Structural design with Alloy

Alcino Cunha

lucid, systematic,
and penetrating
treatment of basic
and dynamic data
structures, sorting,
recursive algorithms,
language structures,
and compiling

PRENTICE-HALL
SERIES IN
AUTOMATIC
COMPUTATION

NIKLAUS WIRTH

Algorithms + Data Structures = Programs

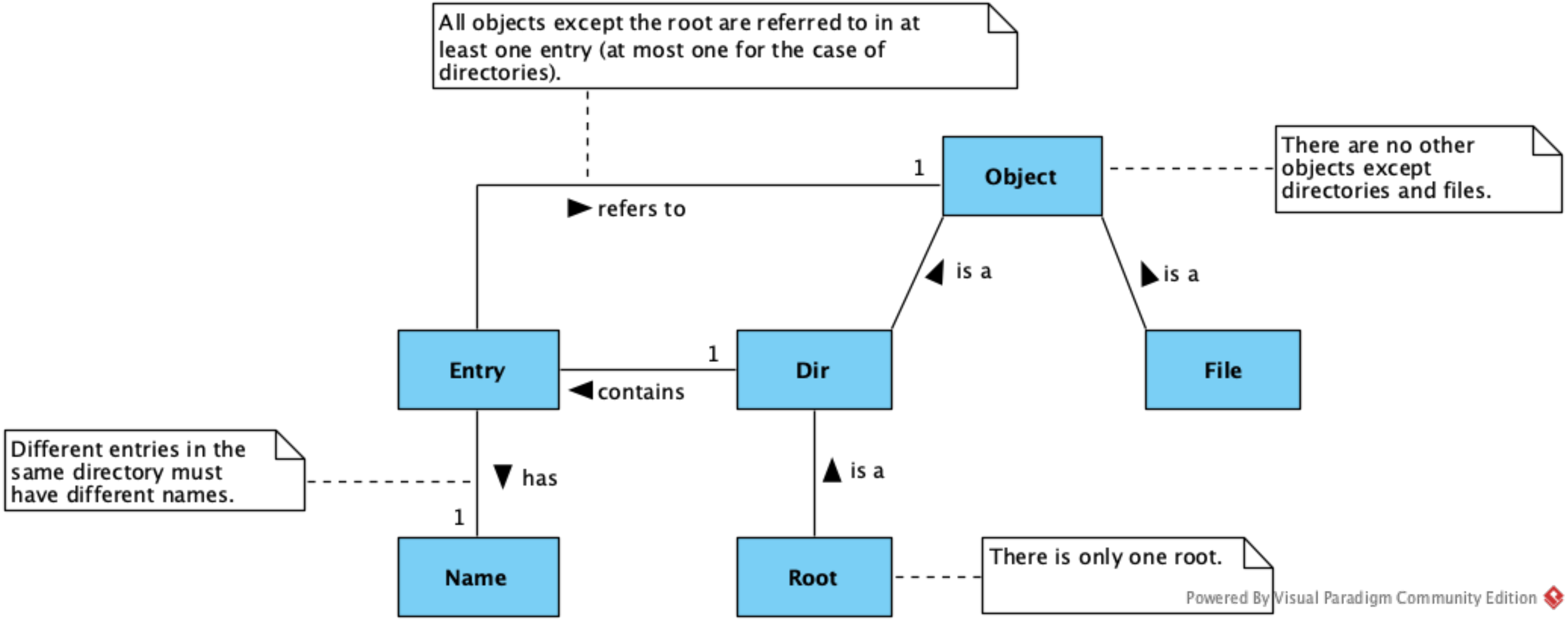
Software structures

- Data structures
- Database schemas
- Architectures
- Network topologies
- Ontologies
- Domain models

Structural design

- Understand entities and their relationships
- Elicit requirements
- Explore alternatives

Domain modeling *a la* UML



Requirement elicitation

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

“The core of software development [...] is the *design* of abstractions. An abstraction is [...] an idea reduced to its essential form.”

–Daniel Jackson

Software Abstractions

Logic, Language, and Analysis

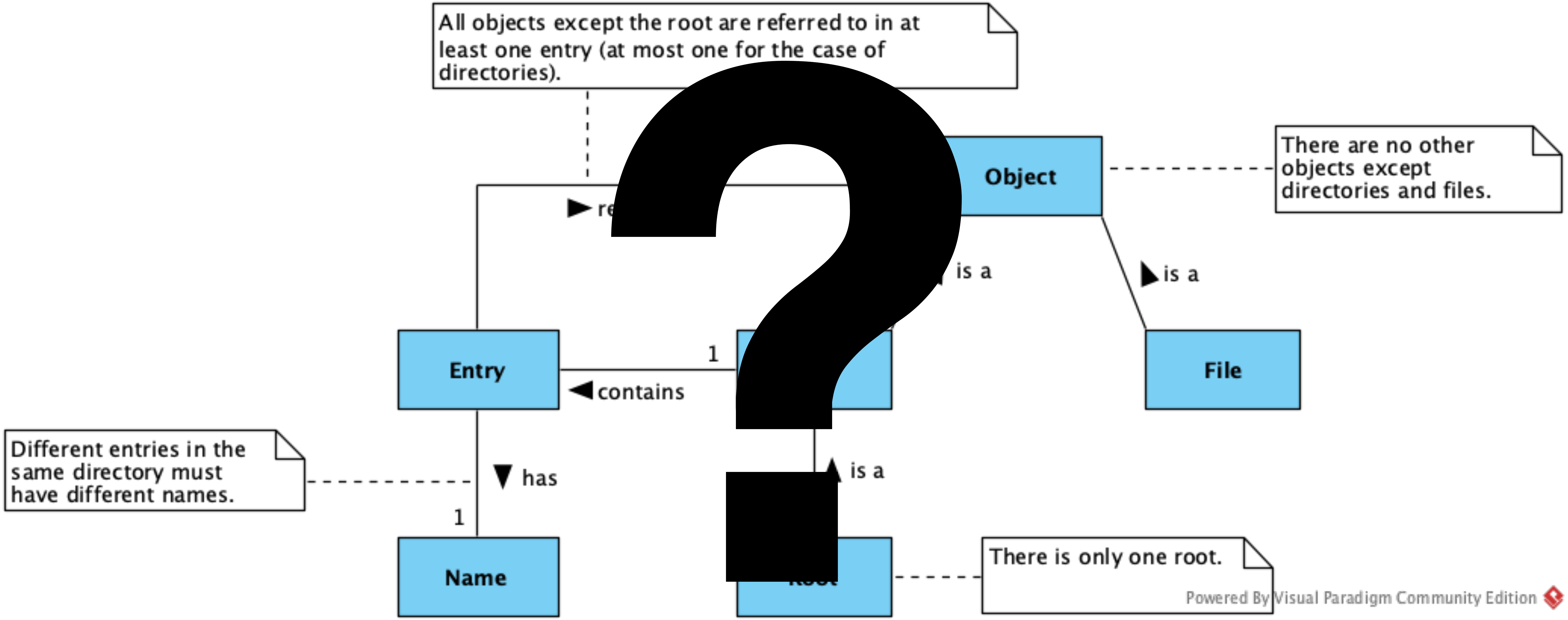
Revised edition

Daniel Jackson

Software design with Alloy

- Alloy is a formal modeling language
- Based on relational logic, an extension of first-order logic
- Models can be automatically analyzed
- Tailored for abstraction - everything is a relation!

Domain modeling with Alloy



First-order logic

Signatures

- Unary predicates are known as *signatures* and are declared with **`sig`**
- Signatures are inhabited by *atoms* from a finite *domain* of discourse
- Signatures can be top-level, extensions, or subsets
- Top-level and extension signatures are disjoint
- Signatures can be abstract, only containing atoms in the extensions
- Signatures can have a multiplicity (**`lone`**, **`some`**, **`one`**)

Top-level signatures

```
sig Object {}
sig Entry {}
sig Name {}
```

$\text{Object} \subseteq D$

$\text{Entry} \subseteq D$

$\text{Name} \subseteq D$

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

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

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

Extension signatures

```
sig Dir extends Object {}
sig File extends Object {}
```

$\text{Dir} \subseteq D$

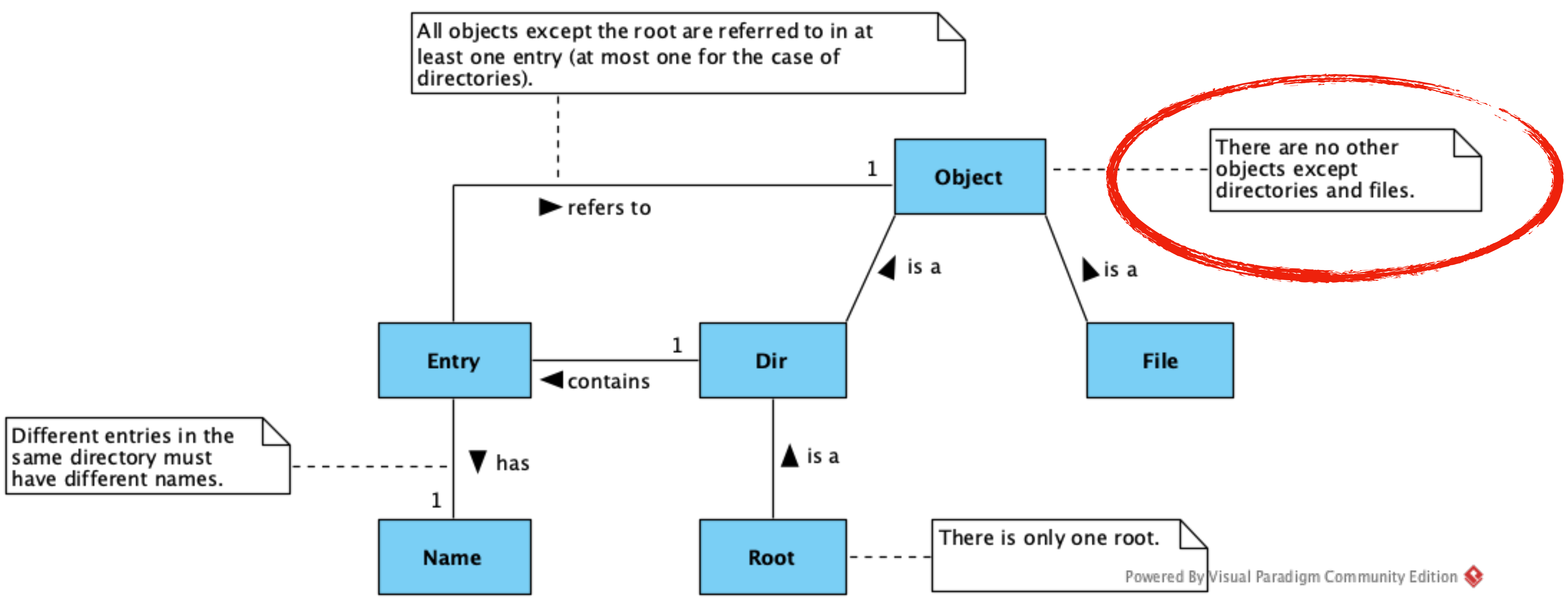
$\text{File} \subseteq D$

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

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

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

Object is abstract



Abstract signatures

```
abstract sig Object {}
sig Dir extends Object {}
sig File extends Object {}
```

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

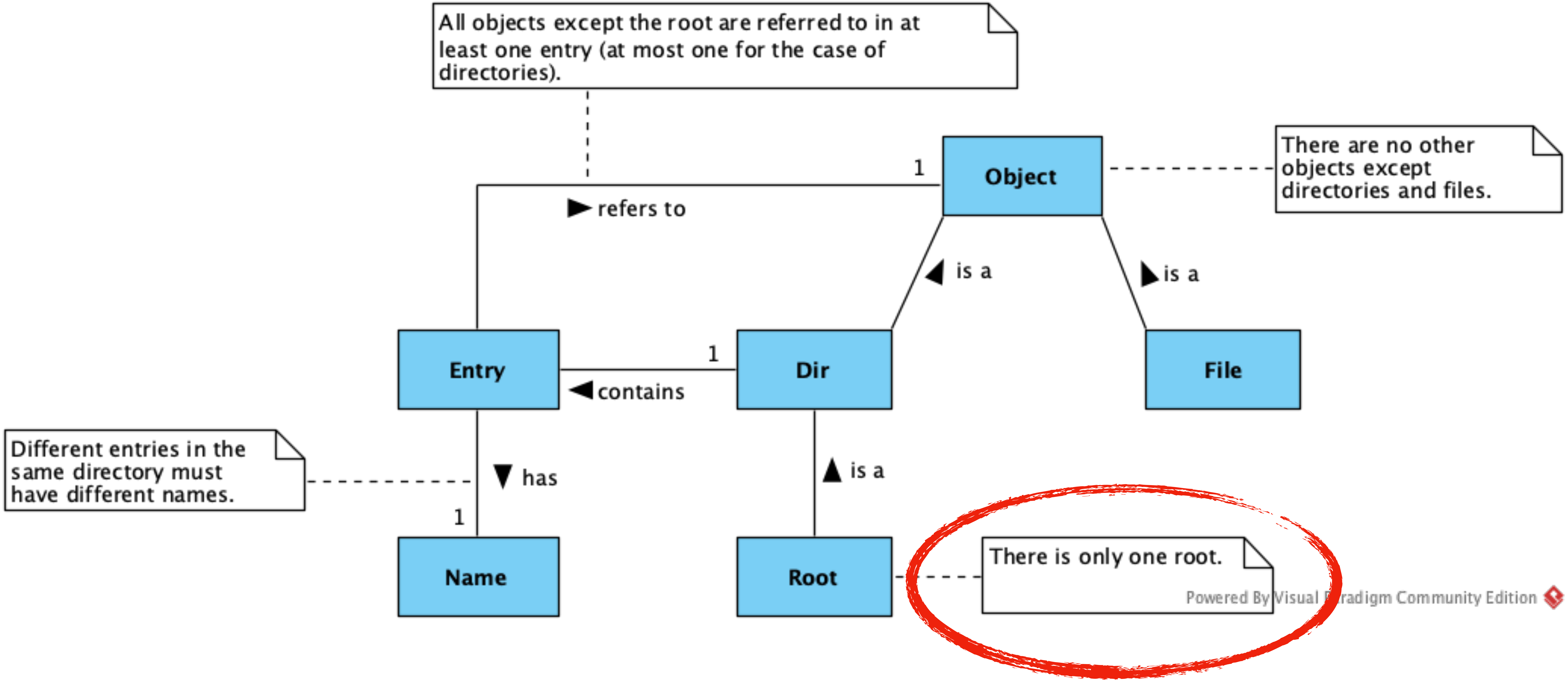
Subset signatures

```
sig Root in Dir {}
```

$$\text{Root} \subseteq D$$

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

There is only one root



Signature multiplicities

```
one sig Root in Dir {}
```

$$\exists x \,.\, \mathrm{Root}(x)$$

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

Fields

- Predicates of arity 2 or more are known as *fields*
- Fields are inhabited by *tuples* of atoms
- Must be declared inside the *domain* signature
- Multiplicities (**`set`**, **`lone`**, **`some`**, **`one`**) can be imposed on the targets
- If no multiplicity is imposed the default is **`one`**

Fields

```
sig Dir {
  contains : set Entry
}
```

$\text{contains} \subseteq D \times D$

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

Fields

```
sig Entry {
  refersTo : one Object,
  has : one Name
}
```

$\text{refersTo} \subseteq D \times D$

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

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

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

$\text{has} \subseteq D \times D$

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

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

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

Facts

- *Facts* specify assumptions

```
fact { φ }
```

- Facts can be named

```
fact Name { φ }
```

- A single fact can have several constraints, one per line

```
fact {
  φ
  ψ
}
```

FOL vs Alloy

$\neg\phi$	`!` ϕ
$\phi \wedge \psi$	ϕ `&&` ψ
$\phi \vee \psi$	ϕ `||` ψ
$\phi \rightarrow \psi$	ϕ `=>` ψ
$(\phi \wedge \psi) \vee (\neg\phi \wedge \theta)$	ϕ `=>` ψ **`else`** θ
$\phi \leftrightarrow \psi$	ϕ `<=>` ψ

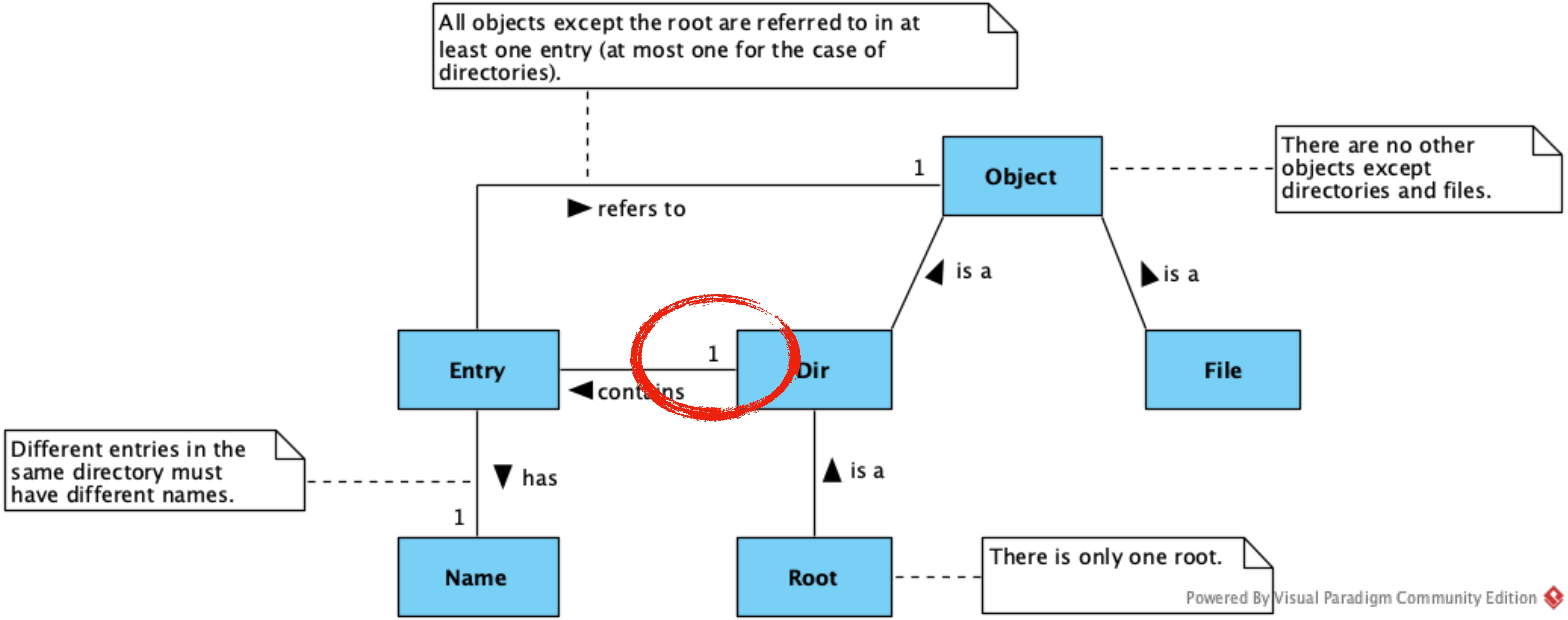
FOL vs Alloy

$\neg\phi$	**`not`** ϕ
$\phi \wedge \psi$	ϕ **`and`** ψ
$\phi \vee \psi$	ϕ **`or`** ψ
$\phi \rightarrow \psi$	ϕ **`implies`** ψ
$(\phi \wedge \psi) \vee (\neg\phi \wedge \theta)$	ϕ **`implies`** ψ **`else`** θ
$\phi \leftrightarrow \psi$	ϕ **`iff`** ψ

FOL vs Alloy

$x = y$	x `=` y
$A(x)$	x **`in`** A
$R(x, y)$	x`->`y **`in`** R
$\forall x \,.\, A(x) \rightarrow \phi$	**`all`** x **`:`** A `|` ϕ
$\exists x \,.\, A(x) \wedge \phi$	**`some`** x **`:`** A `|` ϕ

Each entry is contained in one directory



Each entry is contained in one directory

```
fact {
  // Each entry is contained in one directory
  all x : Entry | some y : Dir | y->x in contains
  all x : Entry, y,z : Dir {
    y->x in contains and z->x in contains implies y = z
  }
}
```

Commands

- Alloy has two types of analysis *commands*:
  - `run { φ }` asks for an *example* that satisfies all facts and φ
  - `check { φ }` asks for a *counter-example* that satisfies all facts by refutes assertion φ
- Likewise facts, commands can be named and can have several constraints, one per line
- In the visualizer it is possible to ask for more examples or counter-examples by pressing *New*

Instances

- Both examples and counter-examples are first-order structures
- In Alloy first-order structures are known as *instances*
- An instance is a valuation to all the signatures and fields
- In an instance "everything is a relation"
    - Signatures are unary relations (sets of unary tuples)
    - Constants are singleton unary relations (sets with one unary tuple)
- By default instances are depicted as graphs

Scopes

- To ensure decidability commands have a *scope*
- The scope imposes a limit on the size of the (finite) domain the Analyzer will exhaustively explore
- The default scope imposes a limit of 3 atoms per top-level signature
- **`for`** can be used to specify a different scope for top-level signatures
- **`but`** can be used to specify different scopes for specific signatures
- **`exactly`** can be used to specify exact scopes

The small scope hypothesis

- If **`run { φ }`** returns an instance then φ is *consistent*, else φ **may be *inconsistent***
  - Could be consistent with a bigger scope!
- If **`check { φ }`** returns an instance then φ is *invalid,* else φ **may be *valid***
  - Could be invalid with a bigger scope!!!
- Anecdotical evidence suggests that most invalid assertions (or consistent predicates) can be refuted (or witnessed) with a small scope

Atoms

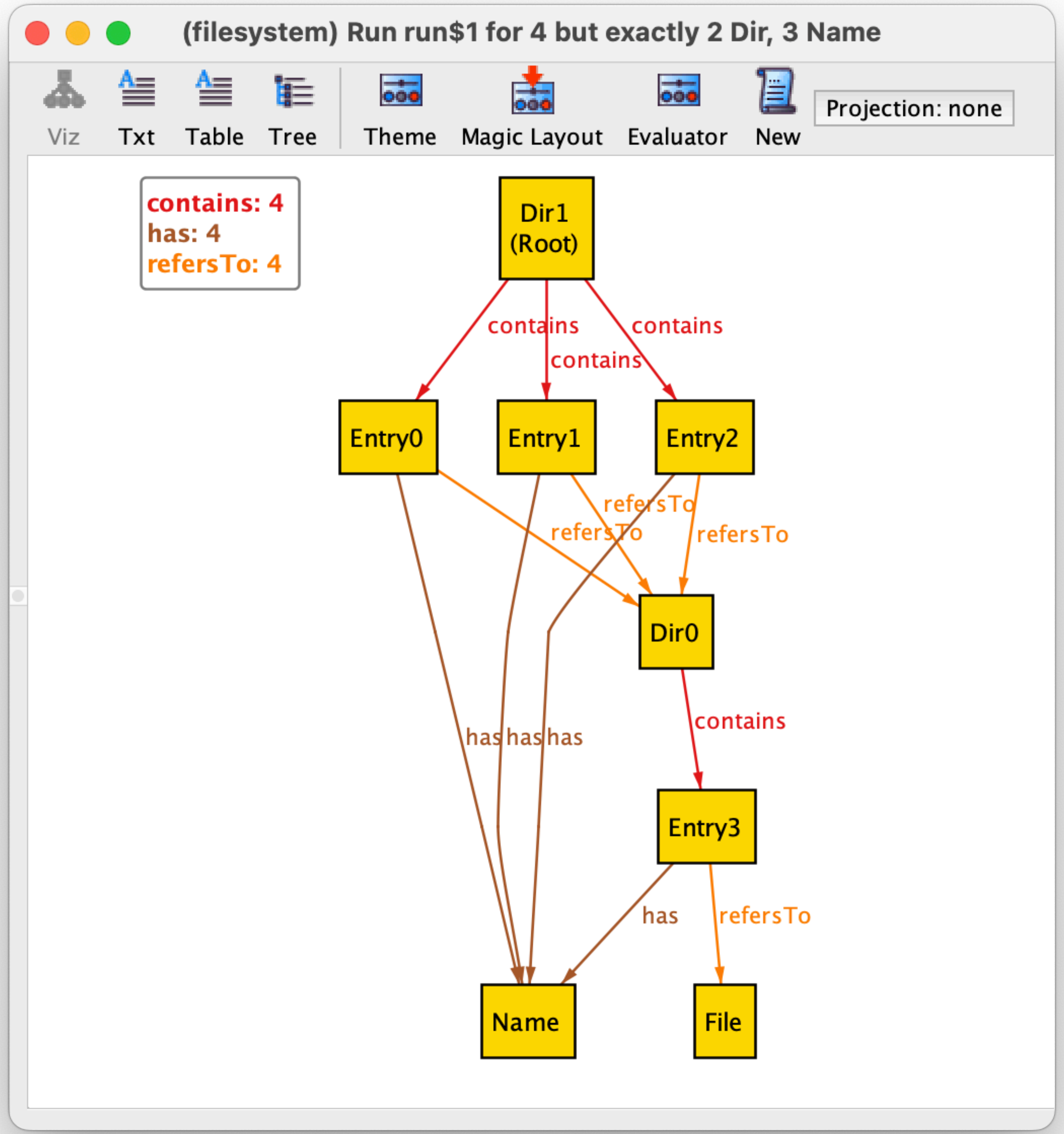
- The universe of discourse contains *atoms*
- Atoms are *uninterpreted* (no semantics)
- Named automatically according to the respective signatures
- Two instances are *isomorphic* (or *symmetric*) if they are equal modulo renaming
- The analysis implements a *symmetry breaking* mechanism to avoid returning isomorphic instances

A simple command

```
run {} for 4 but exactly 2 Dir, 3 Name
```

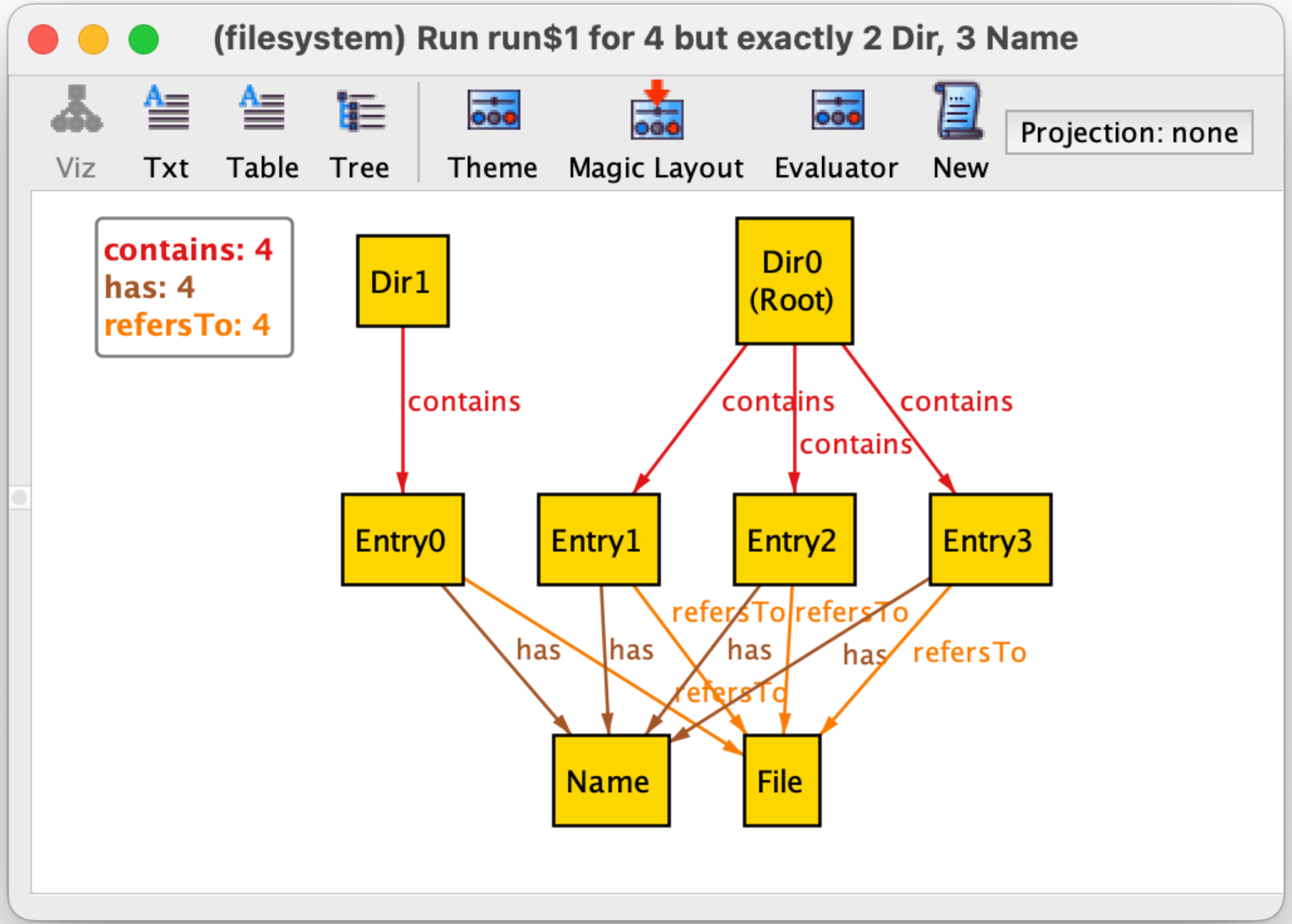
A simple command

```
run {} for 4 but exactly 2 Dir, 3 Name
```



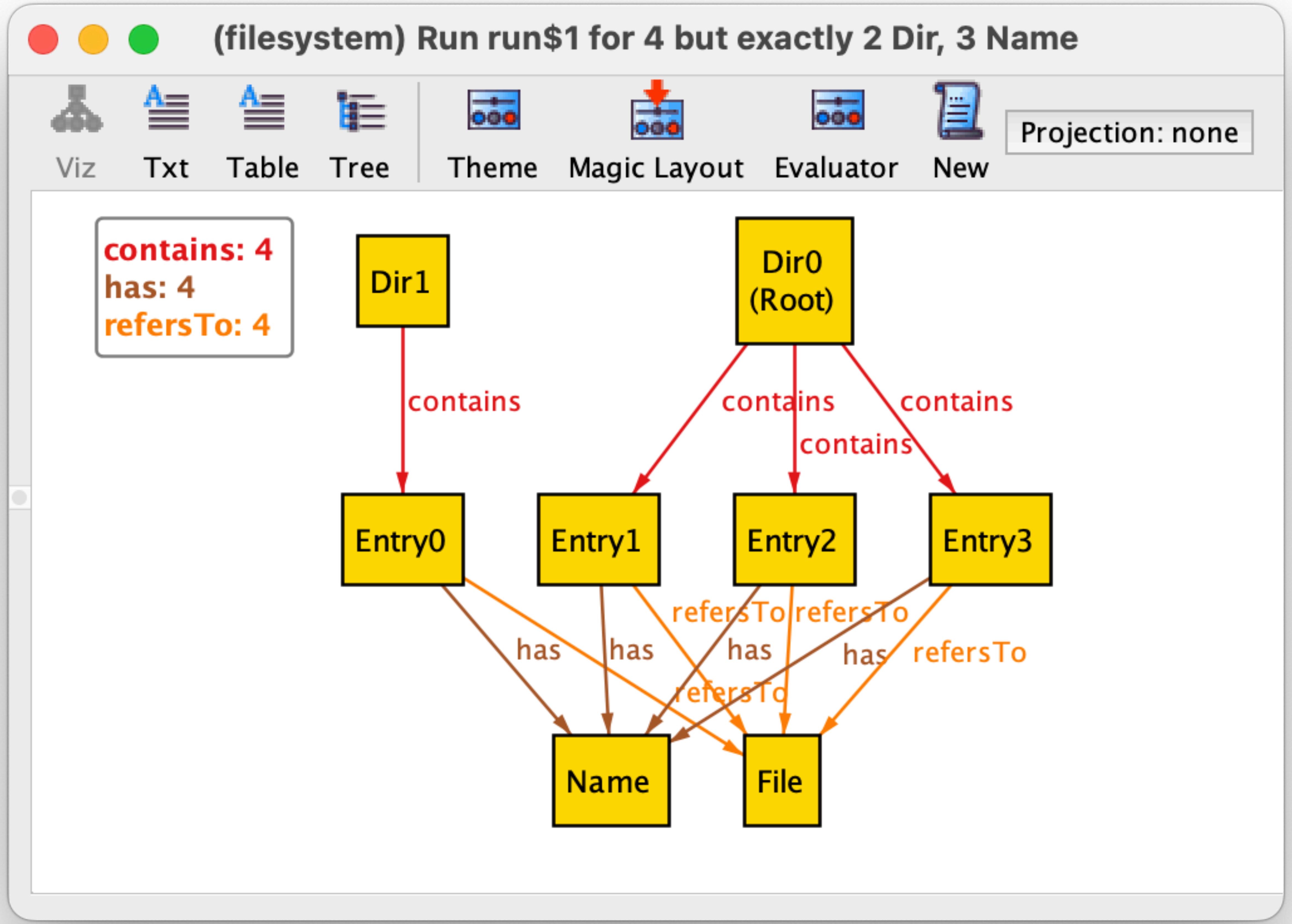
A simple command

```
run {} for 4 but exactly 2 Dir, 3 Name
```



Instances as graphs

Instances as graphs



Instances as relations

```
Object   = {(Dir0),(Dir1),(File)}
Dir      = {(Dir0),(Dir1)}
File     = {(File)}
Root     = {(Dir0)}
Entry    = {(Entry0),(Entry1),(Entry2),(Entry3)}
Name     = {(Name)}
contains = {(Dir1,Entry0),(Dir0,Entry1),(Dir0,Entry2),(Dir0,Entry3)}
refersTo = {(Entry0,File),(Entry1,File),(Entry2,File),(Entry3,File)}
has      = {(Entry0,Name),(Entry1,Name),(Entry2,Name),(Entry3,Name)}
```

Instances as tables

Object
Dir0
Dir1
File

Dir
Dir0
Dir1

Root
Dir0

File
File

Name
Name

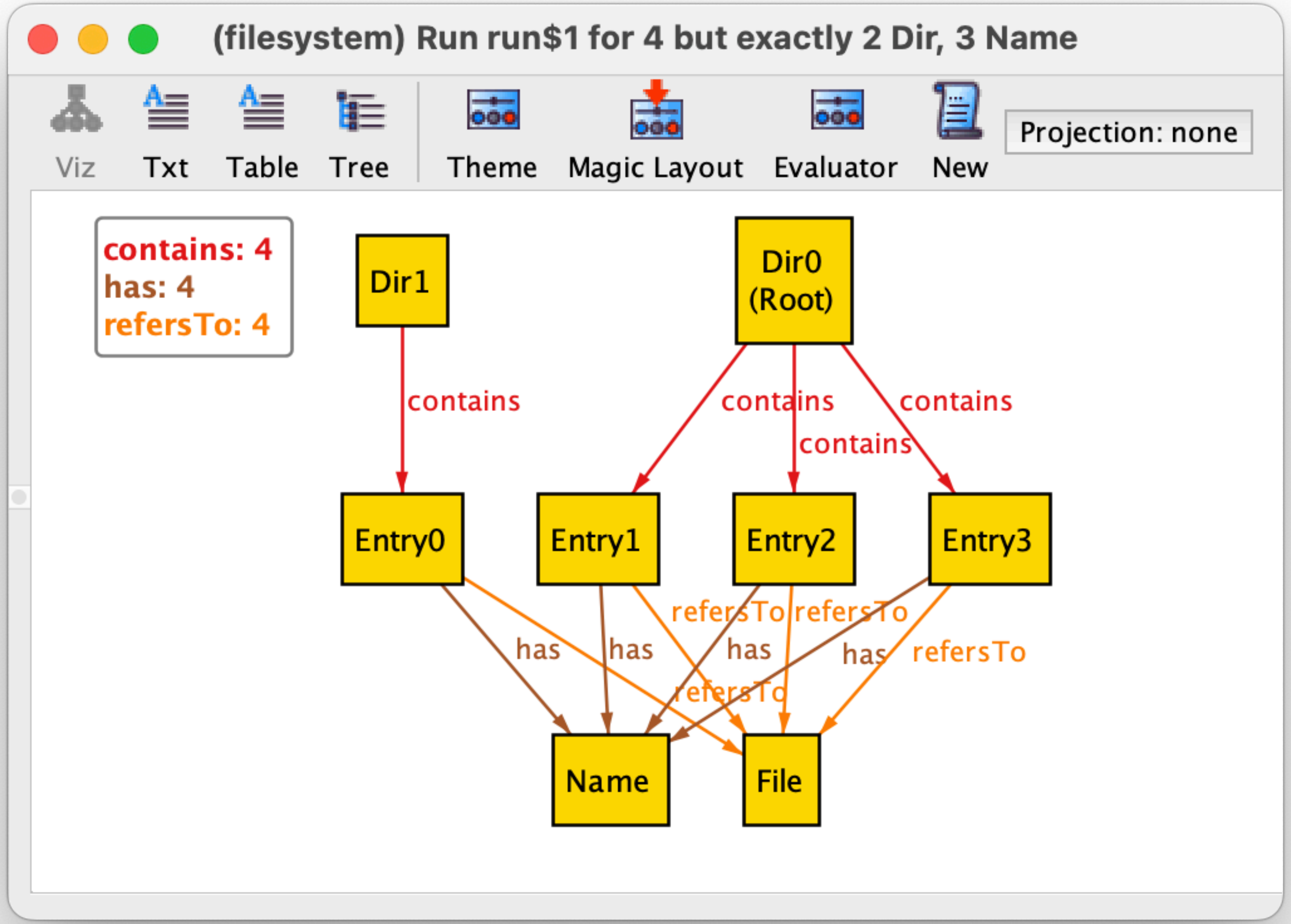
Entry
Entry0
Entry1
Entry2
Entry3

contains	
Dir1	Entry0
Dir0	Entry1
Dir0	Entry2
Dir0	Entry3

refersTo	
Entry0	File
Entry1	File
Entry2	File
Entry3	File

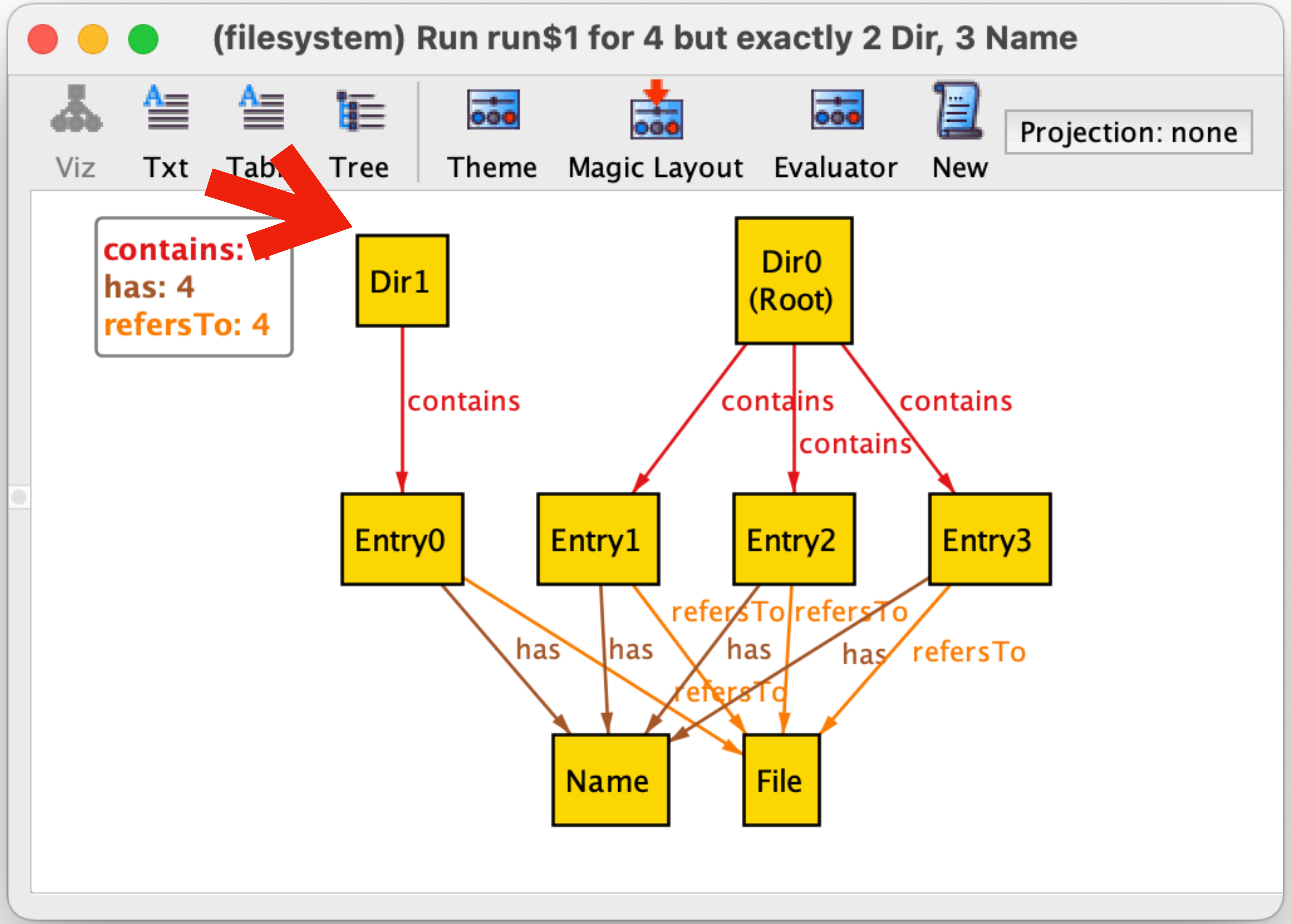
has	
Entry0	Name
Entry1	Name
Entry2	Name
Entry3	Name

Additional requirements



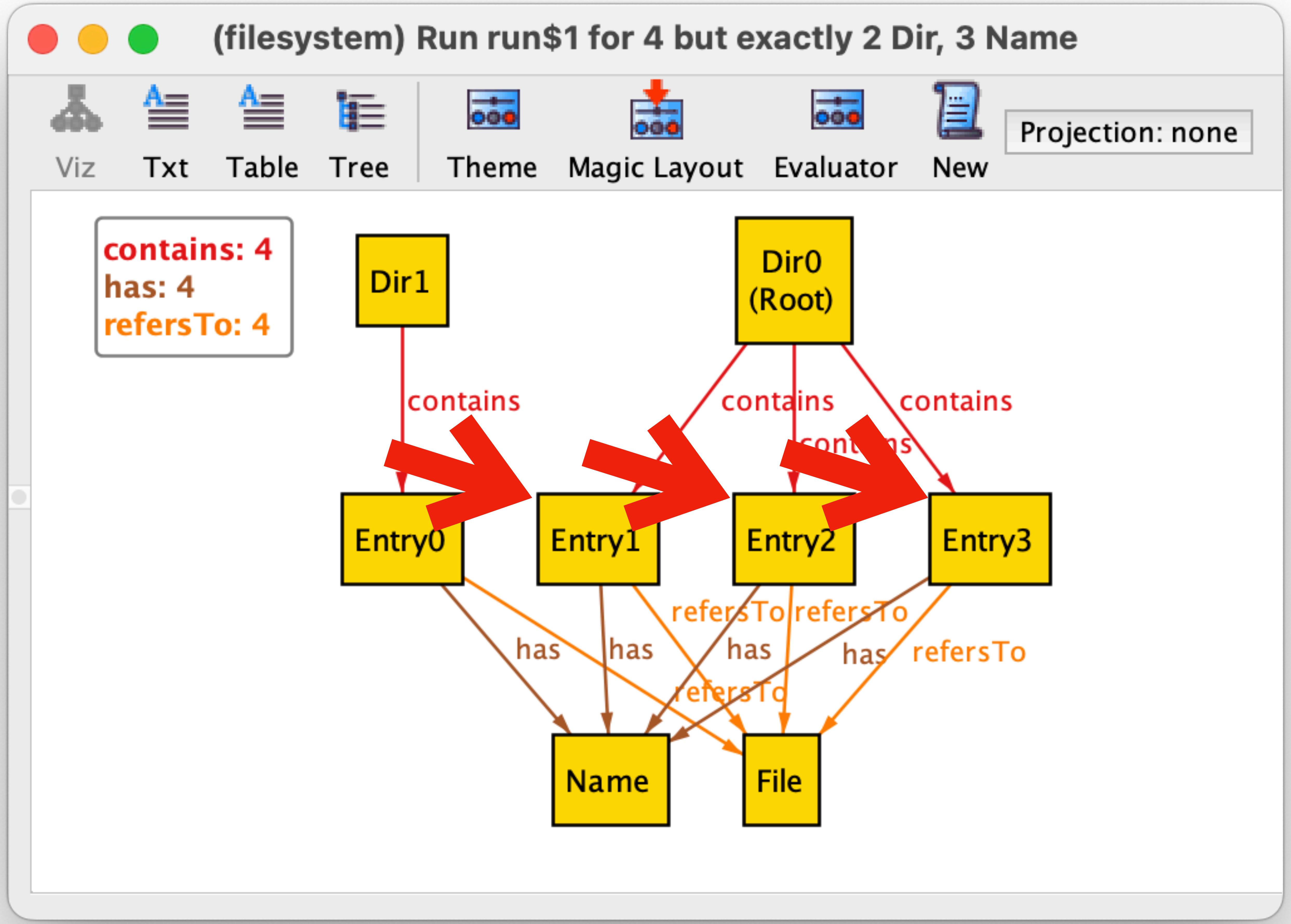
Additional requirements

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



Additional requirements

- All objects except the root are referred to in at least one entry (at most one for the case of directories)
- Different entries in a directory must have different names



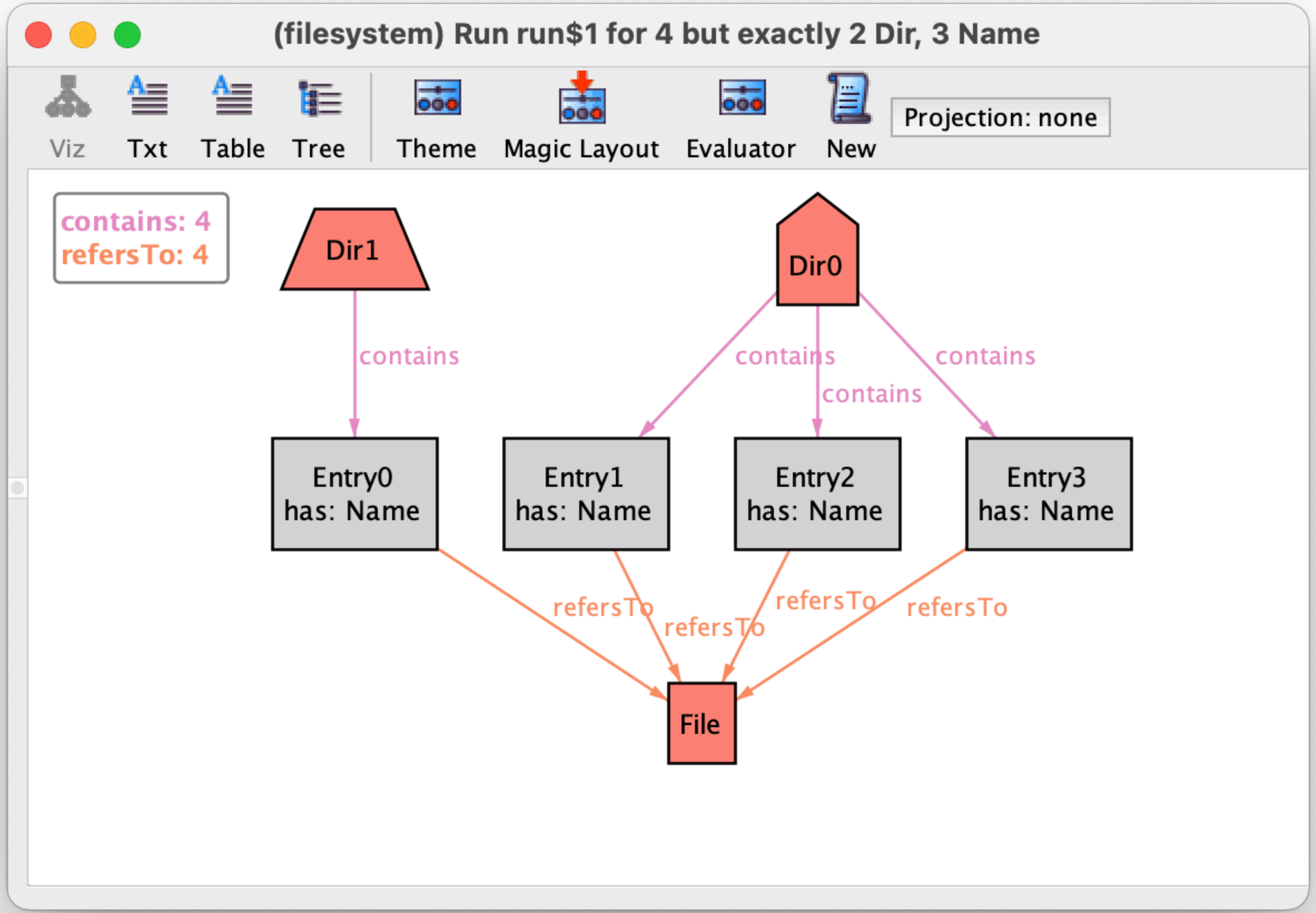
Themes

- The visualizer theme can be customized
- Customization can ease the understanding and help validate the model
- It is possible to customize colors, shapes, visibility, …

Theme customization

DEMO

Theme customization



Additional requirements

```
fact {
  // All directories are referred to in at most one entry
  all x : Dir, y,z : Entry | y->x in refersTo and z->x in refersTo implies y = z

  // The root is not referred in any entry
  all x : Entry, y : Root | x->y not in refersTo

  // All objects except the root are referred to in at least one entry
  all x : Object | x not in Root implies some y : Entry | y->x in refersTo

  // Different entries in a directory must have different names
  all x : Dir, y,z : Entry, w : Name {
    x->y in contains and x->z in contains and y->w in has and z->w in has implies y = z
  }
}
```

Relational logic

Relational logic

- *Relational logic* extends FOL
- Adds operators to combine predicates (relations) into *terms*
- Terms denote derived relations
- Adds transitive closure, which cannot be expressed in FOL

Syntax

$$x, y, z, \ldots \in \mathscr{X}$$
$$P, Q, R, \ldots \in \mathscr{P}$$
$$t, u, \ldots \in \mathbf{Term}_{\mathscr{P}}$$
$$\phi, \psi, \ldots \in \mathbf{Form}_{\mathscr{P}}$$

$$\begin{aligned}
\phi, \psi \doteq\ & t \subseteq u \\
&\mid \top \\
&\mid \bot \\
&\mid (\neg \phi) \\
&\mid (\phi \wedge \psi) \\
&\mid (\phi \vee \psi) \\
&\mid (\phi \to \psi) \\
&\mid (\phi \leftrightarrow \psi) \\
&\mid (\forall x \,.\, \phi) \\
&\mid (\exists x \,.\, \phi)
\end{aligned}$$

$$\begin{aligned}
t, u \doteq\ & x, y, z, \ldots \\
&\mid P, Q, R, \ldots \\
&\mid \varnothing \\
&\mid \mathbf{U} \\
&\mid \mathbf{id} \\
&\mid t \cup u \\
&\mid t \cap u \\
&\mid t \backslash u \\
&\mid t \times u \\
&\mid t \bullet u \\
&\mid t^{\circ} \\
&\mid t^{+}
\end{aligned}$$

Formula semantics

$\mathscr{M}, \mathscr{A} \models \top$		
$\mathscr{M}, \mathscr{A} \not\models \bot$		
$\mathscr{M}, \mathscr{A} \models t \subseteq u$	iff	$\mathscr{V}(t)$ is a subset or equal to $\mathscr{V}(u)$
$\mathscr{M}, \mathscr{A} \models \neg\phi$	iff	$\mathscr{M}, \mathscr{A} \not\models \phi$
$\mathscr{M}, \mathscr{A} \models \phi \wedge \psi$	iff	$\mathscr{M}, \mathscr{A} \models \phi$ and $\mathscr{M}, \mathscr{A} \models \psi$
$\mathscr{M}, \mathscr{A} \models \phi \vee \psi$	iff	$\mathscr{M}, \mathscr{A} \models \phi$ or $\mathscr{M}, \mathscr{A} \models \psi$
$\mathscr{M}, \mathscr{A} \models \phi \rightarrow \psi$	iff	$\mathscr{M}, \mathscr{A} \not\models \phi$ or $\mathscr{M}, \mathscr{A} \models \psi$
$\mathscr{M}, \mathscr{A} \models \phi \leftrightarrow \psi$	iff	$\mathscr{M}, \mathscr{A} \models \phi$ iff $\mathscr{M}, \mathscr{A} \models \psi$
$\mathscr{M}, \mathscr{A} \models \forall x \,.\, \phi$	iff	$\mathscr{M}, \mathscr{A}[x \mapsto a] \models \phi$ for all $a \in D$
$\mathscr{M}, \mathscr{A} \models \exists x \,.\, \phi$	iff	$\mathscr{M}, \mathscr{A}[x \mapsto a] \models \phi$ for some $a \in D$

Term semantics

$$
\begin{aligned}
\mathscr{V}(x) &\doteq \{(\mathscr{A}(x))\} \\
\mathscr{V}(R) &\doteq I(R) \\
\mathscr{V}(\varnothing) &\doteq \{\} \\
\mathscr{V}(\mathbf{U}) &\doteq \{(x) \mid x \in D\} \\
\mathscr{V}(\mathbf{id}) &\doteq \{(x, x) \mid x \in D\} \\
\mathscr{V}(t \cup u) &\doteq \{(x_1, \ldots, x_{|t|}) \mid (x_1, \ldots, x_{|t|}) \in \mathscr{V}(t) \vee (x_1, \ldots, x_{|t|}) \in \mathscr{V}(u)\} \\
\mathscr{V}(t \cap u) &\doteq \{(x_1, \ldots, x_{|t|}) \mid (x_1, \ldots, x_{|t|}) \in \mathscr{V}(t) \wedge (x_1, \ldots, x_{|t|}) \in \mathscr{V}(u)\} \\
\mathscr{V}(t \backslash u) &\doteq \{(x_1, \ldots, x_{|t|}) \mid (x_1, \ldots, x_{|t|}) \in \mathscr{V}(t) \wedge (x_1, \ldots, x_{|t|}) \notin \mathscr{V}(u)\} \\
\mathscr{V}(t \times u) &\doteq \{(x_1, \ldots, x_{|t|}, y_1, \ldots, y_{|u|}) \mid (x_1, \ldots, x_{|t|}) \in \mathscr{V}(t) \wedge (y_1, \ldots, y_{|u|}) \in \mathscr{V}(u)\} \\
\mathscr{V}(t \bullet u) &\doteq \{(x_1, \ldots, x_{|t|-1}, y_2, \ldots, y_{|u|}) \mid (x_1, \ldots, x_{|t|}) \in \mathscr{V}(t) \wedge (y_1, \ldots, y_{|u|}) \in \mathscr{V}(u) \wedge x_{|t|} = y_1\} \\
\mathscr{V}(t^\circ) &\doteq \{(x_1, \ldots, x_{|t|}) \mid (x_{|t|}, \ldots, x_1) \in \mathscr{V}(t)\} \\
\mathscr{V}(t^+) &\doteq \mathscr{V}(t \cup t \bullet t \cup t \bullet t \bullet t \cup \ldots)
\end{aligned}
$$

FOL vs RL

FOL vs RL

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

FOL vs RL

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

$\mathrm{bff} \subseteq \mathrm{friend}$

FOL vs RL

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

bff $\subseteq$ friend

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

FOL vs RL

$\forall x\,.\,\forall y\,.\,\mathrm{bff}(x, y) \rightarrow \mathrm{friend}(x, y)$	$\mathrm{bff} \subseteq \mathrm{friend}$
$\forall x\,.\,\forall y\,.\,\mathrm{friend}(x, y) \rightarrow \mathrm{friend}(y, x)$	$\mathrm{friend} \subseteq \mathrm{friend}^{\circ}$

FOL vs RL

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

bff $\subseteq$ friend

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

friend $\subseteq$ friend°

$\forall x \,.\, \neg \mathrm{friend}(x, x)$

FOL vs RL

$\forall x\,.\,\forall y\,.\,\mathrm{bff}(x, y) \to \mathrm{friend}(x, y)$	$\mathrm{bff} \subseteq \mathrm{friend}$
$\forall x\,.\,\forall y\,.\,\mathrm{friend}(x, y) \to \mathrm{friend}(y, x)$	$\mathrm{friend} \subseteq \mathrm{friend}^{\circ}$
$\forall x\,.\,\neg\mathrm{friend}(x, x)$	$\mathrm{friend} \cap \mathbf{id} \subseteq \varnothing$

FOL vs RL

$\forall x \,.\, \forall y \,.\, \mathrm{bff}(x, y) \rightarrow \mathrm{friend}(x, y)$	$\mathrm{bff} \subseteq \mathrm{friend}$
$\forall x \,.\, \forall y \,.\, \mathrm{friend}(x, y) \rightarrow \mathrm{friend}(y, x)$	$\mathrm{friend} \subseteq \mathrm{friend}^{\circ}$
$\forall x \,.\, \neg \mathrm{friend}(x, x)$	$\mathrm{friend} \cap \mathbf{id} \subseteq \emptyset$
$\forall x \,.\, \forall y \,.\, \mathrm{Ann}(x) \wedge \mathrm{Student}(y) \rightarrow \mathrm{friend}(x, y)$	

FOL vs RL

$$\forall x\,.\,\forall y\,.\,\mathrm{bff}(x,y) \to \mathrm{friend}(x,y) \qquad \mathrm{bff} \subseteq \mathrm{friend}$$

$$\forall x\,.\,\forall y\,.\,\mathrm{friend}(x,y) \to \mathrm{friend}(y,x) \qquad \mathrm{friend} \subseteq \mathrm{friend}^{\circ}$$

$$\forall x\,.\,\neg\mathrm{friend}(x,x) \qquad \mathrm{friend} \cap \mathbf{id} \subseteq \emptyset$$

$$\forall x\,.\,\forall y\,.\,\mathrm{Ann}(x) \wedge \mathrm{Student}(y) \to \mathrm{friend}(x,y) \qquad \mathrm{Ann} \times \mathrm{Student} \subseteq \mathrm{friend}$$

FOL vs RL

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

$\text{bff} \subseteq \text{friend}$

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

$\text{friend} \subseteq \text{friend}^\circ$

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

$\text{friend} \cap \mathbf{id} \subseteq \emptyset$

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

$\text{Ann} \times \text{Student} \subseteq \text{friend}$

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

FOL vs RL

$\forall x \,.\, \forall y \,.\, \text{bff}(x, y) \rightarrow \text{friend}(x, y)$	$\text{bff} \subseteq \text{friend}$
$\forall x \,.\, \forall y \,.\, \text{friend}(x, y) \rightarrow \text{friend}(y, x)$	$\text{friend} \subseteq \text{friend}^{\circ}$
$\forall x \,.\, \neg\text{friend}(x, x)$	$\text{friend} \cap \mathbf{id} \subseteq \emptyset$
$\forall x \,.\, \forall y \,.\, \text{Ann}(x) \wedge \text{Student}(y) \rightarrow \text{friend}(x, y)$	$\text{Ann} \times \text{Student} \subseteq \text{friend}$
$\forall x \,.\, \forall y \,.\, x \neq y \rightarrow \text{friend}(x, y)$	$(\mathbf{U} \times \mathbf{U}) \backslash \mathbf{id} \subseteq \text{friend}$

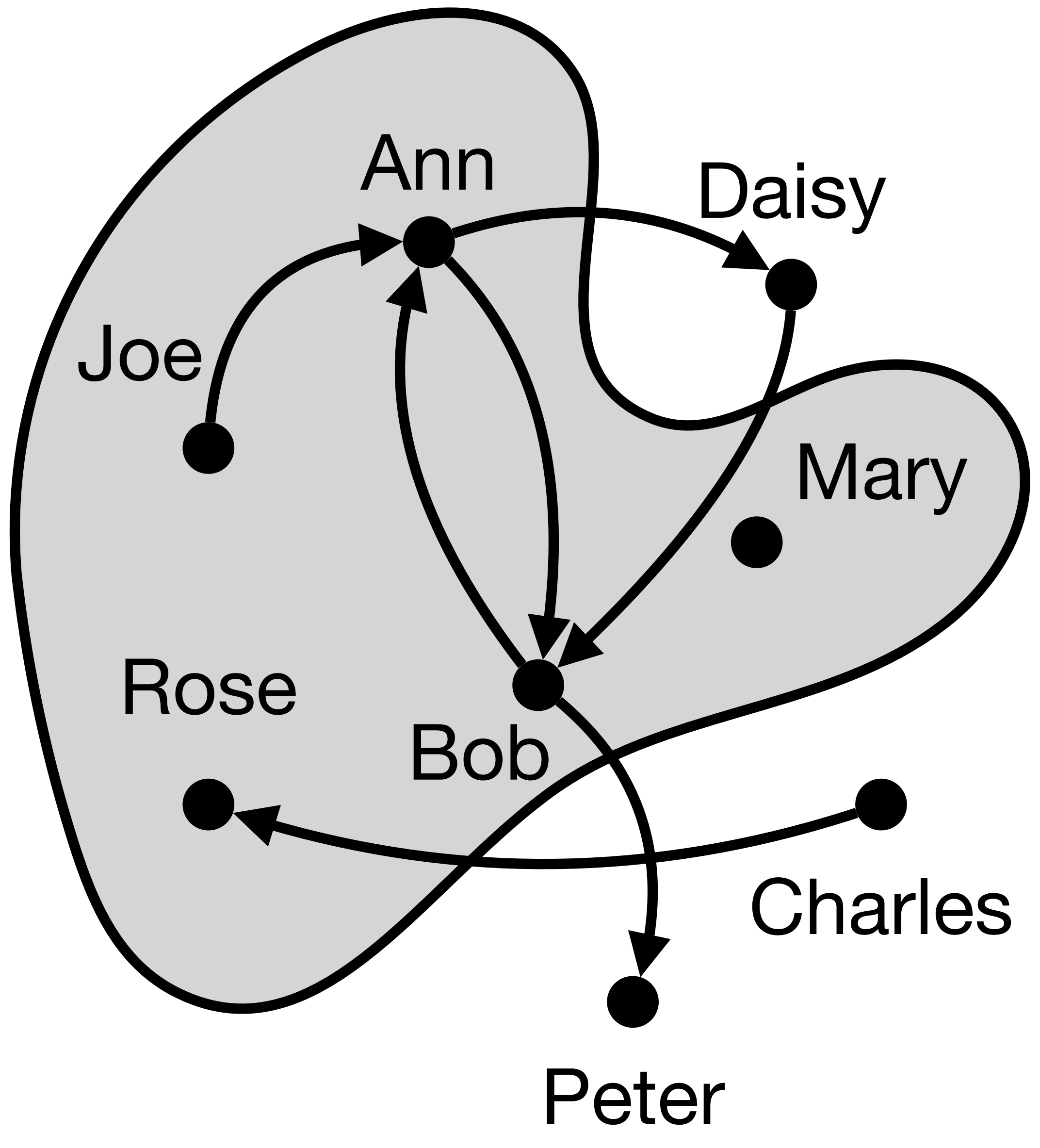
FOL vs RL

$\forall x \,.\, \forall y \,.\, \mathrm{bff}(x, y) \rightarrow \mathrm{friend}(x, y)$	$\mathrm{bff} \subseteq \mathrm{friend}$
$\forall x \,.\, \forall y \,.\, \mathrm{friend}(x, y) \rightarrow \mathrm{friend}(y, x)$	$\mathrm{friend} \subseteq \mathrm{friend}^{\circ}$
$\forall x \,.\, \neg \mathrm{friend}(x, x)$	$\mathrm{friend} \cap \mathbf{id} \subseteq \varnothing$
$\forall x \,.\, \forall y \,.\, \mathrm{Ann}(x) \land \mathrm{Student}(y) \rightarrow \mathrm{friend}(x, y)$	$\mathrm{Ann} \times \mathrm{Student} \subseteq \mathrm{friend}$
$\forall x \,.\, \forall y \,.\, x \neq y \rightarrow \mathrm{friend}(x, y)$	$(\mathbf{U} \times \mathbf{U}) \backslash \mathbf{id} \subseteq \mathrm{friend}$
$\forall x \,.\, \forall y \,.\, x \neq y \rightarrow \mathrm{friend}(x, y)$	

FOL vs RL

$\forall x . \forall y . \mathsf{bff}(x, y) \to \mathsf{friend}(x, y)$	$\mathsf{bff} \subseteq \mathsf{friend}$
$\forall x . \forall y . \mathsf{friend}(x, y) \to \mathsf{friend}(y, x)$	$\mathsf{friend} \subseteq \mathsf{friend}^{\circ}$
$\forall x . \neg \mathsf{friend}(x, x)$	$\mathsf{friend} \cap \mathbf{id} \subseteq \varnothing$
$\forall x . \forall y . \mathsf{Ann}(x) \wedge \mathsf{Student}(y) \to \mathsf{friend}(x, y)$	$\mathsf{Ann} \times \mathsf{Student} \subseteq \mathsf{friend}$
$\forall x . \forall y . x \neq y \to \mathsf{friend}(x, y)$	$(\mathbf{U} \times \mathbf{U}) \backslash \mathbf{id} \subseteq \mathsf{friend}$
$\forall x . \forall y . x \neq y \to \mathsf{friend}(x, y)$	$\forall x . \forall y . x \not\subseteq y \to x \times y \subseteq \mathsf{friend}$

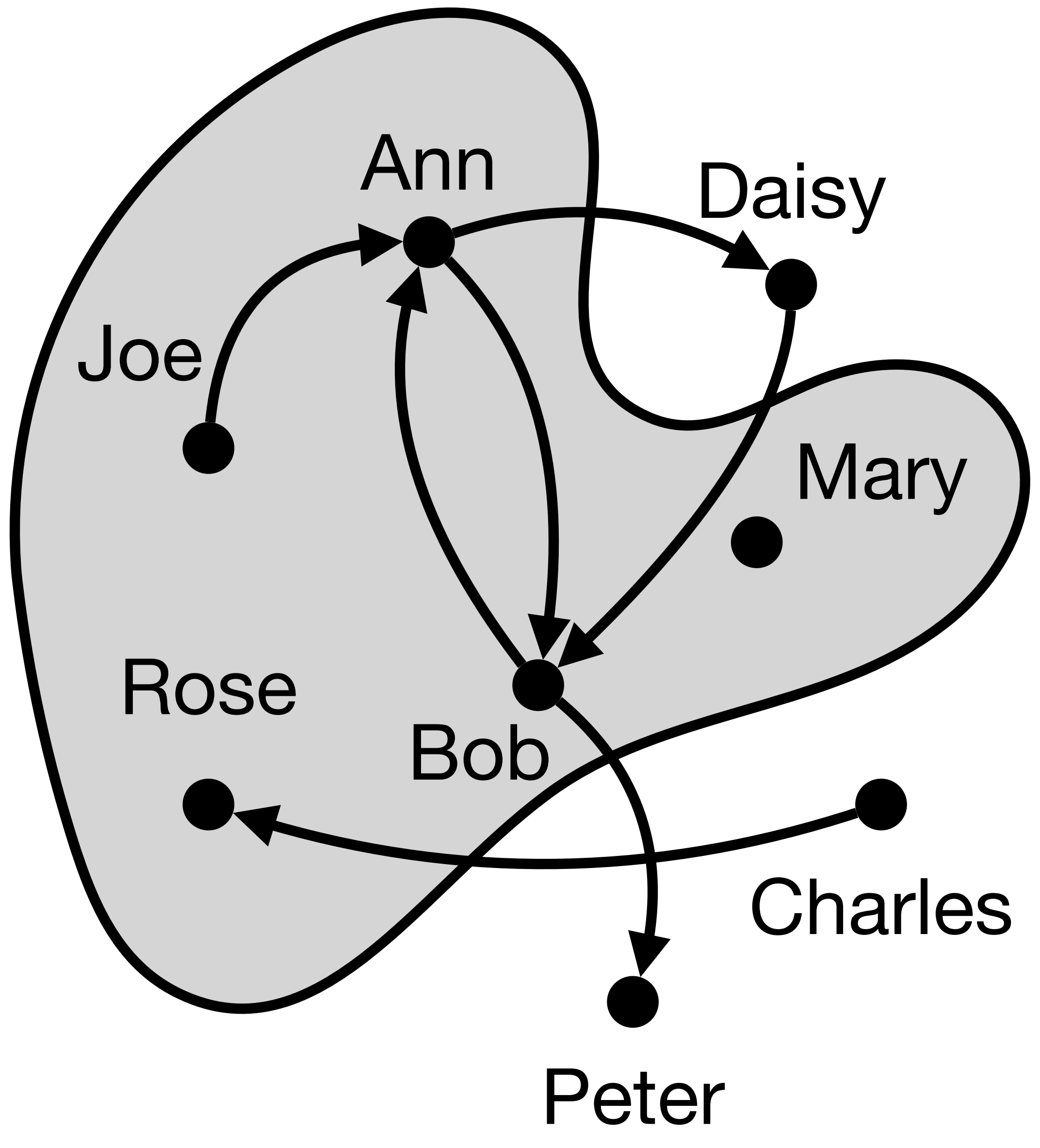
Composition



$$\text{Student} = \{(\text{A}), (\text{M}), (\text{B}), (\text{R}), (\text{J})\}$$

$$\text{friend} = \{(\text{J}, \text{A}), (\text{A}, \text{D}), (\text{A}, \text{B}), (\text{B}, \text{A}), (\text{B}, \text{P}), (\text{D}, \text{B}, (\text{C}, \text{R}))\}$$

Composition

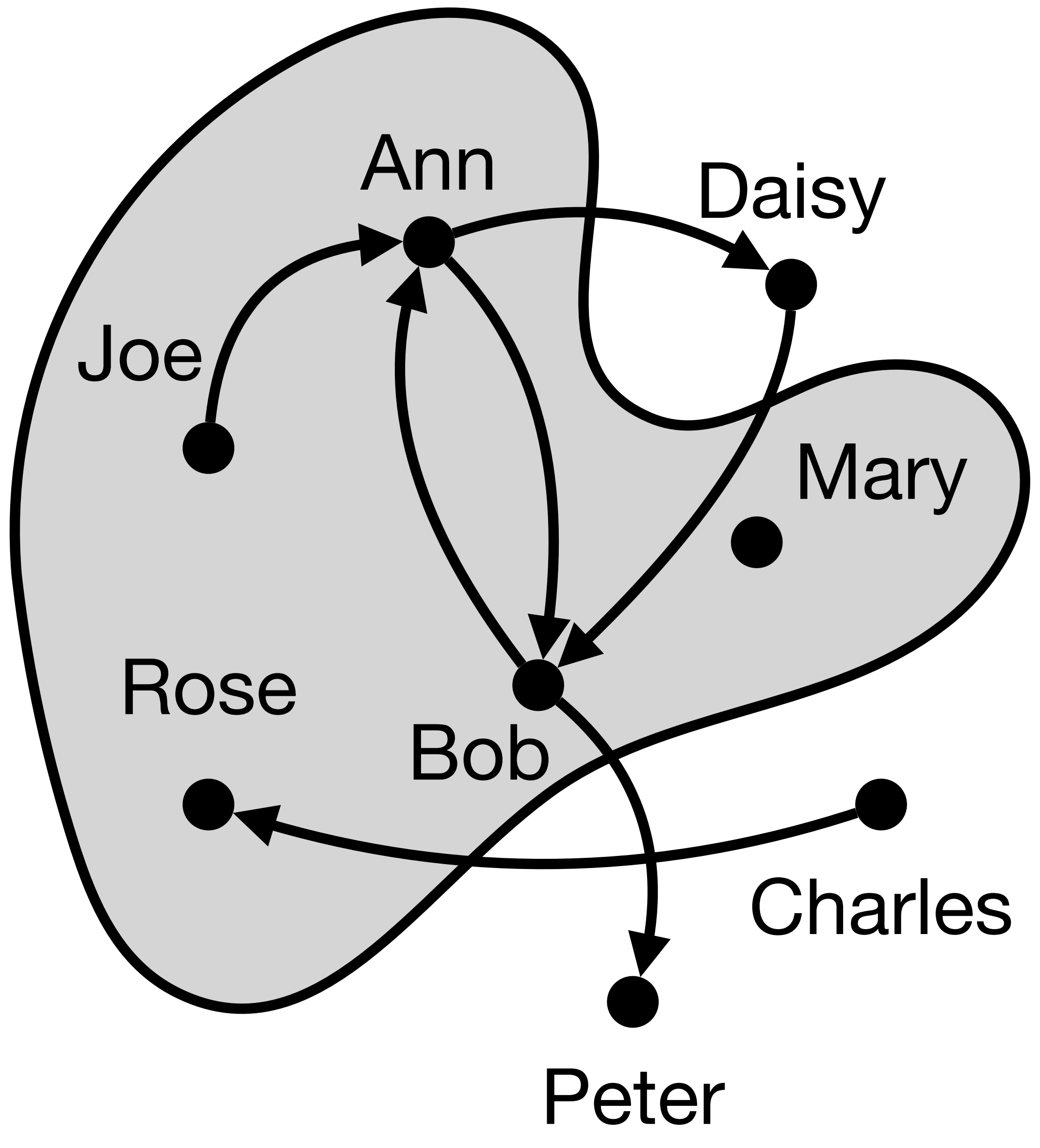


$$\text{Student} = \{(\text{A}), (\text{M}), (\text{B}), (\text{R}), (\text{J})\}$$

$$\text{friend} = \{(\text{J}, \text{A}), (\text{A}, \text{D}), (\text{A}, \text{B}), (\text{B}, \text{A}), (\text{B}, \text{P}), (\text{D}, \text{B}, (\text{C}, \text{R}))\}$$

$$\text{friend} \bullet \text{friend} = \{(\text{J}, \text{D}), (\text{J}, \text{B}), (\text{A}, \text{B}), (\text{A}, \text{P}), (\text{A}, \text{A}), (\text{B}, \text{B}), (\text{B}, \text{D}), (\text{D}, \text{A}), (\text{D}, \text{P})\}$$

Composition



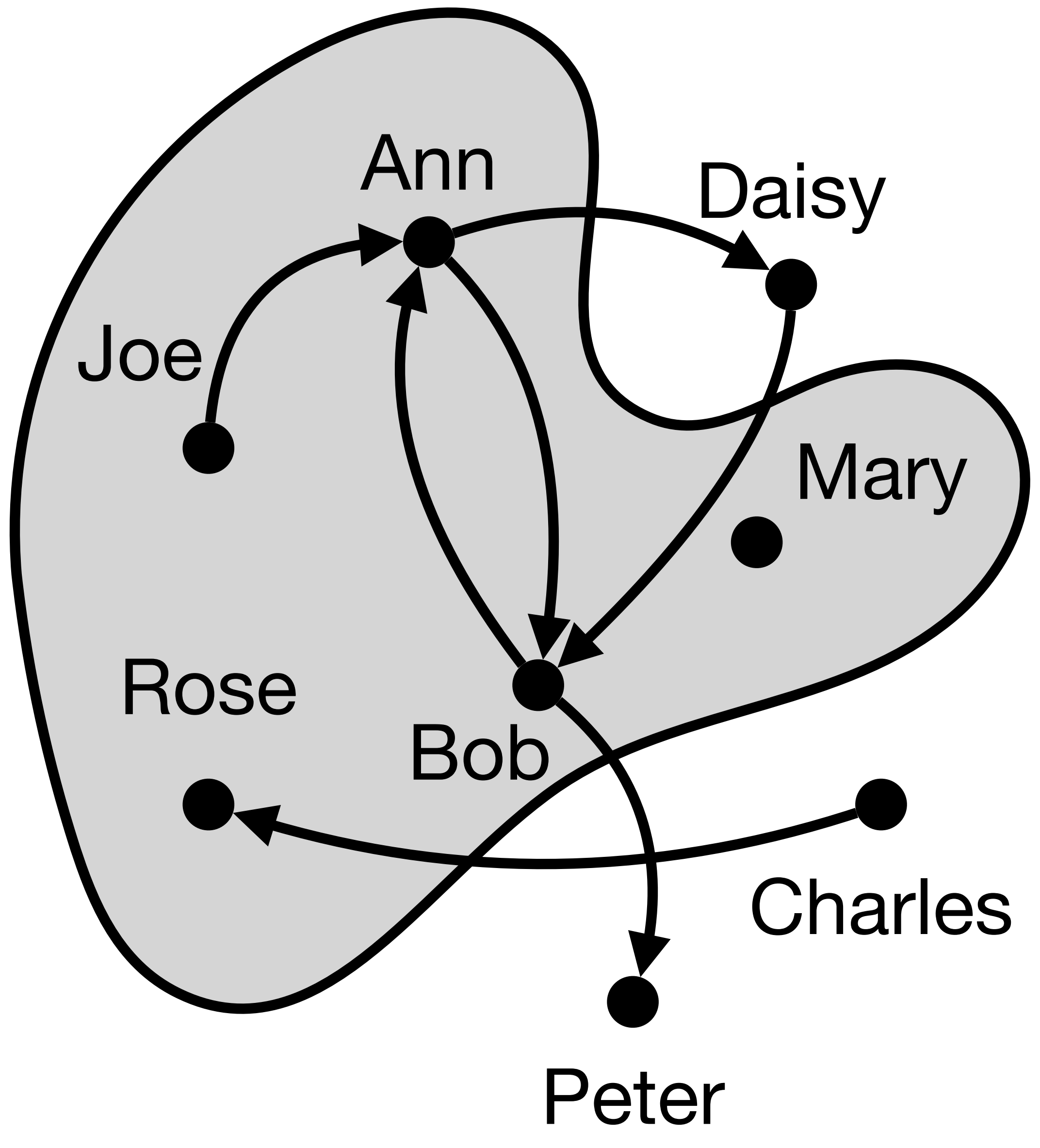
$$\text{Student} = \{(\mathrm{A}), (\mathrm{M}), (\mathrm{B}), (\mathrm{R}), (\mathrm{J})\}$$

$$\text{friend} = \{(\mathrm{J}, \mathrm{A}), (\mathrm{A}, \mathrm{D}), (\mathrm{A}, \mathrm{B}), (\mathrm{B}, \mathrm{A}), (\mathrm{B}, \mathrm{P}), (\mathrm{D}, \mathrm{B}, (\mathrm{C}, \mathrm{R}))\}$$

$$\text{friend} \bullet \text{friend} = \{(\mathrm{J}, \mathrm{D}), (\mathrm{J}, \mathrm{B}), (\mathrm{A}, \mathrm{B}), (\mathrm{A}, \mathrm{P}), (\mathrm{A}, \mathrm{A}), (\mathrm{B}, \mathrm{B}), (\mathrm{B}, \mathrm{D}), (\mathrm{D}, \mathrm{A}), (\mathrm{D}, \mathrm{P})\}$$

$$\text{Ann} \bullet \text{friend} = \{(\mathrm{B}), (\mathrm{D})\}$$

Composition



$$\text{Student} = \{(\text{A}), (\text{M}), (\text{B}), (\text{R}), (\text{J})\}$$

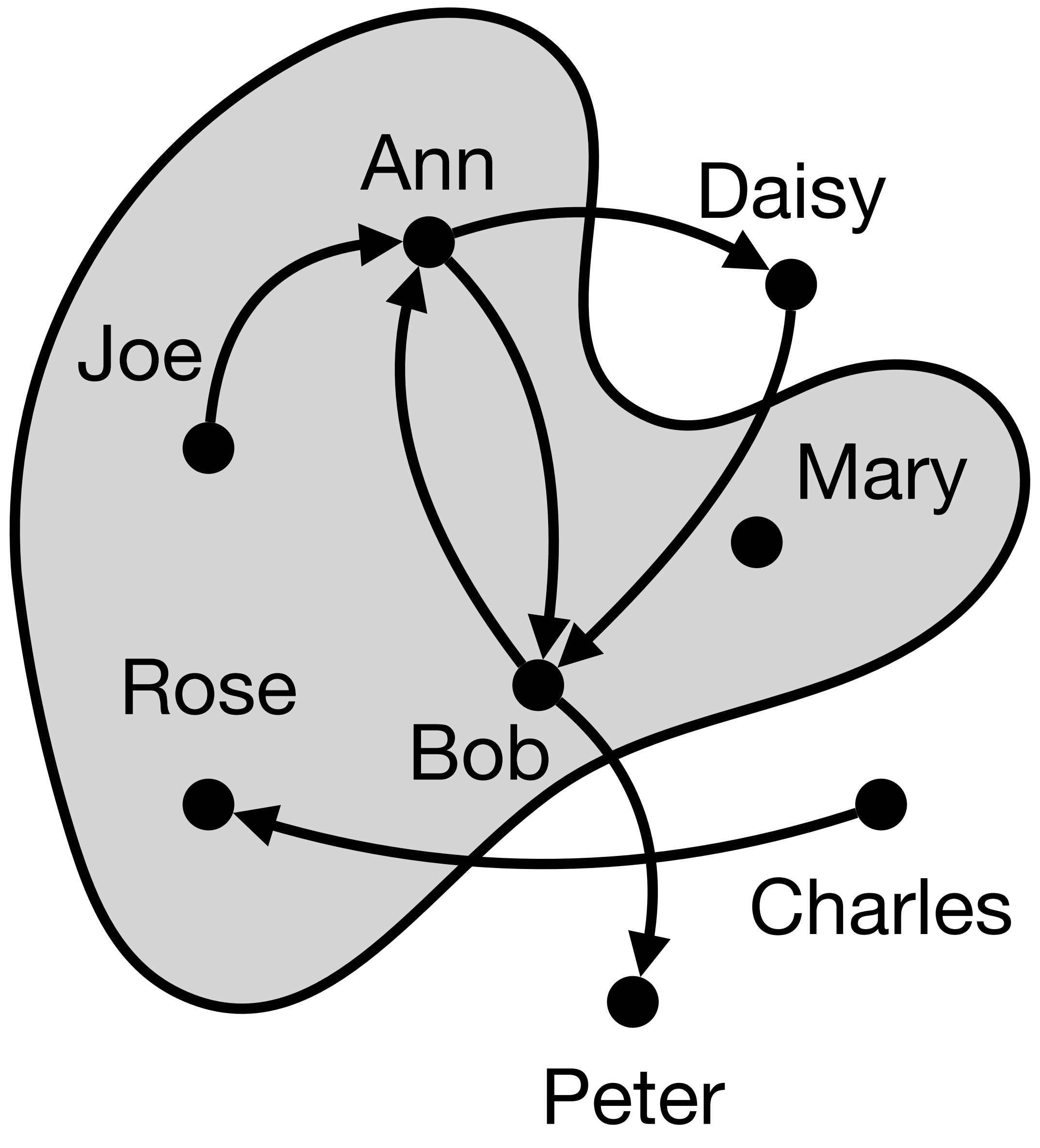
$$\text{friend} = \{(\text{J}, \text{A}), (\text{A}, \text{D}), (\text{A}, \text{B}), (\text{B}, \text{A}), (\text{B}, \text{P}), (\text{D}, \text{B}, (\text{C}, \text{R}))\}$$

$$\text{friend} \bullet \text{friend} = \{(\text{J}, \text{D}), (\text{J}, \text{B}), (\text{A}, \text{B}), (\text{A}, \text{P}), (\text{A}, \text{A}), (\text{B}, \text{B}), (\text{B}, \text{D}), (\text{D}, \text{A}), (\text{D}, \text{P})\}$$

$$\text{Ann} \bullet \text{friend} = \{(\text{B}), (\text{D})\}$$

$$\text{Ann} \bullet \text{friend} \bullet \text{friend} = \{(\text{A}), (\text{B}), (\text{P})\}$$

Composition



$$\text{Student} = \{(\text{A}), (\text{M}), (\text{B}), (\text{R}), (\text{J})\}$$

$$\text{friend} = \{(\text{J}, \text{A}), (\text{A}, \text{D}), (\text{A}, \text{B}), (\text{B}, \text{A}), (\text{B}, \text{P}), (\text{D}, \text{B}, (\text{C}, \text{R}))\}$$

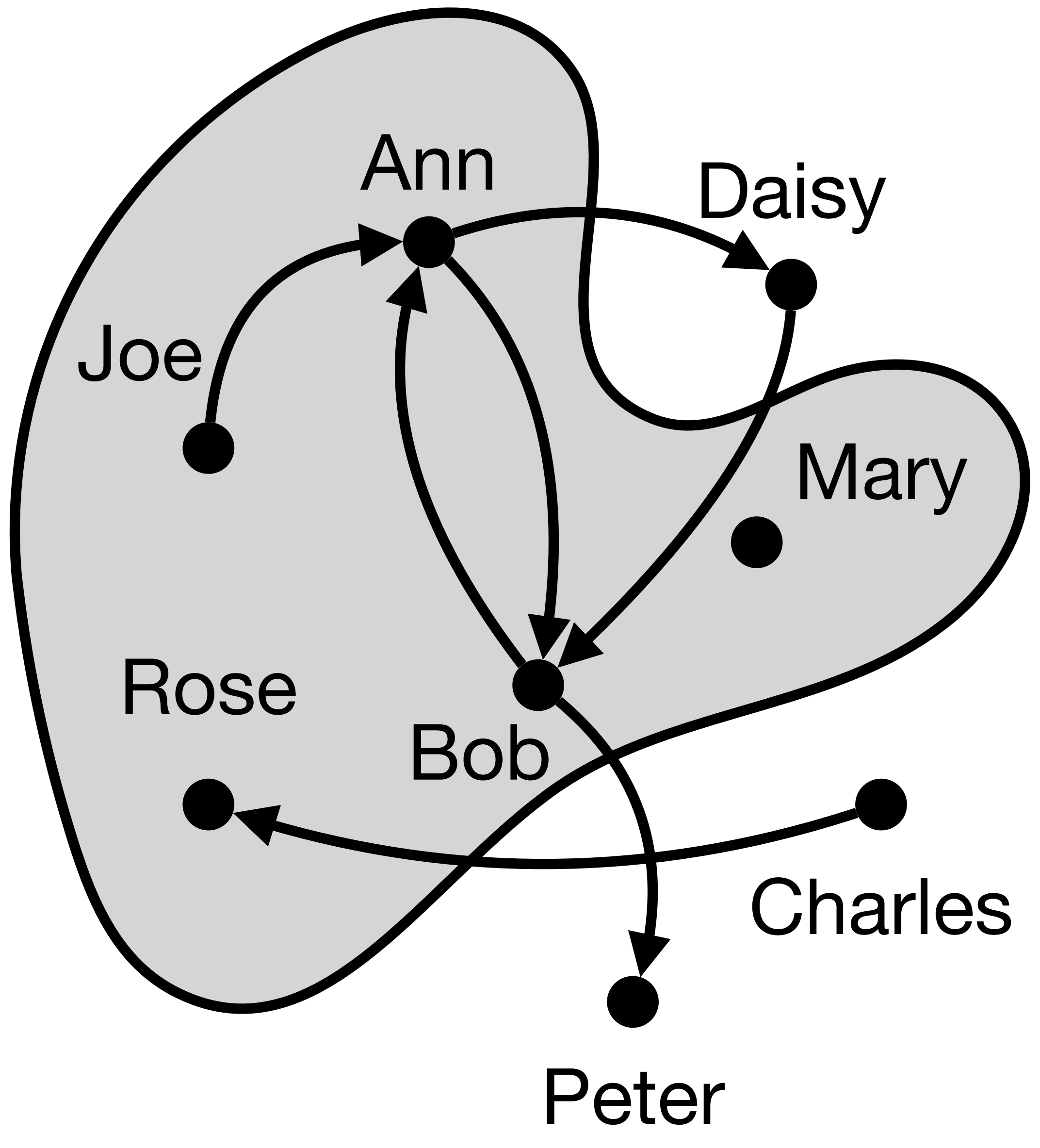
$$\text{friend} \bullet \text{friend} = \{(\text{J}, \text{D}), (\text{J}, \text{B}), (\text{A}, \text{B}), (\text{A}, \text{P}), (\text{A}, \text{A}), (\text{B}, \text{B}), (\text{B}, \text{D}), (\text{D}, \text{A}), (\text{D}, \text{P})\}$$

$$\text{Ann} \bullet \text{friend} = \{(\text{B}), (\text{D})\}$$

$$\text{Ann} \bullet \text{friend} \bullet \text{friend} = \{(\text{A}), (\text{B}), (\text{P})\}$$

$$\text{friend} \bullet \text{Ann} = \{(\text{J}), (\text{B})\}$$

Composition



$$\text{Student} = \{(\text{A}), (\text{M}), (\text{B}), (\text{R}), (\text{J})\}$$

$$\text{friend} = \{(\text{J}, \text{A}), (\text{A}, \text{D}), (\text{A}, \text{B}), (\text{B}, \text{A}), (\text{B}, \text{P}), (\text{D}, \text{B}, (\text{C}, \text{R}))\}$$

$$\text{friend} \bullet \text{friend} = \{(\text{J}, \text{D}), (\text{J}, \text{B}), (\text{A}, \text{B}), (\text{A}, \text{P}), (\text{A}, \text{A}), (\text{B}, \text{B}), (\text{B}, \text{D}), (\text{D}, \text{A}), (\text{D}, \text{P})\}$$

$$\text{Ann} \bullet \text{friend} = \{(\text{B}), (\text{D})\}$$

$$\text{Ann} \bullet \text{friend} \bullet \text{friend} = \{(\text{A}), (\text{B}), (\text{P})\}$$

$$\text{friend} \bullet \text{Ann} = \{(\text{J}), (\text{B})\}$$

$$\text{friend} \bullet \text{Student} = \{(\text{J}), (\text{A}), (\text{D}), (\text{B}), (\text{C})\}$$

FOL vs RL

FOL vs RL

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

FOL vs RL

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

$\mathbf{U} \subseteq \text{friend} \bullet \text{Student}$

FOL vs RL

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

$\mathbf{U} \subseteq \text{friend} \bullet \text{Student}$

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

FOL vs RL

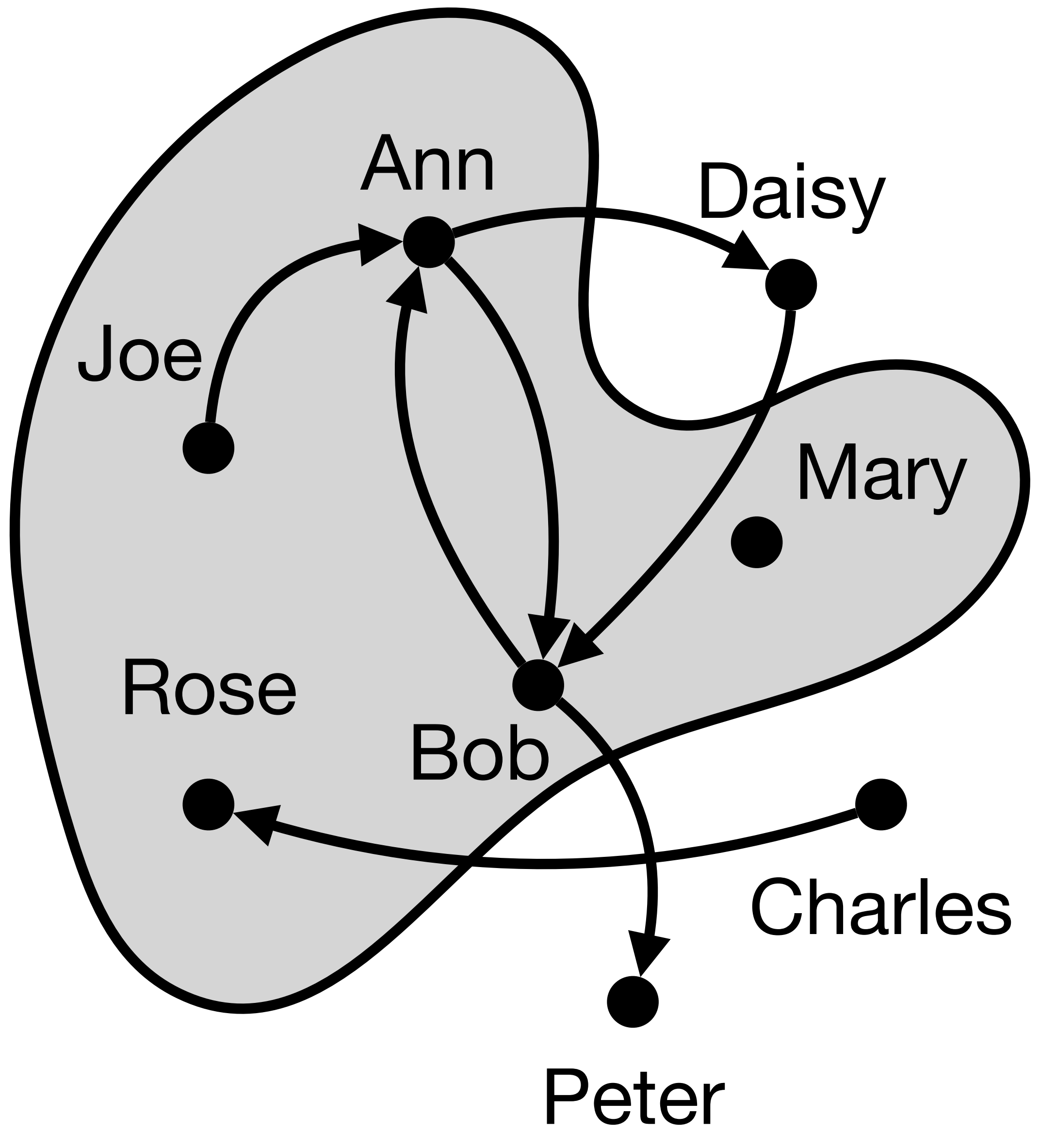
$\forall x \,.\, \exists y \,.\, \text{Student}(y) \land \text{friend}(x, y)$

$\mathbf{U} \subseteq \text{friend} \bullet \text{Student}$

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

$\text{friend} \bullet \text{friend} \subseteq \text{friend}$

Transitive closure



$$\mathrm{Student} = \{(\mathrm{A}), (\mathrm{M}), (\mathrm{B}), (\mathrm{R}), (\mathrm{J})\}$$

$$\mathrm{friend} = \{(\mathrm{J}, \mathrm{A}), (\mathrm{A}, \mathrm{D}), (\mathrm{A}, \mathrm{B}), (\mathrm{B}, \mathrm{A}), (\mathrm{B}, \mathrm{P}), (\mathrm{D}, \mathrm{B}), (\mathrm{C}, \mathrm{R})\}$$

$$\begin{aligned}\mathrm{friend}^+ = \{&(\mathrm{J}, \mathrm{A}), (\mathrm{J}, \mathrm{D}), (\mathrm{J}, \mathrm{B}), (\mathrm{J}, \mathrm{P}), (\mathrm{A}, \mathrm{D}), (\mathrm{A}, \mathrm{B}), (\mathrm{A}, \mathrm{P}), (\mathrm{A}, \mathrm{A}),\\ &(\mathrm{B}, \mathrm{A}), (\mathrm{B}, \mathrm{P}), (\mathrm{B}, \mathrm{D}), (\mathrm{B}, \mathrm{B}), (\mathrm{D}, \mathrm{B}), (\mathrm{D}, \mathrm{A}), (\mathrm{D}, \mathrm{D}), (\mathrm{D}, \mathrm{P}), (\mathrm{C}, \mathrm{R}))\}\end{aligned}$$

FOL $\neq$ RL

Ann is directly or indirectly a friend of everyone

$$\mathbf{U} \subseteq \text{Ann} \bullet \text{friend}^{+}$$

RL in Alloy

$\emptyset$	`none`
$\mathbf{U}$	`univ`
$\mathbf{id}$	`iden`
$t \subseteq u$	t `in` u
$t \cup u$	t `+` u
$t \cap u$	t `&` u
$t \setminus u$	t `-` u
$t \times u$	t `->` u
$t \bullet u$	t `.` u
t°	`~`t
t^{+}	`^`t

Syntactic sugar

t = *u*	*t* **in** *u* **and** *u* **in** *t*
t != *u*	**not** (*t* = *u*)
t **not in** *u*	**not** (*t* **in** *u*)
no *A*	*A* = **none**
some *A*	*A* != **none**
lone *A*	**all** *x*,*y* : *A* $\mid$ *x* = *y*
one *A*	**some** *A* **and** **lone** *A*
A <: *R*	*R* & (*A* -> **univ**)
R :> *A*	*R* & (**univ** -> *A*)
******R*	^*R* + **iden**
all disj *x*,*y* : *A* $\mid$ ϕ	**all** *x*,*y* : *A* $\mid$ *x* != *y* **implies** ϕ
some disj *x*,*y* : *A* $\mid$ ϕ	**some** *x*,*y* : *A* $\mid$ *x* != *y* **and** ϕ

FOL vs RL

```
fact {
  // Each entry is contained in one directory
  all x : Entry | some y : Dir | y->x in contains
  all x : Entry, y,z : Dir {
    y->x in contains and z->x in contains implies y = z
  }
}

fact {
  // Each entry is contained in one directory
  all x : Entry | one contains.x
}
```

FOL vs RL

```
fact {
  // All directories are referred to in at most one entry
  all x : Dir, y,z : Entry {
    y->x in refersTo and z->x in refersTo implies y = z
  }
}

fact {
  // All directories are referred to in at most one entry
  all x : Dir | lone refersTo.x
}
```

FOL vs RL

```
fact {
  // The root is not referred in any entry
  all x : Entry, y : Root | x->y not in refersTo
}

fact {
  // The root is not referred in any entry
  no refersTo.Root
}
```

FOL vs RL

```
fact {
  // All objects except the root are referred to in at least one entry
  all x : Object | x not in Root implies some y : Entry | y->x in refersTo
}

fact {
  // All objects except the root are referred to in at least one entry
  Object-Root in Entry.refersTo
}
```

FOL vs RL

```
fact {
  // Different entries in a directory must have different names
  all x : Dir, y,z : Entry, w : Name {
    x->y in contains and x->z in contains and y->w in has and z->w in has implies y = z
  }
}
```

```
fact {
  // Different entries in a directory must have different names
  all d : Dir, n : Name | lone (d.contains & has.n)
}
```

Everyone has different styles

Everyone has different styles

```
sig Person { style : one Style }
sig Style {}
```

Everyone has different styles

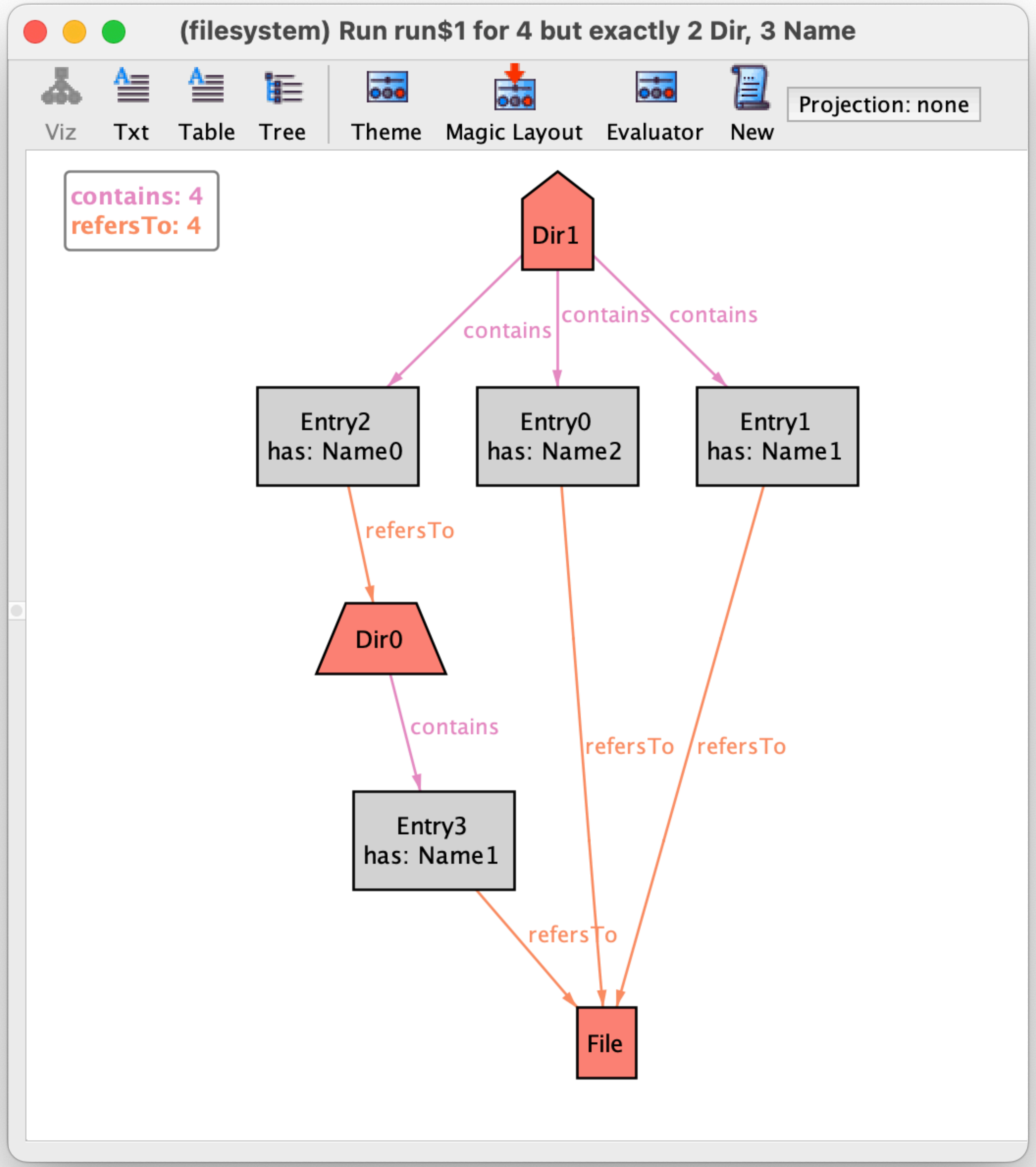
```
sig Person { style : one Style }
sig Style {}

// First order style
all x,y : Person, z : Style | x->z in style and y->z in style implies x=y
```

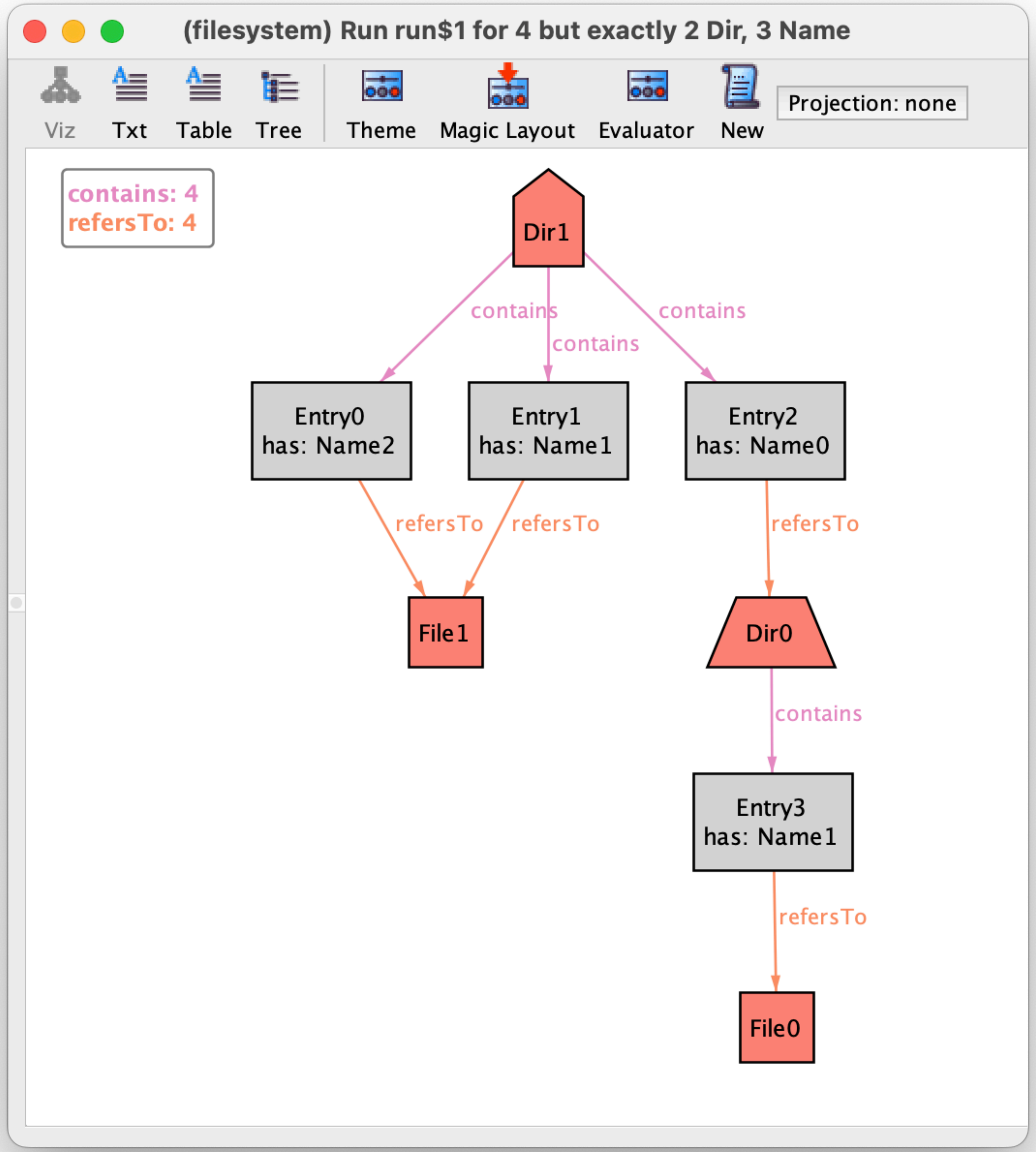
Everyone has different styles

```
sig Person { style : one Style }
sig Style {}

// First order style
all x,y : Person, z : Style | x->z in style and y->z in style implies x=y

// Relational or navigational style
all z : Style | lone style.z

// Point-free style
style.~style in iden
```

Verification

Some instances

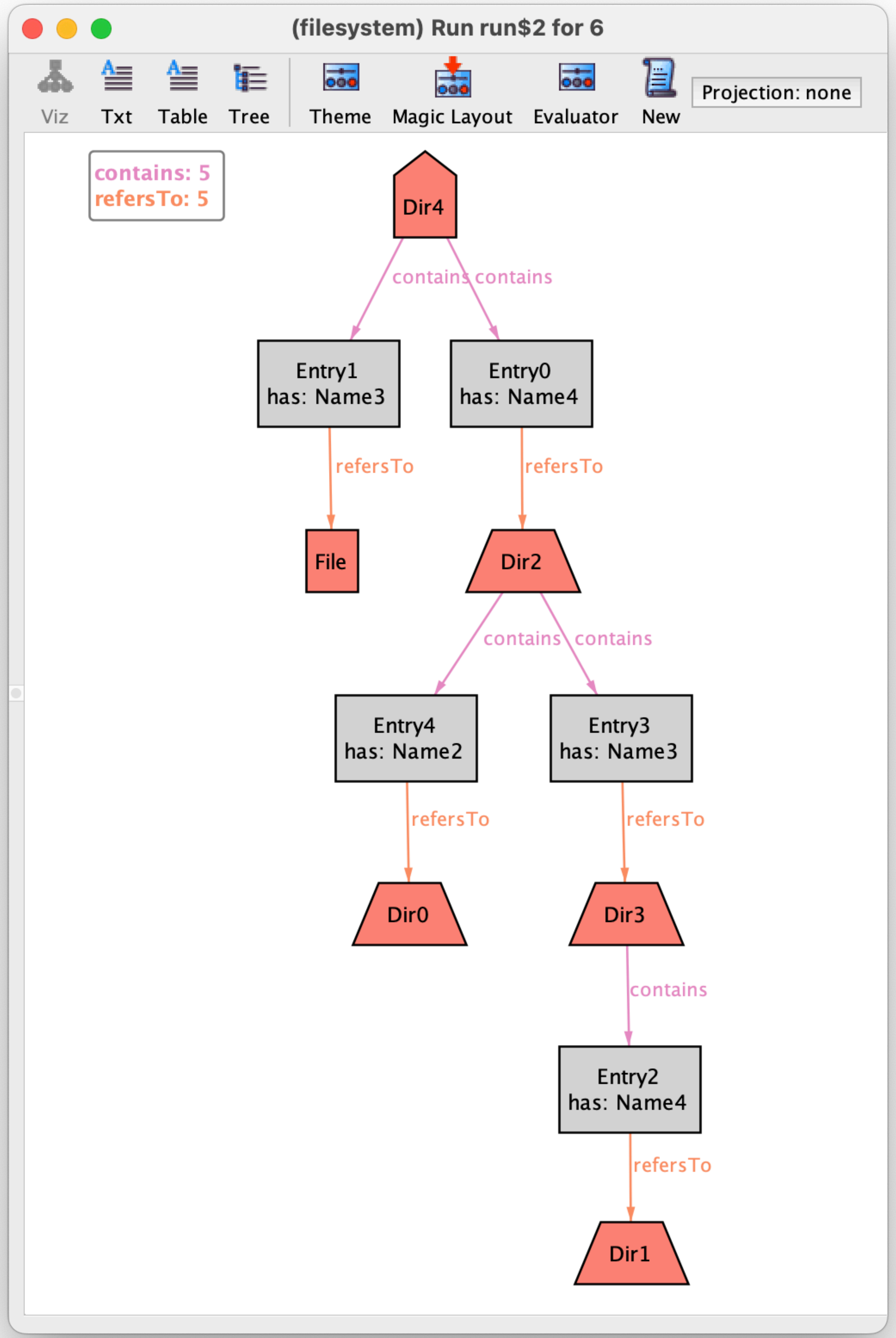




A desirable assertion

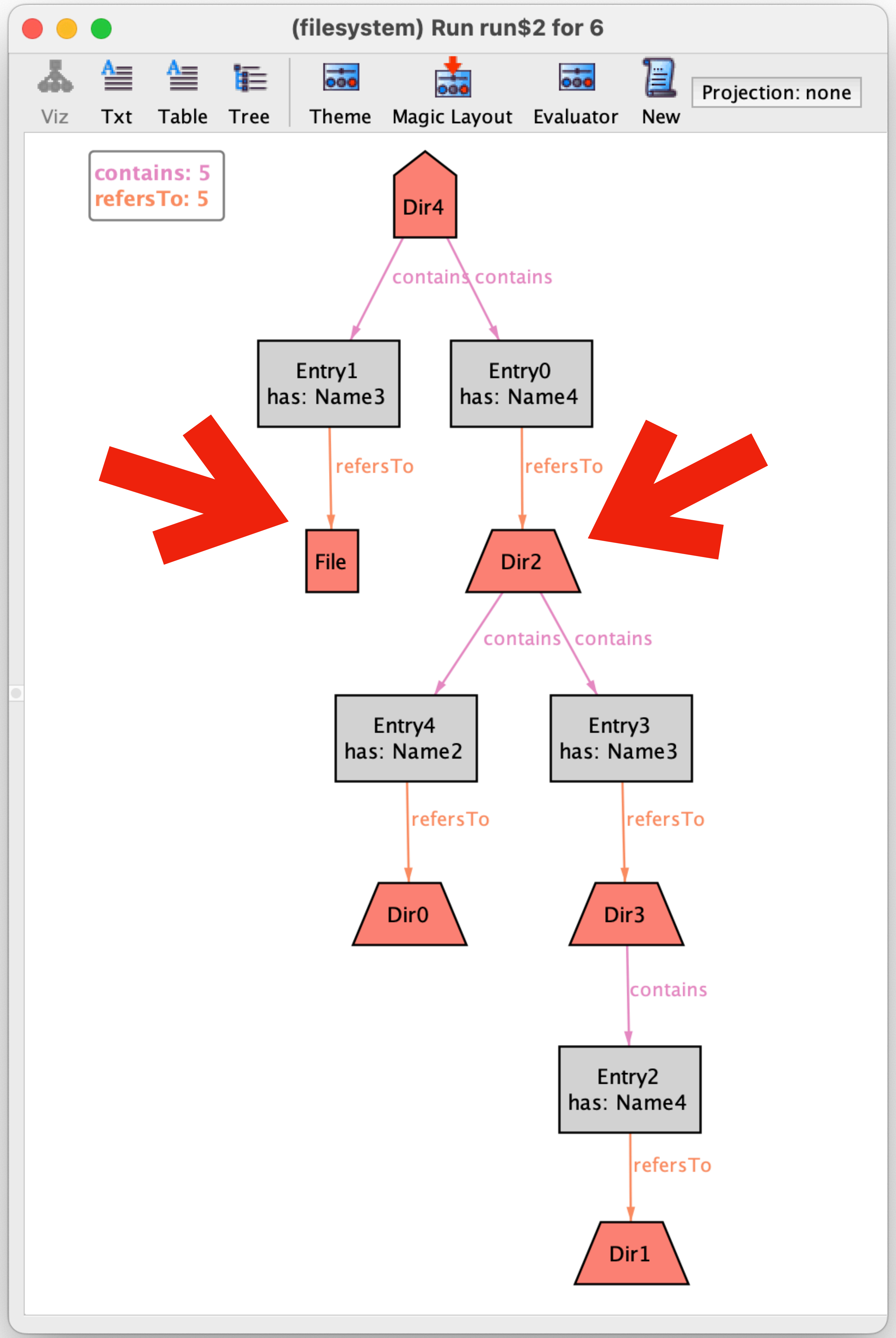
```
assert NoPartitions {
  // All objects are reachable from the root
  ???
}

check NoPartitions
```

Reachable objects



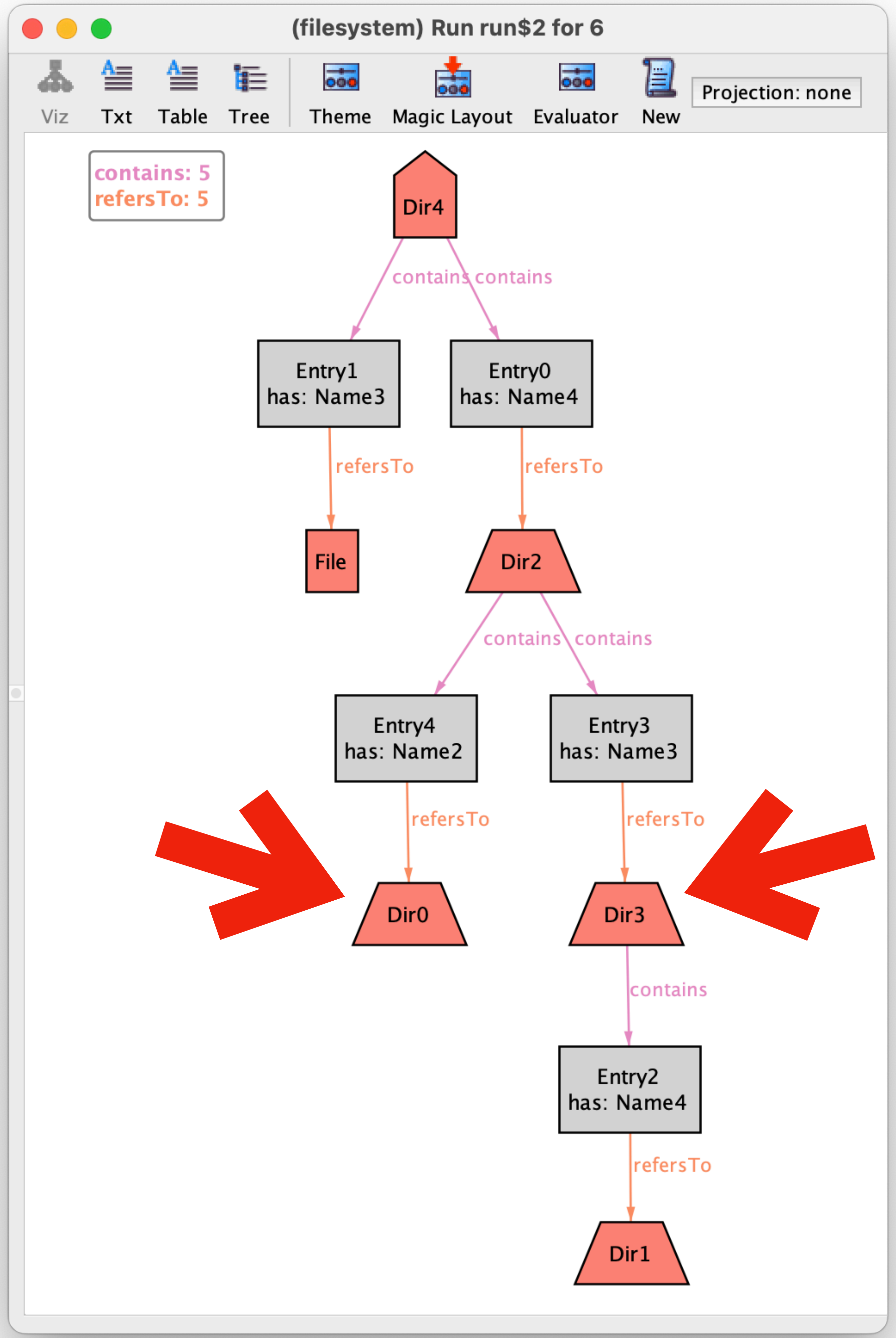
Reachable objects

`Root.contains.refersTo`



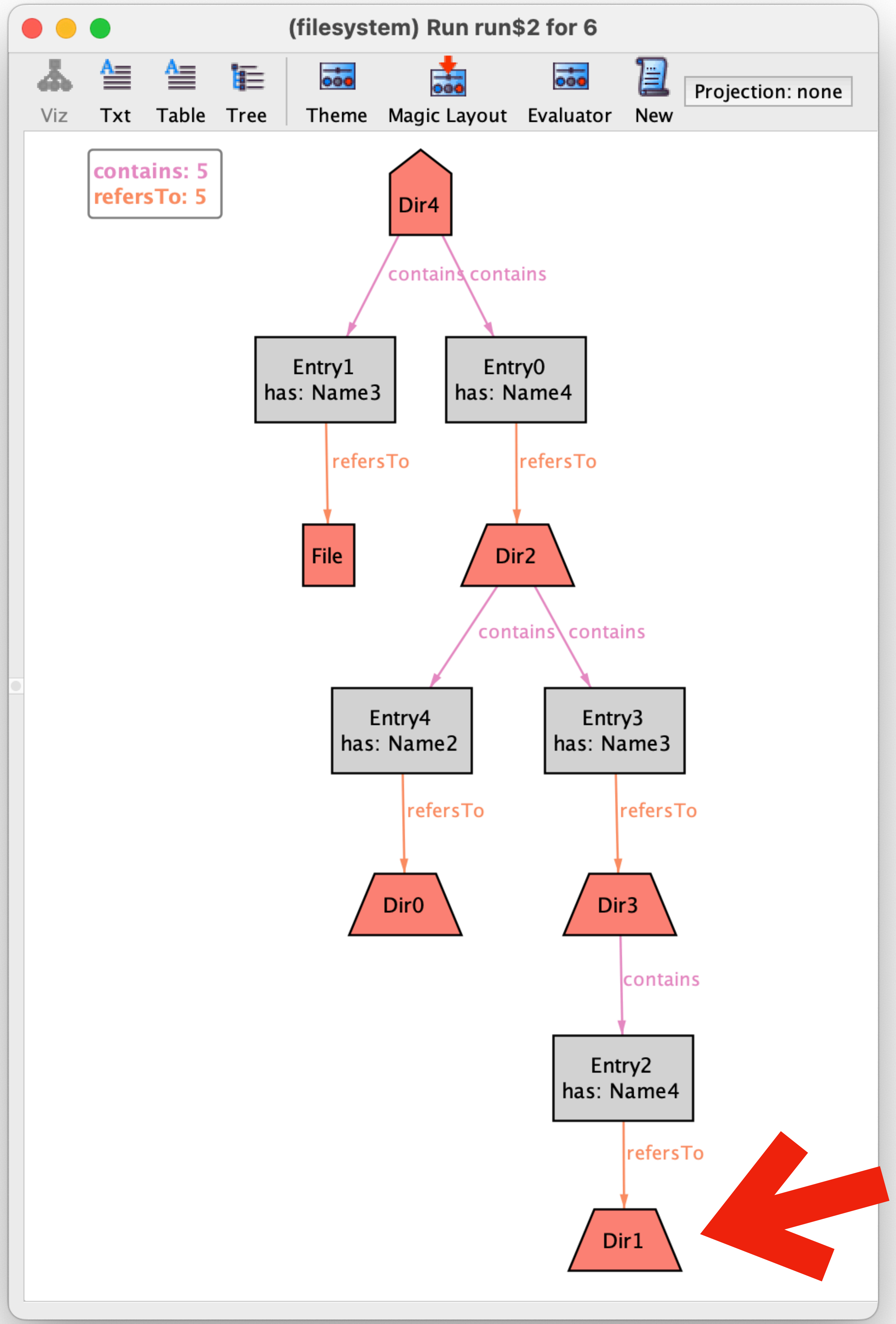
Reachable objects

`Root.contains.refersTo.contains.refersTo`



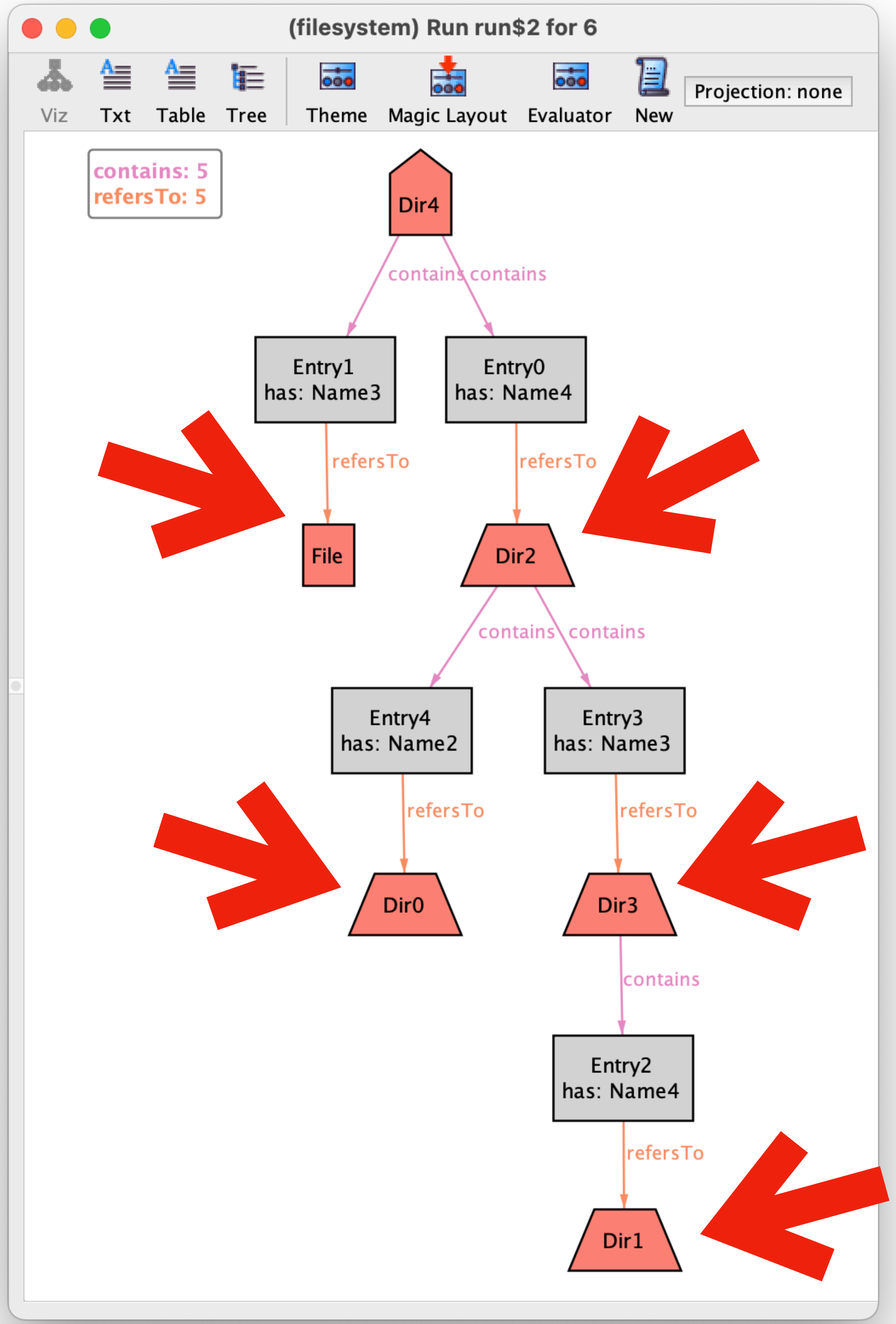
Reachable objects

`Root.contains.refersTo.contains.refersTo.contains.refersTo`



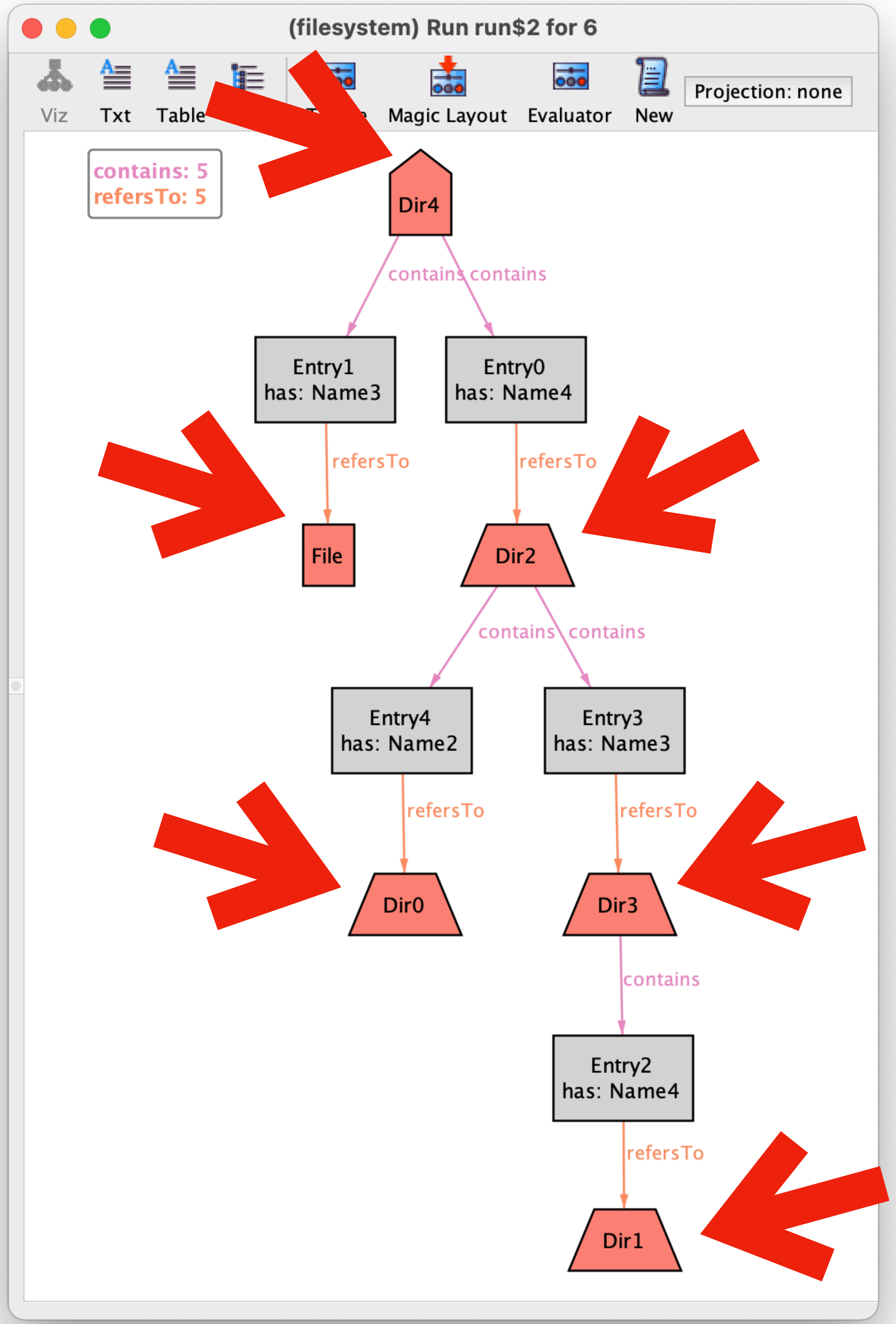
Reachable objects

`Root.^(contains.refersTo)`



Reachable objects

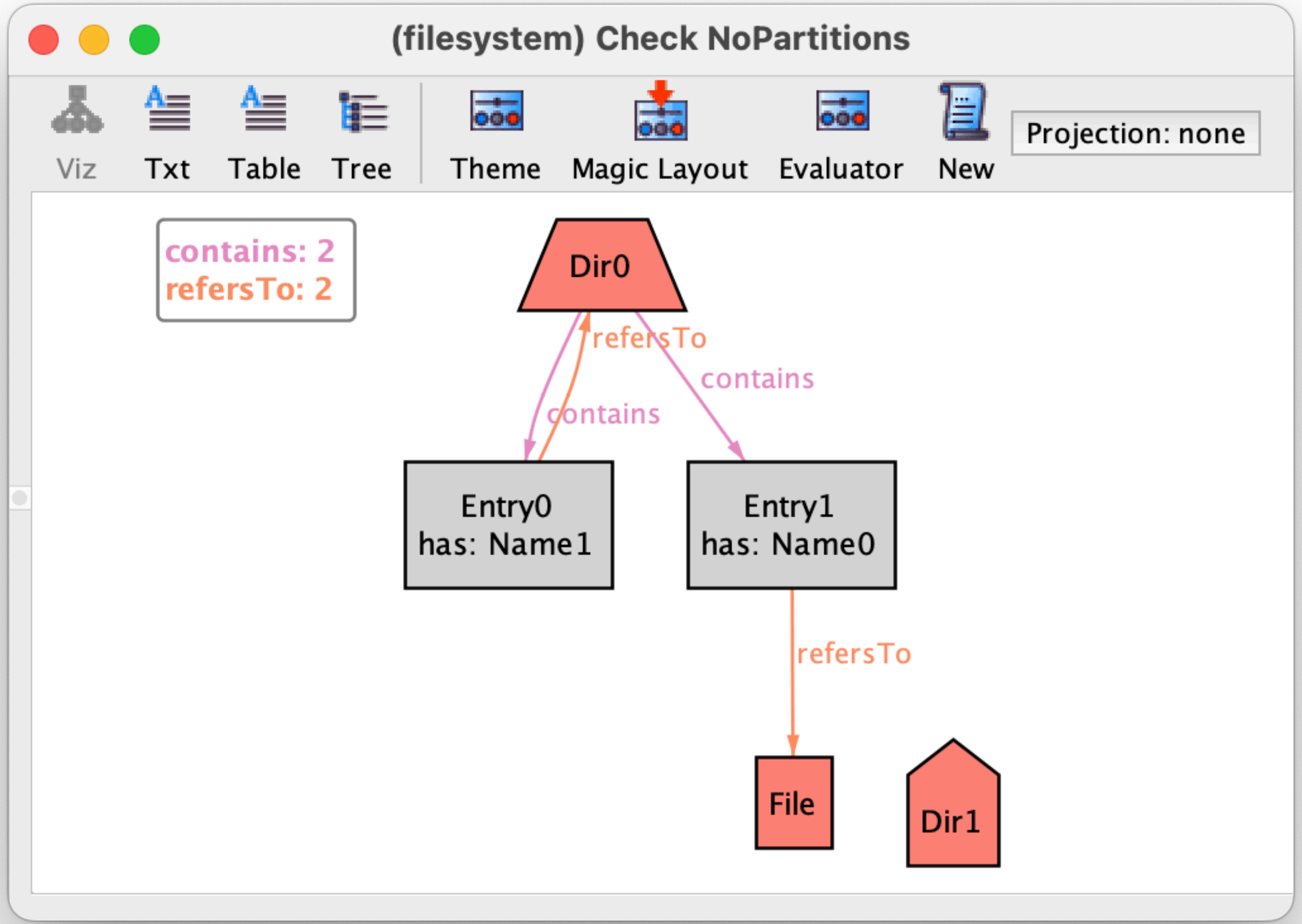
`Root.*(contains.refersTo)`



A desirable assertion

```
assert NoPartitions {
  // All objects are reachable from the root
  Object in Root.*(contains.refersTo)
}

check NoPartitions
```

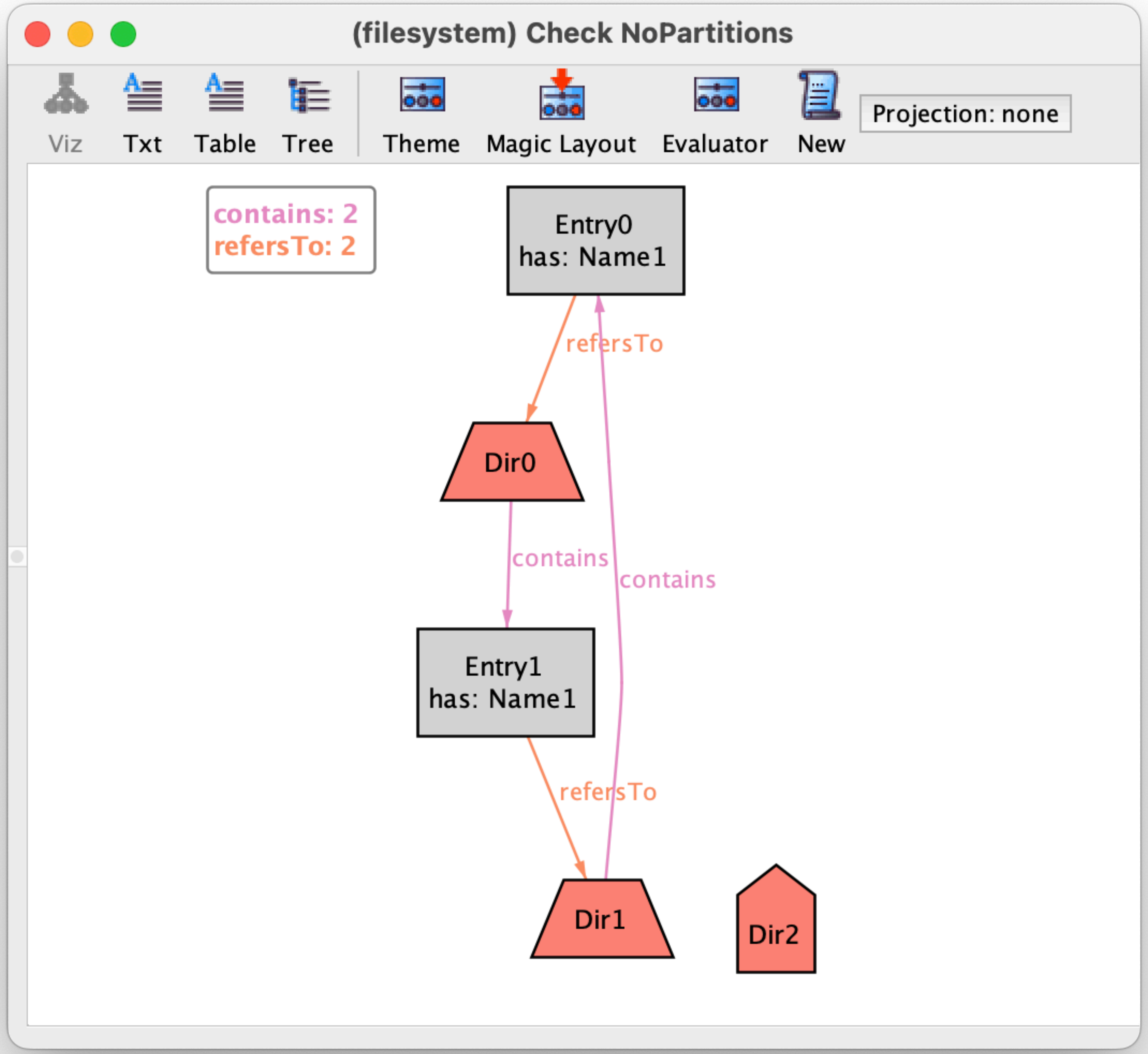
A counter-example



Missing requirement

```
fact {
  // A directory cannot be contained in itself
  all d : Dir | d not in d.contains.refersTo
}
```

Another counter-example



Missing requirement

```
fact {
  // A directory cannot be contained in itself
  all d : Dir | d not in d.^(contains.refersTo)
}
```

```
Executing "Check NoPartitions"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
   586 vars. 37 primary vars. 860 clauses. 3ms.
   No counterexample found. Assertion may be valid. 2ms.
```

Increasing confidence

- Increase the scope of **`check`** commands

```
check NoPartitions for 6
```

- Use **`run`** commands to validate the model
  - Verify that good scenarios are SAT
  - Verify that bad scenarios are UNSAT
  - Use **`expects`** keyword to document expectation

Specifying scenarios

```
run Scenario1 {
    // An empty file system
    Object = Root
} expect 1

run Scenario2 {
    // A file system with only two files with different names
    some disj f1,f2 : File, disj e1,e2 : Entry, disj n1,n2 : Name {
        contains = Root->e1 + Root->e2
        refersTo = e1->f1 + e2->f2
        has      = e1->n1 + e2->n2
    }
} expect 1

run Scenario3 {
    // A file system with only two files with the same name
    some disj f1,f2 : File, disj e1,e2 : Entry, n : Name {
        contains = Root->e1 + Root->e2
        refersTo = e1->f1 + e2->f2
        has      = e1->n  + e2->n
    }
} expect 0
```