From graphs to predicates

Dave Dubin

February 2019

Predicates

- A *predicate* is a kind of expression that represents a proposition.

Predicates

- A *predicate* is a kind of expression that represents a proposition.
- A predicate consists of a *predicate symbol* and one or more *arguments*.

Predicates

- A *predicate* is a kind of expression that represents a proposition.
- A predicate consists of a *predicate symbol* and one or more *arguments*.
- The layout of a predicate expression depends on which syntax we adopt.

Predicates

- A *predicate* is a kind of expression that represents a proposition.
- A predicate consists of a *predicate symbol* and one or more *arguments*.
- The layout of a predicate expression depends on which syntax we adopt.
- Examples: Pb, $Rabc$, Qx, $Tdxn$

Predicates

- A *predicate* is a kind of expression that represents a proposition.
- A predicate consists of a *predicate symbol* and one or more *arguments*.
- The layout of a predicate expression depends on which syntax we adopt.
- Examples: Pb, $Rabc$, Qx, $Tdxn$
- Examples: $Postman(b)$, $Reminds(a, b, c)$, $Quiet(x)$, $Thinking(d, x, n)$

Propositions

- A *proposition* is the abstract content of a simple declarative sentence.

Propositions

- A *proposition* is the abstract content of a simple declarative sentence.
- "Eight is an integer" expresses the same proposition as "Ok estas entjero."

Propositions

- A *proposition* is the abstract content of a simple declarative sentence.
- "Eight is an integer" expresses the same proposition as "Ok estas entjero."
- "Jupiter is larger than Venus" expresses the same proposition as "Jupiter ist größer als die Venus".

Propositions

- A *proposition* is the abstract content of a simple declarative sentence.
- "Eight is an integer" expresses the same proposition as "Ok estas entjero."
- "Jupiter is larger than Venus" expresses the same proposition as "Jupiter ist größer als die Venus".
- Propositions are the bearers of truth values: they are the kinds of things that can be true or false.

Propositions

- A *proposition* is the abstract content of a simple declarative sentence.
- "Eight is an integer" expresses the same proposition as "Ok estas entjero."
- "Jupiter is larger than Venus" expresses the same proposition as "Jupiter ist größer als die Venus".
- Propositions are the bearers of truth values: they are the kinds of things that can be true or false.
- Propositions are the objects of *propositional attitudes*. They are the kinds of things that can be believed, desired, doubted, expected, or feared.

Predicate logic expressions

- Our predicate logic notation uses letters in three different ways.

Predicate logic expressions

- Our predicate logic notation uses letters in three different ways.
- Lower case letters from the beginning of the Latin alphabet represent specific individual things in the domain we're modeling. Think of them like proper names.

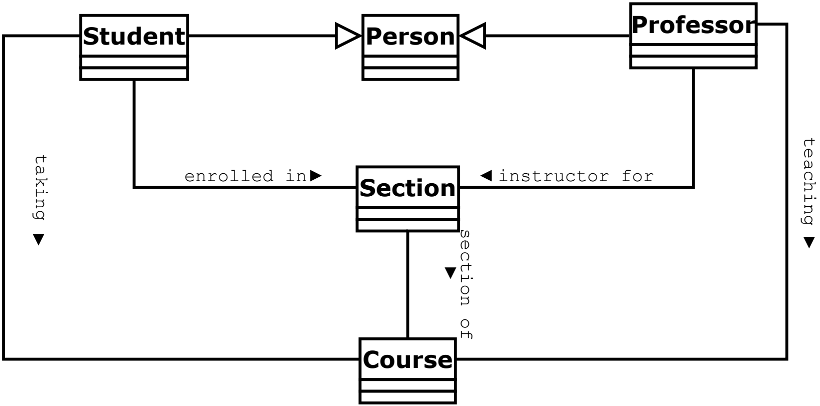
Predicate logic expressions

- Our predicate logic notation uses letters in three different ways.
- Lower case letters from the beginning of the Latin alphabet represent specific individual things in the domain we're modeling. Think of them like proper names.
- Lower case letters from the end of the alphabet (like x and y) are variables that can denote different individuals under different assignments—just like variables in algebraic expressions.

Predicate logic expressions

- Our predicate logic notation uses letters in three different ways.
- Lower case letters from the beginning of the Latin alphabet represent specific individual things in the domain we're modeling. Think of them like proper names.
- Lower case letters from the end of the alphabet (like x and y) are variables that can denote different individuals under different assignments—just like variables in algebraic expressions.
- Capital letters represent properties that an individual might have, classes they might belong to, or relations they might stand in. Think of them like relations in a relational database.

UML Class Diagram



Predicates for courses and sections

- Px means "x is a person."

Predicates for courses and sections

- Px means "x is a person."
- Sx means "x is a student."

Predicates for courses and sections

- Px means "x is a person."
- Sx means "x is a student."
- Rx means "x is a professor."

Predicates for courses and sections

- Px means "x is a person."
- Sx means "x is a student."
- Rx means "x is a professor."
- Cx means "x is a course."

Predicates for courses and sections

- Px means "x is a person."
- Sx means "x is a student."
- Rx means "x is a professor."
- Cx means "x is a course."
- Nx means "x is a section."

Predicates for courses and sections

- *Px* means “x is a person.”
- *Sx* means “x is a student.”
- *Rx* means “x is a professor.”
- *Cx* means “x is a course.”
- *Nx* means “x is a section.”
- *Oxy* means “x is a section of y.”

Predicates for courses and sections

- *Px* means "x is a person."
- *Sx* means "x is a student."
- *Rx* means "x is a professor."
- *Cx* means "x is a course."
- *Nx* means "x is a section."
- *Oxy* means "x is a section of y."
- *Exy* means "x is enrolled in y."

Predicates for courses and sections

- *Px* means "x is a person."
- *Sx* means "x is a student."
- *Rx* means "x is a professor."
- *Cx* means "x is a course."
- *Nx* means "x is a section."
- *Oxy* means "x is a section of y."
- *Exy* means "x is enrolled in y."
- *Ixy* means "x is instructor for y."

Predicates for courses and sections

- Px means "x is a person."
- Sx means "x is a student."
- Rx means "x is a professor."
- Cx means "x is a course."
- Nx means "x is a section."
- Oxy means "x is a section of y."
- Exy means "x is enrolled in y."
- Ixy means "x is instructor for y."
- Txy means "x is taking y."

Predicates for courses and sections

- Px means "x is a person."
- Sx means "x is a student."
- Rx means "x is a professor."
- Cx means "x is a course."
- Nx means "x is a section."
- Oxy means "x is a section of y."
- Exy means "x is enrolled in y."
- Ixy means "x is instructor for y."
- Txy means "x is taking y."
- Gxy means "x is teaching y."

Predicates for courses and sections

- Px means "x is a person."
- Sx means "x is a student."
- Rx means "x is a professor."
- Cx means "x is a course."
- Nx means "x is a section."
- Oxy means "x is a section of y."
- Exy means "x is enrolled in y."
- Ixy means "x is instructor for y."
- Txy means "x is taking y."
- Gxy means "x is teaching y."
- a means "Adamo."

Predicates for courses and sections

- Px means "x is a person."
- Sx means "x is a student."
- Rx means "x is a professor."
- Cx means "x is a course."
- Nx means "x is a section."
- Oxy means "x is a section of y."
- Exy means "x is enrolled in y."
- Ixy means "x is instructor for y."
- Txy means "x is taking y."
- Gxy means "x is teaching y."
- a means "Adamo."
- s means "Sofia."

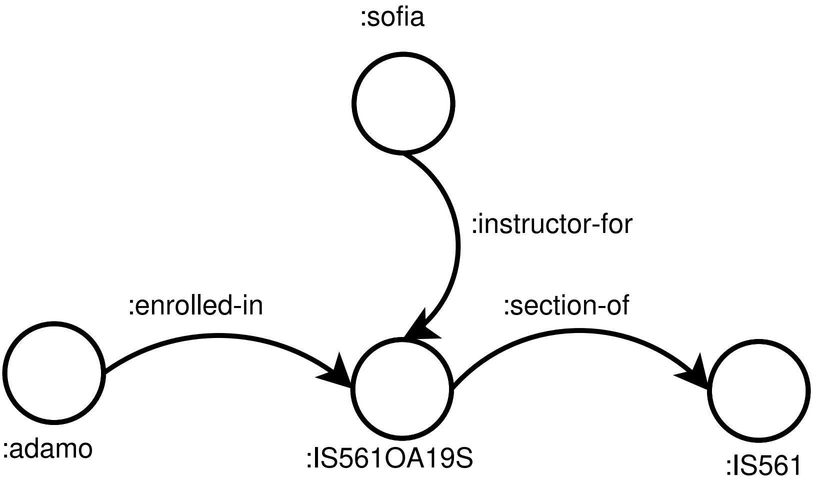
Predicates for courses and sections

- Px means "x is a person."
- Sx means "x is a student."
- Rx means "x is a professor."
- Cx means "x is a course."
- Nx means "x is a section."
- Oxy means "x is a section of y."
- Exy means "x is enrolled in y."
- Ixy means "x is instructor for y."
- Txy means "x is taking y."
- Gxy means "x is teaching y."
- a means "Adamo."
- s means "Sofia."
- m means "IS561"

Predicates for courses and sections

- Px means "x is a person."
- Sx means "x is a student."
- Rx means "x is a professor."
- Cx means "x is a course."
- Nx means "x is a section."
- Oxy means "x is a section of y."
- Exy means "x is enrolled in y."
- Ixy means "x is instructor for y."
- Txy means "x is taking y."
- Gxy means "x is teaching y."
- a means "Adamo."
- s means "Sofia."
- m means "IS561"
- o means "IS561-OA Spring 2019."

Instance diagram



Simple assertions

- Adamo is enrolled in IS561-OA: Eao

Simple assertions

- Adamo is enrolled in IS561-OA: Eao
- Sofia is the instructor for IS561-OA: Iso

Simple assertions

- Adamo is enrolled in IS561-OA: Eao
- Sofia is the instructor for IS561-OA: Iso
- IS561-OA is a section of IS561: Oom

Logical Operators

Symbol	In natural language	Technical name
$\neg$	not	negation
$\wedge$	and	conjunction
$\vee$	or	disjunction
$\rightarrow$	if ... then	implication
$\leftrightarrow$	if and only if	equivalence

Examples of predicate logic expressions

- Predicates take a particular number of arguments, and the order matters. Let Lxy stand for the binary predicate "x loves y," Vx stand for the unary predicate "x is a lover," and the propositional constants r, j, o, d, i stand for Romeo, Juliet, Othello, Desdemona, and Iago, respectively.

Examples of predicate logic expressions

- Predicates take a particular number of arguments, and the order matters. Let Lxy stand for the binary predicate "x loves y," Vx stand for the unary predicate "x is a lover," and the propositional constants r, j, o, d, i stand for Romeo, Juliet, Othello, Desdemona, and Iago, respectively.
- Lrj means "Romeo loves Juliet."

Examples of predicate logic expressions

- Predicates take a particular number of arguments, and the order matters. Let Lxy stand for the binary predicate "x loves y," Vx stand for the unary predicate "x is a lover," and the propositional constants r, j, o, d, i stand for Romeo, Juliet, Othello, Desdemona, and Iago, respectively.
- Lrj means "Romeo loves Juliet."
- $(Lrj \wedge \neg Loi)$ means "Romeo loves Juliet, but Othello doesn't love Iago."

Examples of predicate logic expressions

- Predicates take a particular number of arguments, and the order matters. Let Lxy stand for the binary predicate "x loves y," Vx stand for the unary predicate "x is a lover," and the propositional constants r, j, o, d, i stand for Romeo, Juliet, Othello, Desdemona, and Iago, respectively.
- Lrj means "Romeo loves Juliet."
- $(Lrj \wedge \neg Loi)$ means "Romeo loves Juliet, but Othello doesn't love Iago."
- $\forall x Lxd$ means "everyone loves Desdemona"

Examples of predicate logic expressions

- Predicates take a particular number of arguments, and the order matters. Let Lxy stand for the binary predicate "x loves y," Vx stand for the unary predicate "x is a lover," and the propositional constants r, j, o, d, i stand for Romeo, Juliet, Othello, Desdemona, and Iago, respectively.
- Lrj means "Romeo loves Juliet."
- $(Lrj \wedge \neg Loi)$ means "Romeo loves Juliet, but Othello doesn't love Iago."
- $\forall x Lxd$ means "everyone loves Desdemona"
- $\neg \exists x Lix$ means "Iago loves no one."

Examples of predicate logic expressions

- Predicates take a particular number of arguments, and the order matters. Let Lxy stand for the binary predicate "x loves y," Vx stand for the unary predicate "x is a lover," and the propositional constants r, j, o, d, i stand for Romeo, Juliet, Othello, Desdemona, and Iago, respectively.
- Lrj means "Romeo loves Juliet."
- $(Lrj \wedge \neg Loi)$ means "Romeo loves Juliet, but Othello doesn't love Iago."
- $\forall x Lxd$ means "everyone loves Desdemona"
- $\neg \exists x Lix$ means "Iago loves no one."
- $\forall x \forall y (Vx \rightarrow Lyx)$ means "everyone loves a lover."

Examples of predicate logic expressions

- Predicates take a particular number of arguments, and the order matters. Let Lxy stand for the binary predicate "x loves y," Vx stand for the unary predicate "x is a lover," and the propositional constants r, j, o, d, i stand for Romeo, Juliet, Othello, Desdemona, and Iago, respectively.
- Lrj means "Romeo loves Juliet."
- $(Lrj \wedge \neg Loi)$ means "Romeo loves Juliet, but Othello doesn't love Iago."
- $\forall x Lxd$ means "everyone loves Desdemona"
- $\neg \exists x Lix$ means "Iago loves no one."
- $\forall x \forall y (Vx \rightarrow Lyx)$ means "everyone loves a lover."
- $\forall x (Vx \leftrightarrow \exists z Lxz)$ means "a lover is someone who loves."

Quantifiers have scope

- The scope of a quantifier consists of the logical expression immediately following it. This means that one quantifier can be within the scope of another.

Quantifiers have scope

- The scope of a quantifier consists of the logical expression immediately following it. This means that one quantifier can be within the scope of another.
- Define Sx, Cx, and Txy as meaning x is a student, x is a course, and x takes y, respectively.

Quantifiers have scope

- The scope of a quantifier consists of the logical expression immediately following it. This means that one quantifier can be within the scope of another.
- Define Sx, Cx, and Txy as meaning x is a student, x is a course, and x takes y, respectively.
- The ambiguous English sentence, "Every student was taking a course" expresses two different propositions.

Quantifiers have scope

- The scope of a quantifier consists of the logical expression immediately following it. This means that one quantifier can be within the scope of another.
- Define Sx, Cx, and Txy as meaning x is a student, x is a course, and x takes y, respectively.
- The ambiguous English sentence, "Every student was taking a course" expresses two different propositions.
- We can express the first in logical form as $\exists x(Cx \wedge \forall y(Sy \rightarrow Tyx))$. On this interpretation, there is some particular course (or courses) that every student took.

Quantifiers have scope

- The scope of a quantifier consists of the logical expression immediately following it. This means that one quantifier can be within the scope of another.
- Define Sx, Cx, and Txy as meaning x is a student, x is a course, and x takes y, respectively.
- The ambiguous English sentence, "Every student was taking a course" expresses two different propositions.
- We can express the first in logical form as $\exists x(Cx \wedge \forall y(Sy \rightarrow Tyx))$. On this interpretation, there is some particular course (or courses) that every student took.
- We can express the second as $\forall y(Sy \rightarrow \exists x(Cx \wedge Tyx))$. On this interpretation, every student was taking some course, but no particular course was necessarily taken by every student.

Domain and codomain

- From the UML diagram we see that the domain of “enrolled in” is student and the codomain is section.

Domain and codomain

- From the UML diagram we see that the domain of “enrolled in” is student and the codomain is section.
- So if something x is enrolled in something y, then x must be a student and y must be a section.

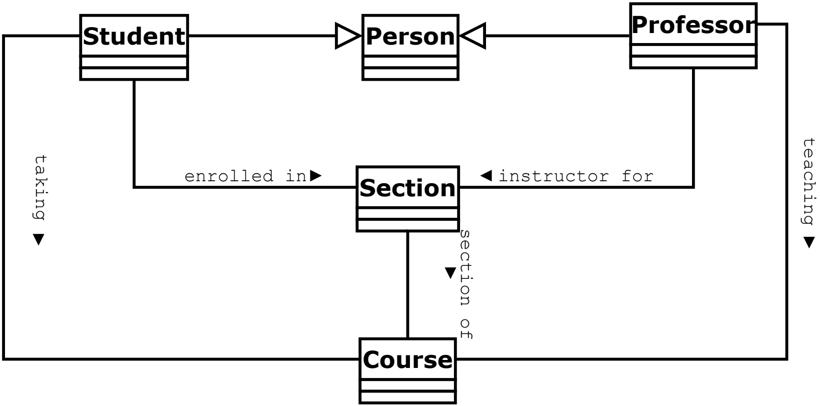
Domain and codomain

- From the UML diagram we see that the domain of "enrolled in" is student and the codomain is section.
- So if something x is enrolled in something y, then x must be a student and y must be a section.
- In other words:

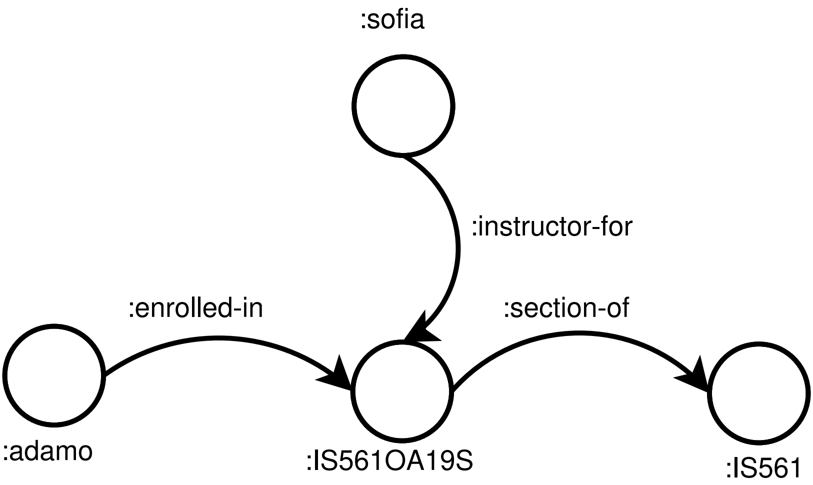
Domain and codomain

- From the UML diagram we see that the domain of "enrolled in" is student and the codomain is section.
- So if something x is enrolled in something y, then x must be a student and y must be a section.
- In other words:
- $\forall x \forall y (Exy \rightarrow (Sx \wedge Ny))$

Back to the UML Class Diagram



Back to the Instance diagram



Other inferences

- The instance diagram only shows students enrolled in sections and professors instructors for sections.

Other inferences

- The instance diagram only shows students enrolled in sections and professors instructors for sections.
- We'd like facts about taking and teaching courses to be deduced from the section enrollment.

Other inferences

- The instance diagram only shows students enrolled in sections and professors instructors for sections.
- We'd like facts about taking and teaching courses to be deduced from the section enrollment.
- $\forall x \forall y \forall z((Exy \wedge Oyz) \rightarrow Txz)$

Other inferences

- The instance diagram only shows students enrolled in sections and professors instructors for sections.
- We'd like facts about taking and teaching courses to be deduced from the section enrollment.
- $\forall x \forall y \forall z ((Exy \land Oyz) \rightarrow Txz)$
- $\forall x \forall y \forall z ((Ixy \land Oyz) \rightarrow Gxz)$