

First-order Logic

Overview

First-order logic

- First-order logic (FOL) uses **quantified variables** to express general properties over a **domain** (or universe) of discourse
- It uses **predicates** to capture relationships between elements of the domain
 - First-order logic is also known as predicate logic
- Unlike PL that sees propositions as indivisible statements, FOL introduces internal structure by reasoning about objects and their relationships

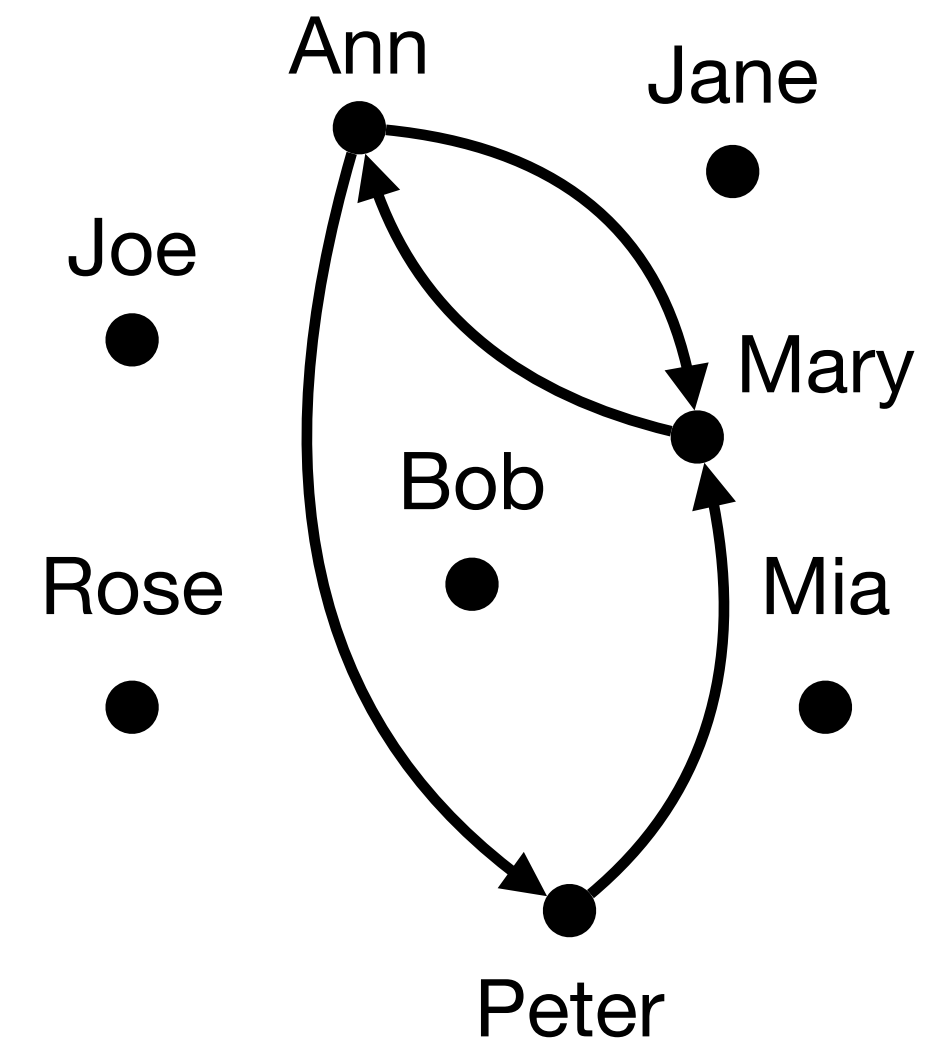
Predicates

- Predicates are **relations**, sets of tuples of elements of the domain
- All tuples have the same length, the **arity** of the predicate
 - The arity of a predicate P will be denoted by $|P|$
- **Binary** predicates (arity 2) represent relationships between two elements
- **Unary** predicates (arity 1) represent sets of elements

Binary predicates

friend = { (Ann, Peter), (Ann, Mary), (Mary, Ann), (Peter, Mary) }

friend	
Ann	Peter
Ann	Mary
Mary	Ann
Peter	Mary

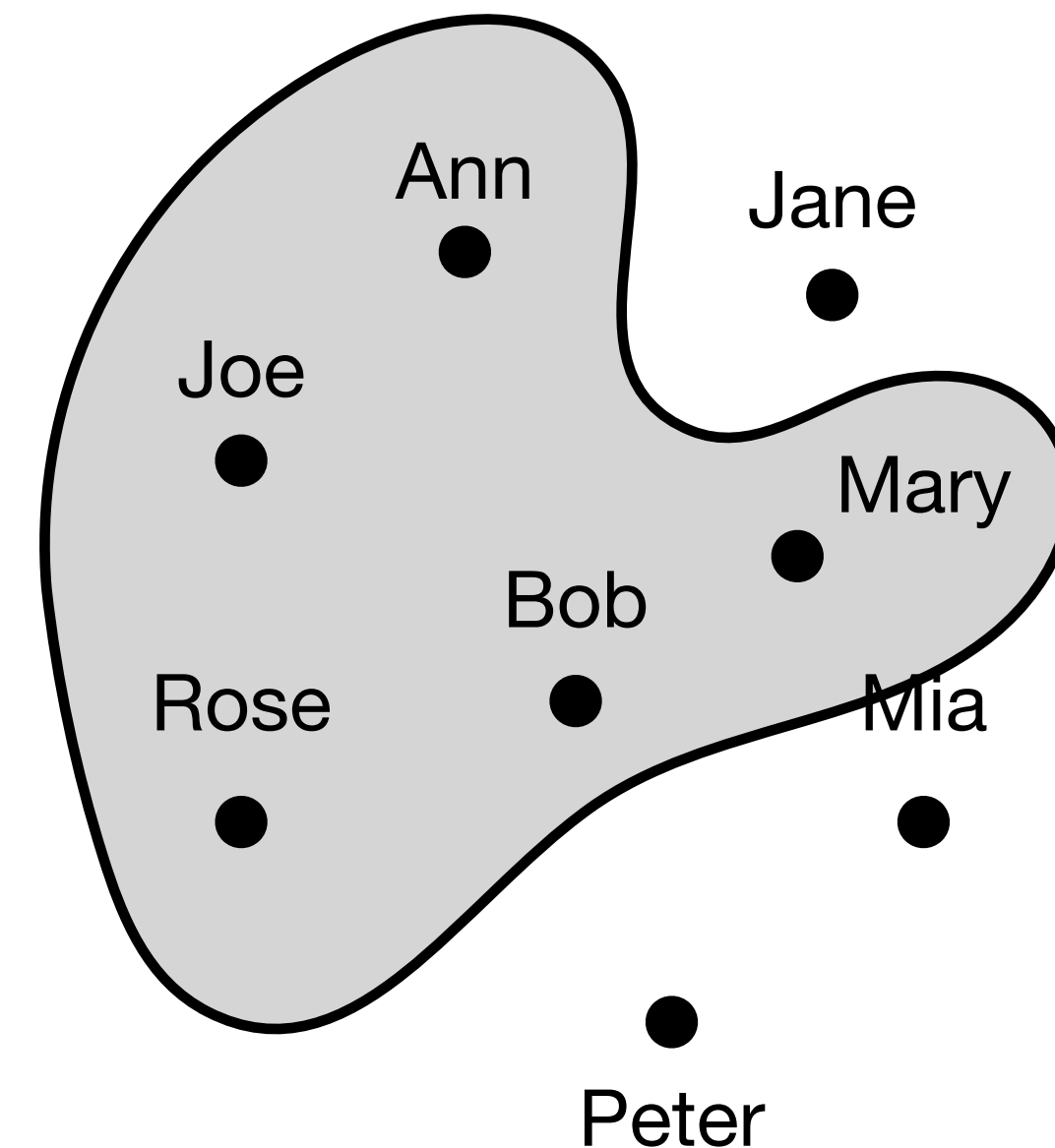


friend(Ann, Peter)

Unary predicates

Student = { (Ann), (Mary), (Joe), (Bob), (Rose) }

Student
Ann
Mary
Joe
Bob
Rose



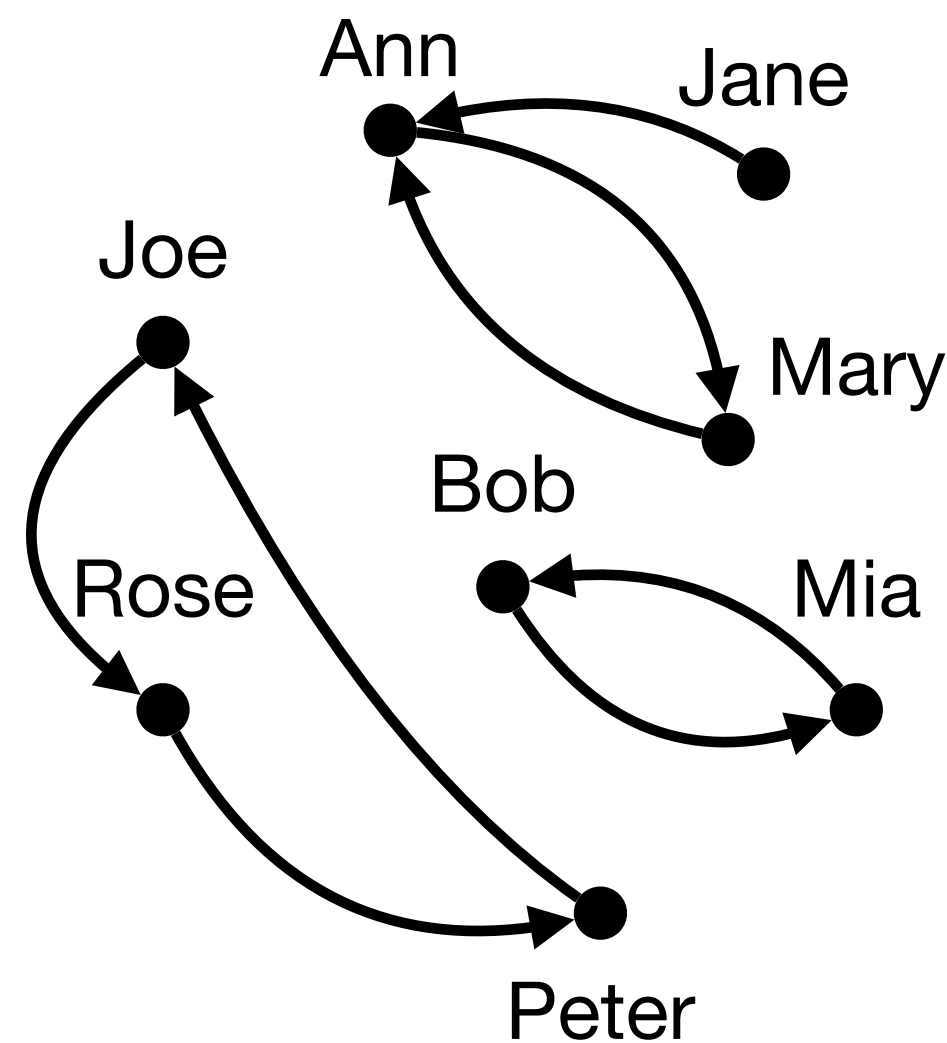
Student(Ann)

Functions, constants, and terms

- **Functions** are special relationships between tuples of elements and exactly *one* element
- The length of the input is the **arity** of the function
- **Constants** denote specific elements of the domain
- With functions, constants, and variables we can build **terms**
- A term denotes a specific element of the domain

Unary functions

$\text{bff} = \{ (\text{Ann}) \mapsto \text{Mary}, (\text{Mary}) \mapsto \text{Ann}, (\text{Peter}) \mapsto \text{Joe}, (\text{Joe}) \mapsto \text{Rose},$
 $(\text{Rose}) \mapsto \text{Peter}, (\text{Bob}) \mapsto \text{Mia}, (\text{Jane}) \mapsto \text{Ann}, (\text{Mia}) \mapsto \text{Bob} \}$



$\text{Rose} = \text{bff}(\text{bff}(\text{Peter}))$

Syntax

- Variables: $x, y, z, \dots$
- Constants: $a, b, c, \dots$
- Functions: $f, g, h, \dots$
- Predicates: $P, Q, R, \dots$
- Logic connectives: $\top, \perp, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall$ (for all), $\exists$ (exists)
- Equality: $=$
- Auxiliary symbols: parenthesis $(,)$, dot $.$

Syntax

	$\phi, \psi \doteq P(t_1, \dots, t_{ P })$	
	$t = u$	
$x, y, z, \dots \in \mathcal{X}$	$\top$	
$a, b, c, \dots \in \mathcal{C}$	$\perp$	
$f, g, h, \dots \in \mathcal{F}$	$(\neg \phi)$	$t, u \doteq x$
$P, Q, R, \dots \in \mathcal{P}$	$(\phi \wedge \psi)$	a
$\mathcal{V} = \mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$	$(\phi \vee \psi)$	$f(t_1, \dots, t_{ f })$
$t, u, \dots \in \mathbf{Term}_{\mathcal{V}}$	$(\phi \rightarrow \psi)$	
$\phi, \psi, \dots \in \mathbf{Form}_{\mathcal{V}}$	$(\phi \leftrightarrow \psi)$	
	$(\forall x. \phi)$	
	$(\exists x. \phi)$	

In the absence of parenthesis, quantifiers have the least precedence, the rest follow PL rules

Examples

- Ann is the bff of Mary $\text{Ann} = \text{bff}(\text{Mary})$
- Ann is a friend of everyone $\forall x . \text{friend}(\text{Ann}, x)$
- Friendship is symmetric $\forall x . \forall y . \text{friend}(x, y) \rightarrow \text{friend}(y, x)$
- Bffs are friends $\forall x . \text{friend}(x, \text{bff}(x))$
- Everyone has a student friend $\forall x . \exists y . \text{Student}(y) \wedge \text{friend}(x, y)$

Functions vs predicates

- Functions and constants simplify the writing of formulas but are not strictly necessary
- A function f of arity n can be represented by a predicate of arity $n + 1$ with additional constraints

$$\forall x . \exists y . f(x, y)$$

$$\forall x . \forall y . \forall z . f(x, y) \wedge f(x, z) \rightarrow y = z$$

- A constant a is a function of arity 0 and can be represented by a predicate of arity 1 with additional constraints

$$\exists x . a(x)$$

$$\forall x . \forall y . a(x) \wedge a(y) \rightarrow x = y$$

Functions vs predicates

$$\forall x . \text{friend}(x, \text{bff}(x))$$

$$\forall x . \forall y . \text{bff}(x, y) \rightarrow \text{friend}(x, y)$$

$$\forall x . \text{friend}(\text{Ann}, x)$$

$$\forall x . \text{Ann}(x) \rightarrow \forall y . \text{friend}(x, y)$$

$$\text{Ann} = \text{bff}(\text{Mary})$$

$$\forall x . \forall y . \text{Ann}(x) \wedge \text{Mary}(y) \rightarrow \text{bff}(x, y)$$

Simplified syntax

$$\begin{aligned}x, y, z, \dots &\in \mathcal{X} \\ P, Q, R, \dots &\in \mathcal{P} \\ \mathcal{V} &= \mathcal{P} \\ \phi, \psi, \dots &\in \mathbf{Form}_{\mathcal{V}}\end{aligned}$$

$$\phi, \psi \doteq P(x_1, \dots, x_{|P|})$$

$$| \ x = y$$

$$| \top$$

$$| \perp$$

$$| (\neg \phi)$$

$$| (\phi \wedge \psi)$$

$$| (\phi \vee \psi)$$

$$| (\phi \rightarrow \psi)$$

$$| (\phi \leftrightarrow \psi)$$

$$| (\forall x . \phi)$$

$$| (\exists x . \phi)$$

Semantics

- To determine the truth value of a formula we need a **structure** $\mathcal{M} = (D, I)$
 - D is a set with the **domain** of discourse
 - I is an **interpretation** for predicates, for each $P \in \mathcal{P}$ we have $I(P) \subseteq D^{|P|}$
- We also need an assignment $\mathcal{A} : \mathcal{X} \mapsto D$ with the value of the free variables
 - A variable is **free** if it is not associated with a quantifier, otherwise it is **bound**
 - A formula without free variables is **closed**
- The fact that ϕ holds under $\mathcal{M}$ with $\mathcal{A}$ is denoted by $\mathcal{M}, \mathcal{A} \models \phi$

Inductive semantics

$$\mathcal{M}, \mathcal{A} \models \top$$

$$\mathcal{M}, \mathcal{A} \not\models \perp$$

$$\mathcal{M}, \mathcal{A} \models P(x_1, \dots, x_n) \quad \text{iff} \quad (\mathcal{A}(x_1), \dots, \mathcal{A}(x_n)) \in I(P)$$

$$\mathcal{M}, \mathcal{A} \models x = y \quad \text{iff} \quad \mathcal{A}(x) \text{ is equal to } \mathcal{A}(y)$$

$$\mathcal{M}, \mathcal{A} \models \neg \phi \quad \text{iff} \quad \mathcal{M}, \mathcal{A} \not\models \phi$$

$$\mathcal{M}, \mathcal{A} \models \phi \wedge \psi \quad \text{iff} \quad \mathcal{M}, \mathcal{A} \models \phi \text{ and } \mathcal{M}, \mathcal{A} \models \psi$$

$$\mathcal{M}, \mathcal{A} \models \phi \vee \psi \quad \text{iff} \quad \mathcal{M}, \mathcal{A} \models \phi \text{ or } \mathcal{M}, \mathcal{A} \models \psi$$

$$\mathcal{M}, \mathcal{A} \models \phi \rightarrow \psi \quad \text{iff} \quad \mathcal{M}, \mathcal{A} \not\models \phi \text{ or } \mathcal{M}, \mathcal{A} \models \psi$$

$$\mathcal{M}, \mathcal{A} \models \phi \leftrightarrow \psi \quad \text{iff} \quad \mathcal{M}, \mathcal{A} \models \phi \text{ iff } \mathcal{M}, \mathcal{A} \models \psi$$

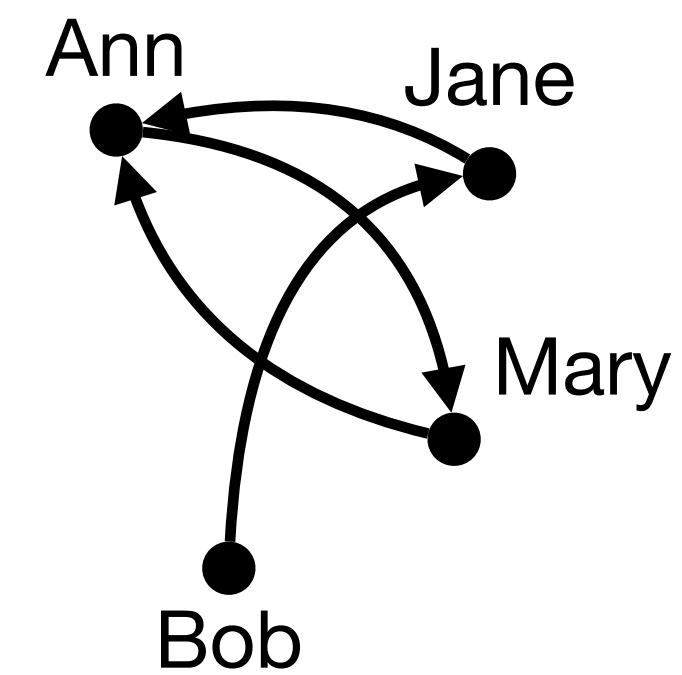
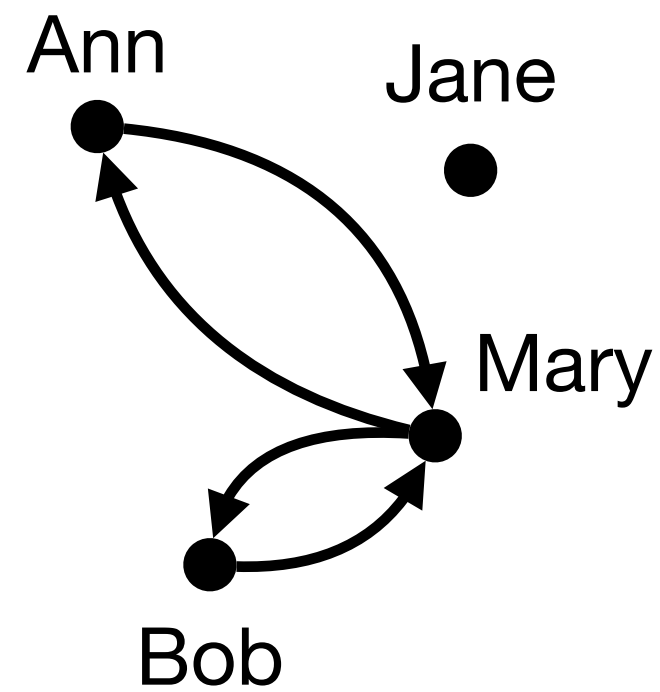
$$\mathcal{M}, \mathcal{A} \models \forall x. \phi \quad \text{iff} \quad \mathcal{M}, \mathcal{A}[x \mapsto a] \models \phi \text{ for all } a \in D$$

$$\mathcal{M}, \mathcal{A} \models \exists x. \phi \quad \text{iff} \quad \mathcal{M}, \mathcal{A}[x \mapsto a] \models \phi \text{ for some } a \in D$$

$\mathcal{A}[x \mapsto a]$ extends $\mathcal{A}$ by assigning a to x

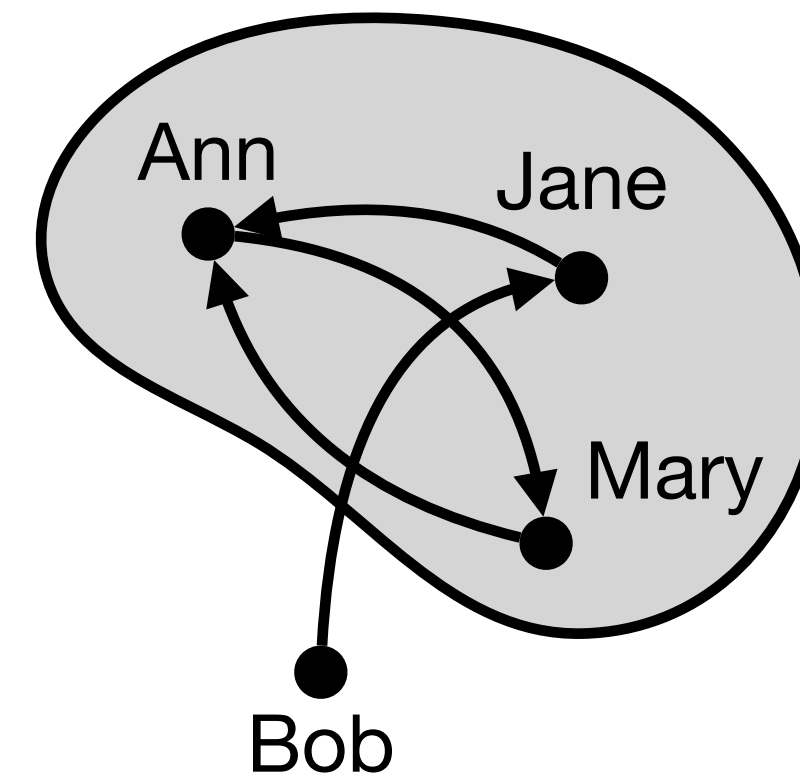
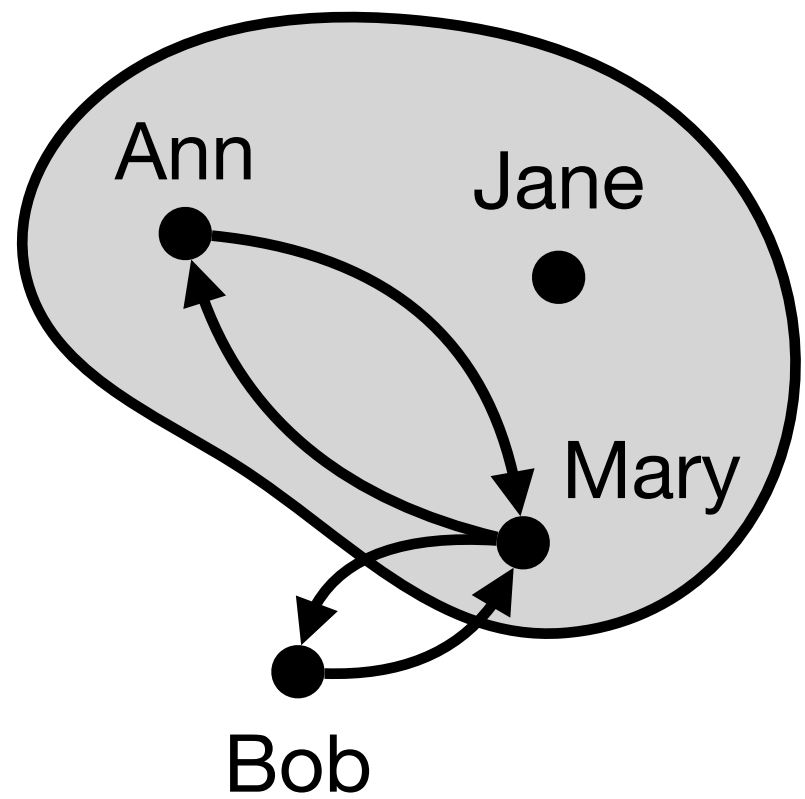
Example

$$\forall x . \forall y . \text{friend}(x, y) \rightarrow \text{friend}(y, x)$$



Example

$$\forall x . \exists y . \text{Student}(y) \wedge \text{friend}(x, y)$$



Terminology

- A formula ϕ is
 - **valid** or a **tautology** iff it holds under *all* interpretations with *all* assignments, and we write $\models \phi$
 - **satisfiable** iff it holds under *some* interpretation with *some* assignment
 - **unsatisfiable** or a **contradiction** iff it does *not* hold under *all* interpretations with *all* assignments
 - **refutable** iff it does *not* hold under *some* interpretation with *some* assignment
- A formula ϕ is valid iff $\neg\phi$ is unsatisfiable

Consequence and equivalence

- ϕ is a **consequence** of ψ (or ψ **entails** ϕ), denoted by $\psi \models \phi$, if for every $\mathcal{A}$ and every $\mathcal{M}$ that $\mathcal{M}, \mathcal{A} \models \psi$, $\mathcal{M}, \mathcal{A} \models \phi$ also holds
- ϕ and ψ are **equivalent**, denoted by $\psi \equiv \phi$, if $\phi \models \psi$ and $\psi \models \phi$
- $\phi \models \psi$ iff $\models \phi \rightarrow \psi$ and $\phi \equiv \psi$ iff $\models \phi \leftrightarrow \psi$

Basic equivalences

$$\forall x . \phi \wedge \psi \equiv (\forall x . \phi) \wedge (\forall x . \psi)$$

$$\forall x . \forall y . \phi \equiv \forall y . \forall x . \phi$$

$$\neg(\forall x . \phi) \equiv \exists x . \neg\phi$$

$$\exists x . \phi \vee \psi \equiv (\exists x . \phi) \vee (\exists x . \psi)$$

$$\exists x . \exists y . \phi \equiv \exists y . \exists x . \phi$$

$$\neg(\exists x . \phi) \equiv \forall x . \neg\phi$$

$$\phi \wedge \forall x . \psi \equiv \forall x . \phi \wedge \psi \text{ (if } x \text{ not free in } \phi) \quad \phi \vee \exists x . \psi \equiv \exists x . \phi \vee \psi \text{ (if } x \text{ not free in } \phi)$$

All the equivalences from PL still hold

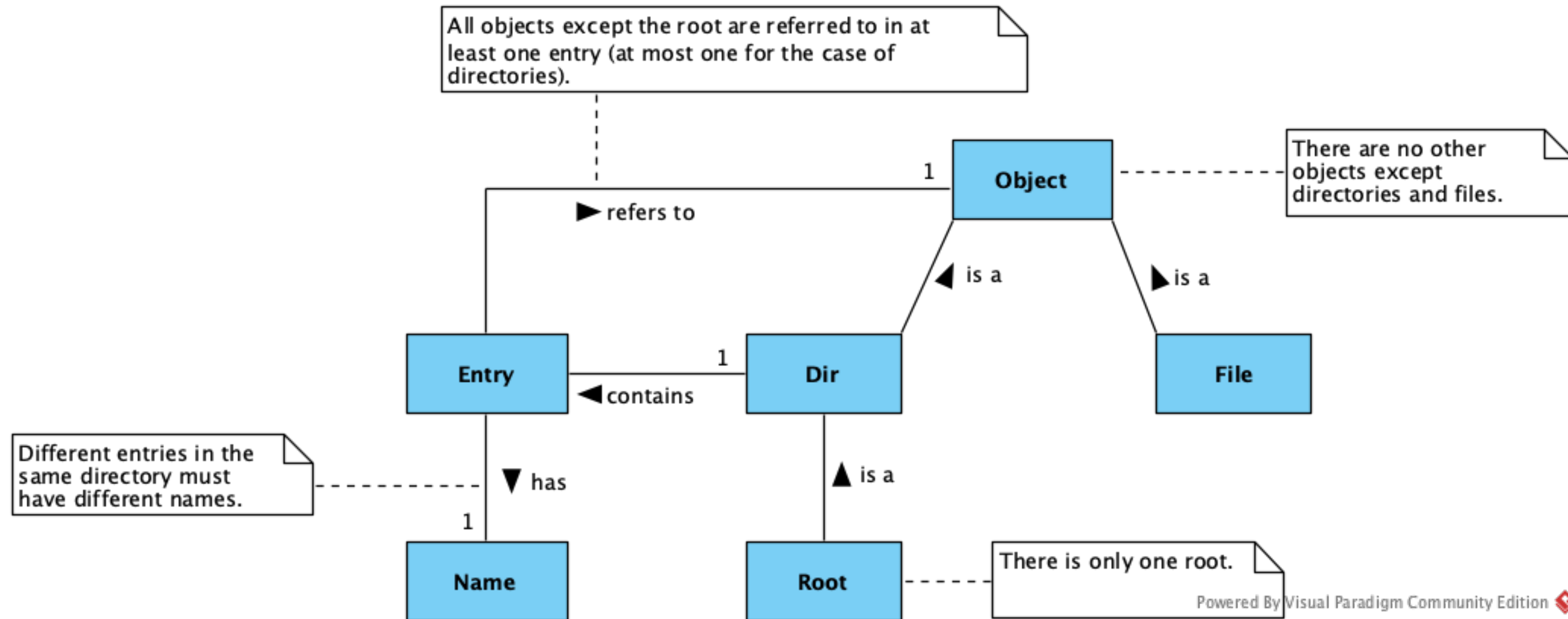
Decidability

- The decision problem “Is ϕ satisfiable?” when ϕ is in FOL is **undecidable**
 - Solvers for FOL may not terminate or answer with an unknown
- There are two strategies to “recover” decidability
 - **Bound** the domain of analysis up to a finite size (effectively reducing FOL to PL) and use SAT solvers
 - Work on subsets of FOL (**theories**) that are decidable (e.g. linear arithmetic) and use SMT solvers



Domain modelling

Domain modeling *a la* UML



Domain model analysis

- Are the requirements consistent?
- Any forgotten or redundant requirements?
- Do the requirements entail all the expected properties?

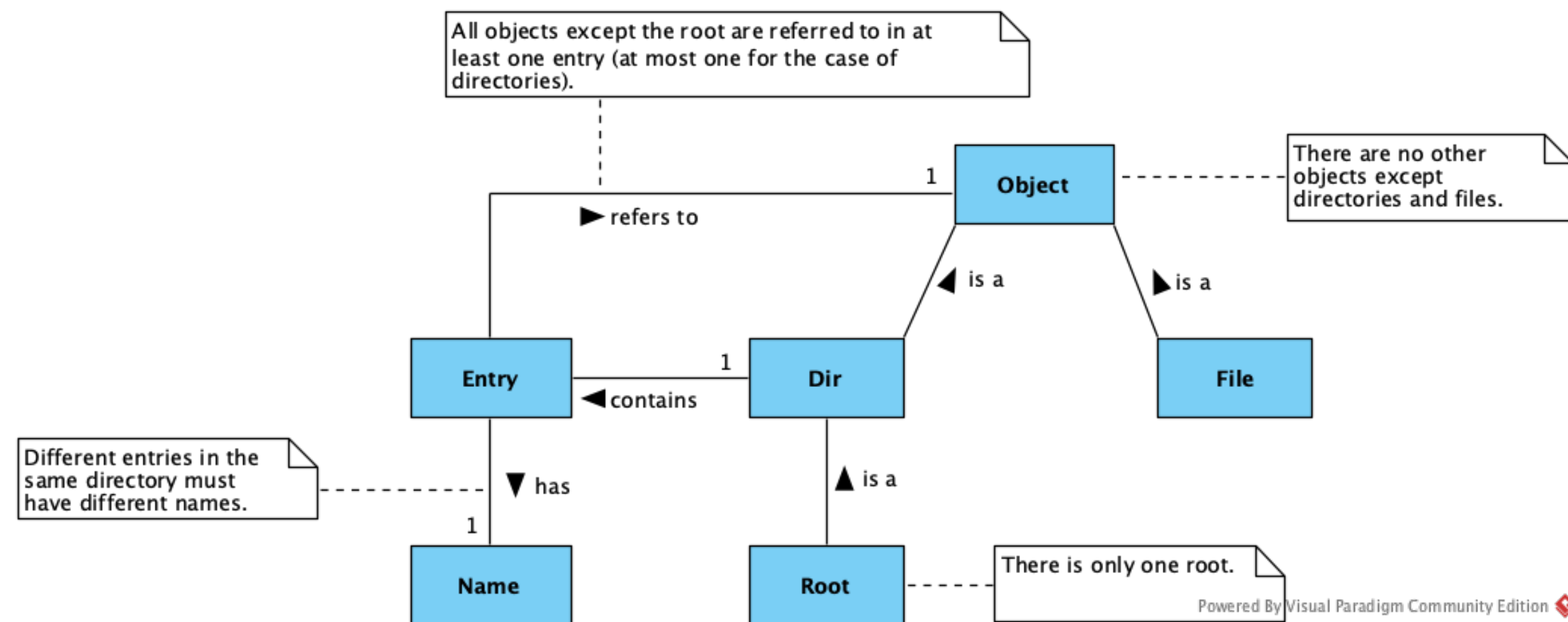
Entities

- The “is a” relationship denotes a specialization or extension
- Formally it can be seen as a subset relationship
- All entities in a diagram not related by “is a” are disjoint
 - Top-level entities are disjoint
 - Multiple extensions of the same entity are disjoint

Formalizing entities

- Each entity can be formalized by a unary predicate
- Constraints should be added to specify
 - The subset relationship of extension signatures
 - The disjointness of all other signatures

Formalizing entities



$\text{Object} \subseteq D$

$\text{File} \subseteq D$

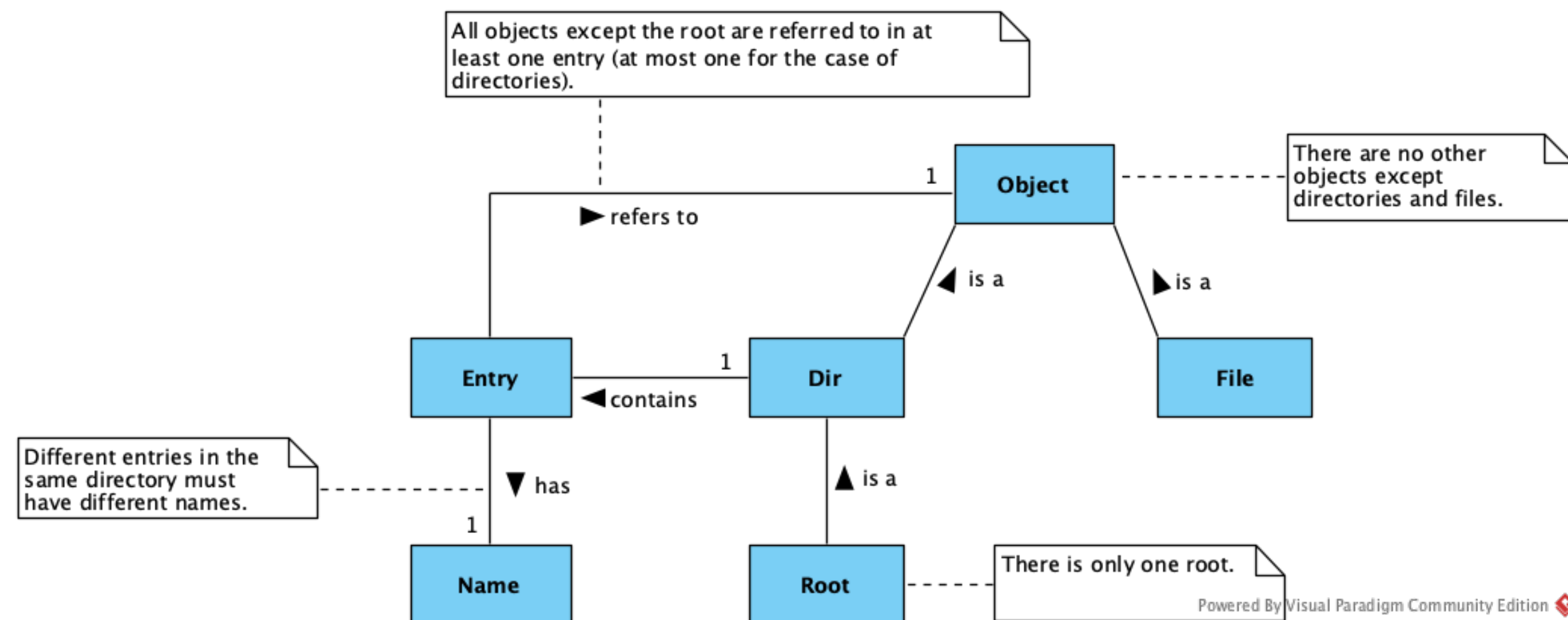
$\text{Dir} \subseteq D$

$\text{Root} \subseteq D$

$\text{Entry} \subseteq D$

$\text{Name} \subseteq D$

Formalizing entities



$$\forall x . \neg(\text{Object}(x) \wedge \text{Entry}(x))$$

$$\forall x . \neg(\text{Object}(x) \wedge \text{Name}(x))$$

$$\forall x . \neg(\text{Name}(x) \wedge \text{Entry}(x))$$

$$\forall x . \text{File}(x) \rightarrow \text{Object}(x)$$

$$\forall x . \text{Dir}(x) \rightarrow \text{Object}(x)$$

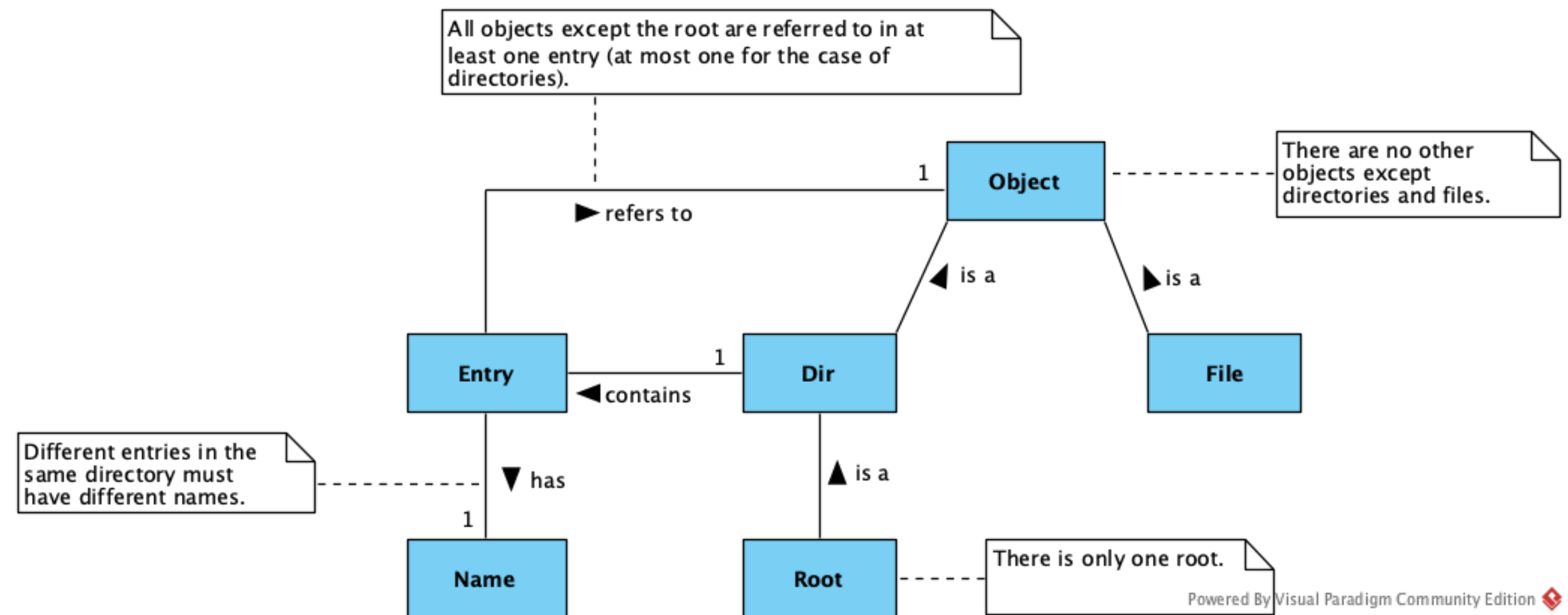
$$\forall x . \neg(\text{File}(x) \wedge \text{Dir}(x))$$

$$\forall x . \text{Root}(x) \rightarrow \text{Dir}(x)$$

Formalizing associations

- Each association relationship can be formalized by a predicate
- Constraints should be added to specify
 - The type of related entities
 - The multiplicity restrictions at each end

Formalizing associations



contains $\subseteq D \times D$

refersTo $\subseteq D \times D$

has $\subseteq D \times D$

Syntactic sugar

$$\forall x : A . \phi \quad \equiv \quad \forall x . A(x) \rightarrow \phi$$

$$\exists x : A . \phi \quad \equiv \quad \exists x . A(x) \wedge \phi$$

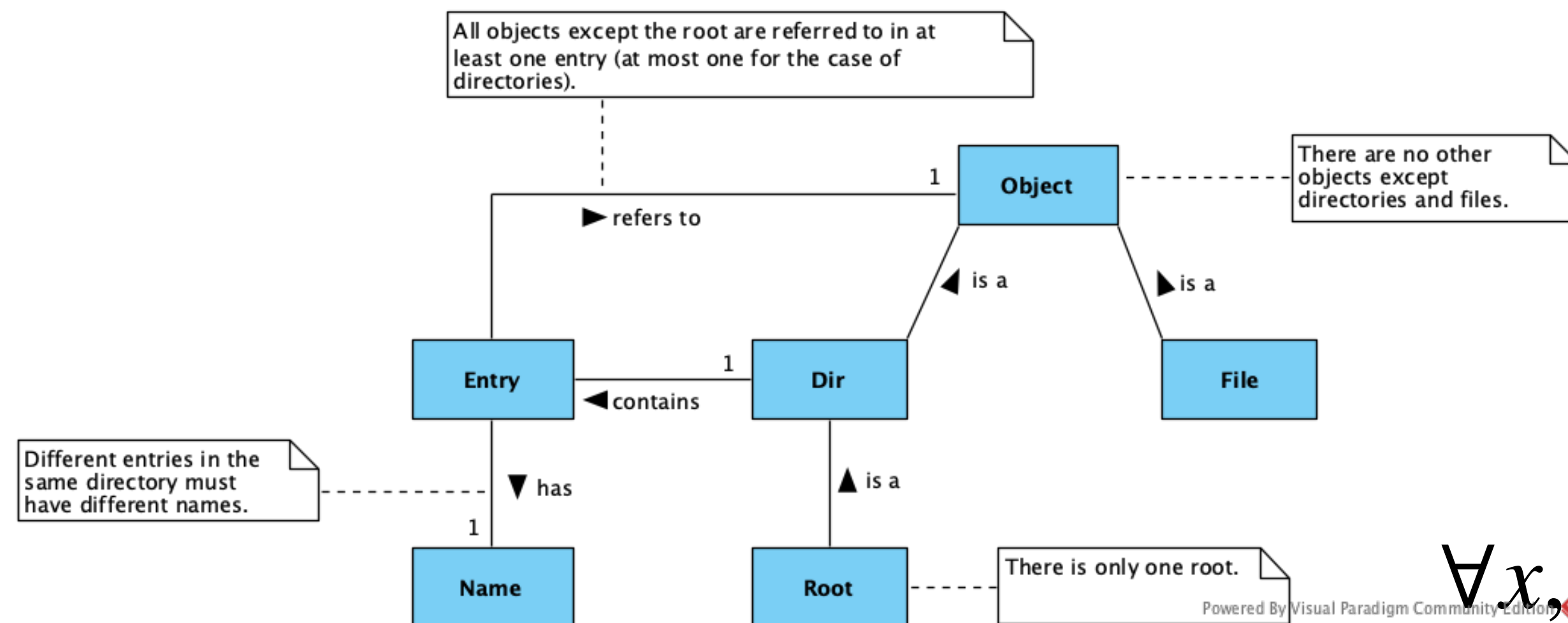
$$\forall x : A, y : B . \phi \quad \equiv \quad \forall x : A . \forall y : B . \phi$$

$$\forall x, y : A . \phi \quad \equiv \quad \forall x : A, y : A . \phi$$

$$\exists x : A, y : B . \phi \quad \equiv \quad \exists x : A . \exists y : B . \phi$$

$$\exists x, y : A . \phi \quad \equiv \quad \exists x : A, y : A . \phi$$

Formalizing associations



$$\forall x, y . \text{contains}(x, y) \rightarrow \text{Dir}(x) \wedge \text{Entry}(y)$$

$$\forall x, y . \text{refersTo}(x, y) \rightarrow \text{Entry}(x) \wedge \text{Object}(y)$$

$$\forall x, y . \text{has}(x, y) \rightarrow \text{Entry}(x) \wedge \text{Name}(y)$$

$$\forall x : \text{Entry} . \exists y . \text{has}(x, y)$$

$$\forall x, y, z . \text{has}(x, y) \wedge \text{has}(x, z) \rightarrow y = z$$

$$\forall x : \text{Entry} . \exists y . \text{refersTo}(x, y)$$

$$\forall x, y, z . \text{refersTo}(x, y) \wedge \text{refersTo}(x, z) \rightarrow y = z$$

$$\forall x : \text{Entry} . \exists y . \text{contains}(y, x)$$

$$\forall x, y, z . \text{contains}(y, x) \wedge \text{contains}(z, x) \rightarrow y = z$$

Specifying requirements

- There is only one root

$$\exists x . \text{Root}(x)$$

$$\forall x, y : \text{Root} . x = y$$

- There are no other objects except directories and files

$$\forall x : \text{Object} . \text{Dir}(x) \vee \text{File}(x)$$

- Different entries in the same directory must have different names

$$\forall x, y, z, w . \text{contains}(x, y) \wedge \text{contains}(x, z) \wedge \text{has}(y, w) \wedge \text{has}(z, w) \rightarrow y = z$$

Specifying requirements

- All objects except the root are referred to in at least one entry (at most one for the case of directories)

- All objects except the root are referred to in at least one entry

$$\forall x : \text{Object} . \neg \text{Root}(x) \rightarrow \exists y . \text{refersTo}(y, x)$$

- The root is not referred in any entry

$$\forall x : \text{Entry}, y : \text{Root} . \neg \text{refersTo}(x, y)$$

- All directories are referred to in at most one entry

$$\forall x : \text{Dir} . \forall y, z . \text{refersTo}(y, x) \wedge \text{refersTo}(z, x) \rightarrow y = z$$

Domain model analysis

- Are the requirements consistent?
 - Check if the specified requirements are satisfiable
- Any forgotten or redundant requirements?
 - Ask for satisfiable interpretations and inspect the respective scenarios
- Do the requirements entail all the expected properties?
 - Check whether the requirements entail expected properties

A possible interpretation

Object = { (Root), (Dir), (File) }

Dir = { (Root), (Dir) }

File = { (File) }

Root = { (Root) }

Entry = { (Entry1), (Entry2), (Entry3) }

Name = { (Name1), (Name2) }

refersTo = { (Entry1, File), (Entry2, Dir), (Entry3, Dir) }

contains = { (Root, Entry1), (Root, Entry2), (Dir, Entry3) }

has = { (Entry1, Name1), (Entry2, Name2), (Entry3, Name1) }

A possible interpretation

Object = { (Root), (Dir), (File) }

Dir = { (Root), (Dir) }

File = { (File) }

Root = { (Root) }

Entry = { (Entry1), (Entry2), (Entry3) }

Name = { (Name1), (Name2) }

refersTo = { (Entry1, File), (Entry2, Dir), (Entry3, Dir) }

contains = { (Root, Entry1), (Root, Entry2), (Dir, Entry3) }

has = { (Entry1, Name1), (Entry2, Name2), (Entry3, Name1) }

Limitations

- Tedious to formalize entities and associations
- Scenarios can be difficult to understand
- Analysis can diverge



Alloy

Alloy

- ~~Tedious to formalize entities and associations~~
 - High-level syntax to declare sets and relations
- ~~Scenarios can be difficult to understand~~
 - Graphical depiction of scenarios
- ~~Analysis can diverge~~
 - The domain of analysis is bounded with a *scope*

Scenario depiction

