"Consequently he who wishes to attain to human perfection, must therefore first study Logic, next the various branches of Mathematics in their proper order, then Physics, and lastly Metaphysics."

Maimonides 1135–1204

"Logic is the beginning of wisdom, not the end."

Leonard Nimoy
"Star Trek VI: The Undiscovered Country" 1991

Individuals and Relations

- It is useful to view the world as consisting of individuals (entitles, objects, things) with properties of individuals and relations among individuals.

Individuals and Relations

- It is useful to view the world as consisting of individuals (entitles, objects, things) with properties of individuals and relations among individuals.
- Features are made from properties of individuals and relations among individuals.
  E.g, Argentina is a county.
  Argentina won FIFA World cup in 2022.

Individuals and Relations

- It is useful to view the world as consisting of individuals (entitles, objects, things) with properties of individuals and relations among individuals.
- Features are made from properties of individuals and relations among individuals.
  E.g, Argentina is a county.
  Argentina won FIFA World cup in 2022.
- Reasoning in terms of individuals and relationships can be simpler than reasoning in terms of features, if we can express general knowledge that covers all individuals.

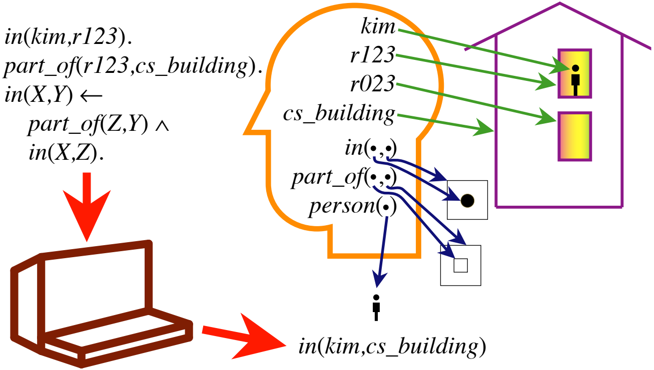
Individuals and Relations

- It is useful to view the world as consisting of individuals (entitles, objects, things) with properties of individuals and relations among individuals.
- Features are made from properties of individuals and relations among individuals.
  E.g, Argentina is a county.
  Argentina won FIFA World cup in 2022.
- Reasoning in terms of individuals and relationships can be simpler than reasoning in terms of features, if we can express general knowledge that covers all individuals.
- Sometimes you may know some individual exists, but not which one.

Individuals and Relations

- It is useful to view the world as consisting of individuals (entitles, objects, things) with properties of individuals and relations among individuals.
- Features are made from properties of individuals and relations among individuals.
  E.g, Argentina is a county.
  Argentina won FIFA World cup in 2022.
- Reasoning in terms of individuals and relationships can be simpler than reasoning in terms of features, if we can express general knowledge that covers all individuals.
- Sometimes you may know some individual exists, but not which one.
- Sometimes there are infinitely many individuals we want to refer to (e.g., set of all integers, or the set of all stacks of blocks).

Role of Semantics in Automated Reasoning



- Users can have meanings for symbols in their head.
  They tell the computer what is true.
- The computer doesn't need to know these meanings to derive logical consequence.
- Users can interpret any answers according to their meaning.

Predicate Calculus

Predicate calculus, or just predicate logic, extends propositional calculus in two ways:

- atoms have structure and can include constants and logical variables
- quantification of logical variables.

Predicate Calculus

Predicate calculus, or just predicate logic, extends propositional calculus in two ways:

- atoms have structure and can include constants and logical variables
- quantification of logical variables.

Syntactic convention of Datalog / Prolog:

- variables start with an upper-case letter.
- constants, predicates and functions start with a lower-case letter.

In mathematics, variables typically are x, y, and z.

A Syntax of Datalog and First-order Logic

- A variable starts with upper-case letter.
  E.g., *A*, *Person*, *Country*.

A Syntax of Datalog and First-order Logic

- A variable starts with upper-case letter.
  E.g., *A*, *Person*, *Country*.
- A constant starts with lower-case letter or is a sequence of digits (numeral).
  E.g., *argentina*, 2022, *fifa_world_cup*.

A Syntax of Datalog and First-order Logic

- A variable starts with upper-case letter.
  E.g., *A*, *Person*, *Country*.
- A constant starts with lower-case letter or is a sequence of digits (numeral).
  E.g., *argentina*, 2022, *fifa_world_cup*.
- A predicate symbol starts with lower-case letter.
  E.g., *won*, *plays_for*, *parent*.

A Syntax of Datalog and First-order Logic

- A variable starts with upper-case letter.
  E.g., *A*, *Person*, *Country*.
- A constant starts with lower-case letter or is a sequence of digits (numeral).
  E.g., *argentina*, 2022, *fifa_world_cup*.
- A predicate symbol starts with lower-case letter.
  E.g., *won*, *plays_for*, *parent*.
- A term is either a variable or a constant.

A Syntax of Datalog and First-order Logic

- A variable starts with upper-case letter.
  E.g., A, $Person$, $Country$.
- A constant starts with lower-case letter or is a sequence of digits (numeral).
  E.g., $argentina$, 2022, $fifa\_world\_cup$.
- A predicate symbol starts with lower-case letter.
  E.g., won, $plays\_for$, $parent$.
- A term is either a variable or a constant.
- An atomic symbol (atom) is of the form p or $p(t_1, \ldots, t_n)$ where p is a predicate symbol and t_i are terms.
  E.g., $won(argentina,\ fifa\_world\_cup,\ 2022)$.

A Syntax of Datalog and First-order Logic

- A variable starts with upper-case letter.
  E.g., A, $Person$, $Country$.
- A constant starts with lower-case letter or is a sequence of digits (numeral).
  E.g., $argentina$, 2022, $fifa\_world\_cup$.
- A predicate symbol starts with lower-case letter.
  E.g., won, $plays\_for$, $parent$.
- A term is either a variable or a constant.
- An atomic symbol (atom) is of the form p or $p(t_1, \ldots, t_n)$ where p is a predicate symbol and t_i are terms.
  E.g., $won(argentina,\ fifa\_world\_cup,\ 2022)$.
- Logical connectives: $\neg$ (not), $\wedge$ (and), $\vee$ (or), $\leftarrow$ (if), $\rightarrow$ (implies), $\leftrightarrow$ (equivalence)

A Syntax of Datalog and First-order Logic

- A variable starts with upper-case letter.
  E.g., A, $Person$, $Country$.
- A constant starts with lower-case letter or is a sequence of digits (numeral).
  E.g., $argentina$, 2022, $fifa\_world\_cup$.
- A predicate symbol starts with lower-case letter.
  E.g., won, $plays\_for$, $parent$.
- A term is either a variable or a constant.
- An atomic symbol (atom) is of the form p or $p(t_1, \ldots, t_n)$ where p is a predicate symbol and t_i are terms.
  E.g., $won(argentina,\ fifa\_world\_cup,\ 2022)$.
- Logical connectives: $\neg$ (not), $\wedge$ (and), $\vee$ (or), $\leftarrow$ (if), $\rightarrow$ (implies), $\leftrightarrow$ (equivalence)
- Quantification: $\forall$ (for all), $\exists$ (there exists)

Syntax of Datalog

- A definite clause is either an atomic symbol (a fact) or a rule of the form:

$$\underbrace{a}_{\text{head}} \leftarrow \underbrace{b_1 \wedge \cdots \wedge b_m}_{\text{body}}$$

where a and b_i are atomic symbols.

Syntax of Datalog

- A definite clause is either an atomic symbol (a fact) or a rule of the form:

$$\underbrace{a}_{\text{head}} \leftarrow \underbrace{b_1 \land \cdots \land b_m}_{\text{body}}$$

  where a and b_i are atomic symbols.
- query is of the form $?b_1 \land \cdots \land b_m$.

Syntax of Datalog

- A definite clause is either an atomic symbol (a fact) or a rule of the form:

$$\underbrace{a}_{\text{head}} \leftarrow \underbrace{b_1 \wedge \cdots \wedge b_m}_{\text{body}}$$

  where a and b_i are atomic symbols.
- query is of the form $?b_1 \wedge \cdots \wedge b_m$.
- knowledge base is a set of definite clauses.

Example Data

The relation

Course	Section	Time	Room
cs111	7	830	dp101
cs422	2	1030	cc208
cs502	1	1230	dp202

Example Data

The relation

Course	Section	Time	Room
cs111	7	830	dp101
cs422	2	1030	cc208
cs502	1	1230	dp202

can be represented by the facts:

$$scheduled(cs111, 7, 830, dp101).$$
$$scheduled(cs422, 2, 1030, cc208).$$
$$scheduled(cs502, 1, 1230, dp202).$$

Example Data

The relation

Course	Section	Time	Room
cs111	7	830	dp101
cs422	2	1030	cc208
cs502	1	1230	dp202

can be represented by the facts:

$$scheduled(cs111, 7, 830, dp101).$$
$$scheduled(cs422, 2, 1030, cc208).$$
$$scheduled(cs502, 1, 1230, dp202).$$

A student is busy when they have a class:

$$busy(StudentNum, Time) \leftarrow$$
$$enrolled(StudentNum, Course, Section) \land$$
$$scheduled(Course, Section, Time, Room).$$

Example Rules

$$
\begin{aligned}
& in(kim, R) \leftarrow \\
& \quad teaches(kim, cs322) \land \\
& \quad in(cs322, R). \\
& grandfather(william, X) \leftarrow \\
& \quad father(william, Y) \land \\
& \quad parent(Y, X). \\
& slithy(toves) \leftarrow \\
& \quad mimsy \land borogroves \land \\
& \quad outgrabe(mome, Raths).
\end{aligned}
$$

Semantics: General Idea

A semantics specifies the meaning of sentences in the language.
An interpretation specifies:

- what objects (individuals) are in the world

Semantics: General Idea

A semantics specifies the meaning of sentences in the language.

An interpretation specifies:

- what objects (individuals) are in the world
- the correspondence between symbols in the computer and objects & relations in world

Semantics: General Idea

A semantics specifies the meaning of sentences in the language.

An interpretation specifies:

- what objects (individuals) are in the world
- the correspondence between symbols in the computer and objects & relations in world
  - constants denote individuals

Semantics: General Idea

A semantics specifies the meaning of sentences in the language.
An interpretation specifies:

- what objects (individuals) are in the world
- the correspondence between symbols in the computer and objects & relations in world
  - constants denote individuals
  - predicate symbols denote relations

Formal Semantics

An interpretation is a triple $I = \langle D, \phi, \pi \rangle$, where

- D, the domain, is a nonempty set. Elements of D are individuals.

Formal Semantics

An interpretation is a triple $I = \langle D, \phi, \pi \rangle$, where

- D, the domain, is a nonempty set. Elements of D are individuals.
- ϕ is a mapping that assigns to each constant an element of D. Constant c denotes individual $\phi(c)$.

Formal Semantics

An interpretation is a triple $I = \langle D, \phi, \pi \rangle$, where

- D, the domain, is a nonempty set. Elements of D are individuals.
- ϕ is a mapping that assigns to each constant an element of D. Constant c denotes individual $\phi(c)$.
- π is a mapping that assigns to each n-ary predicate symbol a relation: a function from D^n into $\{TRUE, FALSE\}$.

Example Interpretation

Constants: *phone*, *pencil*, *telephone*.

Example Interpretation

Constants: *phone*, *pencil*, *telephone*.
Predicate Symbol: *noisy* (unary), *left_of* (binary).

Example Interpretation

Constants: *phone*, *pencil*, *telephone*.
Predicate Symbol: *noisy* (unary), *left_of* (binary).

- $D = \{$✂, ☎, ✎$\}$.

Example Interpretation

Constants: *phone*, *pencil*, *telephone*.
Predicate Symbol: *noisy* (unary), *left_of* (binary).

- $D = \{$✄, ☎, ✎$\}$.
- $\phi(phone) =$ ☎, $\phi(pencil) =$ ✎, $\phi(telephone) =$ ☎.

Example Interpretation

Constants: *phone*, *pencil*, *telephone*.
Predicate Symbol: *noisy* (unary), *left_of* (binary).

- $D = \{$✂, ☎, ✎$\}$.
- $\phi(phone) =$ ☎, $\phi(pencil) =$ ✎, $\phi(telephone) =$ ☎.
- $\pi(noisy)$:

⟨✂⟩	*FALSE*	⟨☎⟩	*TRUE*	⟨✎⟩	*FALSE*

Example Interpretation

Constants: *phone*, *pencil*, *telephone*.
Predicate Symbol: *noisy* (unary), *left_of* (binary).

- *D* = {✂, ☎, ✎}.
- φ(*phone*) = ☎, φ(*pencil*) = ✎, φ(*telephone*) = ☎.
- π(*noisy*):

⟨✂⟩	*FALSE*	⟨☎⟩	*TRUE*	⟨✎⟩	*FALSE*

  π(*left_of*):

⟨✂, ✂⟩	*FALSE*	⟨✂, ☎⟩	*TRUE*	⟨✂, ✎⟩	*TRUE*
⟨☎, ✂⟩	*FALSE*	⟨☎, ☎⟩	*FALSE*	⟨☎, ✎⟩	*TRUE*
⟨✎, ✂⟩	*FALSE*	⟨✎, ☎⟩	*FALSE*	⟨✎, ✎⟩	*FALSE*

Important points to note

- The domain D can contain real objects. (e.g., a person, a room, a course). D can't necessarily be stored in a computer.

Important points to note

- The domain D can contain real objects. (e.g., a person, a room, a course). D can't necessarily be stored in a computer.
- $\pi(p)$ specifies whether the relation denoted by the n-ary predicate symbol p is true or false for each n-tuple of individuals.

Important points to note

- The domain D can contain real objects. (e.g., a person, a room, a course). D can't necessarily be stored in a computer.
- $\pi(p)$ specifies whether the relation denoted by the n-ary predicate symbol p is true or false for each n-tuple of individuals.
- If predicate symbol p has no arguments, then $\pi(p)$ is either *TRUE* or *FALSE*.

Truth in an interpretation

A constant c denotes in I the individual $\phi(c)$.

Truth in an interpretation

A constant c denotes in I the individual $\phi(c)$.
Ground (variable-free) atom $p(t_1, \ldots, t_n)$ is

- true in interpretation I

Truth in an interpretation

A constant c denotes in I the individual $\phi(c)$.
Ground (variable-free) atom $p(t_1, \ldots, t_n)$ is

- true in interpretation I if $\pi(p)(\langle\phi(t_1), \ldots, \phi(t_n)\rangle) = \mathit{TRUE}$ in interpretation I and

Truth in an interpretation

A constant c denotes in I the individual $\phi(c)$.
Ground (variable-free) atom $p(t_1, \ldots, t_n)$ is

- true in interpretation I if $\pi(p)(\langle \phi(t_1), \ldots, \phi(t_n) \rangle) = \mathit{TRUE}$ in interpretation I and
- false otherwise.

Truth in an interpretation

A constant c denotes in I the individual $\phi(c)$.
Ground (variable-free) atom $p(t_1, \ldots, t_n)$ is

- true in interpretation I if $\pi(p)(\langle \phi(t_1), \ldots, \phi(t_n) \rangle) = \mathit{TRUE}$ in interpretation I and
- false otherwise.

Ground clause $h \leftarrow b_1 \wedge \ldots \wedge b_m$ is false in interpretation I if h is false in I and each b_i is true in I, and is true in interpretation I otherwise.

Example Truths

In the interpretation given before, which of following are true?

$noisy(phone)$
$noisy(telephone)$
$noisy(pencil)$
$left\_of(phone, pencil)$
$left\_of(phone, telephone)$
$noisy(phone) \leftarrow left\_of(phone, telephone)$
$noisy(pencil) \leftarrow left\_of(phone, telephone)$
$noisy(pencil) \leftarrow left\_of(phone, pencil)$
$noisy(phone) \leftarrow noisy(telephone) \wedge noisy(pencil)$

Example Truths

In the interpretation given before, which of following are true?

$noisy(phone)$	true
$noisy(telephone)$	true
$noisy(pencil)$	false
$left\_of(phone, pencil)$	true
$left\_of(phone, telephone)$	false
$noisy(phone) \leftarrow left\_of(phone, telephone)$	true
$noisy(pencil) \leftarrow left\_of(phone, telephone)$	true
$noisy(pencil) \leftarrow left\_of(phone, pencil)$	false
$noisy(phone) \leftarrow noisy(telephone) \wedge noisy(pencil)$	true

Models and logical consequences

- A knowledge base, KB, is true in interpretation I if and only if every clause in KB is true in I.

Models and logical consequences

- A knowledge base, KB, is true in interpretation I if and only if every clause in KB is true in I.
- A model of a set of clauses is an interpretation in which all the clauses are true.

Models and logical consequences

- A knowledge base, KB, is true in interpretation I if and only if every clause in KB is true in I.
- A model of a set of clauses is an interpretation in which all the clauses are true.
- If KB is a set of clauses and g is a conjunction of atoms, g is a logical consequence of KB, written $KB \models g$, if g is true in every model of KB.

Models and logical consequences

- A knowledge base, KB, is true in interpretation I if and only if every clause in KB is true in I.
- A model of a set of clauses is an interpretation in which all the clauses are true.
- If KB is a set of clauses and g is a conjunction of atoms, g is a logical consequence of KB, written $KB \models g$, if g is true in every model of KB.
- That is, $KB \models g$ if there is no interpretation in which KB is true and g is false.

User's view of Semantics

1. Choose a task domain: intended interpretation.
2. Associate constants with individuals you want to name.
3. For each relation you want to represent, associate a predicate symbol in the language.

User's view of Semantics

1. Choose a task domain: intended interpretation.
2. Associate constants with individuals you want to name.
3. For each relation you want to represent, associate a predicate symbol in the language.
4. Tell the system clauses that are true in the intended interpretation: axiomatizing the domain.

User's view of Semantics

1. Choose a task domain: intended interpretation.
2. Associate constants with individuals you want to name.
3. For each relation you want to represent, associate a predicate symbol in the language.
4. Tell the system clauses that are true in the intended interpretation: axiomatizing the domain.
5. Ask questions about the intended interpretation.

User's view of Semantics

1. Choose a task domain: intended interpretation.
2. Associate constants with individuals you want to name.
3. For each relation you want to represent, associate a predicate symbol in the language.
4. Tell the system clauses that are true in the intended interpretation: axiomatizing the domain.
5. Ask questions about the intended interpretation.
6. If $KB \models g$, then

User's view of Semantics

1. Choose a task domain: intended interpretation.
2. Associate constants with individuals you want to name.
3. For each relation you want to represent, associate a predicate symbol in the language.
4. Tell the system clauses that are true in the intended interpretation: axiomatizing the domain.
5. Ask questions about the intended interpretation.
6. If $KB \models g$, then g must be true in the intended interpretation.

Computer's view of semantics

- The computer doesn't have access to the intended interpretation.

Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.

Computer's view of semantics

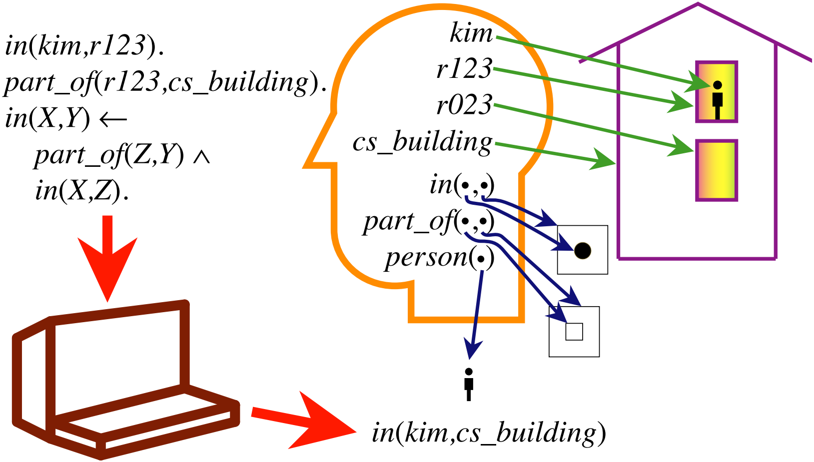
- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.
- The computer can determine if a formula is a logical consequence of KB.

Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.
- The computer can determine if a formula is a logical consequence of KB.
- If $KB \models g$ then g must be true in the intended interpretation.

Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.
- The computer can determine if a formula is a logical consequence of KB.
- If $KB \models g$ then g must be true in the intended interpretation.
- If $KB \not\models g$ then there is a model of KB in which g is false. As far as the computer is concerned, this could be the intended interpretation.


in(kim,r123).
part_of(r123,cs_building).
in(X,Y) ←
part_of(Z,Y) ∧
in(X,Z).
kim
r123
r023
cs_building
in(•,•)
part_of(•,•)
person(•)
in(kim,cs_building)

Variables

- A variable assignment is a function from variables into the domain.
- Given an interpretation and a variable assignment,
  each term denotes an individual and
  each clause is either true or false.

Variables

- A variable assignment is a function from variables into the domain.
- Given an interpretation and a variable assignment,
  each term denotes an individual and
  each clause is either true or false.
- Variables are universally quantified in the scope of a clause.
- A clause containing variables is true in an interpretation if it is true for all variable assignments.

Queries and Answers

A query is a way to ask if a body is a logical consequence of the knowledge base:

$$?b_1 \wedge \cdots \wedge b_m.$$

An answer is either

- an instance of the query that is a logical consequence of the knowledge base KB, or
- no if no instance is a logical consequence of KB.

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building).$	

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building).$	*no*
$?in(kim, r023).$	

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \land in(X, Z). \end{cases}$$

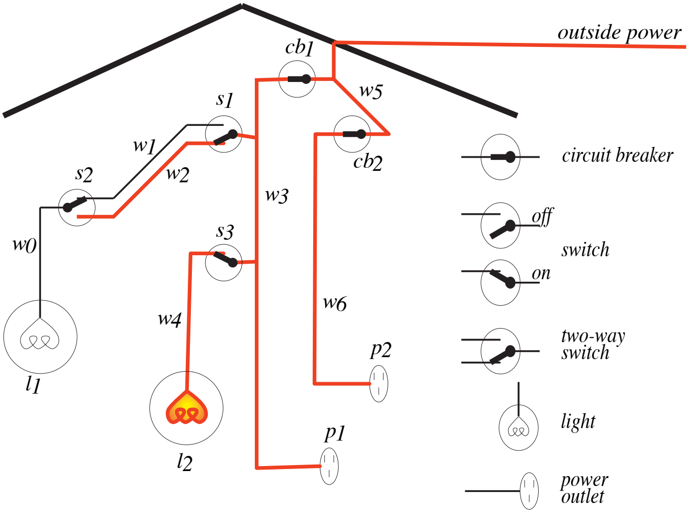
Query	Answer
$?part\_of(r123, B).$	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building).$	no
$?in(kim, r023).$	no
$?in(kim, B).$	

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building).$	no
$?in(kim, r023).$	no
$?in(kim, B).$	$in(kim, r123)$
	$in(kim, cs\_building)$

Electrical Environment



Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light
$light(l_1)$. $light(l_2)$.
% $down(S)$ is true if switch S is down
$down(s_1)$. $up(s_2)$. $up(s_3)$.
% $ok(D)$ is true if D is not broken
$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

$?light(l_1)$. $\Longrightarrow$

Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light
$light(l_1)$. $light(l_2)$.
% $down(S)$ is true if switch S is down
$down(s_1)$. $up(s_2)$. $up(s_3)$.
% $ok(D)$ is true if D is not broken
$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

$?light(l_1)$. $\Longrightarrow$ *yes*
$?light(l_6)$. $\Longrightarrow$

Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light
$light(l_1)$. $light(l_2)$.
% $down(S)$ is true if switch S is down
$down(s_1)$. $up(s_2)$. $up(s_3)$.
% $ok(D)$ is true if D is not broken
$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

$?light(l_1)$. $\Longrightarrow$ *yes*
$?light(l_6)$. $\Longrightarrow$ *no*
$?up(X)$. $\Longrightarrow$

Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light
$light(l_1)$. $light(l_2)$.
% $down(S)$ is true if switch S is down
$down(s_1)$. $up(s_2)$. $up(s_3)$.
% $ok(D)$ is true if D is not broken
$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

$?light(l_1)$. $\Longrightarrow$ yes
$?light(l_6)$. $\Longrightarrow$ no
$?up(X)$. $\Longrightarrow$ $up(s_2)$, $up(s_3)$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W).$ $\Longrightarrow$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W). \quad \Longrightarrow \quad W = w_1$

$?connected\_to(w_1, W). \quad \Longrightarrow$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W).$ $\Longrightarrow$ $W = w_1$

$?connected\_to(w_1, W).$ $\Longrightarrow$ no

$?connected\_to(Y, w_3).$ $\Longrightarrow$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W).$ $\Longrightarrow$ $W = w_1$

$?connected\_to(w_1, W).$ $\Longrightarrow$ no

$?connected\_to(Y, w_3).$ $\Longrightarrow$ $Y = w_2,\ Y = w_4,\ Y = p_1$

$?connected\_to(X, W).$ $\Longrightarrow$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W).$ $\Longrightarrow$ $W = w_1$

$?connected\_to(w_1, W).$ $\Longrightarrow$ no

$?connected\_to(Y, w_3).$ $\Longrightarrow$ $Y = w_2,\ Y = w_4,\ Y = p_1$

$?connected\_to(X, W).$ $\Longrightarrow$ $X = w_0, W = w_1, \ldots$

% $lit(L)$ is true if the light L is lit

$$lit(L) \leftarrow light(L) \land ok(L) \land live(L).$$

% $live(C)$ is true if there is power coming into C

$$live(Y) \leftarrow$$
$$connected\_to(Y, Z) \land$$
$$live(Z).$$
$$live(outside).$$

This is a recursive definition of *live*.

Recursion and Mathematical Induction

$$above(X, Y) \leftarrow on(X, Y).$$
$$above(X, Y) \leftarrow on(X, Z) \wedge above(Z, Y).$$

This can be seen as:

- Recursive definition of *above*: prove *above* in terms of a base case (*on*) or a simpler instance of itself; or
- Way to prove *above* by mathematical induction: the base case is when there are no blocks between X and Y, and if you can prove *above* when there are n blocks between them, you can prove it when there are $n + 1$ blocks.

Limitations

- Suppose you had a database using the relation:

  $$enrolled(S, C)$$

  which is true when student S is enrolled in course C.
- Can you define the relation:

  $$empty\_course(C)$$

  which is true when course C has no students enrolled in it?
- Why? or Why not?

Limitations

- Suppose you had a database using the relation:

  $$enrolled(S, C)$$

  which is true when student S is enrolled in course C.
- Can you define the relation:

  $$empty\_course(C)$$

  which is true when course C has no students enrolled in it?
- Why? or Why not?
  $empty\_course(C)$ doesn't logically follow from a set of *enrolled* relation because there are always models where someone is enrolled in a course!