First-Order Logic

Part One

Recap from Last Time

Recap So Far

- A ***propositional variable*** is a variable that is either true or false.
- The ***propositional connectives*** are as follows:
  - Negation: $\neg p$
  - Conjunction: $p \land q$
  - Disjunction: $p \lor q$
  - Implication: $p \to q$
  - Biconditional: $p \leftrightarrow q$
  - True: $\top$
  - False: $\bot$

Take out a sheet of paper!

What's the truth table for the → connective?

What's the negation of $p \to q$?

New Stuff!

First-Order Logic

What is First-Order Logic?

- ***First-order logic*** is a logical system for reasoning about properties of objects.
- Augments the logical connectives from propositional logic with
  - ***predicates*** that describe properties of objects,
  - ***functions*** that map objects to one another, and
  - ***quantifiers*** that allow us to reason about multiple objects.

Some Examples

$$Likes(You, Eggs) \land Likes(You, Tomato) \rightarrow Likes(You, Shakshuka)$$

$$Learns(You, History) \lor ForeverRepeats(You, History)$$

$$In(MyHeart, Havana) \land TookBackTo(Him, Me, EastAtlanta)$$

$$Likes(You, Eggs) \land Likes(You, Tomato) \rightarrow Likes(You, Shakshuka)$$

$$Learns(You, History) \lor ForeverRepeats(You, History)$$

$$In(MyHeart, Havana) \land TookBackTo(Him, Me, EastAtlanta)$$

These blue terms are called *constant symbols*. Unlike propositional variables, they refer to *objects*, not propositions.

$$Likes(You, Eggs) \land Likes(You, Tomato) \rightarrow Likes(You, Shakshuka)$$

$$Learns(You, History) \lor ForeverRepeats(You, History)$$

$$In(MyHeart, Havana) \land TookBackTo(Him, Me, EastAtlanta)$$

The red things that look like function calls are called *predicates*. Predicates take objects as arguments and evaluate to true or false.

$$Likes(You, Eggs) \wedge Likes(You, Tomato) \rightarrow Likes(You, Shakshuka)$$

$$Learns(You, History) \vee ForeverRepeats(You, History)$$

$$In(MyHeart, Havana) \wedge TookBackTo(Him, Me, EastAtlanta)$$

What remains are traditional propositional connectives. Because each predicate evaluates to true or false, we can connect the truth values of predicates using normal propositional connectives.

Reasoning about Objects

- To reason about objects, first-order logic uses ***predicates***.
- Examples:

$$Cute(Quokka)$$

$$ArgueIncessantly(Democrats, Republicans)$$

- Applying a predicate to arguments produces a proposition, which is either true or false.
- Typically, when you're working in FOL, you'll have a list of predicates, what they stand for, and how many arguments they take. It'll be given separately than the formulas you write.

First-Order Sentences

- Sentences in first-order logic can be constructed from predicates applied to objects:

$$Cute(a) \to Dikdik(a) \lor Kitty(a) \lor Puppy(a)$$

$$Succeeds(You) \leftrightarrow Practices(You)$$

$$x < 8 \to x < 137$$

The less-than sign is just another predicate. Binary predicates are sometimes written in *infix notation* this way.

Numbers are not "built in" to first-order logic. They're constant symbols just like "You" and "a" above.

Equality

- First-order logic is equipped with a special predicate **=** that says whether two objects are equal to one another.
- Equality is a part of first-order logic, just as → and ¬ are.
- Examples:

$$TomMarvoloRiddle = LordVoldemort$$

$$MorningStar = EveningStar$$

- Equality can only be applied to ***objects***; to state that two ***propositions*** are equal, use ↔.

Let's see some more examples.

$$FavoriteMovieOf(You) \neq FavoriteMovieOf(Date) \land$$
$$StarOf(FavoriteMovieOf(You)) = StarOf(FavoriteMovieOf(Date))$$

$$FavoriteMovieOf(You) \neq FavoriteMovieOf(Date) \land StarOf(FavoriteMovieOf(You)) = StarOf(FavoriteMovieOf(Date))$$

These purple terms are *functions*. Functions take objects as input and produce objects as output.

Functions

- First-order logic allows ***functions*** that return objects associated with other objects.
- Examples:

$$ColorOf(Money)$$

$$MedianOf(x, y, z)$$

$$x + y$$

- As with predicates, functions can take in any number of arguments, but always return a single value.
- Functions evaluate to ***objects***, not ***propositions***.

Objects and Predicates

- When working in first-order logic, be careful to keep objects (actual things) and propositions (true or false) separate.
- You cannot apply connectives to objects:

$$\mathit{Venus} \to \mathit{TheSun}$$

- You cannot apply functions to propositions:

$$\mathit{StarOf}(\mathit{IsRed}(\mathit{Sun}) \land \mathit{IsGreen}(\mathit{Mars}))$$

- Ever get confused? *Just ask!*

The Type-Checking Table

	… operate on …	… and produce
Connectives (↔, ∧, etc.) …	propositions	a proposition
Predicates (=, etc.) …	objects	a proposition
Functions …	objects	an object

One last (and major) change

Some muggle is intelligent.

$$\exists m.\ (Muggle(m) \land Intelligent(m))$$

∃ is the ***existential quantifier*** and says "for some choice of m, the following is true."

The Existential Quantifier

- A statement of the form

  $$\exists x.\ \textit{some-formula}$$

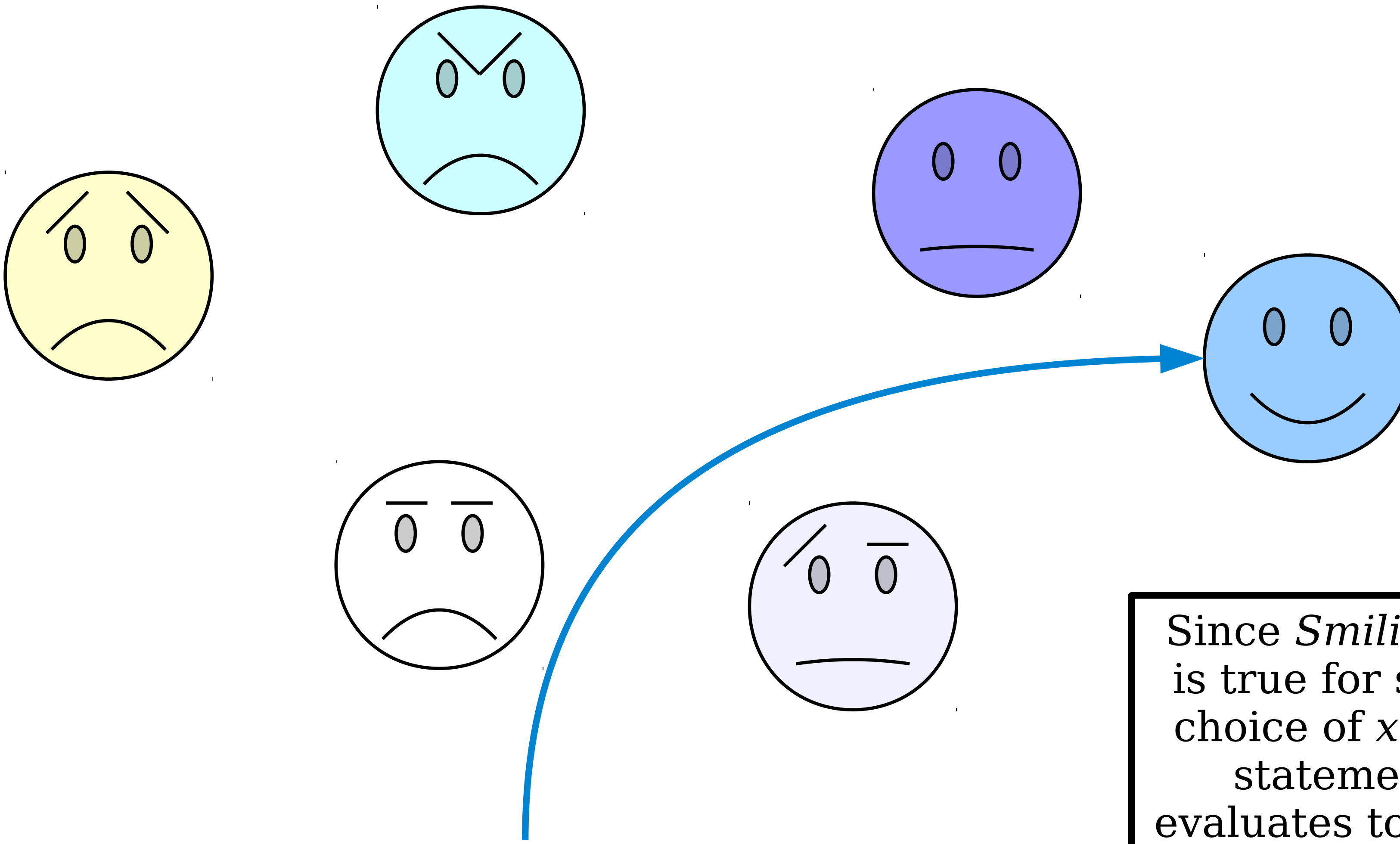
  is true if there exists a choice of x where ***some-formula*** is true when that x is plugged into it.
- Examples:

  $\exists x.\ (Even(x) \land Prime(x))$

  $\exists x.\ (TallerThan(x, me) \land LighterThan(x, me))$

  $(\exists w.\ Will(w)) \to (\exists x.\ Way(x))$

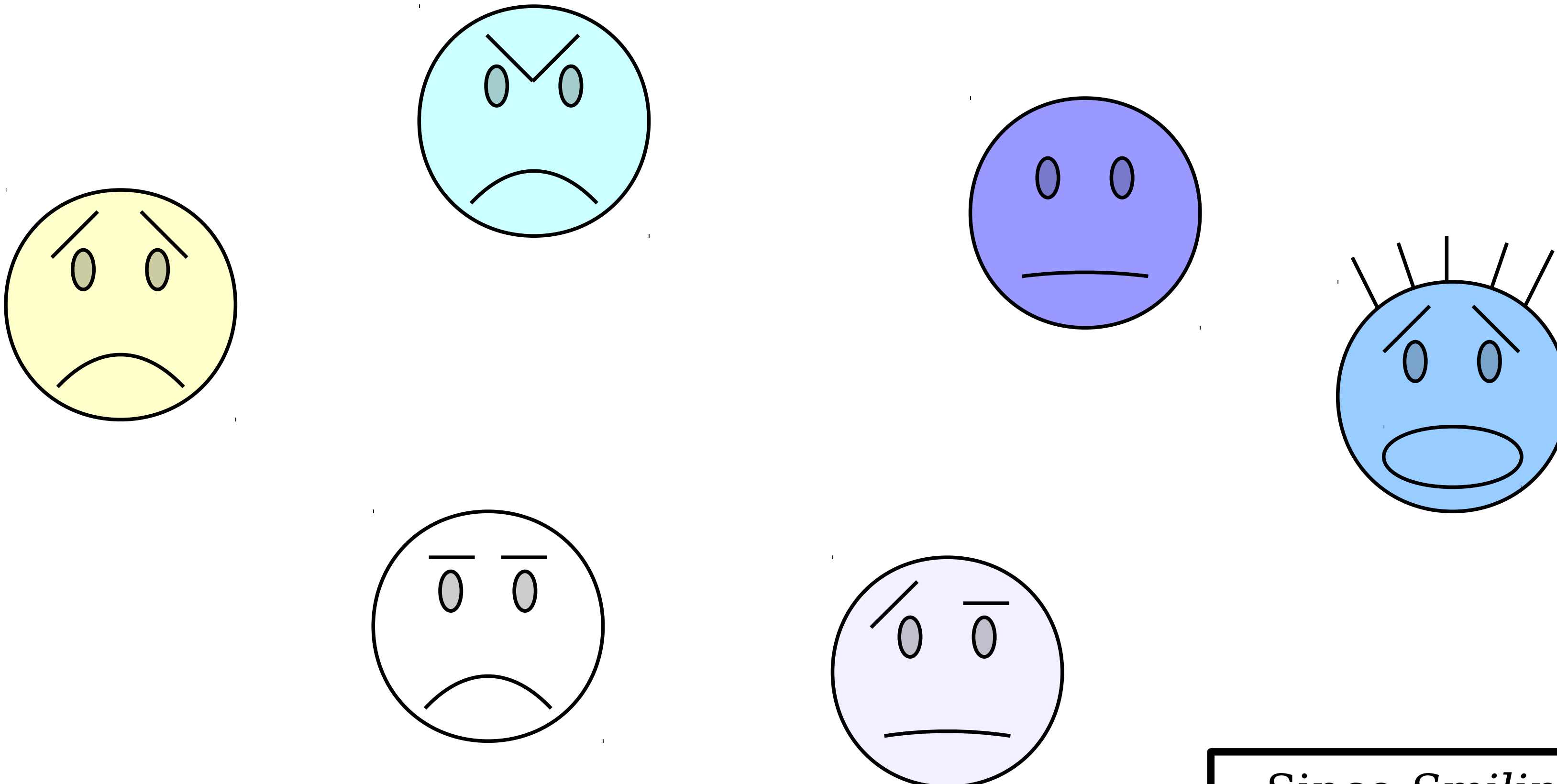
The Existential Quantifier



$\exists x.\ Smiling(x)$

Since *Smiling*(*x*) is true for some choice of *x*, this statement evaluates to true.

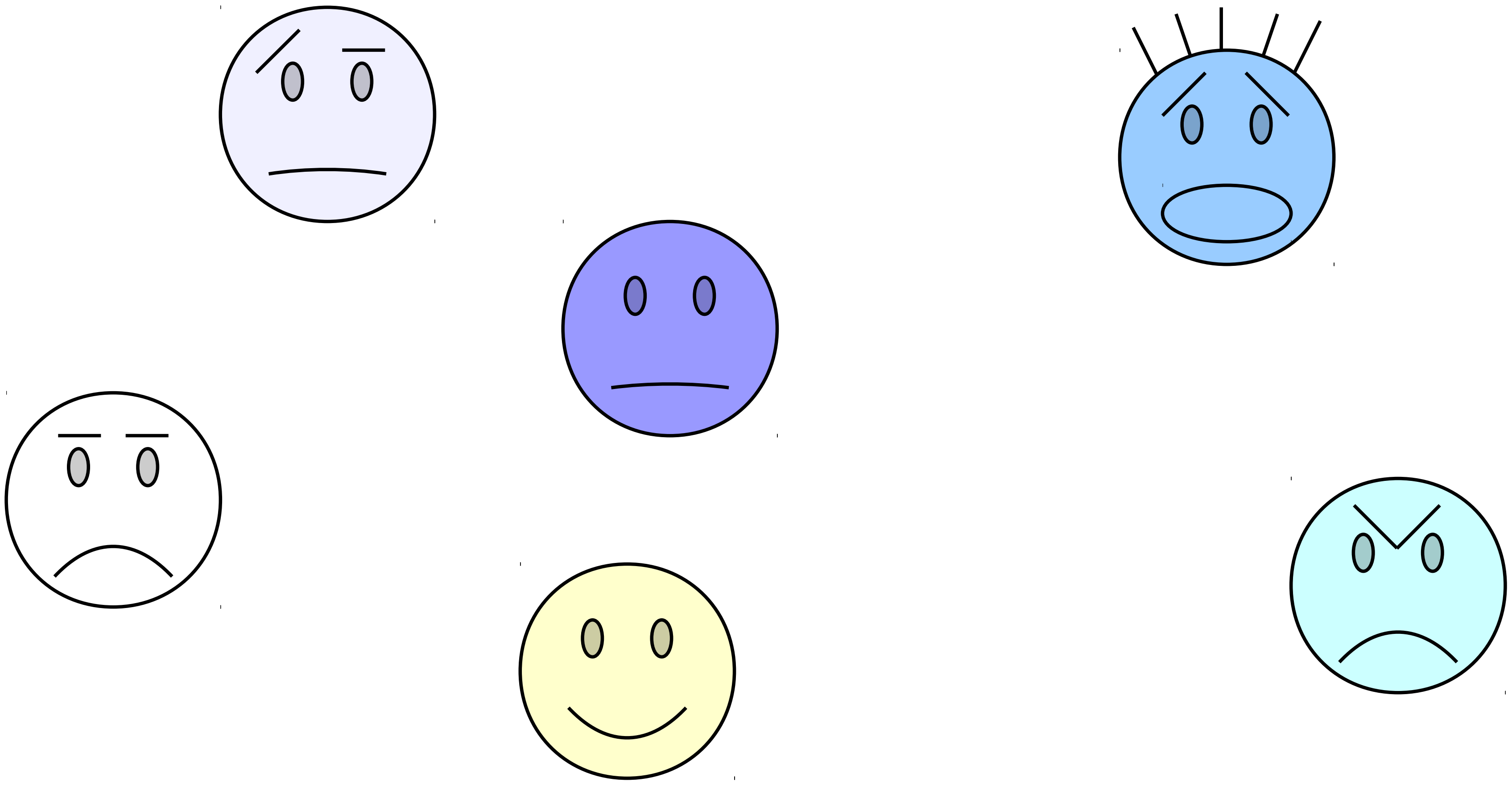
The Existential Quantifier



$\exists x.\ \mathit{Smiling}(x)$

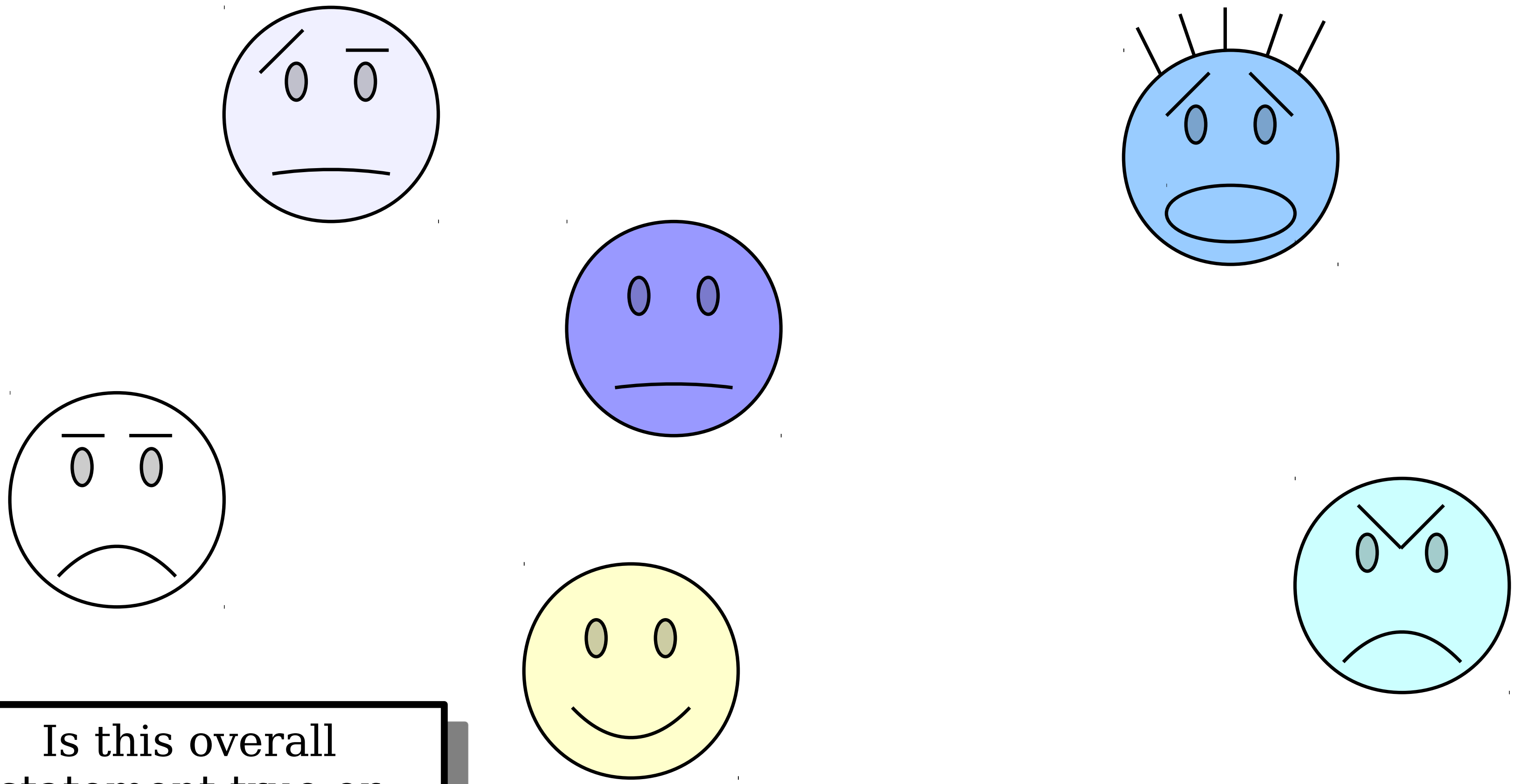
Since $\mathit{Smiling}(x)$ is not true for any choice of x, this statement evaluates to false.

The Existential Quantifier



$$(\exists x.\ Smiling(x)) \rightarrow (\exists y.\ WearingHat(y))$$

The Existential Quantifier



Is this overall statement true or false?

~~$(\exists x.\ Smiling(x)) \rightarrow (\exists y.\ WearingHat(y))$~~

Fun with Edge Cases

Existentially-quantified statements are false in an empty world, since nothing exists, period!

~~$\exists x.\ Smiling(x)$~~

Some Technical Details

Variables and Quantifiers

- Each quantifier has two parts:
  - the variable that is introduced, and
  - the statement that's being quantified.
- The variable introduced is scoped just to the statement being quantified.

$$(\exists x.\ Loves(You,\ x)) \land (\exists y.\ Loves(y,\ You))$$

The variable *x* just lives here.

The variable *y* just lives here.

Variables and Quantifiers

- Each quantifier has two parts:
  - the variable that is introduced, and
  - the statement that's being quantified.
- The variable introduced is scoped just to the statement being quantified.

$$(\exists x.\ \mathit{Loves}(\mathit{You}, x)) \land (\exists x.\ \mathit{Loves}(x, \mathit{You}))$$

The variable x just lives here.

A different variable, also named x, just lives here.

Operator Precedence (Again)

- When writing out a formula in first-order logic, quantifiers have precedence just below $\neg$.
- The statement

$$\exists x.\ P(x) \land R(x) \land Q(x)$$

is parsed like this:

$$(\exists x.\ P(x)) \ \land\ (R(x) \land Q(x))$$

- This is syntactically invalid because the variable x is out of scope in the back half of the formula.
- To ensure that x is properly quantified, explicitly put parentheses around the region you want to quantify:

$$\exists x.\ (P(x) \land R(x) \land Q(x))$$

"For any natural number n,
n is even if and only if n^2 is even"

$$\forall n.\ (n \in \mathbb{N} \to (Even(n) \leftrightarrow Even(n^2)))$$

∀ is the ***universal quantifier*** and says "for any choice of n, the following is true."

The Universal Quantifier

- A statement of the form

  $$\forall x.\ \textit{some-formula}$$

  is true if, for every choice of x, the statement ***some-formula*** is true when x is plugged into it.

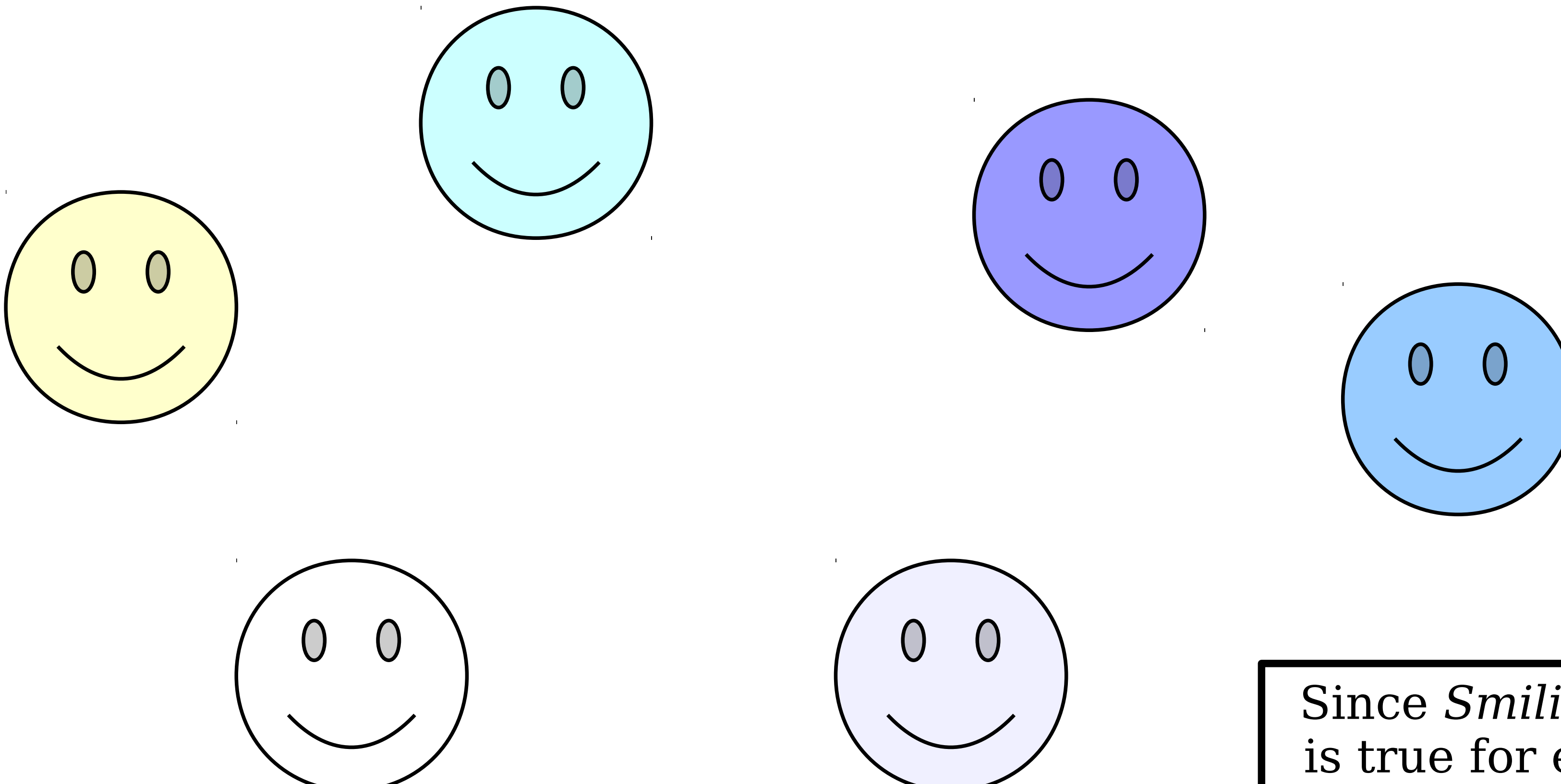
- Examples:

  $\forall p.\ (Puppy(p) \to Cute(p))$

  $\forall a.\ (EatsPlants(a) \lor EatsAnimals(a))$

  $Tallest(SultanKösen) \to$
  $\quad \forall x.\ (SultanKösen \neq x \to ShorterThan(x, SultanKösen))$

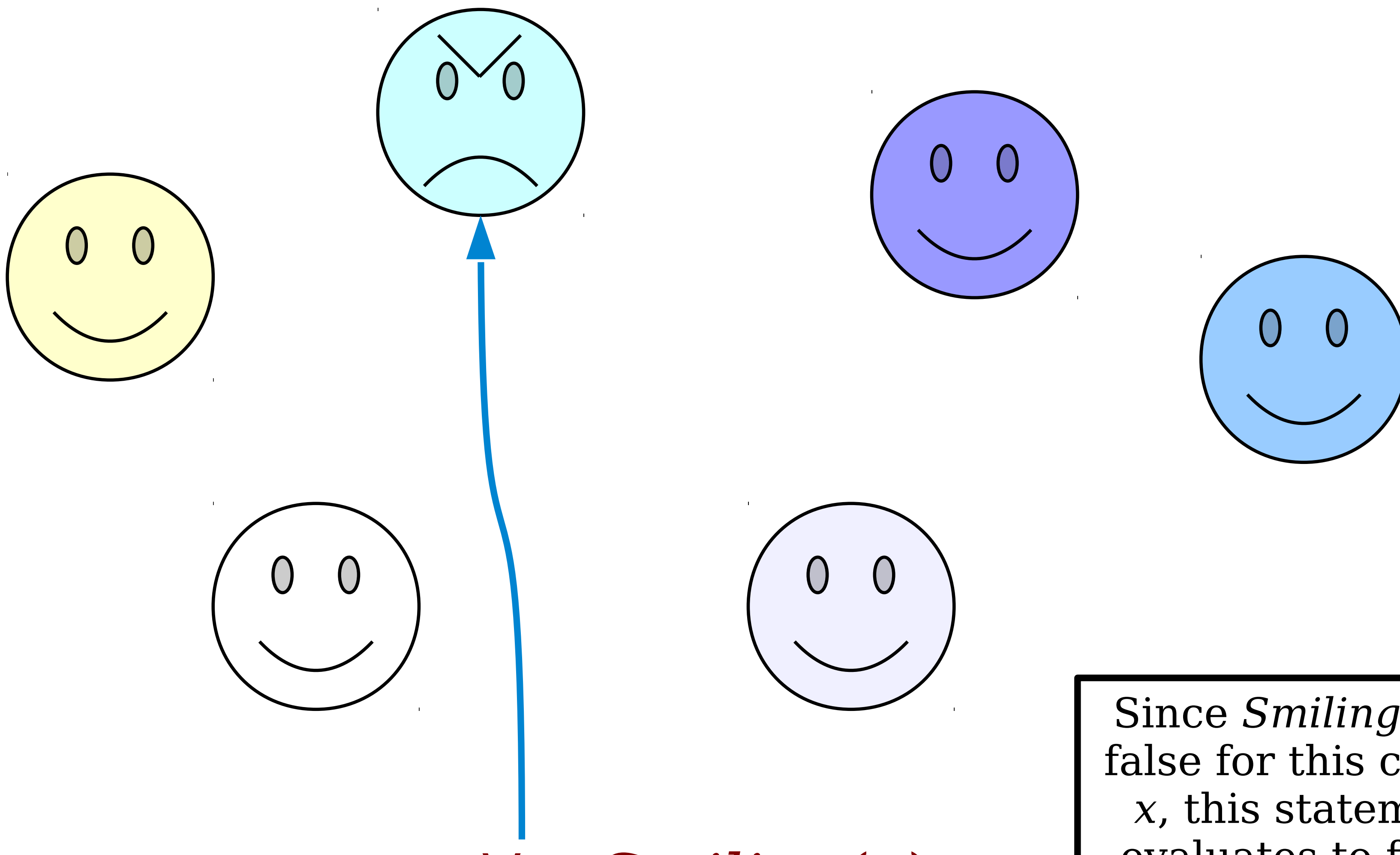
The Universal Quantifier



Since *Smiling*(x) is true for every choice of x, this statement evaluates to true.

$$\forall x.\ \mathit{Smiling}(x)$$

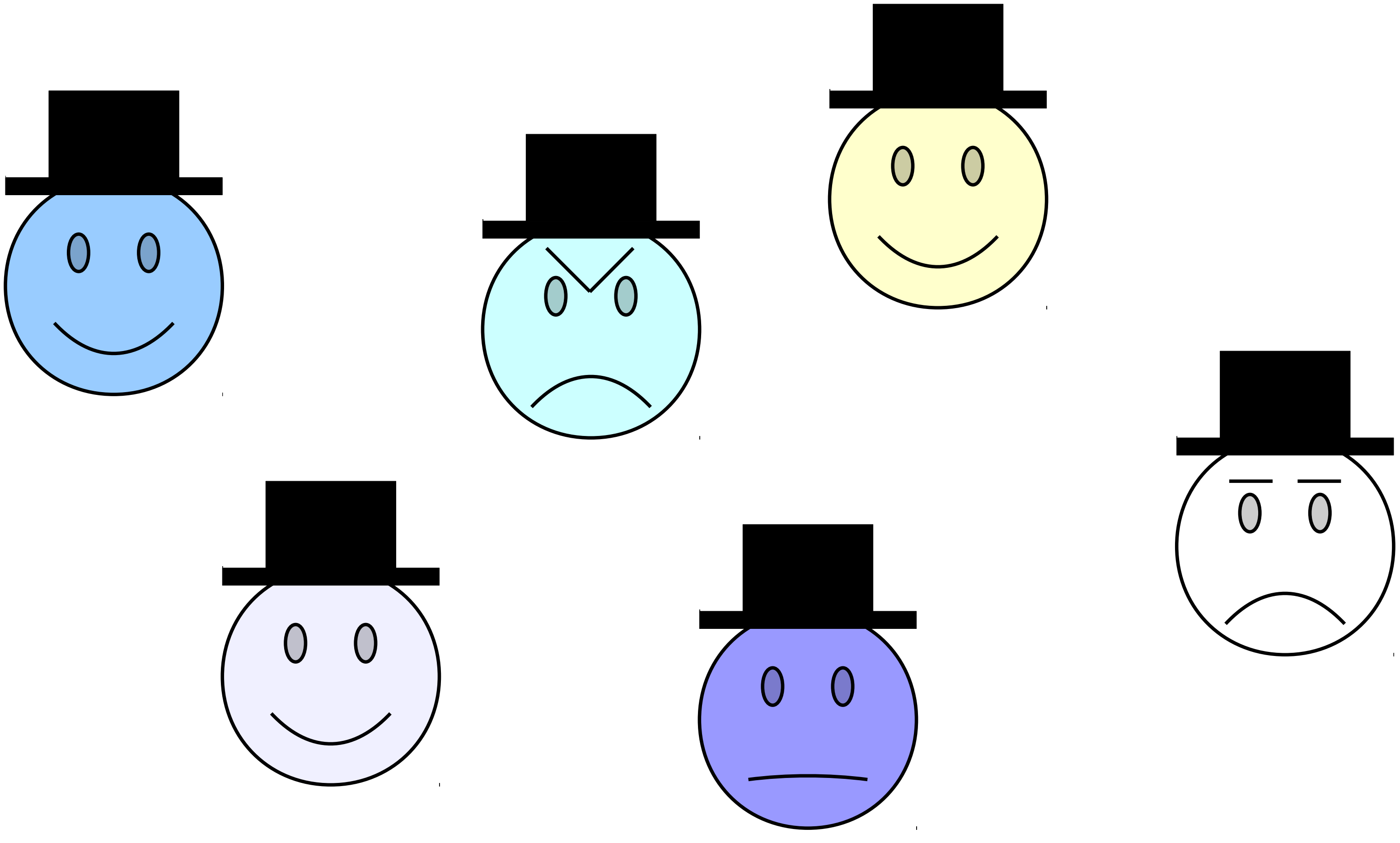
The Universal Quantifier



~~$\forall x.\ Smiling(x)$~~

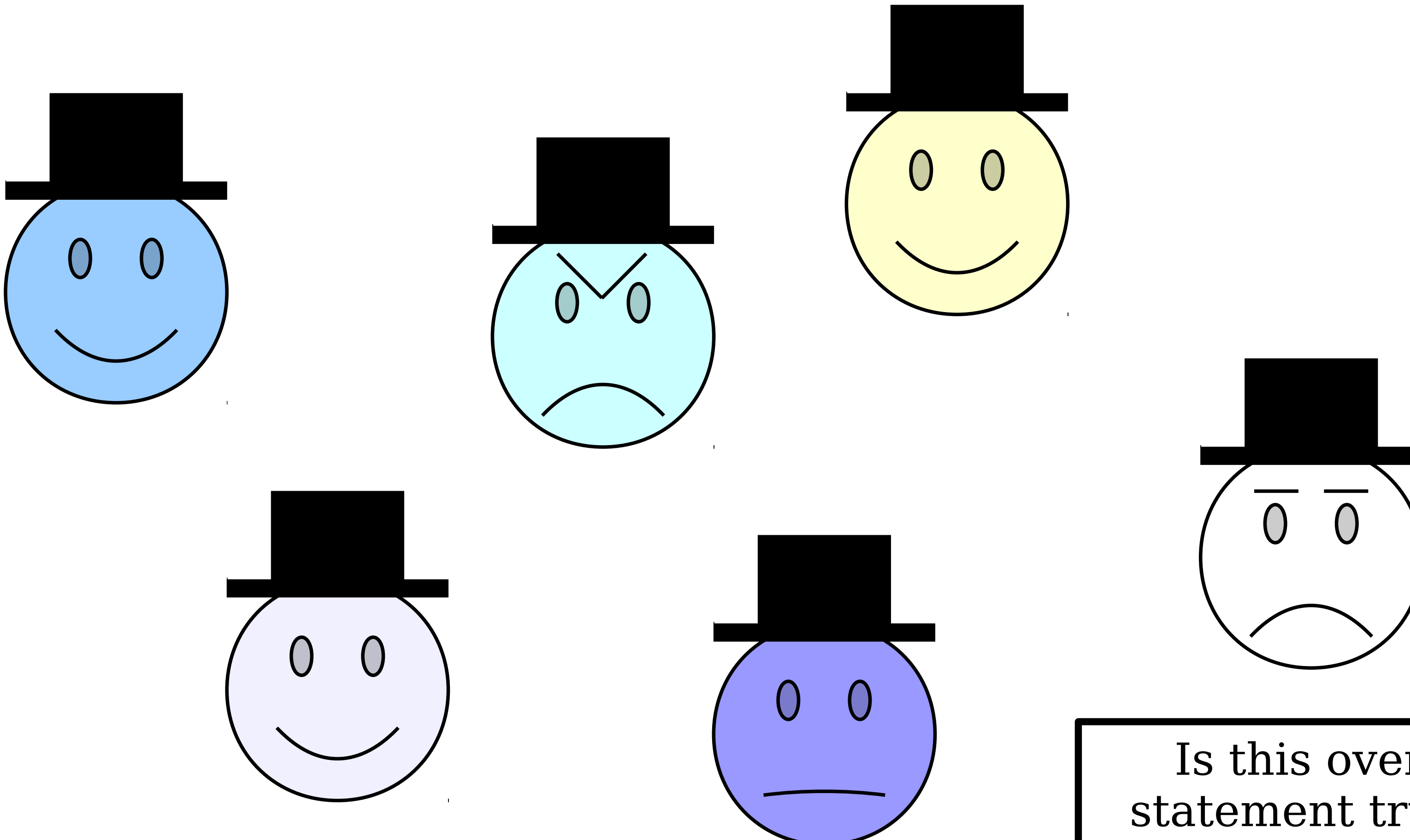
Since $Smiling(x)$ is false for this choice x, this statement evaluates to false.

The Universal Quantifier



$$(\forall x.\ Smiling(x)) \rightarrow (\forall y.\ WearingHat(y))$$

The Universal Quantifier



Is this overall statement true or false in this scenario?

$$(\forall x.\ Smiling(x)) \rightarrow (\forall y.\ WearingHat(y))$$

Fun with Edge Cases

Universally-quantified statements are said to be ***vacuously true*** in empty worlds.

$$\forall x.\ Smiling(x)$$

Time-Out for Announcements!

Stanford Daily Tech Team

- The Stanford Daily is looking for students for their tech team.
- In their own words: “At The Daily, you’ll have the opportunity to take on a wide gamut of projects — building our website and mobile app, creating special custom formats and features for use in our articles, helping us leverage analytics, or managing our data and server architecture. You’ll also have plenty of opportunities to start your own projects.”
- Apply online at https://bit.ly/3CE0ym0 by 5PM on Friday, October 1st.

Your Questions

“What areas of math should I focus on for the CS AI track and what is the minimum depth I should go in math?”

If you're looking at AI, I'd recommend focusing on linear algebra and probability theory.

I can't overstate how important linear algebra is across basically all branches of CS. Math 51 is a great start. Math 104 or Math 113 are great follow-up classes.

For probability, CS109 is a great launching point and is probably enough for much of what you'll do. If you want to learn more, you can look at classes in Math and Stats to back that up.

More generally, *here's a list of the math electives* for CS and some advice about how to pick them.

“What's something that you wish you did/took advantage of as a Stanford undergrad?”

Stanford is an amazing institution with world-class departments in basically every field. I'm really happy with the classes I took, though in retrospect I should have branched out a bit more and taken classes across more departments. It's harder to learn creative writing, art history, political philosophy, etc. once you graduate, though it's definitely still possible.

I also can't understate just how impressive a group of people you are and how lucky you are to get to live, study, and work with each other. Make lasting friendships with one another and go out of your way to meet each other.

"Many tech companies want to hear about projects students have worked on, but after taking 106A and 106B, I feel like I don't have any to show. Do you have advice for finding time to start projects or classes that focus on projects?"

If you're early on in CS, it's completely normal to not have a lot of project experience – after all, you're just getting started! Feel free to talk about what you've worked on for your classes. That's perfectly fine! Just make sure to delimit what part you did and what part was in the starter files. As you take more CS classes, you'll naturally start building up this kind of experience. Project-based classes in graphics, HCI, systems, and AI are great for this.

Some companies specifically ask about side projects – things you've done in your spare time. IMHO, that's not a great question to ask. I know many great engineers who basically don't code outside of work. But if you do want to do a project, make it something you actually are interested in. Otherwise it's really easy to burn out.

Back to CS103!

Translating into First-Order Logic

Translating Into Logic

- First-order logic is an excellent tool for manipulating definitions and theorems to learn more about them.
- Need to take a negation? Translate your statement into FOL, negate it, then translate it back.
- Want to prove something by contrapositive? Translate your implication into FOL, take the contrapositive, then translate it back.

Translating Into Logic

- When translating from English into first-order logic, we recommend that you

  think of first-order logic as a mathematical programming language.

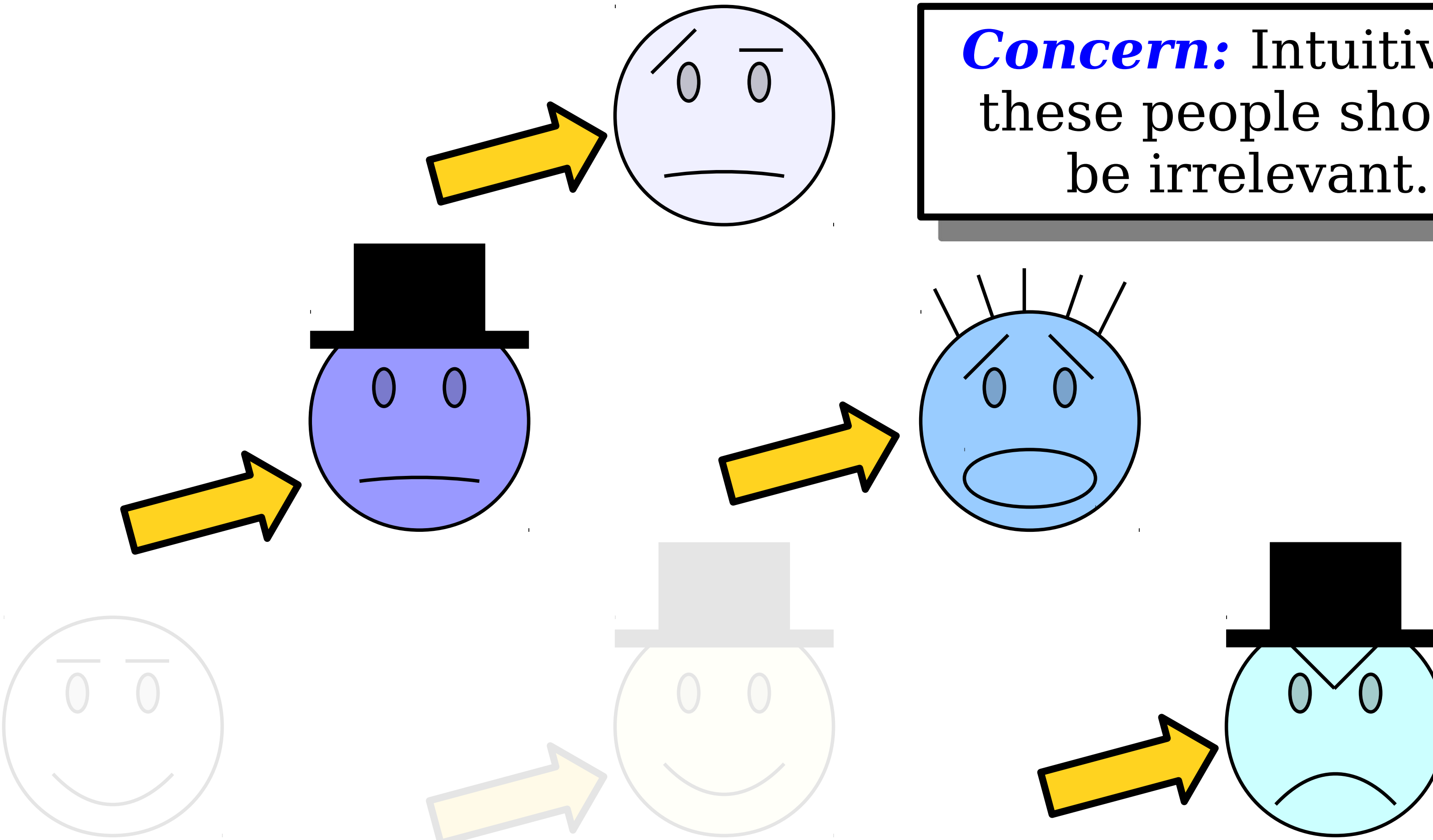
- Your goal is to learn how to combine basic concepts (quantifiers, connectives, etc.) together in ways that say what you mean.

Using the predicates

- *Smiling*(x), which states that x is smiling, and
- *WearingHat*(x), which states that x is wearing a hat,

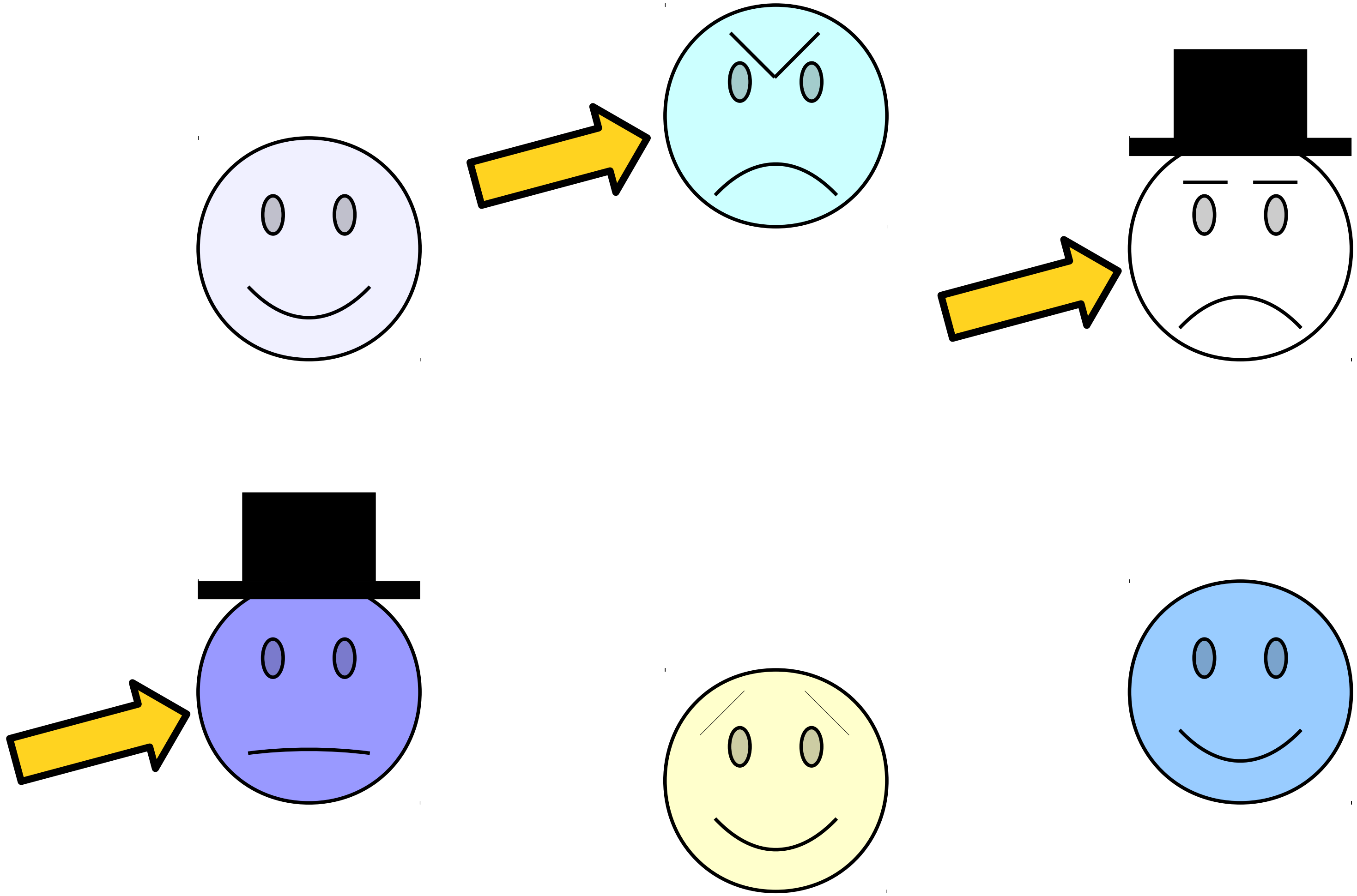
write a sentence in first-order logic that says

some smiling person wears a hat.

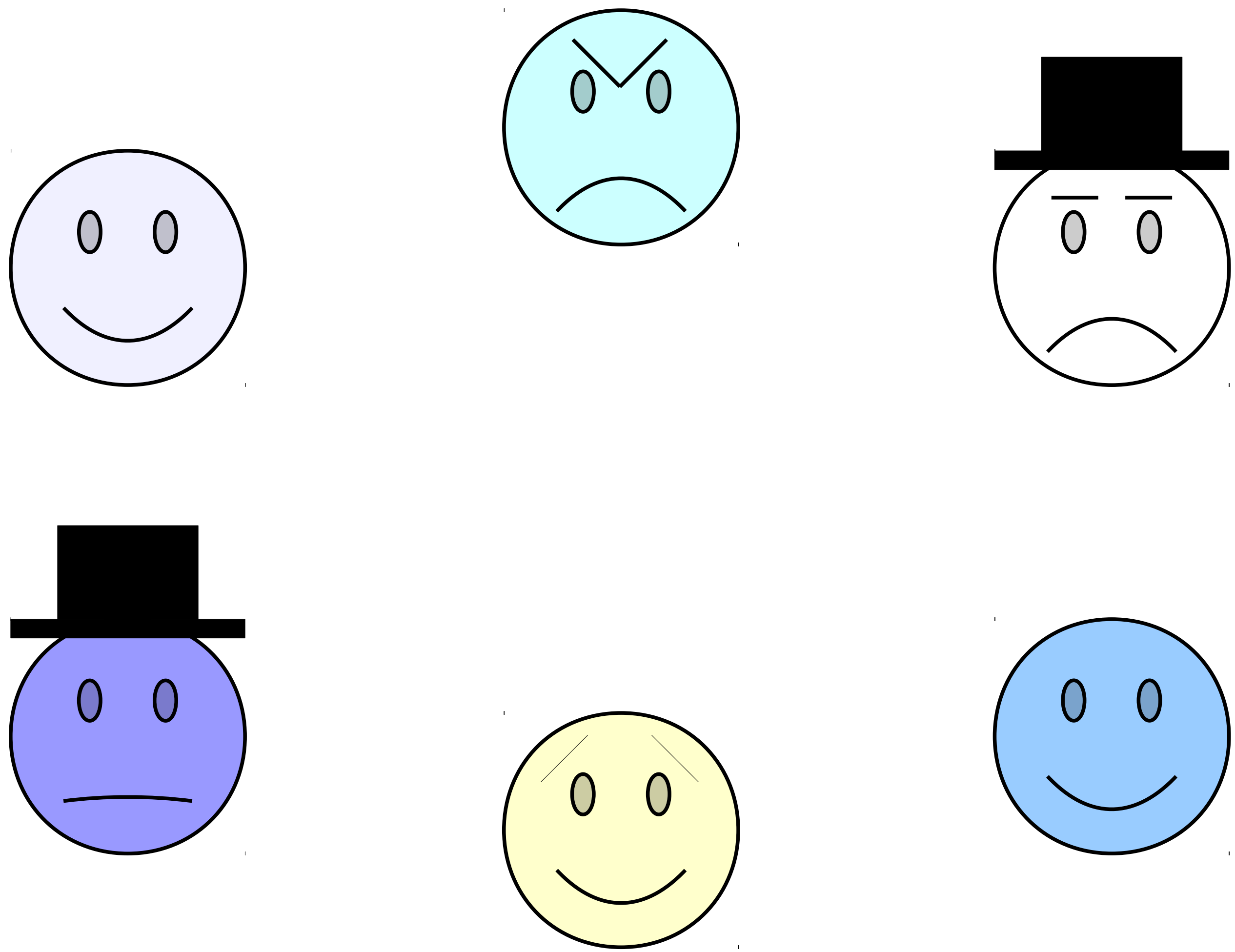


Concern: Intuitively, these people should be irrelevant.

"Some smiling person wears a hat."	***True***
$\exists x. (Smiling(x) \land WearingHat(x))$	***True***
$\exists x. (Smiling(x) \to WearingHat(x))$	***True***



"Some smiling person wears a hat."	***False***
$\exists x. (Smiling(x) \land WearingHat(x))$	***False***
$\exists x. (Smiling(x) \to WearingHat(x))$	***True***



"Some smiling person wears a hat."	***False***
$\exists x.\ (Smiling(x) \land WearingHat(x))$	***False***
~~$\exists x.\ (Smiling(x) \to WearingHat(x))$~~	***True***

“Some *P* is a *Q*”

translates as

$$\exists x.\ (P(x) \land Q(x))$$

Useful Intuition:

Existentially-quantified statements are false unless there's a positive example.

$$\exists x.\ (P(x) \land Q(x))$$

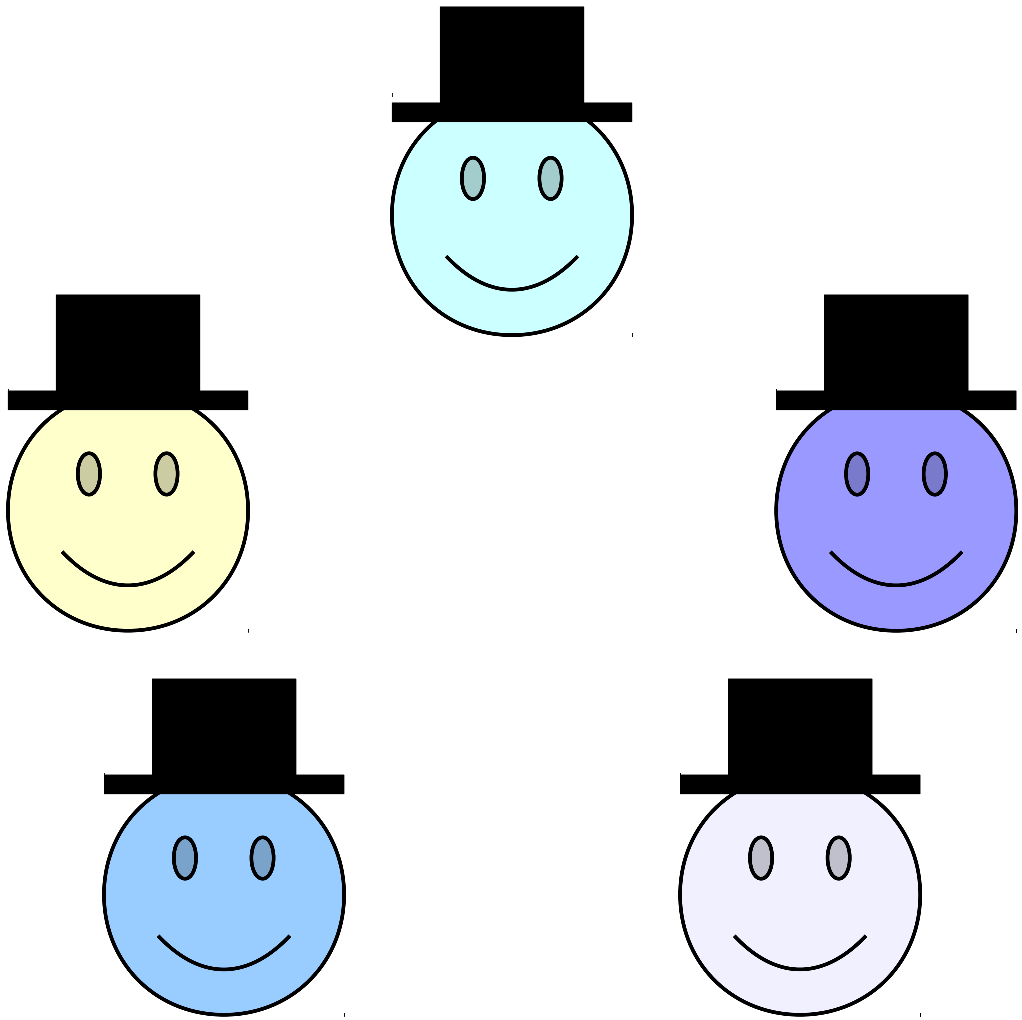
If x is an example, it *must* have property P on top of property Q.

Using the predicates

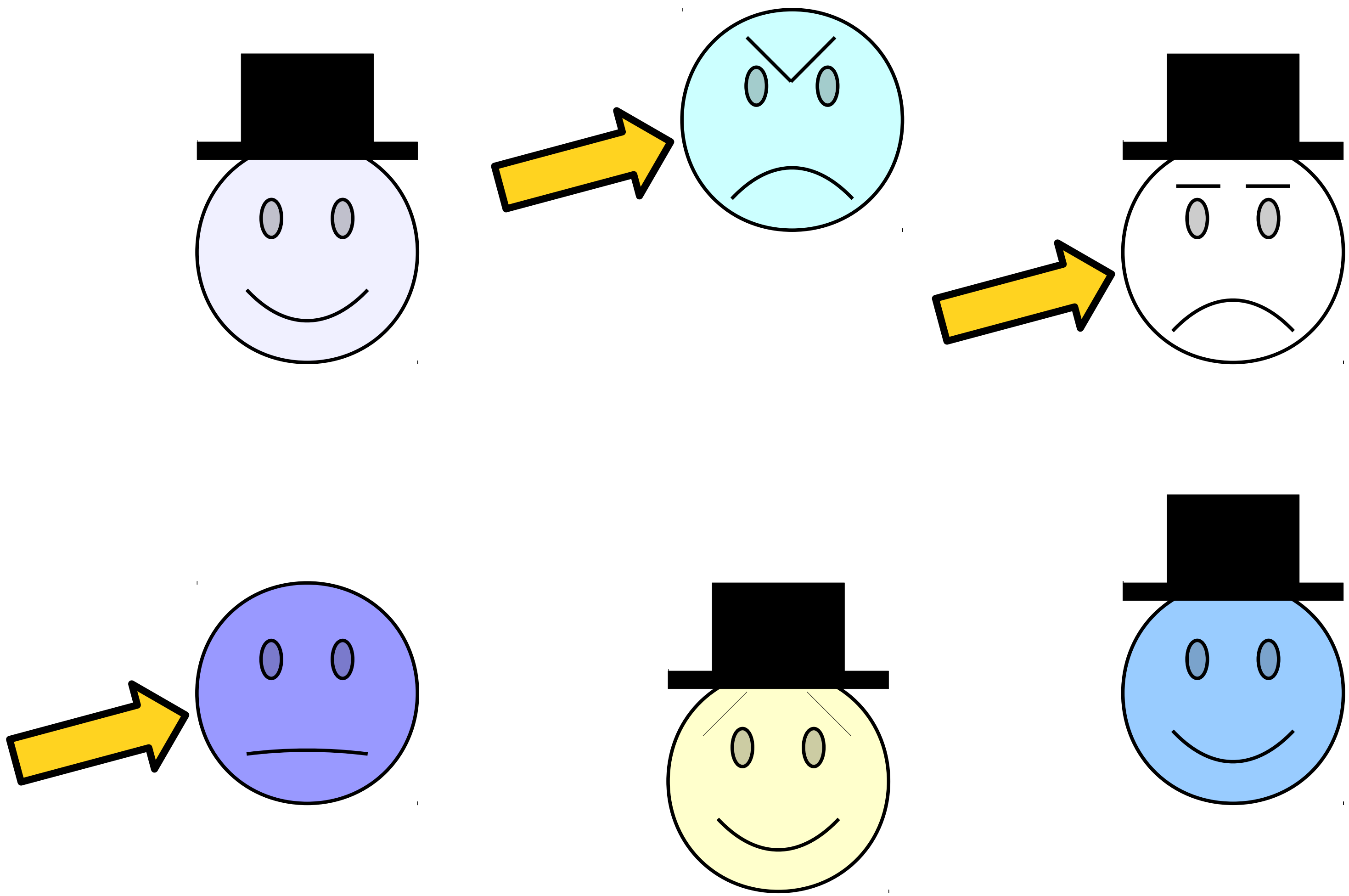
- *Smiling*(x), which states that x is smiling, and
- *WearingHat*(x), which states that x is wearing a hat,

write a sentence in first-order logic that says

every smiling person wears a hat.



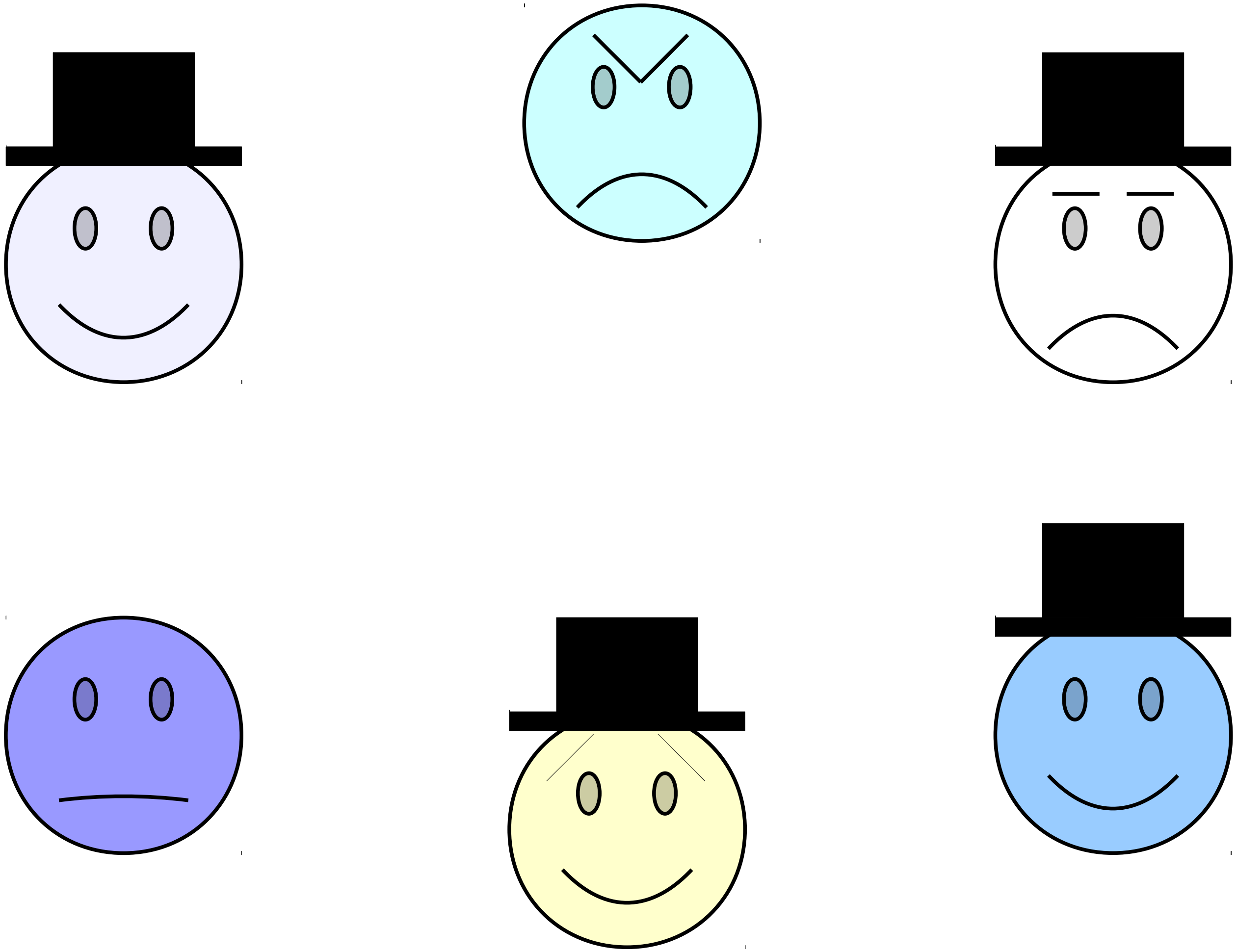
"Every smiling person wears a hat."	***True***
$\forall x. (Smiling(x) \land WearingHat(x))$	***True***
$\forall x. (Smiling(x) \to WearingHat(x))$	***True***



“Every smiling person wears a hat.” ***True***

$\forall x.\ (Smiling(x) \land WearingHat(x))$ ***False***

$\forall x.\ (Smiling(x) \to WearingHat(x))$



"Every smiling person wears a hat."	***True***
~~$\forall x.\ (Smiling(x) \land WearingHat(x))$~~	***False***
$\forall x.\ (Smiling(x) \to WearingHat(x))$	***True***

“All *P*'s are *Q*'s”

translates as

$$\forall x.\ (P(x) \to Q(x))$$

Useful Intuition:

Universally-quantified statements are true unless there's a counterexample.

$$\forall x. \, (P(x) \to Q(x))$$

If x is a counterexample, it *must* have property P but not have property Q.

Good Pairings

- The ∀ quantifier *usually* is paired with →.

$$\forall x.\ (P(x) \to Q(x))$$

- The ∃ quantifier *usually* is paired with ∧.

$$\exists x.\ (P(x) \land Q(x))$$

- In the case of ∀, the → connective prevents the statement from being *false* when speaking about some object you don't care about.
- In the case of ∃, the ∧ connective prevents the statement from being *true* when speaking about some object you don't care about.

Next Time

- ***First-Order Translations***
  - How do we translate from English into first-order logic?
- ***Quantifier Orderings***
  - How do you select the order of quantifiers in first-order logic formulas?
- ***Negating Formulas***
  - How do you mechanically determine the negation of a first-order formula?
- ***Expressing Uniqueness***
  - How do we say there's just one object of a certain type?