) is true.

The Universal Quantifier $\forall$

- Example:

Let the u.d. of x be parking spaces at UTD.

Let P(x) be the predicate “x is full”.

Then the universal quantification of P(x), $\forall x\, P(x)$, is the proposition:

- “All parking spaces at UTD are full”
- i.e., “Every parking space at UTD is full.”
- i.e., “For each parking space at UTD, that space is full.”

The Universal Quantifier $\forall$

- Example
  - All kings are persons
    - $\forall x \ \ King(x) \Rightarrow Person(x)$

- Extended interpretation:
  - $x \rightarrow$ Richard the Lionheart
  - $x \rightarrow$ King John
  - $x \rightarrow$ Richard's left leg
  - $x \rightarrow$ John's left leg,
  - $x \rightarrow$ the crown.

The Universal Quantifier ∀

- $\forall x$ King(x) ⇒ Person(x) is true if the sentence King(x) ⇒ Person(x) is true under each of the five extended interpretations

- Richard the Lionheart is a king ⇒ Richard the Lionheart is a person
- King John is a king ⇒ King John is a person.
- Richard's left leg is a king ⇒ Richard's left leg is a person.
- John's left leg is a king ⇒ John's left leg is a person
- The crown is a king ⇒ the crown is a person

The Universal Quantifier $\forall$

- A common mistake using $\forall$
  - $\forall x \;\; King(x) \wedge Person(x)$

- “Richard the Lionheart is a king $\wedge$ Richard the Lionheart is a person”
- “King John is a king $\wedge$ King John is a person”
- “Richard’s left leg is a king $\wedge$ Richard’s left leg is a person”
- ...

The Existential Quantifier $\exists$

- Example:

Let the u.d. of x be parking spaces at UTD.

Let P(x) be the predicate "x is full".

Then the existential quantification of P(x), $\exists x\, P(x)$, is the proposition:

- "Some parking spaces at UTD are full"
- i.e., "There is a parking space at UTD is full."
- i.e., "At least one parking space at UTD is full."

The Existential Quantifier ∃

- Example
  - King John has a crown on his head
    - $\exists x \ \ Crown(x) \land OnHead(x, John)$
- At least one of the following is true:
  - Richard the Lionheart is a crown ∧ Richard the Lionheart is on John's head;
  - King John is a crown ∧ King John is on John's head
  - Richard's left leg is a crown ∧ Richard's left leg is on John's head;
  - John's left leg is a crown ∧ John's left leg is on John's head;
  - The crown is a crown ∧ the crown is on John's head.

Free and Bound Variables

An expression like P(x) is said to have a free variable x (meaning, x is undefined).

A quantifier (either ∀ or ∃) operates on an expression having one or more free variables, and binds one or more of those variables, to produce an expression having one or more bound variables.

Example of Binding

- $P(x, y)$ has 2 free variables, x and y
- $\forall x\, P(x, y)$ has 1 free variable, and one bound variable. [**Which is which?**]
- "$P(x)$, where $x = 3$" is another way to bind x.
- An expression with zero free variable is a proposition.
- An expression with **one or more free variables** is still a predicate: e.g. Let $Q(y) = \forall x\, P(x, y)$

Nesting of Quantifiers

- Example: Let the u.d. of x & y be people.
- Let $L(x, y)$ = "x likes y" (how many free vars?)

- Then $\exists y\, L(x, y)$ = "There is someone whom x likes." (how many free vars are there?)

- The $\forall x(\exists y\, L(x, y))$ = "Everyone has someone whom they like."

(how many free vars are there?)

Quantifier Exercise

- If $R(x, y)$ = “x relies upon y”, express the following in unambiguous English:
- $\forall x(\exists y\ R(x, y))$:
  - Everyone has someone to rely on
- $\exists y(\forall x\ R(x, y))$:
  - There’s a poor overburdened soul whom everyone relies upon (including himself)!
- $\exists x(\forall y\ R(x, y))$:
  - There’s some needy person who relies upon everybody (including himself).
- $\forall y(\exists x\ R(x, y))$:
  - Everyone has someone who relies upon them.
- $\forall x(\forall y\ R(x, y))$:
  - Everyone relies upon everybody, (including themselves)!

More to Know About Binding

- $\forall x\ \exists x\ P(x)$ - x is not a free variable in $\exists x\ P(x)$, therefore the $\forall x$ binding isn't used.
- $(\forall x\ P(x)) \land Q(x)$ - The variable x is outside of the scope of the $\forall x$ quantifier, and is therefore free.
  - Not a complete proposition!
- $(\forall x\ P(x)) \land (\exists x\ Q(x))$ - This is legal, because there are 2 different x's!

Quantifier Equivalence Laws

- Definitions of quantifiers: If u.d. = a, b, c, ...

$$\forall x\ P(x) \Leftrightarrow P(a) \wedge P(b) \wedge P(c) \wedge \ldots$$

$$\exists x\ P(x) \Leftrightarrow P(a) \vee P(b) \vee P(c) \vee \ldots$$

- "Everyone likes ice cream" means "there is no one who does not like ice cream":

- $\forall x$ Likes(x, IceCream) is equivalent to $\neg\exists x\ \neg$Likes(x, IceCream)

Quantifier Equivalence Laws

- Definitions of quantifiers: If u.d. = a, b, c, ...

$$\forall x\ P(x) \Leftrightarrow P(a) \land P(b) \land P(c) \land \ldots$$

$$\exists x\ P(x) \Leftrightarrow P(a) \lor P(b) \lor P(c) \lor \ldots$$

- From those, we can prove the laws:

$$\forall x\ P(x) \Leftrightarrow \neg\exists x\ \neg P(x)$$

$$\exists x\ P(x) \Leftrightarrow \neg\forall x\ \neg P(x)$$

Which propositional equivalence laws can be used to prove this?

De Morgan's

Logic Equivalence

- $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ De Morgan
- $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ De Morgan

- $(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivity of $\wedge$ over $\vee$
- $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of $\vee$ over $\wedge$

Quantifier Equivalence Laws

- Definitions of quantifiers: If u.d. = a, b, c, ...

  $\forall x\, P(x) \Leftrightarrow P(a) \wedge P(b) \wedge P(c) \wedge \ldots$

  $\exists x\, P(x) \Leftrightarrow P(a) \vee P(b) \vee P(c) \vee \ldots$

- From those, we can prove the laws:

  $\forall x\, P(x) \Leftrightarrow \neg\exists x\ \neg P(x)$

  $\exists x\, P(x) \Leftrightarrow \neg\forall x\ \neg P(x)$

$\neg(\neg P(a) \vee \neg P(b) \vee \neg P(c) \vee \ldots) = P(a) \wedge P(b) \wedge P(c) \wedge \ldots$

$\neg(\neg P(a) \wedge \neg P(b) \wedge \neg P(c) \wedge \ldots) = P(a) \vee P(b) \vee P(c) \vee$

Equality

- Another way to make atomic sentences,
  - Father (John) = Henry

- The negation of equality can be used to insist that two terms are not the same object.
  - $\exists$ x, y Brother (x, Richard) $\wedge$ Brother (y, Richard)
  - $\exists$ x, y Brother (x, Richard) $\wedge$ Brother (y, Richard) $\wedge$ $\neg$(x = y)

Use of First-order Logic

- Sentences are added to a knowledge base using TELL
  - TELL(KB, King(John)) .
  - TELL(KB, Person(Richard))
  - TELL(KB, $\forall$ x King(x) $\Rightarrow$ Person(x)) .
- We can ask questions using ASK
  - ASK(KB, King(John))
  - ASK(KB, $\exists$x Person(x)) .

Use of First-order Logic

- The value of x makes the sentence true
  - **ASKVARS**(KB, Person(x))
- **Binding list**
  - {x/John}
  - {x/Richard}.
- Cannot be always achieved
  - King(John) $\vee$ King(Richard) is true
  - No binding for x for the query $\exists x$ King(x)

Constructing Natural Numbers

- **Peano Axioms**
  - Predicate: NatNum(x)
  - Constant symbol: 0
  - Function symbol: S (successor)

Constructing Natural Numbers

- **Peano Axioms**
  - NatNum(0)
  - $\forall n \;\; \mathrm{NatNum}(n) \Rightarrow \mathrm{NatNum}(S(n))$
- S(n) is the **successor function**
  - $\forall n \qquad 0 \neq S(n)$
  - $\forall m, n \quad m \neq n \Rightarrow S(m) \neq S(n)$.
- Addition
  - $\forall\, m \qquad \mathrm{NatNum}(m) \Rightarrow +(0, m) = m$.
  - $\forall\, m, n \quad \mathrm{NatNum}(m) \wedge \mathrm{NatNum}(n) \Rightarrow +(S(m), n) = S(+(m, n))$.
  - $\forall\, m, n \quad \mathrm{NatNum}(m) \wedge \mathrm{NatNum}(n) \Rightarrow (m + 1) + n = (m + n) + 1$.