CS 480

Introduction to Artificial Intelligence

February 21, 2023

Announcements / Reminders

- **Please follow the Week 06 To Do List instructions**

- **Programming Assignment #01 due on ~~Monday (02/20/23) at 11:59 PM CST~~ Saturday (02/25/23) at 11:59 PM CST**

Plan for Today

- Forward Chaining
- Predicate / First-Order Logic

Definite Clauses

A sentence can be called a definite clause if and only if it is a disjunction of literals of which EXACTLY one is positive. For example:

$$(\neg p \vee \neg q \vee r)$$

is a definite clause.

This:

$$(x \vee \neg y \vee z)$$

is NOT a definite clause (more than one positive literal)

Horn Clauses

A sentence can be called a Horn clause if and only if it is a disjunction of literals of which AT MOST one is positive. For example:

$$(\neg p \vee \neg q \vee r)$$

is a Horn clause. This:

$$(x \vee \neg y \vee z)$$

is NOT a Horn clause. However, this:

$$(\neg d \vee \neg e \vee \neg f)$$

is a Horn clause (goal clause $\rightarrow$ no positive literals).

Definite / Horn Clauses: Why Bother?

Reasons to use definite / Horn clauses:

- **resolution of two Horn clauses, yields a Horn clause**
- **definite clauses can be rewritten as implications:**

$$(\neg p \vee \neg q \vee r) \equiv (p \wedge q) \Rightarrow r$$

- **inference with Horn clauses can be realized using two human-friendly (-understandable) algorithms: forward- and backward-chaining**
- **deciding entailment with Horn clauses is** $O(|KB|)$

Definite / Horn Clauses: Why Bother?

Reasons to use definite / Horn clauses:

- **resolution of two Horn clauses, yields a Horn clause**
- **definite clauses can be rewritten as implications:**

$$(\neg p \vee \neg q \vee r) \equiv (p \wedge q) \Rightarrow r$$

- **inference with Horn clauses can be realized using two human-friendly (-understandable) algorithms: forward- and backward-chaining**
- **deciding entailment with Horn clauses is** $O(|KB|)$

Types of Horn Clauses

Types of Horn clauses (at most one positive literal):

Type of Horn clause	Disjunction form	Implication form	Read in English as
Definite clause	$(\neg p \vee \neg q \vee \ldots \vee \neg t \vee u)$	$(p \wedge q \wedge \ldots \wedge t) \Rightarrow u$	assume that, if p and q and ... and t all hold, then also u holds Rules | If .... then
Fact / Unit Clause	u	$T \Rightarrow u$	assume that u holds
Goal clause	$(\neg p \vee \neg q \vee \ldots \vee \neg t)$	$(p \wedge q \wedge \ldots \wedge t) \Rightarrow \bot$	show that p and q and ... and t all hold

$$(\neg p \vee \neg q \vee \ldots \vee \neg t \vee u) \equiv \neg(p \wedge \neg q \wedge \ldots \wedge \neg t) \vee u$$

Because (Implication elimination reversed) $\neg a \vee b \equiv a \Rightarrow b$:

$$\neg(p \wedge \neg q \wedge \ldots \wedge \neg t) \vee u \equiv (p \wedge \neg q \wedge \ldots \wedge \neg t) \Rightarrow u$$

Also: $(\neg p \vee \neg q \vee \ldots \vee \neg t \vee u) \equiv (\text{head/consequence} \vee \text{body/premise})$

Definite Clause and Modus Ponens

Modus Ponens

$$P \Rightarrow Q$$

$$Q$$

$$\therefore Q$$

Modus Ponens (textbook)

$$(P \Rightarrow Q), (Q)$$

$$(Q)$$

Definite Clause and Modus Ponens

Modus Ponens

$$(p \wedge \neg q \wedge \ldots \wedge \neg t) \Rightarrow u$$

$$u$$

$$\therefore u$$

Modus Ponens (textbook)

$$((p \wedge \neg q \wedge \ldots \wedge \neg t) \Rightarrow u), (u)$$

$$(u)$$

Forward Chaining Algorithm

Entailment can be verified with Forward Chaining:

- **set up your Knowledge Base** KB
- **set up your query** Q
- **start with known facts (say** A **and** B**):**
  - A **and** B **are automatically considered "inferred"**
  - **are they a part of some implication** $\mathrm{A} \wedge \mathrm{B} \Rightarrow \mathrm{X}$**?**
  - **if yes,** X **is now considered "inferred"**
- **Repeat until:**
  - Q **is "inferred", or**
  - **no further inferences can be made**

Forward Chaining: Pseudocode

function PL-FC-ENTAILS?(KB, q) **returns** *true* or *false*
  inputs: KB, the knowledge base, a set of propositional definite clauses
    q, the query, a proposition symbol
  count $\leftarrow$ a table, where *count*[c] is initially the number of symbols in clause c's premise
  inferred $\leftarrow$ a table, where *inferred*[s] is initially *false* for all symbols
  queue $\leftarrow$ a queue of symbols, initially symbols known to be true in KB

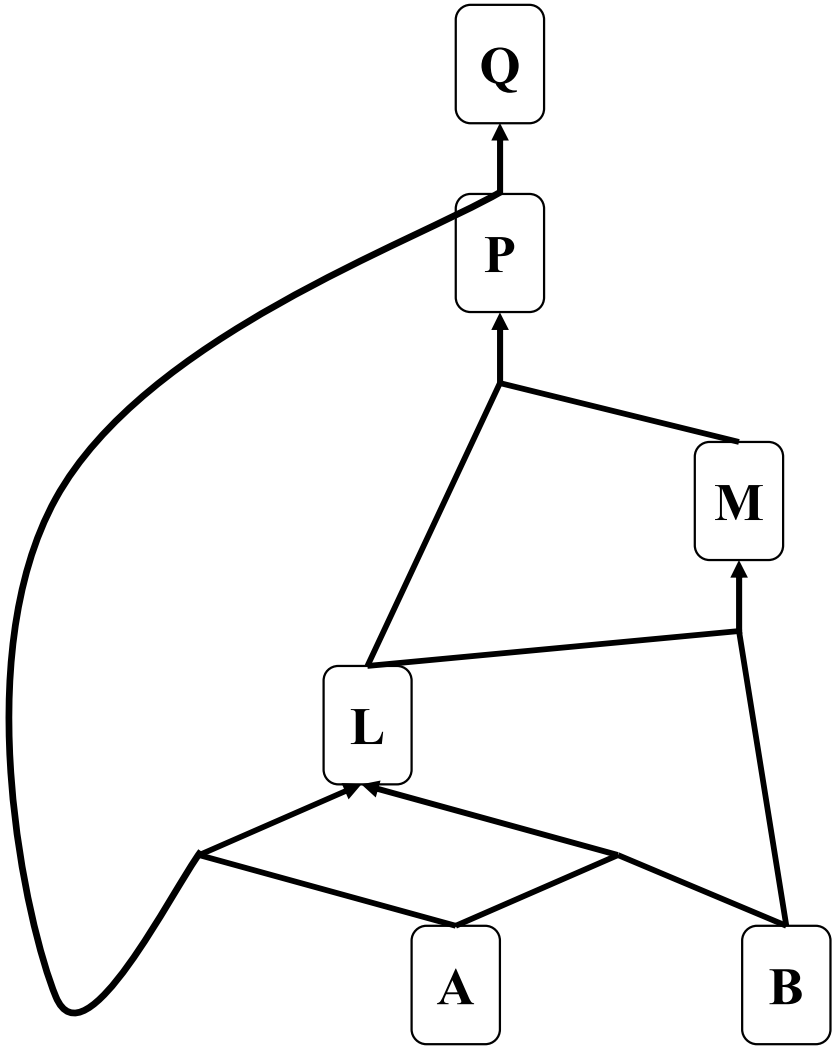
  while *queue* is not empty **do**
    $p \leftarrow$ POP(*queue*)
    if $p = q$ **then return** *true*
    if *inferred*[p] = *false* **then**
      inferred[p] $\leftarrow$ *true*
      for each clause c in KB where p is in c.PREMISE **do**
        decrement *count*[c]
        if *count*[c] = 0 **then** add c.CONCLUSION to *queue*
  return *false*

Forward Chaining Algorithm

Knowledge Base KB:

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Inferred



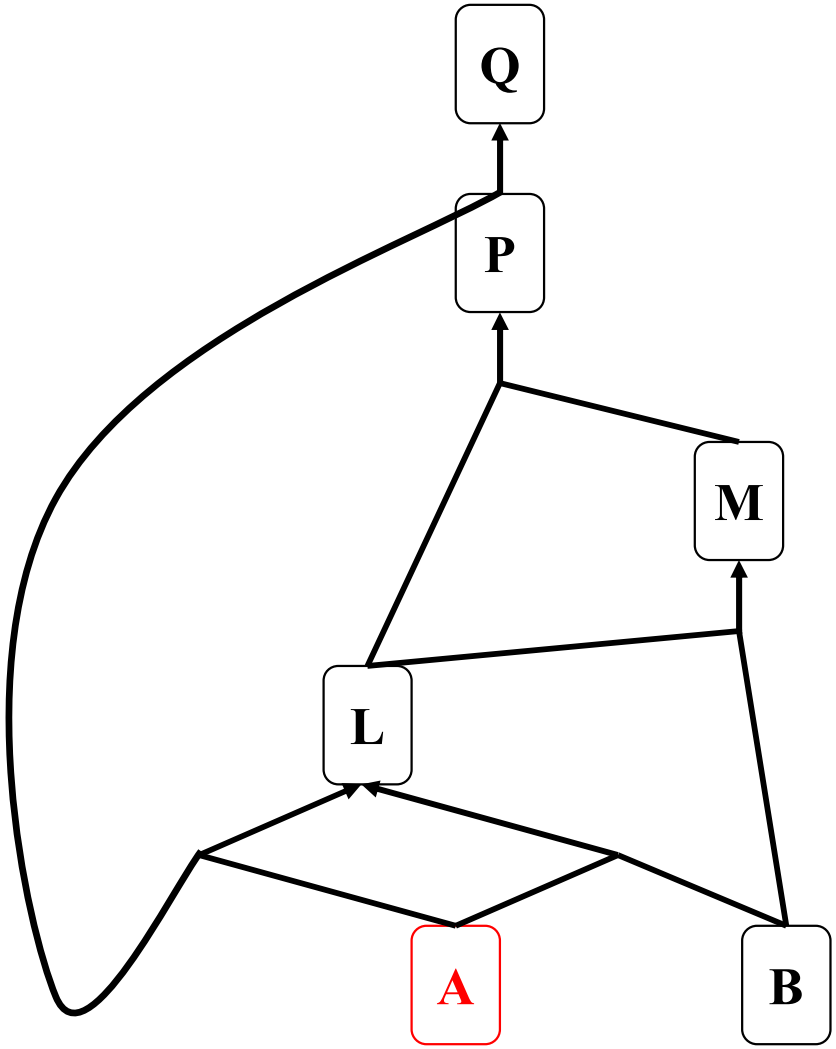
Forward Chaining Algorithm

Knowledge Base KB:

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Inferred

A (because it is a fact)



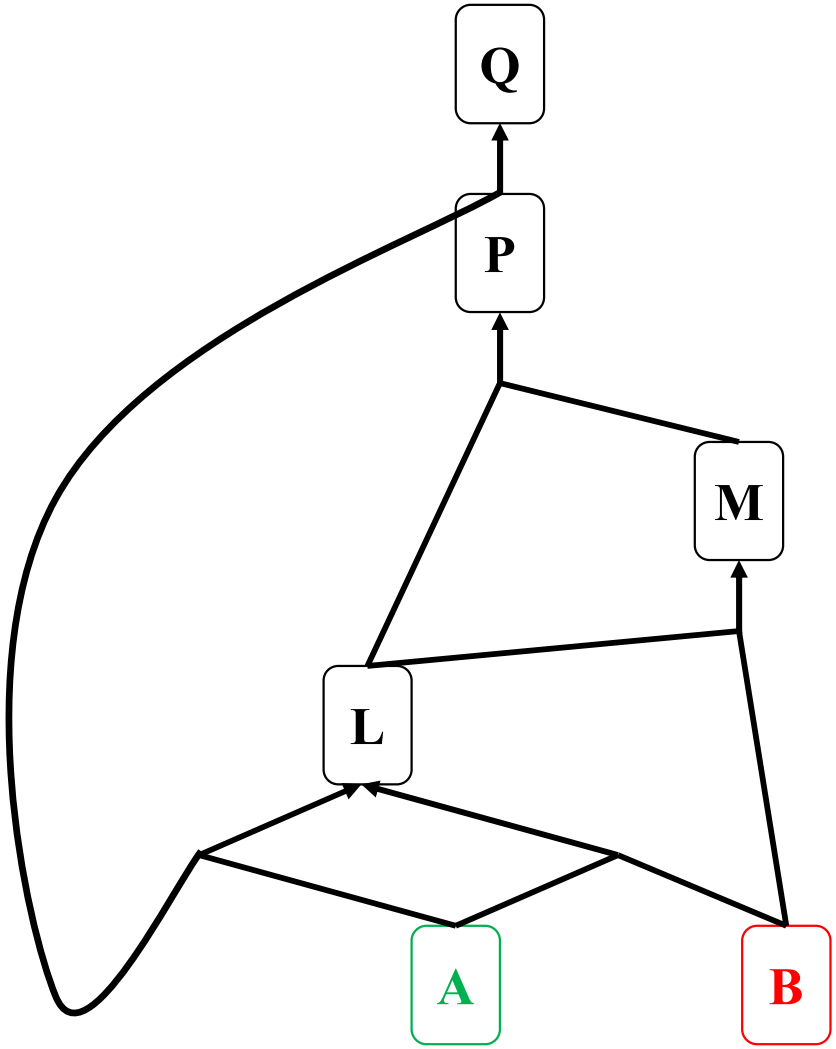
Forward Chaining Algorithm

Knowledge Base KB:

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Inferred:

A
B (because it is a fact)



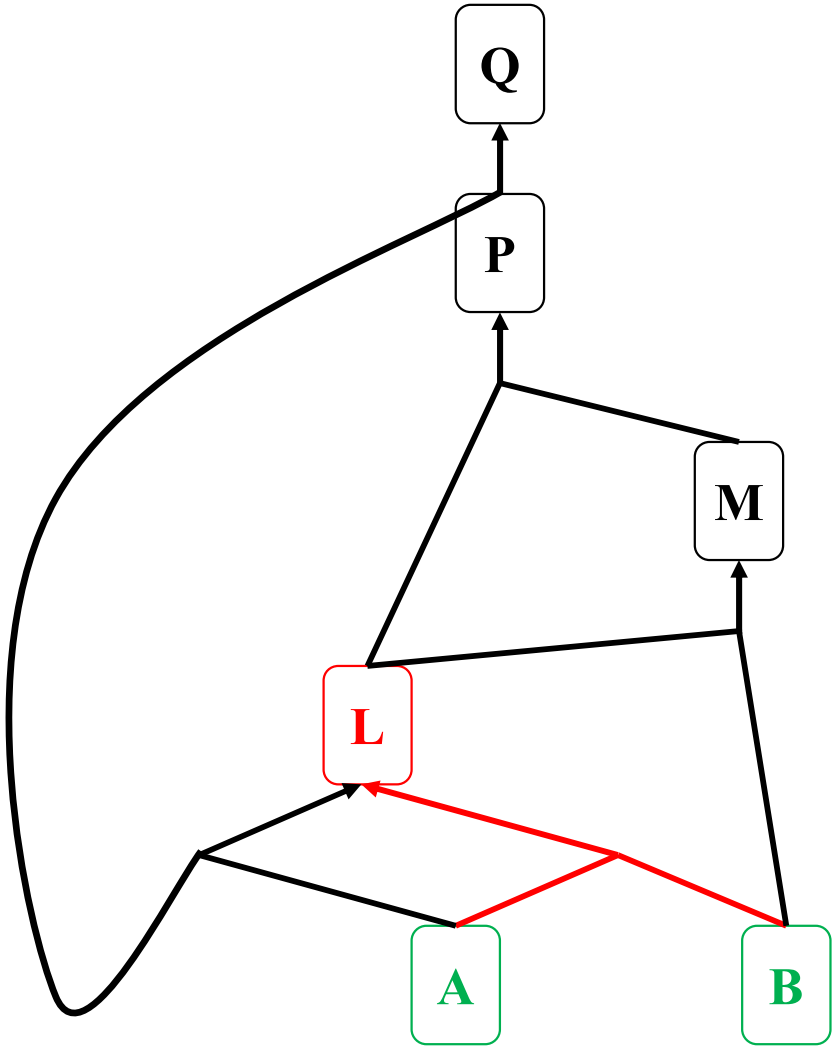
Forward Chaining Algorithm

Knowledge Base KB:

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Inferred:

A
B
L (because $A \wedge B \Rightarrow L$)



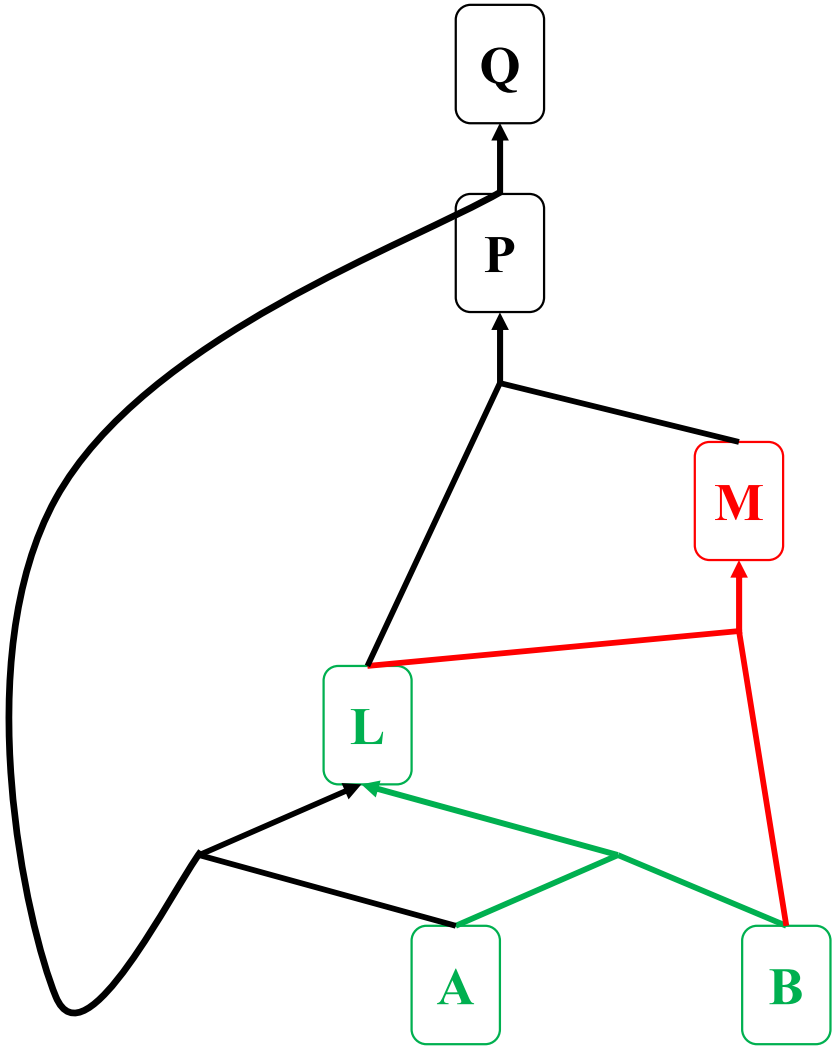
Forward Chaining Algorithm

Knowledge Base KB:

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Inferred:

A
B
L (because $A \wedge B \Rightarrow L$)
M (because $B \wedge L \Rightarrow M$)



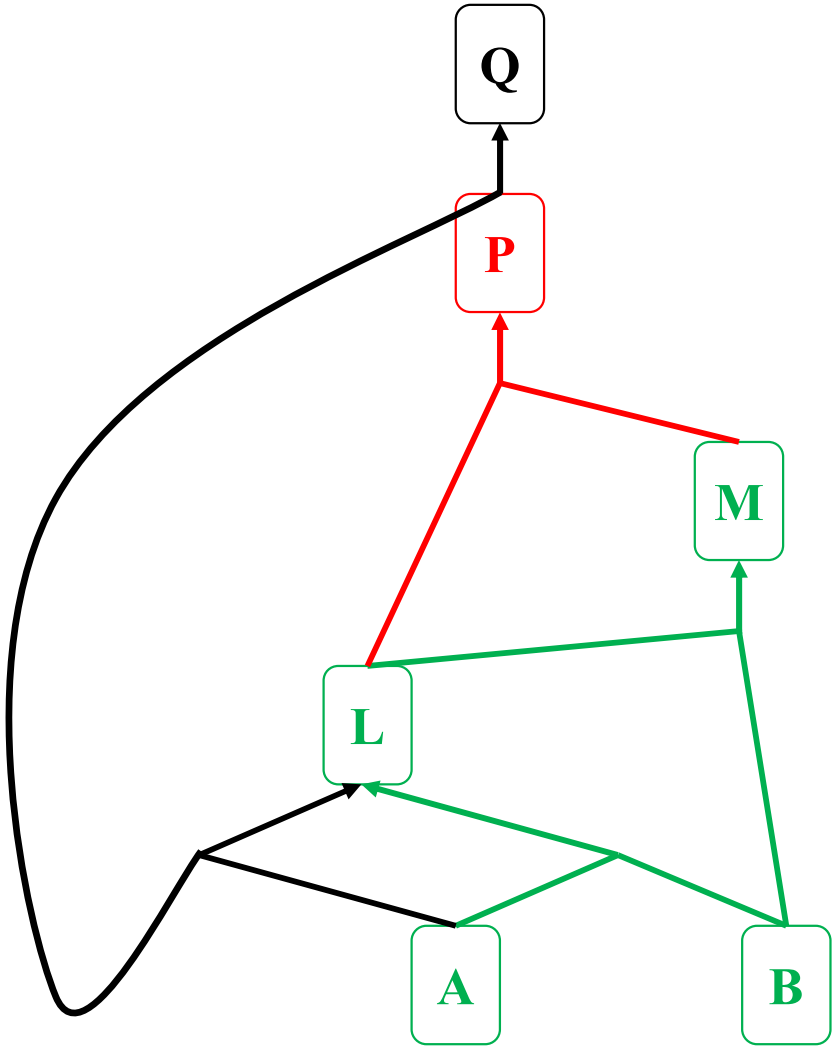
Forward Chaining Algorithm

Knowledge Base KB:

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Inferred:

A
B
L (because $A \wedge B \Rightarrow L$)
M (because $B \wedge L \Rightarrow M$)
P (because $L \wedge M \Rightarrow P$)



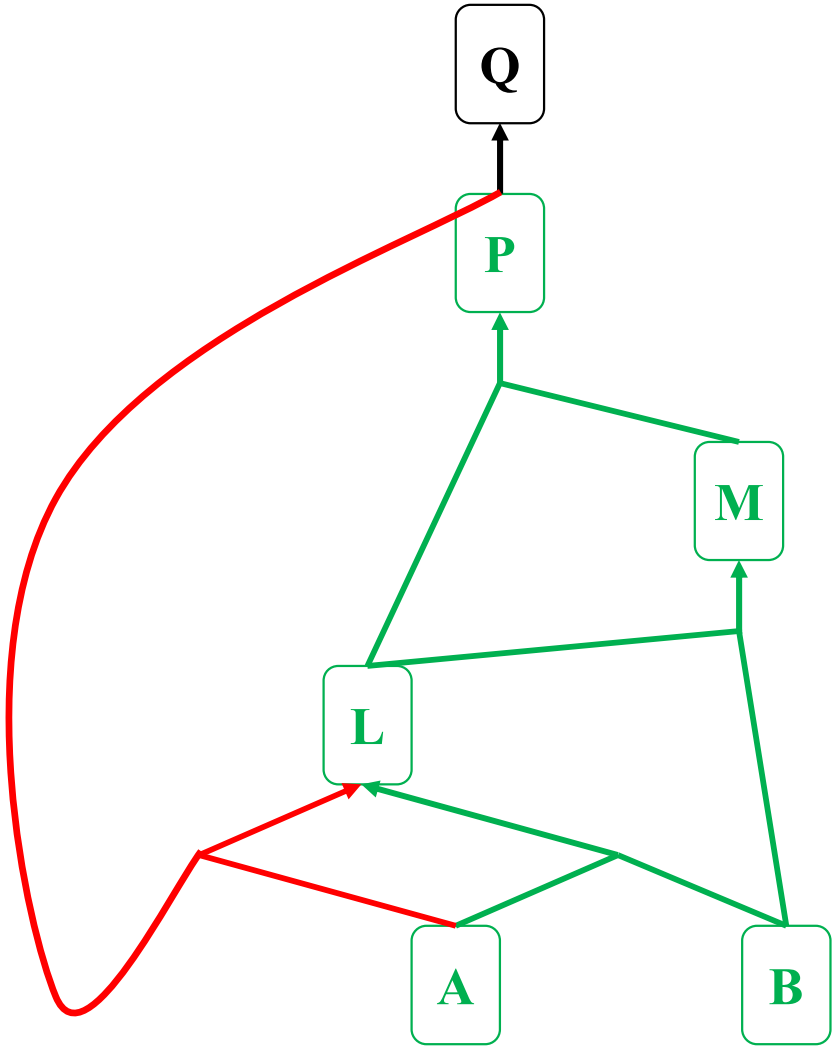
Forward Chaining Algorithm

Knowledge Base KB:

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$ (note: L is already inferred)
$A \wedge B \Rightarrow L$
A
B

Inferred:

A
B
L (because $A \wedge B \Rightarrow L$)
M (because $B \wedge L \Rightarrow M$)
P (because $L \wedge M \Rightarrow P$)



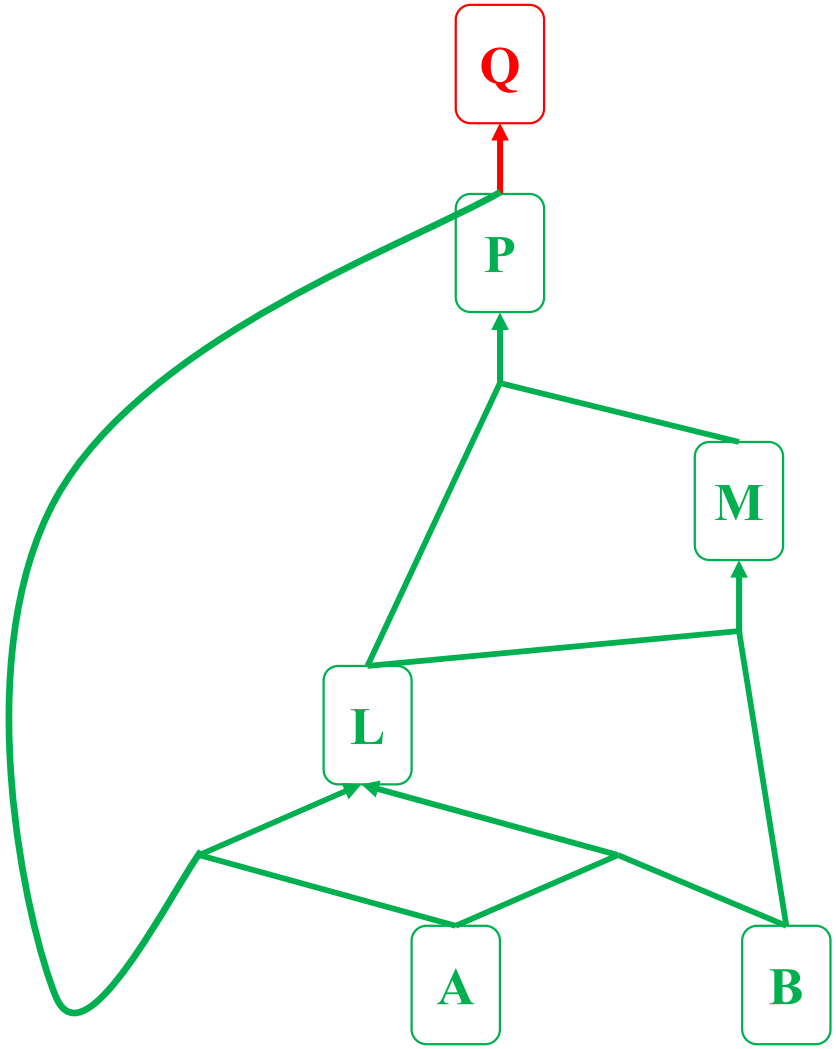
Forward Chaining Algorithm

Knowledge Base KB:

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Inferred:

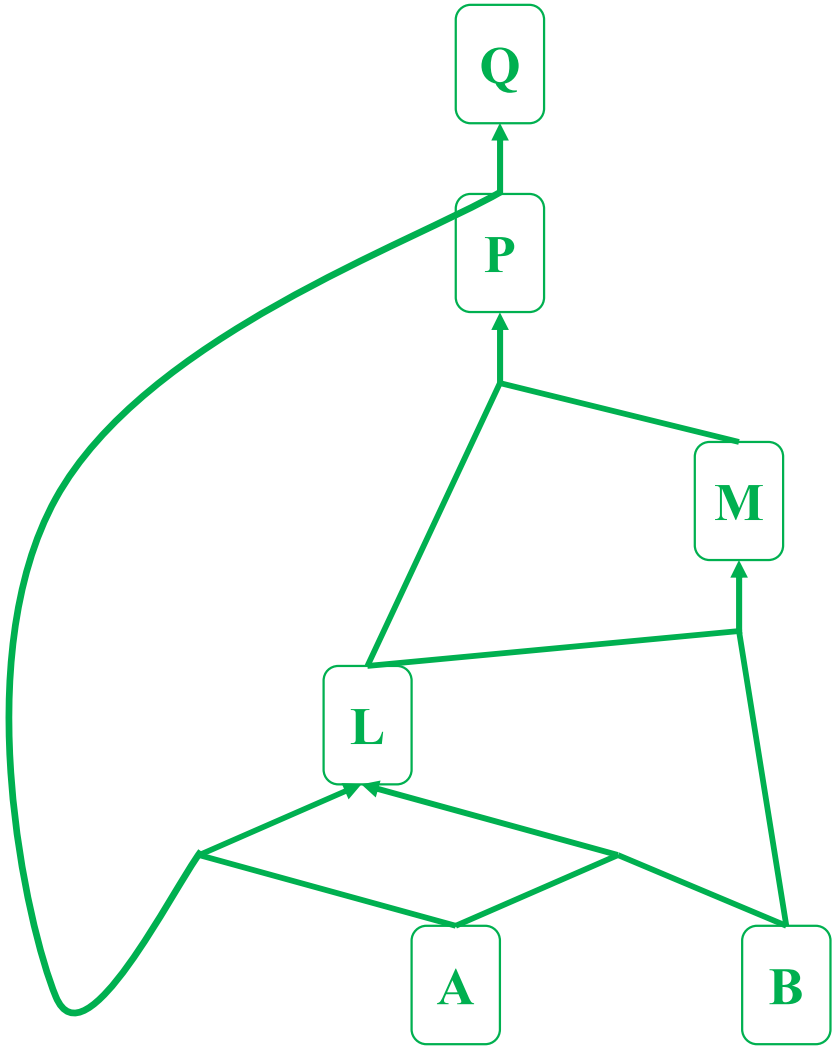
A
B
L (because $A \wedge B \Rightarrow L$)
M (because $B \wedge L \Rightarrow M$)
P (because $L \wedge M \Rightarrow P$)
Q (because $P \Rightarrow Q$)



Forward Chaining Algorithm

Knowledge Base KB:

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B



Inferred:

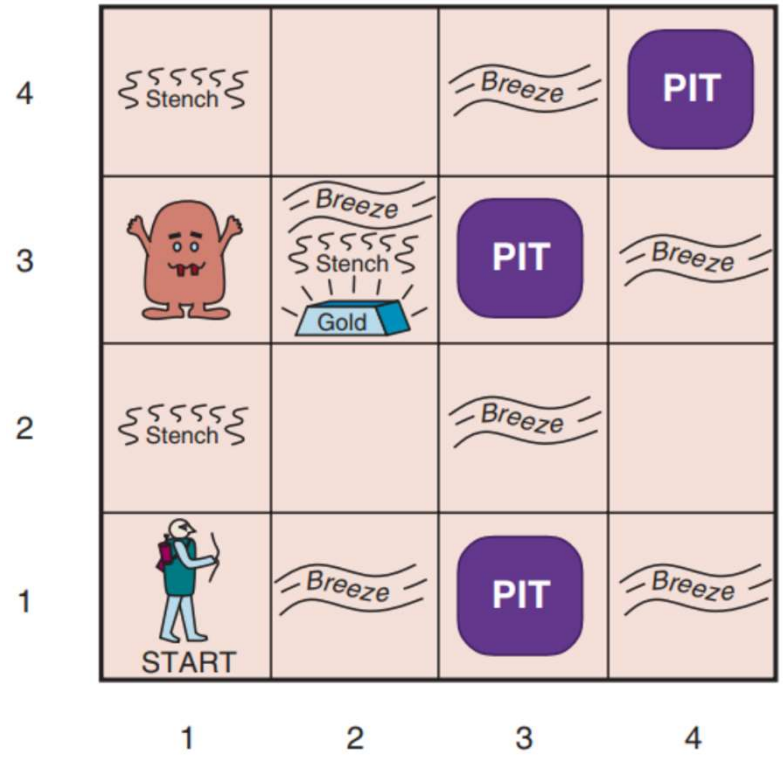
A
B
L (because $A \wedge B \Rightarrow L$)
M (because $B \wedge L \Rightarrow M$)
P (because $L \wedge M \Rightarrow P$)
Q (because $P \Rightarrow Q$)
Q is inferred, therefore KB entails Q

Propositional Logic Weaknesses

Wumpus World is represented by several variables in propositional logic:

- $B_{x,y}$: **there is a breeze in square (x,y)**
- $P_{x,y}$: **there is a pit in square (x,y)**
- $S_{x,y}$: **there is a stench in square (x,y)**
- $W_{x,y}$: **there is a wumpus in square (x,y)**
- $G_{x,y}$: **there is gold in square (x,y)**
- $L_{x,y}$: **agent is located in square (x, y)**

Either one of those can be either true or false. There is 4 x 4 variables for every type.



What if this Wumpus World was a N x N grid and N was a really large number. How many variables we would have?

Propositional Logic Weaknesses

Similarly, consider the following English sentence:

"robot R514 is standing at coordinates (123, 4501)"

It could be represented by a following propositional logic variable:

robot_R514_is_standing_at_coordinates_(123_4501)

and a true **/** false **value. What's the problem?**

Propositional Logic Weaknesses

How about relationships such as this one:

"robot R514 is to the left of robot R89"

It could be represented by a following propositional logic variable:

robot_R514_is_to_the_left_of_R89

and a true **/** false **value. What's the problem?**

Propositional Logic Weaknesses

How about relationships such as this one:

"robot R514 is to the left of robot R89"

It would be much easier to represent it in a form similar to this one:

toTheLeftOf(R514, R89)

Propositional Logic Weaknesses

Propositional logic:

- **CAN represent facts**
- **CANNOT represent objects**
- **CANNOT represent relationships between objects**
- **is overall not very expressive, for example it is impossible to transform this sentence into propositional logic:**

"Some people live in Illinois."

A More Expressive Formal Language

A more expressive formal language could be obtained by augmenting propositional logic with:

- **objects:**
  - **people, houses, humbers, theories, colors, basketball games, wars, etc.**
- **relations:**
  - **unary relations (properties): red, round, bogus, etc.**
  - **n-ary relations: brother of, bigger than, inside, part of, etc.**
- **functions:**
  - **relations in which there is only one "value" for a given "input"**
  - **father of, best friend, one more than, beginning of, etc.**

Predicate / First-Order Logic

Predicate Logic is an extension of Propositional Logic. It is a formal language in which propositions are expressed in terms of predicates, variables, and quantifiers.

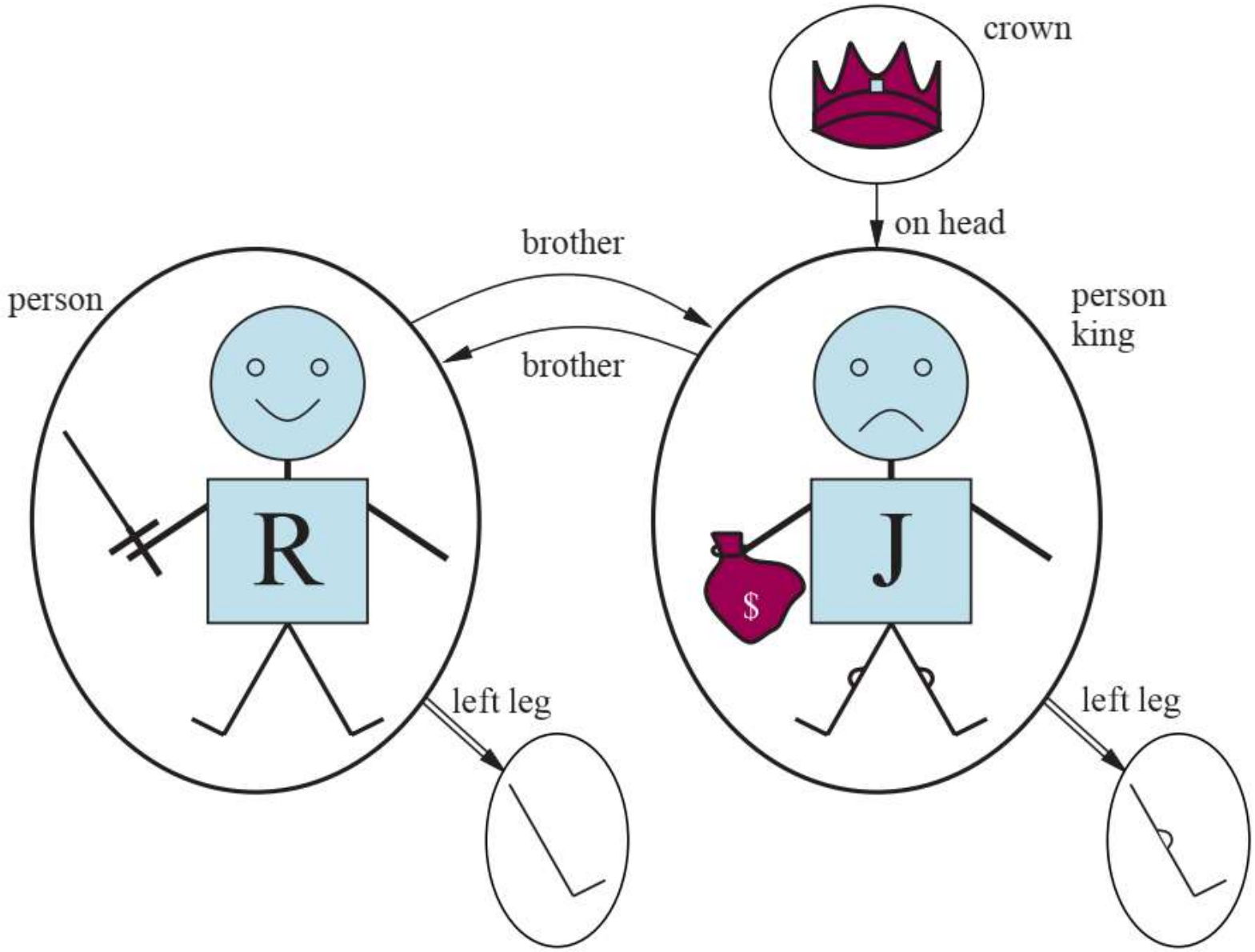
Predicates

In logic, a predicate is a symbol which represents a property or a relation. For example:

- **in the sentence $\mathrm{P}(\mathrm{a})$, the symbol P is a predicate which applies to the individual constant a,**
- **in the formula $\mathrm{R}(\mathrm{a},\mathrm{b})$ the symbol R is a predicate which applies to the individual constants a and b.**

In the semantics of logic, predicates are interpreted as relations.

Predicate Logic: Model of the World



Predicate Logic: Model and Objects

Domain D

lukeSkywalker

genghisKhan

wyattEarp

gun

sword

lightSaber

Symbols: Refresher

Symbol	Name	Alternative symbols*	Should be read
$\neg$	Negation	$\sim$, !	not
$\wedge$	(Logical) conjunction	$\bullet$, &	and
$\vee$	(Logical) disjunction	+, $\|\|$	or
$\Rightarrow$	(Material) implication	$\rightarrow$, $\supset$	implies
$\Leftrightarrow$	(Material) equivalence	$\leftrightarrow$, $\equiv$, iff	if and only if
$\top$	Tautology	T, 1, $\blacksquare$	truth
$\bot$	Contradiction	F, 0, $\square$	falsum empty clause
$\forall$	Universal quantification		for all; for any; for each
$\exists$	Existential quantification		there exist

* you can encounter it elsewhere in literature

Predicate Logic Syntax: Symbols

Predicate calculus symbols include:

- **truth symbols: true and false**
- **constant symbols**
- **variable symbols**
- **predicate symbols**
- **function symbols**

Syntax: Constant

A constant c (c - symbol) represent objects

- **for example:**
  - lightSaber
  - kingJohn
  - lukeSkywalker
  - box
  - table
  - crown

Predicate Logic: Model and Objects

Domain D

lukeSkywalker

genghisKhan

wyattEarp

gun

sword

lightSaber

Existing constants in this representation are

D = {lukeSkywalker, genghisKhan, wyattEarp, gun, sword, lightSaber}

Syntax: Variable

A variable v (v - symbol) can be used to represent classes of objects or properties in the world.

- **for example:**
  - weapon
  - character
  - person
  - height
  - distance

Predicate Logic: Model and Objects

Domain D

lukeSkywalker

genghisKhan

wyattEarp

gun

sword

lightSaber

Possible variables for this model: person **and** weapon

D_{person} = {lukeSkywalker, genghisKhan, wyattEarp} D_{weapon} = {gun, sword, lightSaber}

Syntax: Predicate Symbols

A predicate p (p - symbol) maps object(s) to a boolean value.

$p(argument_1, argument_2, ..., argument_N)$ **or just p**

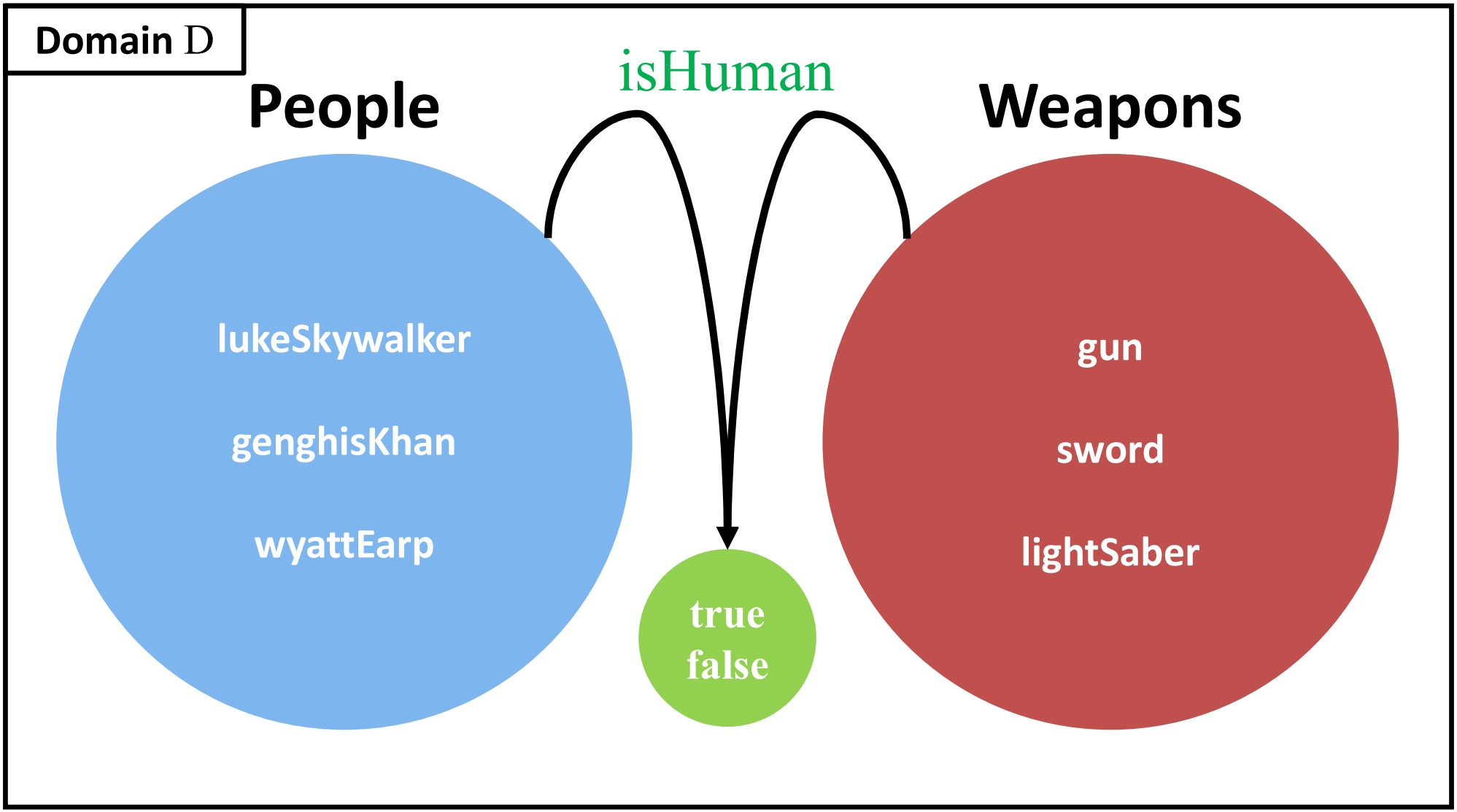
Predicates:

- **take objects as arguments**
- **output boolean values true / false**
- **for example:**

happy(student) **would output value** false **or** true

wasRaining(today) **would output value** true

Predicates: Example



isHuman(lukeSkywalker) = true

isHuman(sword) = false

Syntax: Function Symbols

A function f (f - symbol) maps object(s) to an object.

$$f(argument_1, argument_2, ..., argument_N)$$

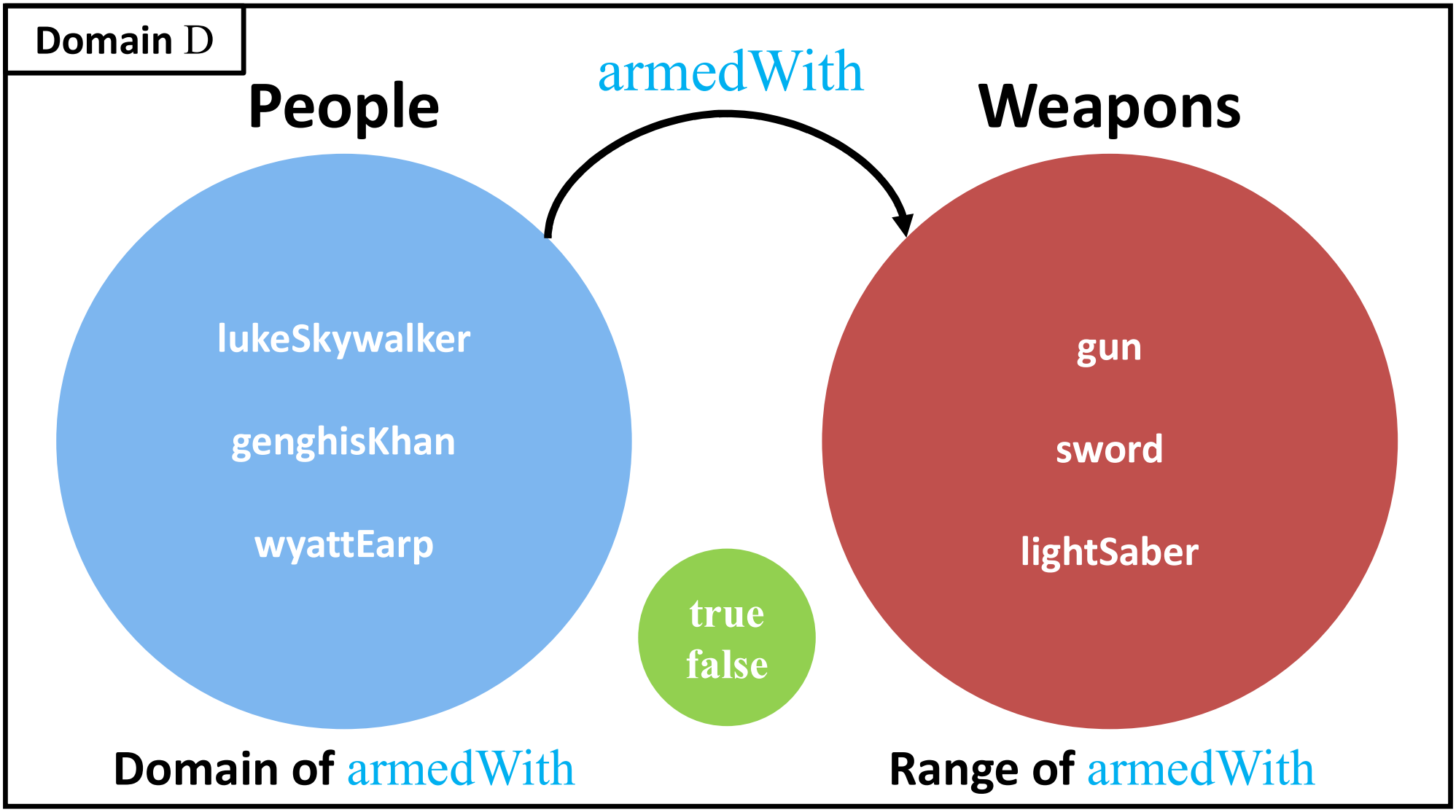
Unlike predicates, functions:

- **take objects as arguments**
- **output objects**
- **for example:**

currentCourse(student) **would output object** cs480

instructor(cs480) **would output object** Jacek

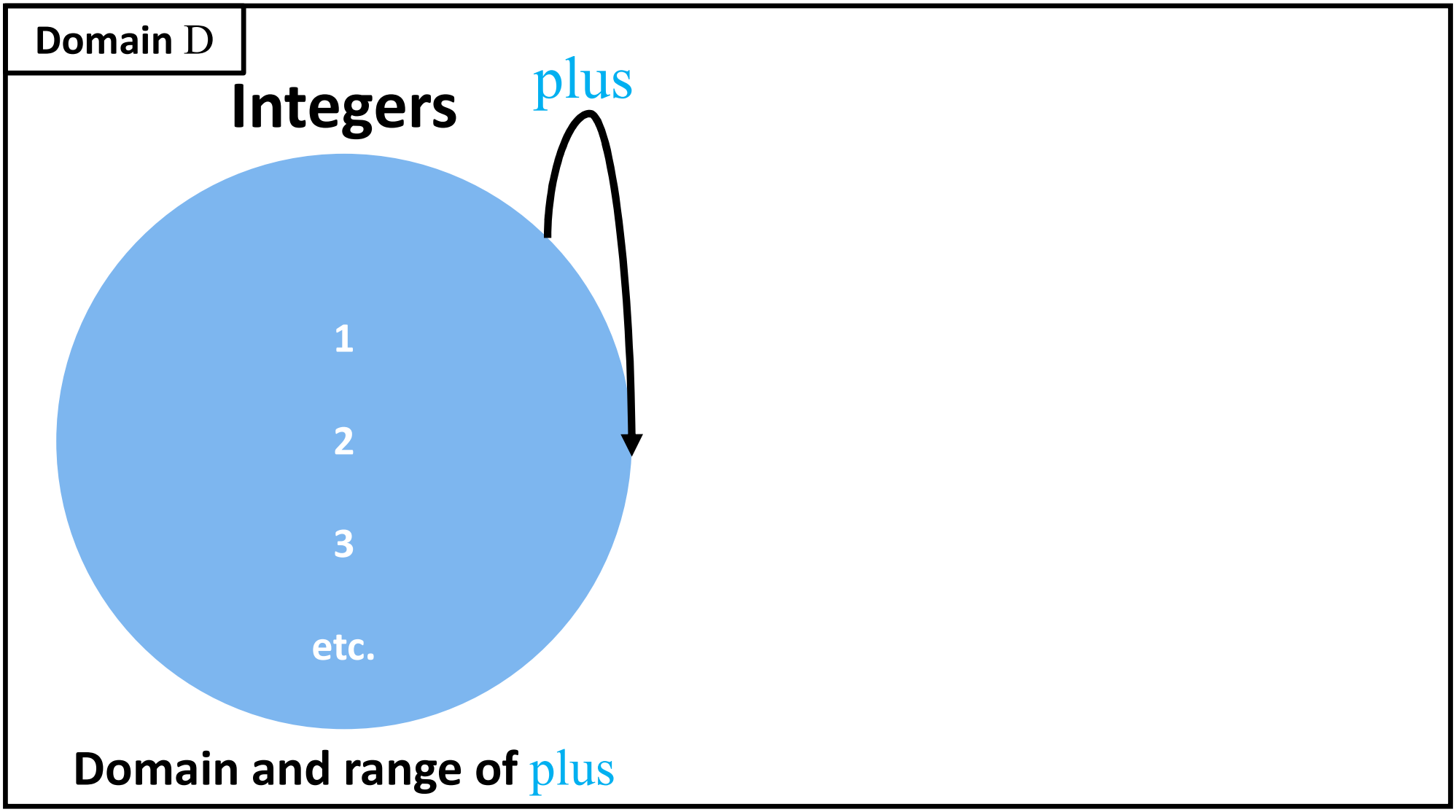
Functions: Example



armedWith(lukeSkywalker) = lightSaber

armedWith(genghisKhan) = sword

Functions: Example



$$\text{plus}(2,3) = 5$$

Syntax: Terms

A term is a logical expression that refers to an object. Terms can be:

- **constants:** lukeSkywalker
- **variables:** son
- **complex term - a function symbol followed by a parenthesized list of terms as arguments:**

father(lukeSkywalker) **or**

father(brother(princessLeia))

Both output an OBJECT: darthVader

Syntax: Complex Sentences

Complex sentences can be constructed using logical connectives (just like in propositional logic). For example:

$$\neg \text{brother}(\text{leftLeg}(\text{Richard}), \text{John})$$

$$\text{brother}(\text{Richard}, \text{John}) \wedge \text{brother}(\text{John}, \text{Richard})$$

$$\text{king}(\text{Richard}) \vee \text{king}(\text{John})$$

$$\neg \text{king}(\text{Richard}) \Rightarrow \text{king}(\text{John})$$

Syntax: Universal Quantifier

Quantifiers allow expressing properties of collections of objects.

Universal quantifier ("for all") indicates that a sentence is true for all possible values of the variable. For example:

$$\forall x \; likes(x, cake)$$

- $\forall$ **is the universal quantifier symbol**
- x **is a variable**
- $likes(x, cake)$ **is a sentence**

Syntax: Existential Quantifier

Quantifiers allow expressing properties of collections of objects.

Existential quantifier ("there exists") indicates that a sentence is true for at least one value of the the variable. For example:

$$\exists x \; likes(x, cake)$$

- $\exists$ **is the existential quantifier symbol**
- x **is a variable**
- $likes(x, cake)$ **is a sentence**

Syntax: Equality

In predicate logic we can ise the equality symbol to indicate that two terms refer to the same object.

For example:

$$\text{father}(\text{John}) = \text{Henry}$$

This sentence means that the output of function father(John) **is the same as object (constant)** Henry**.**

Syntax: Summary

$$
\begin{array}{rcl}
\textit{Sentence} & \rightarrow & \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} & \rightarrow & \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} & \rightarrow & (\ \textit{Sentence}\ ) \\
& \mid & \neg\ \textit{Sentence} \\
& \mid & \textit{Sentence} \wedge \textit{Sentence} \\
& \mid & \textit{Sentence} \vee \textit{Sentence} \\
& \mid & \textit{Sentence} \Rightarrow \textit{Sentence} \\
& \mid & \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
& \mid & \textit{Quantifier}\ \textit{Variable}, \ldots\ \textit{Sentence} \\
\\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \\
& \mid & \textit{Constant} \\
& \mid & \textit{Variable} \\
\\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\textit{Predicate} & \rightarrow & \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} & \rightarrow & \textit{Mother} \mid \textit{LeftLeg} \mid \cdots \\
\text{OPERATOR PRECEDENCE} & : & \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{array}
$$

Predicate Calculus: Interpretation

Let the domain D be a non-empty set. An *interpretation* I over D is an assignment of the entities in D to each of the constant, variable, predicate, and function symbols of a predicate calculus expression, such that:

- **each constant is assigned an element of D,**
- **each variable is assigned to non-empty subset of D; these are the allowable substitutions for that variable,**
- **each function f of arity m is defined on m arguments of D and defines mapping from D^m into D,**
- **each predicate p of arity n is defined on n arguments from D and defines a mapping from D^n into $\{true, false\}$.**

Predicate Calculus: Sentences

Predicate calculus sentences are created according to following rules:

- **every atomic sentence** s **is a sentence**
- **if** s **a sentence, so it its negation** $\neg s$**,**
- **if** s_1 **and** s_2 **are sentences, then so is their conjunction** $s_1 \wedge s_2$**,**
- **if** s_1 **and** s_2 **are sentences, then so is their disjunction** $s_1 \vee s_2$**,**
- **if** s_1 **and** s_2 **are sentences, then so is their implication** $s_1 \Rightarrow s_2$**,**
- **if** s_1 **and** s_2 **are sentences, then so is their equivalence** $s_1 \Rightarrow s_2$**,**
- **if** x **is a variable** s **a sentence, then** $\forall x$ **is a sentence,**
- **if** x **is a variable** s **a sentence, then** $\exists x$ **is a sentence.**

Predicate Calculus: Evaluation

Assume a sentence E and an interpretation I for E over a non-empty domain D. The truth value for E is determined by:

- **the value of each constant is the element of D it is assigned to by I,**
- **the value of a variable is the set of elements of D it is assigned to by I,**
- **the value of a function expression is that element of D obtained by evaluating the function for the parameter values I assigned ,**
- **the value of "true" is true and "false" is false,**
- **The value of an atomic sentence is either true or false, as determined by the interpretation I.**

Predicate Calculus: Evaluation

- **the value of negation of a sentence is** true **if the value of the sentence is** false **(is false if the value of the sentence is** true**),**
- **the value of the conjunction of two sentences is** true **if the value of both sentences is** true **and is** false **otherwise**
- **similarly, the truth value of expressions using** $\vee$, $\Rightarrow$, **and** $\Leftrightarrow$ **is determined using related truth tables,**

- **The value of** $\forall x\ s$ **is** true **if** s **is** true **for all assignments to** x **under** I**, and it is** false **otherwise,**
- **The value of** $\exists x\ s$ **is** true **if** s **is** true **if there is an assignment to** x **under** I**, and it is** false **otherwise,**

First-Order Logic Sentences: Examples

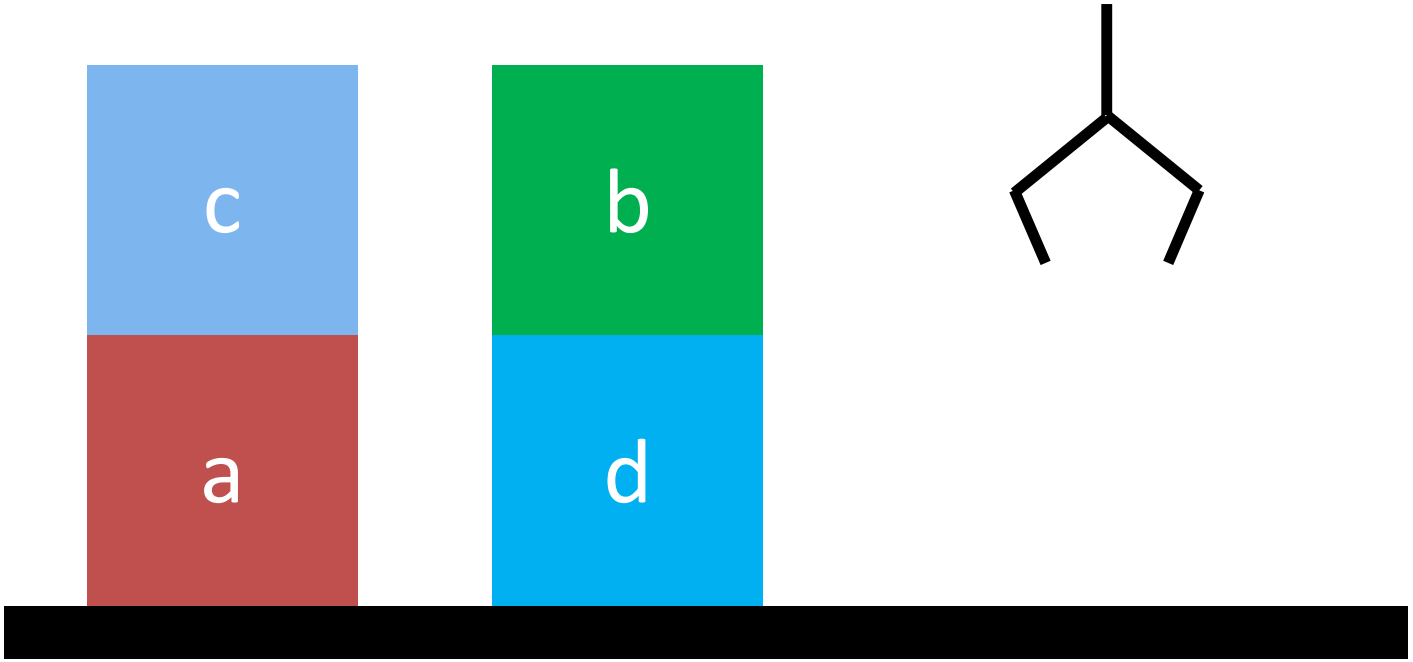
Sentence in First-Order Logic	Sentence in English
$\forall x\ frog(x) \Rightarrow green(x)$	All frogs are green. (For all xs being a frog implies being green)
$\forall x\ frog(x) \wedge brown(x) \Rightarrow big(x)$	All brown frogs are big. (For all xs being a frog and brown implies being big)
$\forall x\ likes(x, cake)$	Everyone likes cake. (All xs like cake.)
$\neg\forall x\ likes(x, cake)$	Not everyone likes cake. (Not all xs like cake.)
$\exists x\ \forall y\ likes(y, x)$	There is something that everyone likes. (There exists something x that all ys like.)
$\exists x\ \forall y\ likes(x, y)$	There is someone that likes everyone. (There exists an x that is likes all ys.)
$\forall x\ \exists y\ likes(y, x)$	Everything is liked by someone. (For all xs there exists a y that likes them.)
$\forall x\ \exists y\ likes(x, y)$	Everyone likes something. (For all xs there exists a y that is being liked by x.)

First-Order Logic Sentences: Examples

Sentence in First-Order Logic	Sentence in English
$\forall x\ customer(x) \Rightarrow likes(bob, x)$	Bob likes every customer. (For all xs being a customer implies being liked by Bob.)
$\exists x\ customer(x) \wedge likes(x, bob)$	There is a customer who likes Bob. (There exist an x who is a customer and likes Bob.)
$\exists x\ baker(x) \wedge \forall y\ customer(y) \Rightarrow likes(x, y)$	There is a baker who likes all customers. (There exist an x who is a baker such that for all ys being a customer, implies that y is liked by x.)
$\forall x\ older(mother(x), x)$ *	Every mother is older than her child. (For all xs, mother of x is older than x.)
$\forall x\ older(mother(mother(x)), x)$ *	Every grandmother is older than her daugther's child. (For all xs, mother of mother of x is older than x.)
$brother(x, bob) \Rightarrow \exists y\ parent(y, x) \wedge parent(y, bob)$	If x is Bob's brother, x and Bob must have the same parent.
$\forall x \forall y \forall z\ f(x, y) \wedge f(y, z) \Rightarrow f(x, z)$	f is a transitive relation.

* mother is function symbol here

Predicate Calculus: Model Example

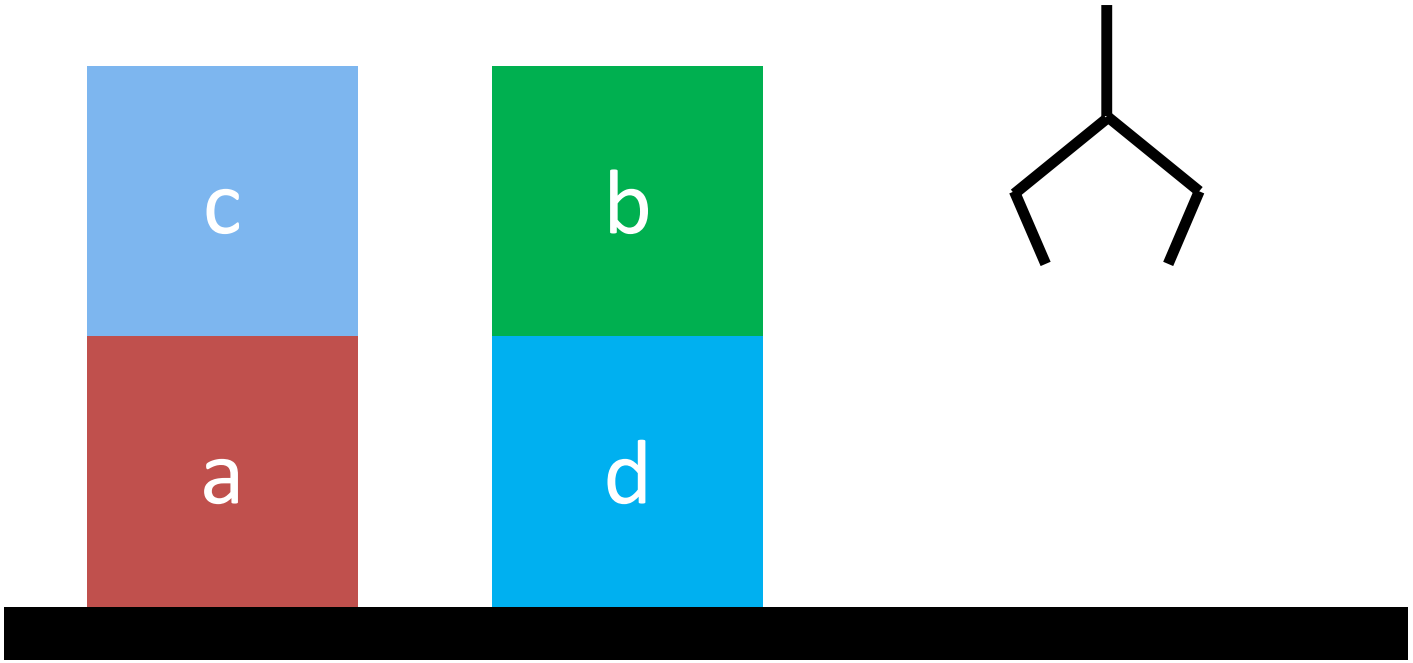


Predicate calculus description of the world:

on(c, a)
on(b, d)
onTable(a)
onTable(d)

clear(b)
clear(c)
manipulatorEmpty

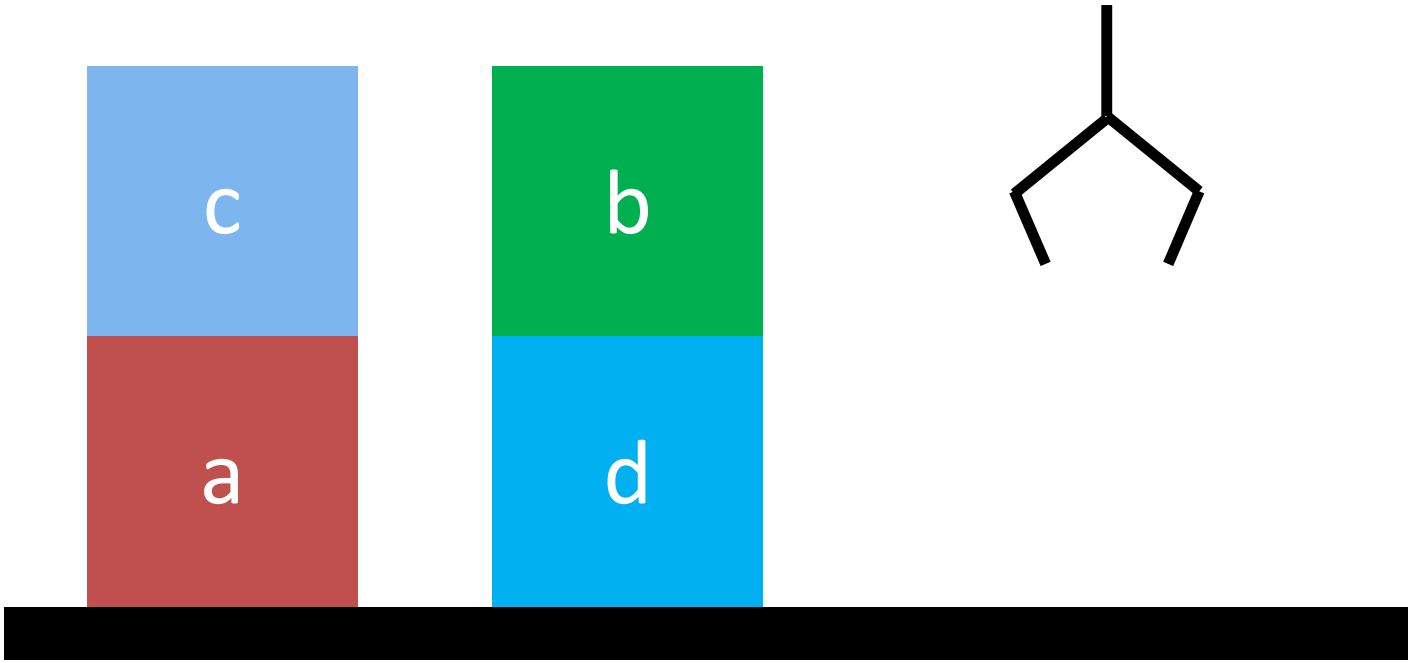
Predicate Calculus: Objects



Objects:

table
blocks a, b, c, d
manipulator

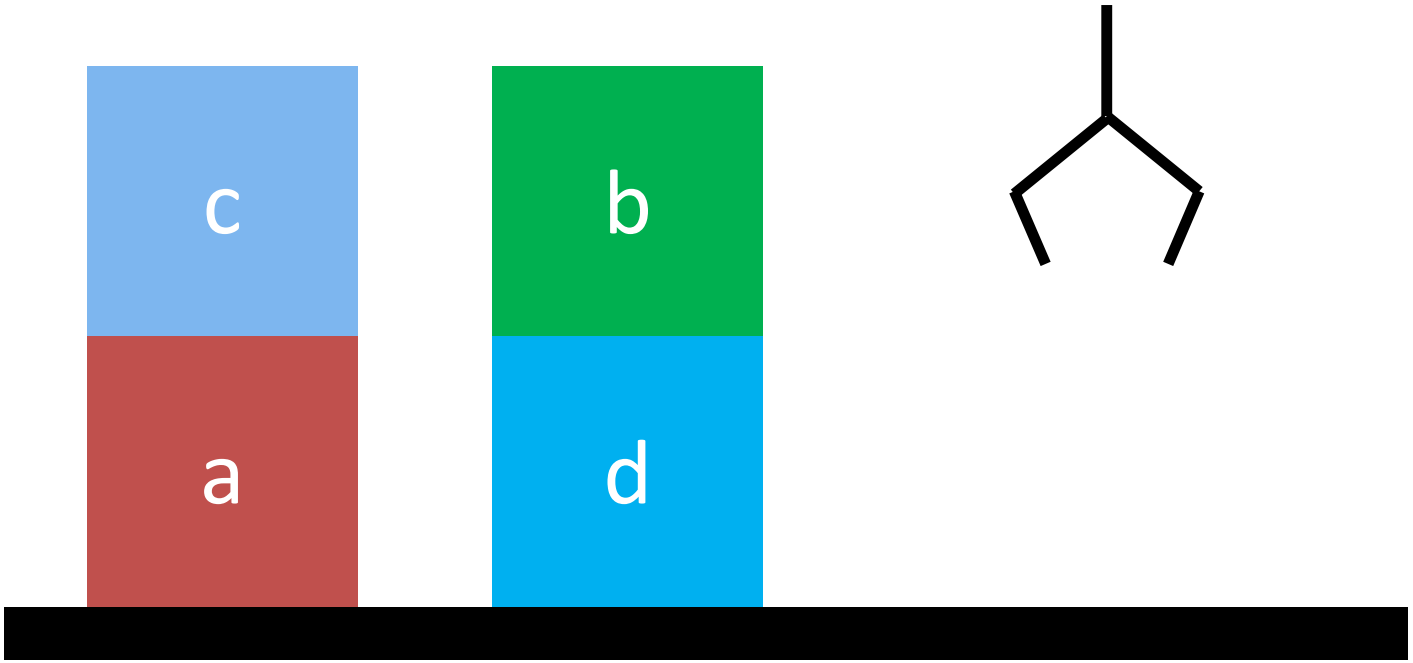
Predicate Calculus: Predicates



Predicates:

on(x, y)	**asserts that object x is on top of y**
onTable(x)	**asserts that object x is on the table**
clear(x)	**asserts that no objects is on top of x**
manipulatorEmpty	**asserts the status of the manipulator**

Model of the World



Predicate calculus description of the world (model):

on(c, a)
on(b, d)
onTable(a)
onTable(d)

clear(b)
clear(c)
manipulatorEmpty

Quantifier Nesting

Quantifiers can be nested to obtain more complex expressions. For example:

$$\forall x\ \forall y\ \text{brother}(x, y) \Rightarrow \text{sibling}(x, y)$$

means "Brothers are siblings". Here

$$\forall x\ \forall y\ \text{sibling}(y, x) \Leftrightarrow \text{sibling}(x, y)$$

a symmetric relationship is expressed.

Quantifier Nesting: Ordering

When quantifiers are nested it is important to pay attention to ordering. For example:

$$\forall x \, \exists y \, \text{loves}(x, y)$$

means "Everybody loves somebody". Here

$$\exists x \, \forall y \, \text{loves}(y, x)$$

we have "There exists someone who is loved by everyone".

Quantifier Nesting (Use Parentheses)

Quantifiers can be nested to obtain more complex expressions. For example:

$$\forall x\ (\forall y\ \text{brother}(x, y) \Rightarrow \text{sibling}(x, y))$$

means "Brothers are siblings". Here

$$\forall x\ (\forall y\ \text{sibling}(y, x) \Leftrightarrow \text{sibling}(x, y))$$

a symmetric relationship is expressed.

Quantifier Nesting (Use Parentheses)

When quantifiers are nested it is important to pay attention to ordering. For example:

$$\forall x\ (\exists y\ \text{loves}(x, y))$$

means "Everybody loves somebody". Here

$$\exists x\ (\forall y\ \text{loves}(y, x))$$

we have "There exists someone who is loved by everyone".

Quantifier Nesting: Variable Names

Certain quantified sentences may be confusing:

$$\forall x\ (\text{crown}(x) \vee (\exists x\ \text{brother}(\text{Richard}, x))$$

Variable x is used twice:

- **universally quantified x in** $\forall x\ (\text{crown}(x) \vee$
- **existientially quantified x in** $\exists x\ \text{brother}(\text{Richard}, x)$
- **x and x are NOT the same (different "context")**

Rule:

variable belongs to innermost quantifier that mentions it.

Universal Quantifier: Conjuctions

Universal quantifier ("for all") indicates that a sentence is true for all possible values of the variable. For example:

$$\forall x \; likes(x, cake)$$

is true if $likes(x, cake)$ **is** true **for all interpretations of variable** x**. Assuming that**

$$x \in \{x_1, x_2, ..., x_n\}$$

we can rewrite $\forall x \; likes(x, cake)$ **as:**

$$likes(x_1, cake) \wedge likes(x_2, cake) \wedge ... \wedge likes(x_n, cake)$$

Existential Quantifier: Disjunctions

Existential quantifier ("there exists") indicates that a sentence is true for at least one value of the the variable. For example:

$$\exists x \; likes(x, cake)$$

is true if $likes(x, cake)$ **is** true **for at least one interpretation of variable** x**. Assuming that**

$$x \in \{x_1, x_2, ..., x_n\}$$

we can rewrite $\exists x \; likes(x, cake)$ **as:**

$$likes(x_1, cake) \vee likes(x_2, cake) \vee ... \vee likes(x_n, cake)$$

Universal/Existential Quantifiers

We assumed that $x \in \{x_1, x_2, ..., x_n\}$ **and then we rewrote** $\forall x\ likes(x, cake)$ **as:**

$$likes(x_1, cake) \wedge likes(x_2, cake) \wedge ... \wedge likes(x_n, cake)$$

and $\exists x\ likes(x, cake)$ **as:**

$$likes(x_1, cake) \vee likes(x_2, cake) \vee ... \vee likes(x_n, cake)$$

From De Morgan's rules we can obtain the following equivalence:

$$\forall x\ likes(x, cake) \equiv \neg \exists x\ \neg likes(x, cake)$$

"Everyone likes cake" ≡ "Nobody dislikes cake"

Universal/Existential Q. Equivalences

Selected equivalences:

$$\forall x\,(P(x) \wedge Q(x)) \equiv \forall x\,(P(x)) \wedge \forall x\,(Q(x))$$

$$\exists x\,(P(x) \vee Q(x)) \equiv \exists x\,(P(x)) \vee \exists x\,(Q(x))$$

$$\neg[\exists x\,(N(x))] \equiv \forall x\,(\neg N(x))$$

$$\neg[\forall x\,(N(x))] \equiv \exists x\,(\neg N(x))$$

Quantifiers: Scope of Quantification

Consider the following sentence:

$$\forall x\,(P(x) \wedge Q(x))$$

Scope of quantification for variable x

Variable x is universally quantified in both $P(x)$ and $Q(x)$.

In this sentence:

$$\exists x\,(P(x) \vee Q(y) \Rightarrow R(x))$$

Scope of quantification for variable x

Variable x is existentionally quantified in both $P(x)$ and $R(x)$.