Logic: first-order logic

Limitations of propositional logic

Alice and Bob both know arithmetic.

$\text{AliceKnowsArithmetic} \wedge \text{BobKnowsArithmetic}$

All students know arithmetic.

$\text{AliceIsStudent} \rightarrow \text{AliceKnowsArithmetic}$

$\text{BobIsStudent} \rightarrow \text{BobKnowsArithmetic}$

$\ldots$

Every even integer greater than 2 is the sum of two primes.

???

- If the goal of logic is to be able to express facts in the world in a compact way, let us ask ourselves if propositional logic is enough.
- Some facts can be expressed in propositional logic, but it is very clunky, having to instantiate many different formulas. Others simply can't be expressed at all, because we would need to use an infinite number of formulas.

Limitations of propositional logic

All students know arithmetic.

AliceIsStudent $\rightarrow$ AliceKnowsArithmetic

BobIsStudent $\rightarrow$ BobKnowsArithmetic

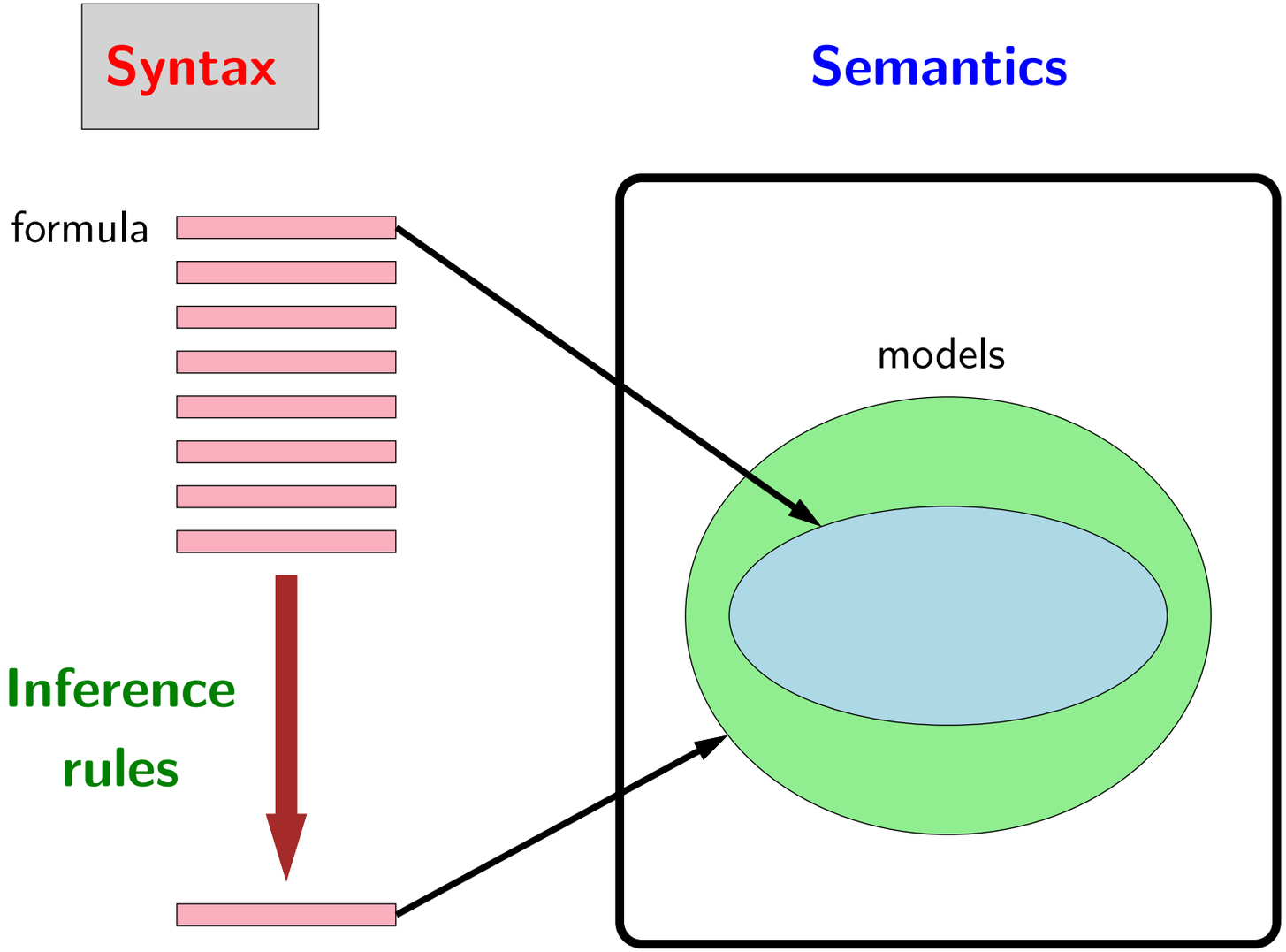
...

Propositional logic is very clunky. What's missing?

- Objects and predicates: propositions (e.g., AliceKnowsArithmetic) have more internal structure (alice, Knows, arithmetic)
- Quantifiers and variables: *all* is a quantifier which applies to each person, don't want to enumerate them all...

- What's missing? The key conceptual observation is that the world is not just a bunch of atomic facts, but that each fact is actually made out of **objects** and **predicates** on those objects.
- Once facts are decomposed in this way, we can use **quantifiers** and **variables** to implicitly define a huge (and possibly infinite) number of facts with one compact formula. Again, where logic excels is the ability to represent complex things via simple means.

First-order logic



- We will now introduce **first-order logic**, which will address the representational limitations of propositional logic.
- Remember to define a logic, we need to talk about its syntax, its semantics (interpretation function), and finally inference rules that we can use to operate on the syntax.

First-order logic: examples

Alice and Bob both know arithmetic.

$$\mathsf{Knows}(\mathsf{alice}, \mathsf{arithmetic}) \wedge \mathsf{Knows}(\mathsf{bob}, \mathsf{arithmetic})$$

All students know arithmetic.

$$\forall x \, \mathsf{Student}(x) \rightarrow \mathsf{Knows}(x, \mathsf{arithmetic})$$

- Before formally defining things, let's look at two examples. First-order logic is basically propositional logic with a few more symbols.

Syntax of first-order logic

Terms (refer to objects):

- Constant symbol (e.g., arithmetic)
- Variable (e.g., x)
- Function of terms (e.g., $\text{Sum}(3, x)$)

Formulas (refer to truth values):

- Atomic formulas (atoms): predicate applied to terms (e.g., $\text{Knows}(x, \text{arithmetic})$)
- Connectives applied to formulas (e.g., $\text{Student}(x) \rightarrow \text{Knows}(x, \text{arithmetic})$)
- Quantifiers applied to formulas (e.g., $\forall x\, \text{Student}(x) \rightarrow \text{Knows}(x, \text{arithmetic})$)

- In propositional logic, everything was a formula (or a connective). In first-order logic, there are two types of beasts: terms and formulas. There are three types of terms: constant symbols (which refer to specific objects), variables (which refer to some unspecified object to be determined by quantifiers), and functions (which is a function applied to a set of arguments which are themselves terms).
- Given the terms, we can form atomic formulas, which are the analogue of propositional symbols, but with internal structure (e.g., terms).
- From this point, we can apply the same connectives on these atomic formulas, as we applied to propositional symbols in propositional logic. At this level, first-order logic looks very much like propositional logic.
- Finally, to make use of the fact that atomic formulas have internal structure, we have **quantifiers**, which are really the whole point of first-order logic!

Quantifiers

Universal quantification ($\forall$):

Think conjunction: $\forall x\, P(x)$ is like $P(A) \wedge P(B) \wedge \cdots$

Existential quantification ($\exists$):

Think disjunction: $\exists x\, P(x)$ is like $P(A) \vee P(B) \vee \cdots$

Some properties:

- $\neg \forall x\, P(x)$ equivalent to $\exists x\, \neg P(x)$
- $\forall x\, \exists y\, \mathsf{Knows}(x, y)$ different from $\exists y\, \forall x\, \mathsf{Knows}(x, y)$

- There are two types of quantifiers: universal and existential. These are basically glorified ways of doing conjunction and disjunction, respectively.
- For crude intuition, we can think of conjunction and disjunction as very nice syntactic sugar, which can be rolled out into something that looks more like propositional logic. But quantifiers aren't just sugar, and it is important that they be compact, for sometimes the variable being quantified over can take on an infinite number of objects.
- That being said, the conjunction and disjunction intuition suffices for day-to-day guidance. For example, it should be intuitive that pushing the negation inside a universal quantifier (conjunction) turns it into a existential (disjunction), which was the case for propositional logic (by de Morgan's laws). Also, one cannot interchange universal and existential quantifiers any more than one can swap conjunction and disjunction in propositional logic.

Natural language quantifiers

Universal quantification ($\forall$):

Every student knows arithmetic.

$\forall x\, \mathsf{Student}(x) \rightarrow \mathsf{Knows}(x, \mathsf{arithmetic})$

Existential quantification ($\exists$):

Some student knows arithmetic.

$\exists x\, \mathsf{Student}(x) \wedge \mathsf{Knows}(x, \mathsf{arithmetic})$

Note the different connectives!

- Universal and existential quantifiers naturally correspond to the words *every* and *some*, respectively. But when converting English to formal logic, one must exercise caution.
- *Every* can be thought of as taking two arguments P and Q (e.g., *student* and *knows arithmetic*). The connective between P and Q is an implication (not conjunction, which is a common mistake). This makes sense because when we talk about every P, we are only restricting our attention to objects x for which $P(x)$ is true. Implication does exactly that.
- On the other hand, the connective for existential quantification is conjunction, because we're asking for an object x such that $P(x)$ and $Q(x)$ both hold.

Some examples of first-order logic

There is some course that every student has taken.

$$\exists y\, \mathsf{Course}(y) \wedge [\forall x\, \mathsf{Student}(x) \rightarrow \mathsf{Takes}(x, y)]$$

Every even integer greater than 2 is the sum of two primes.

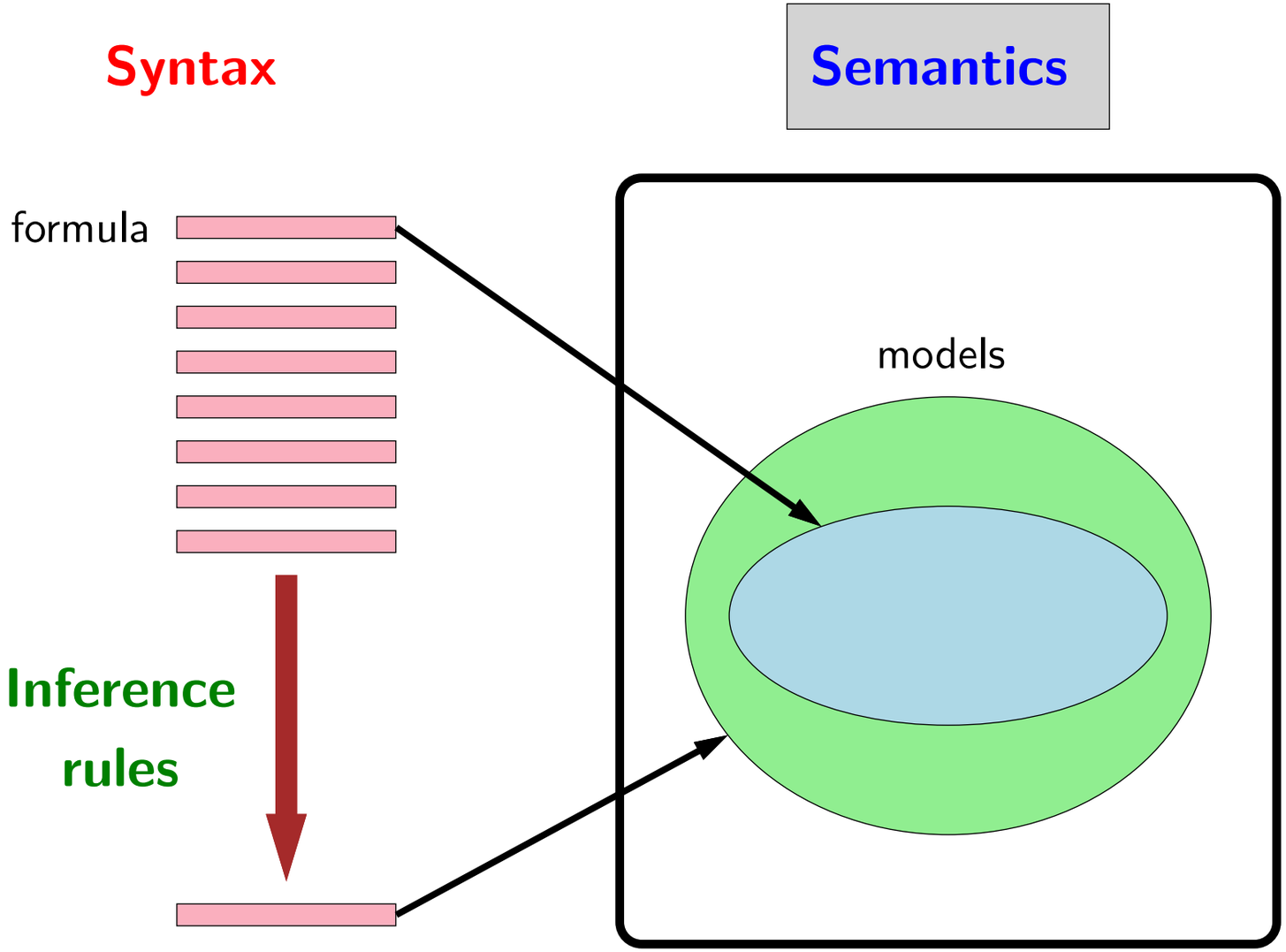
$$\forall x\, \mathsf{EvenInt}(x) \wedge \mathsf{Greater}(x, 2) \rightarrow \exists y\, \exists z\, \mathsf{Equals}(x, \mathsf{Sum}(y, z)) \wedge \mathsf{Prime}(y) \wedge \mathsf{Prime}(z)$$

If a student takes a course and the course covers a concept, then the student knows that concept.

$$\forall x\, \forall y\, \forall z\, (\mathsf{Student}(x) \wedge \mathsf{Takes}(x, y) \wedge \mathsf{Course}(y) \wedge \mathsf{Covers}(y, z)) \rightarrow \mathsf{Knows}(x, z)$$

- Let's do some more examples of converting natural language to first-order logic. Remember the connectives associated with existential and universal quantification!
- Note that some English words such as *a* can trigger both universal or existential quantification, depending on context. In *A student took CS221*, we have existential quantification, but in *if a student takes CS221, ...*, we have universal quantification.
- Formal logic clears up the ambiguities associated with natural language.

First-order logic



- So far, we've only presented the syntax of first-order logic, although we've actually given quite a bit of intuition about what the formulas mean. After all, it's hard to talk about the syntax without at least a hint of semantics for motivation.
- Now let's talk about the formal semantics of first-order logic.

Models in first-order logic

Recall a model represents a possible situation in the world.

Propositional logic: Model w maps propositional symbols to truth values.

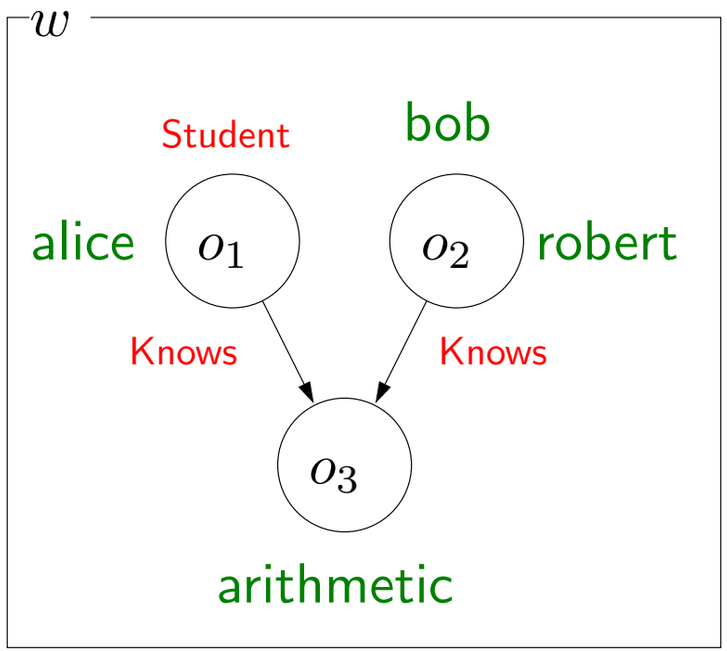
$$w = \{\text{AliceKnowsArithmetic} : 1, \text{BobKnowsArithmetic} : 0\}$$

First-order logic: ?

- Recall that a model in propositional logic was just an assignment of truth values to propositional symbols.
- A natural candidate for a model in first-order logic would then be an assignment of truth values to grounded atomic formula (those formulas whose terms are constants as opposed to variables). This is almost right, but doesn't talk about the relationship between constant symbols.

Graph representation of a model

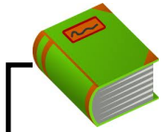
If only have unary and binary predicates, a model w can be represented as a directed graph:



- Nodes are objects, labeled with constant symbols
- Directed edges are binary predicates, labeled with predicate symbols; unary predicates are additional node labels

- A better way to think about a first-order model is that there are a number of objects in the world ($o_1, o_2, \ldots$); think of these as nodes in a graph. Then we have predicates between these objects. Predicates that take two arguments can be visualized as labeled edges between objects. Predicates that take one argument can be visualized as node labels (but these are not so important).
- So far, the objects are unnamed. We can access individual objects directly using constant symbols, which are labels on the nodes.

Models in first-order logic



Definition: model in first-order logic

A model w in first-order logic maps:

- constant symbols to objects

$w(\text{alice}) = o_1, w(\text{bob}) = o_2, w(\text{arithmetic}) = o_3$

- predicate symbols to tuples of objects

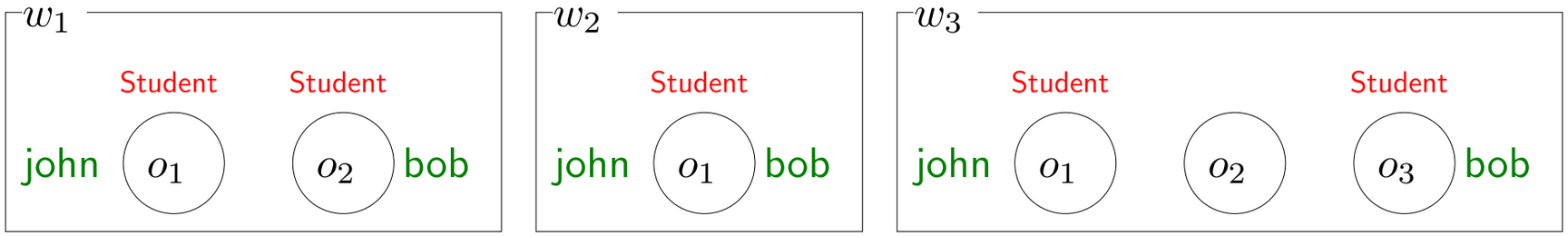
$w(\text{Knows}) = \{(o_1, o_3), (o_2, o_3), \ldots\}$

- Formally, a first-order model w maps constant symbols to objects and predicate symbols to tuples of objects (2 for binary predicates).

A restriction on models

John and Bob are students.

$$\text{Student}(\text{john}) \wedge \text{Student}(\text{bob})$$



- Unique names assumption: Each object has **at most one** constant symbol. This rules out w_2.
- Domain closure: Each object has **at least one** constant symbol. This rules out w_3.

Point:

constant symbol ⟷ object

- Note that by default, two constant symbols can refer to the same object, and there can be objects which no constant symbols refer to. This can make life somewhat confusing. Fortunately, there are two assumptions that people sometimes make to simplify things.
- The unique names assumption says that there's at most one way to refer to an object via a constant symbol. Domain closure says there's at least one. Together, they imply that there is a one-to-one relationship between constant symbols in syntax-land and objects in semantics-land.

Propositionalization

If one-to-one mapping between constant symbols and objects (**unique names** and **domain closure**),

first-order logic is syntactic sugar for propositional logic:

Knowledge base in first-order logic

$$\text{Student}(\text{alice}) \wedge \text{Student}(\text{bob})$$
$$\forall x\, \text{Student}(x) \rightarrow \text{Person}(x)$$
$$\exists x\, \text{Student}(x) \wedge \text{Creative}(x)$$

Knowledge base in propositional logic

$$\text{Studentalice} \wedge \text{Studentbob}$$
$$(\text{Studentalice} \rightarrow \text{Personalice}) \wedge (\text{Studentbob} \rightarrow \text{Personbob})$$
$$(\text{Studentalice} \wedge \text{Creativealice}) \vee (\text{Studentbob} \wedge \text{Creativebob})$$

Point: use any inference algorithm for propositional logic!

- If a one-to-one mapping really exists, then we can **propositionalize** all our formulas, which basically unrolls all the quantifiers into explicit conjunctions and disjunctions.
- The upshot of this conversion, is that we're back to propositional logic, and we know how to do inference in propositional logic (either using model checking or by applying inference rules). Of course, propositionalization could be quite expensive and not the most efficient thing to do.