Mathematical Knowledge Representation and Reasoning in the Wolfram Language

Ian Ford
Wolfram Research Inc.

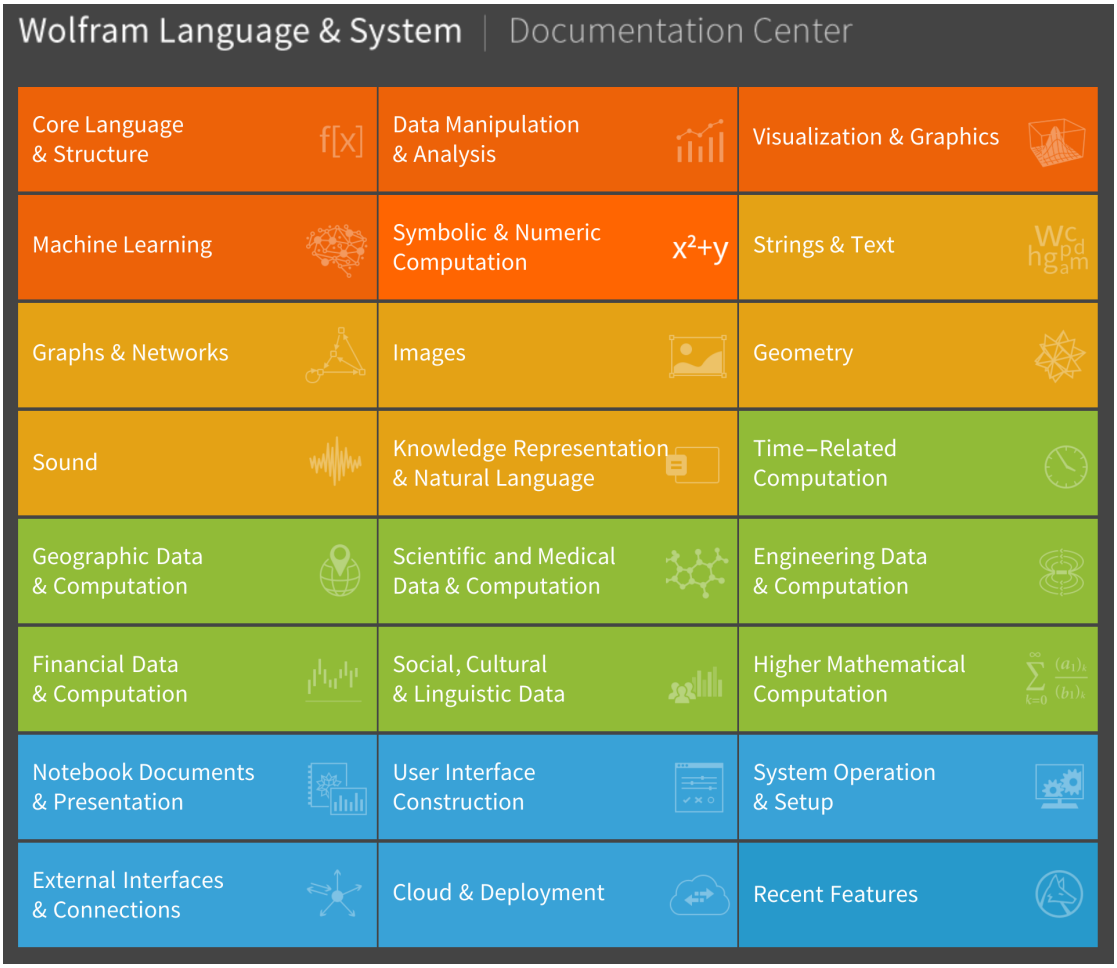
What is the Wolfram Language?

It is:

- a symbolic, general-purpose computational language

It is not:

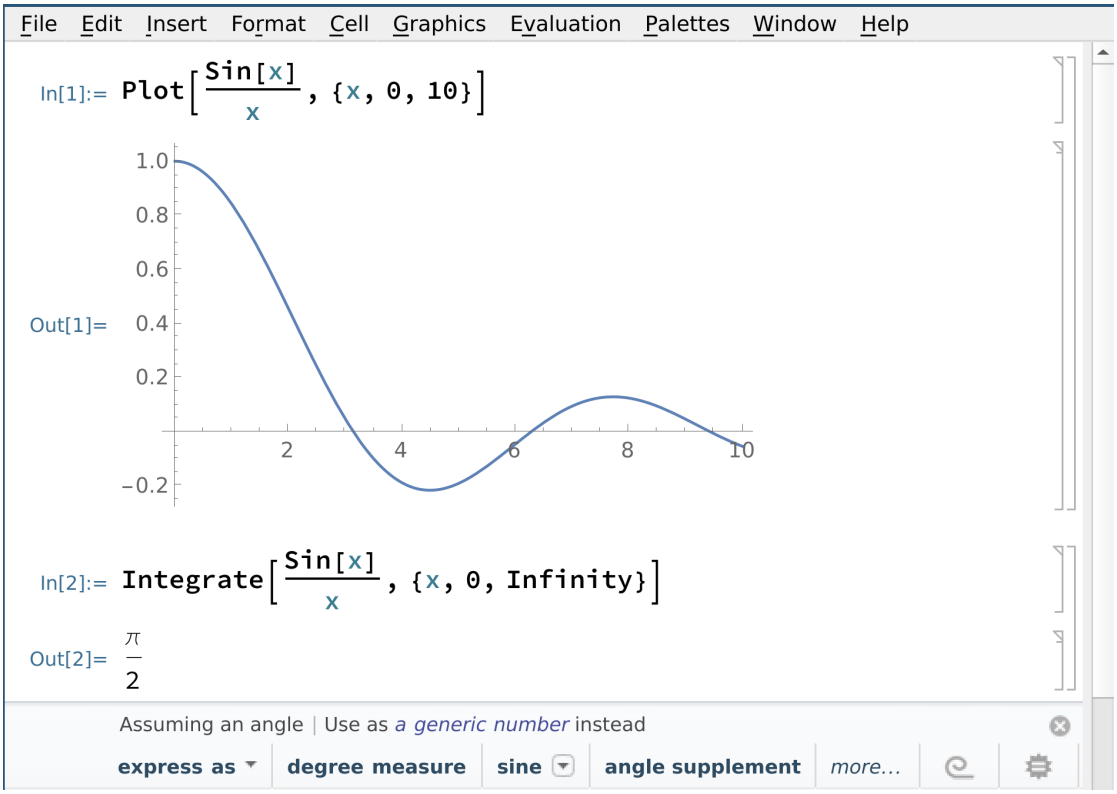
- merely a computer algebra system



Where do you use the Wolfram Language?

Mathematica

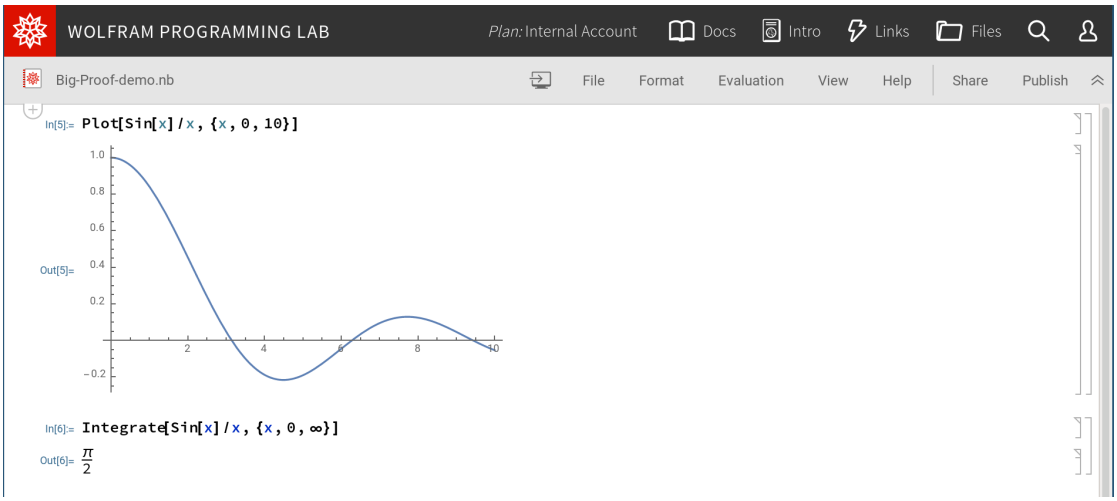
- desktop front-end using computational notebooks
- https://wolfram.com/engine
- This presentation was written in Mathematica!



Where do you use the Wolfram Language?

Wolfram Programming Lab

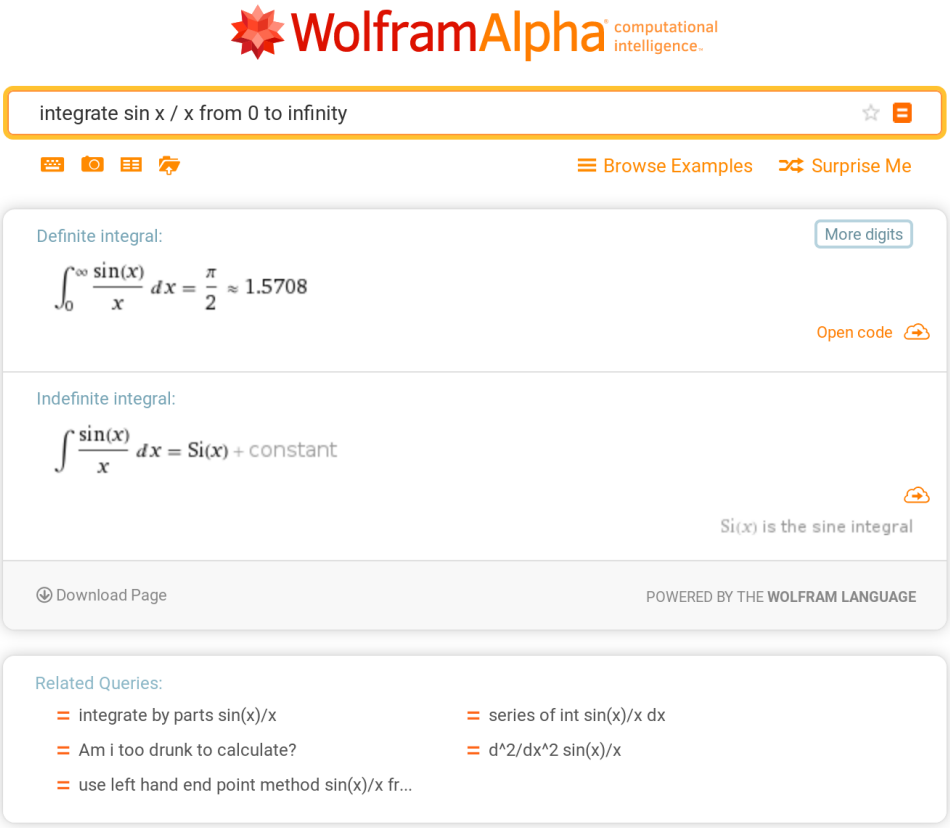
- web front-end also using computational notebooks
- https://www.wolframcloud.com



Where do you use the Wolfram Language?

Wolfram|Alpha

- web front-end for Wolfram Knowledgebase using the Wolfram Language and natural language processing (NLP)
- https://www.wolframalpha.com



Wolfram Language Syntax

Atomic data “types”

- Integer, Rational, Real, Complex

$$\left\{42, -\frac{1}{12}, 6.62607 \times 10^{-34}, 1 + i\right\}$$

- String

```
"Hello, world!"
```

- Image, Image3D, Audio

- Symbol

```
{Plot, Integrate, foo}
```

Wolfram Language Syntax

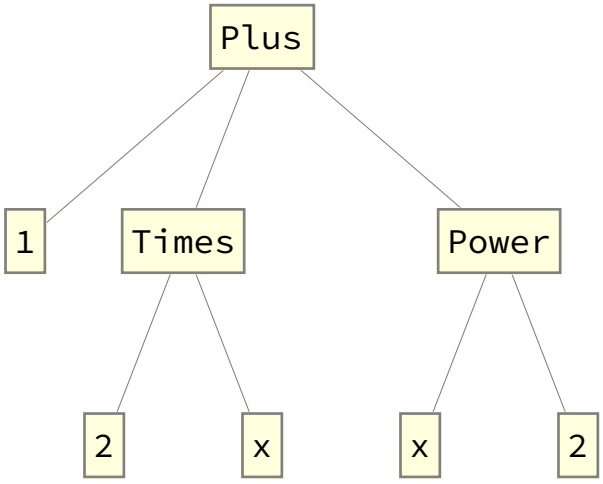
Everything is an expression

f[a, b, ...]

- f is the *head* of the expression
- a, b, … are the *arguments* / *parts* of the expression
- The *head* and *parts* of an expression can be *atomic* or *compound* expressions

In[ ]:= **TreeForm[$x^2 + 2\,x + 1$]**

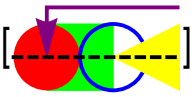
Out[ ]//TreeForm=



Wolfram Language Syntax

Everything is an expression

- Graphics are just expressions that format as graphical images

```
In[ ]:= FullForm[  ]
```

```
Out[ ]//FullForm=
Graphics[List[RGBColor[0, 1, 0], Thickness[Large], Rectangle[List[0, -1], List[2, 1]],
  List[RGBColor[1, 0, 0], Disk[List[0, 0]]], List[RGBColor[0, 0, 1], Circle[List[2, 0]]],
  List[RGBColor[1, 1, 0], Polygon[List[List[2, 0], List[4, 1], List[4, -1]]]],
  List[RGBColor[0.5`, 0, 0.5`], Arrowheads[Large],
   Arrow[List[List[4, Rational[3, 2]], List[0, Rational[3, 2]], List[0, 0]]],
   List[GrayLevel[0], Dashing[List[Small, Small]], Line[List[List[-1, 0], List[4, 0]]]]]]
```

- Even this notebook is an expression

```
In[ ]:= NotebookGet[EvaluationNotebook[]] // Short[#, 20] &
```

```
Out[ ]//Short= Notebook[{Cell[CellGroupData[{Cell[
                 Mathematical Knowledge Representation and Reasoning in the Wolfram Language,
                 Title, CellChangeTimes → {{3.76944 × 10^9, 3.76944 × 10^9}},
                 TextAlignment → Center], ≪9≫}, Open]]},
          WindowSize → {629, 686}, WindowMargins → {{7, Automatic}, {7, Automatic}},
          FrontEndVersion →
           12.0 for Linux x86 (64-bit) (April 8, 2019),
          StyleDefinitions → Default.nb]
```

Programming in the Wolfram Language

Functions are defined using pattern matching and replacement rules:

```
In[ ]:= f[{x_, y_}] := x + y
        f[_Plus] := Block[{},
          Echo["Addition detected"];
          42
         ]

In[ ]:= f[{1, 2}]

Out[ ]= 3

In[ ]:= f[1 + x]

  » Addition detected

Out[ ]= 42
```

Wolfram Knowledgebase

https://www.wolfram.com/knowledgebase

Computable knowledge across thousands of domains:

Major coverage areas

- Units & Measures »
- Dates, Times & Calendars »
- Words & Languages »
- Places & Geography »
- Socioeconomic Data »
- Money & Finance »
- People & History »
- Culture & Media »
- Art & Design »
- Music »
- Math & Statistics »
- Physics & Chemistry »
- Astronomy »
- Life Sciences »
- Earth Sciences »
- Weather & Meteorology »
- Engineering »
- Materials »
- Health & Medicine »
- Food & Nutrition »
- Educational Data »
- Organizations »
- Sports & Games »
- Shopping & Products »
- Transportation »
- Web & Computer Systems »
- Technological Systems »

Wolfram Knowledgebase

Entities

Entities represent real-world objects, people, etc. as well as abstract concepts:

Edinburgh Castle CASTLE

Entities have a *type* and are identified by their *standard name*:

```
In[ ]:= Edinburgh Castle CASTLE // FullForm
Out[ ]//FullForm=
Entity["Castle", "EdinburghCastle"]
```

Entity Properties

Each entity type has a list of supported *properties*, and entities of that type have *values* for those properties:

```
In[ ]:= EntityProperties["Castle"]
Out[ ]= {coordinates, country, elevation,
  image, latitude, longitude, name, position}
```

```
In[ ]:= EntityValue[Edinburgh Castle CASTLE, image]
```

Out[ ]=

Mathematical Knowledge

Famous Math Problems

```
In[ ]:= RandomEntity["FamousMathProblem", 10]
```

Out[ ]= {Hilbert's eighth problem, strong Goldbach conjecture, two envelopes paradox, composite number problem, 196-algorithm termination problem, Bayes' theorem, prime number theorem, Legendre's conjecture, Hilbert's fourteenth problem, Hilbert's sixth problem}

Out[20]=

Entity	abc conjecture
associated people	{David Masser, Joseph Oesterlé}
associated equations	{max[Abs[a], Abs[b], Abs[c]] ≤ Subscript[C, ϵ] Product[p^(1 + ϵ), p | a b c]}
formal statement	ForAll[ϵ, ϵ > 0, Exists[Subscript[C, ϵ], ForAll[{a, b, c}, (a | b | c)∈ Integers && GCD[a, b, c] == 1 && a + b == c, Max[Abs[a], Abs[b], Abs[c]] <= Subscript[C, ϵ] Exp[DivisorSum[a b c, (1+ϵ) Log[#1] &, # ∈ Primes &]]]]
classes	{mathematical conjectures, unsolved mathematics problems}
implied theorems	{Fermat's last theorem, Mordell conjecture}
formulation date	Year: 1985
statement	For any ϵ > 0, there exists a constant Subscript[C, ϵ] such that for any three relatively prime integers a, b, c satisfying a + b == c, the inequality max[Abs[a], Abs[b], Abs[c]] ≤ Subscript[C, ϵ] Product[p^(1+ϵ), p | a b c] holds.
unsolved	True

Mathematical Knowledge

Function Spaces

```
In[•]:= RandomEntity["FunctionSpace", 5]
```

```
Out[•]= {vanishing mean oscillation space VMO(ℝ^n, d'x^n), continuous space C^∞([a, b]),
  Bochner space L^∞(T;X), Lebesgue space L^p(𝔻, d'λ^2), harmonic Hardy space h^2(𝔻)}
```

Out[37]=

Entity	Lebesgue space $L^2(\mathbb{R}^n, d'x^n)$
alternate names	{2-Lebesgue space on $\mathbb{R}^n$, space of square-integrable complex-valued functions on $\mathbb{R}^n$}
associated people	{David Hilbert, Erhard Schmidt, Frigyes Riesz, Henri-Léon Lebesgue}
Bessel inequality	Function[{f, ϕ}, $\sum_{\phi \in$ PureMath`OrthonormalSubset[{{LebesgueL, {{Reals, n}, {LebesgueMeasure, n}}}, 2}]$}$ Abs[PureMath`InnerProduct[{f, ϕ}, {{LebesgueL, {{Reals, n}, {LebesgueMeasure, n}}}, 2}]] ≤ Norm[f, {{LebesgueL, {{Reals, n}, {LebesgueMeasure, n}}}, 2}]]
Cauchy-Schwarz inequality	Function[{f1, f2}, Abs[PureMath`InnerProduct[{f1, f2}, {{LebesgueL, {{Reals, n}, {LebesgueMeasure, n}}}, 2}]] ≤ Norm[f1, {{LebesgueL, {{Reals, n}, {LebesgueMeasure, n}}}, 2}] Norm[f2, {{LebesgueL, {{Reals, n}, {LebesgueMeasure, n}}}, 2}]]
classes	{generalized Lebesgue space, homogeneous space, space of measurable functions, real domain space}
dual space	Lebesgue space $L^2(\mathbb{R}^n, d'x^n)$
inner product	Function[{f1, f2}, Integrate[f1[x] Conjugate[f2][x], {x, PureMath`MeasureSpace[$\mathbb{R}^n$, PureMath`LebesgueMeasure[n]]}]]
isomorphic spaces	{Bergman space $A^2(\mathbb{D}, d'\lambda^2)$, harmonic Bergman space $a^2(\mathbb{D}, d'\lambda^2)$, sequence space $l^2(\mathbb{Z}^+)$}
measure space	PureMath`MeasureSpace[$\mathbb{R}^n$, PureMath`LebesgueMeasure[n]]
norm	Function[f, $\sqrt{\text{Integrate}[\text{Abs}[f[x]]^2, \{x, \text{PureMath`MeasureSpace}[\mathbb{R}^n, \text{PureMath`LebesgueMeasure}[n]]\}]}$]
related results	{Banach-Saks theorem, Lax-Milgram theorem, Littlewood-Paley decomposition, Minkowski's inequalities, Plancherel's theorem, Poincaré-Friedrichs inequalities, Poincaré inequality, Riesz-Fischer theorem, Stampacchia theorem}

Mathematical Knowledge

Function Spaces

Mathematical Knowledge

Function Spaces

Find function spaces associated with David Hilbert:

```
In[ ]:= EntityList[
          Entity["FunctionSpace", "AssociatedPeople" → ContainsAny[{David Hilbert PERSON}]]]
```

```
Out[ ]= {Lebesgue space L^0(𝔻, dλ^2), Lebesgue space L^2(𝔻, dλ^2), Lebesgue space L^∞(𝔻, dλ^2),
         Lebesgue space L^2(ℝ^n, dx^n), sequence space c_0(ℤ^+), sequence space l^0(ℤ^+),
         sequence space l^2(ℤ^+), sequence space l^∞(ℤ^+), sequence space l^p(ℤ^+)}
```

Show the $L^p(\mathbb{R}^n)$ is reflexive:

```
In[ ]:= lp = Lebesgue space L^p(ℝ^n, dx^n) FUNCTION SPACE ;
```

```
In[ ]:= lq = EntityValue[lp, "DualSpace"]
```

```
Out[ ]= Lebesgue space L^(p/(-1+p))(ℝ^n, dx^n)
```

```
In[ ]:= EntityValue[lq, "DualSpace"]
```

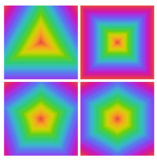
```
Out[ ]= Lebesgue space L^(p/(-1+p) (-1+p/(-1+p)))(ℝ^n, dx^n)
```

```
In[ ]:= % // Simplify
```

```
Out[ ]= Lebesgue space L^p(ℝ^n, dx^n)
```

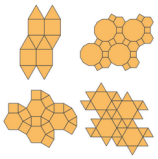
Mathematical Knowledge

https://www.wolfram.com/language/12/math-entities/

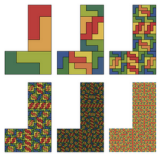


Visualize Plane Curves and Their Properties »

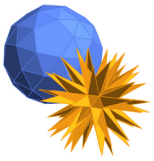
Laminae of a Different Shape »



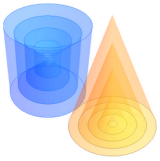
Exploring Periodic Tilings »



Explore Nonperiodic Tilings »

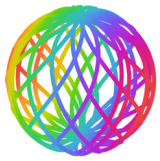


Explore Isoperimetric Quotient of Polyhedra »

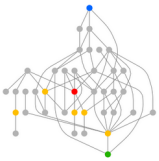


Most Efficient Containers of Given Shape »

Visualize Eponymous Surfaces »

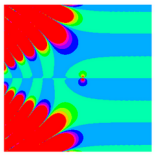


Visualize Properties of Space Curves »

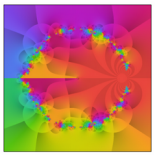


Explore Hierarchical Structure of Topological Space Types »

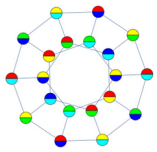
Explore Function Spaces »



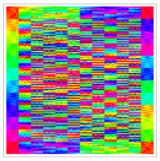
Investigate the Riemann Hypothesis »



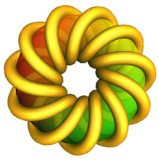
Visualize Continued Fraction Identities »



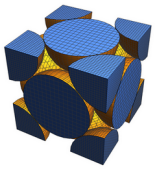
Fractional Coloring of a Graph »



Explore Group Relations and Generators »



Explore Torus Knots »



Compute Lattice Packing Densities »

Equational Logic

One of the most powerful tools for mathematical computation in the Wolfram Language is `FullSimplify`, which can be used to simplify

- polynomials

```
In[ ]:= FullSimplify[x^3 - 6 x^2 + 11 x - 6]
Out[ ]= (-3 + x) (-2 + x) (-1 + x)
```

- hyperbolic expressions

```
In[ ]:= FullSimplify[Cosh[x] - Sinh[x]]
Out[ ]= e^-x
```

- special functions

```
In[ ]:= FullSimplify[Csc[Pi v] (BesselI[-v, z] - BesselI[v, z]) / 2]
Out[ ]= BesselK[v, z] / π
```

- and equations subject to some axioms

```
In[ ]:= FullSimplify[
         ForAll[x, Exists[y, g[x, y] == e]],
         ForAll[{x, y, z}, g[x, g[y, z]] == g[g[x, y], z] && g[e, x] == x && g[inv[x], x] == e]
        ]
Out[ ]= True
```

Equational Logic

As of version 12 of the Wolfram Language, we can now generate proofs of such equations using `FindEquationalProof`:

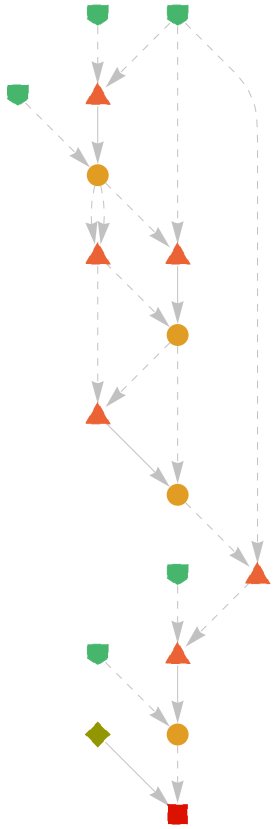
```
In[ ]:= proof = FindEquationalProof[
          ForAll[x, Exists[y, g[x, y] == e]],
          ForAll[{x, y, z}, g[x, g[y, z]] == g[g[x, y], z] && g[e, x] == x && g[inv[x], x] == e]
        ]
```

```
Out[ ]= ProofObject[ Logic: EquationalLogic  Steps: 17
                     Theorem: ∀x ∃y g[x, y] == e ]
```

We can visualize the proof:

```
In[ ]:= proof["ProofGraph"]
```

Out[ ]=



Equational Logic

Axiomatic Theories

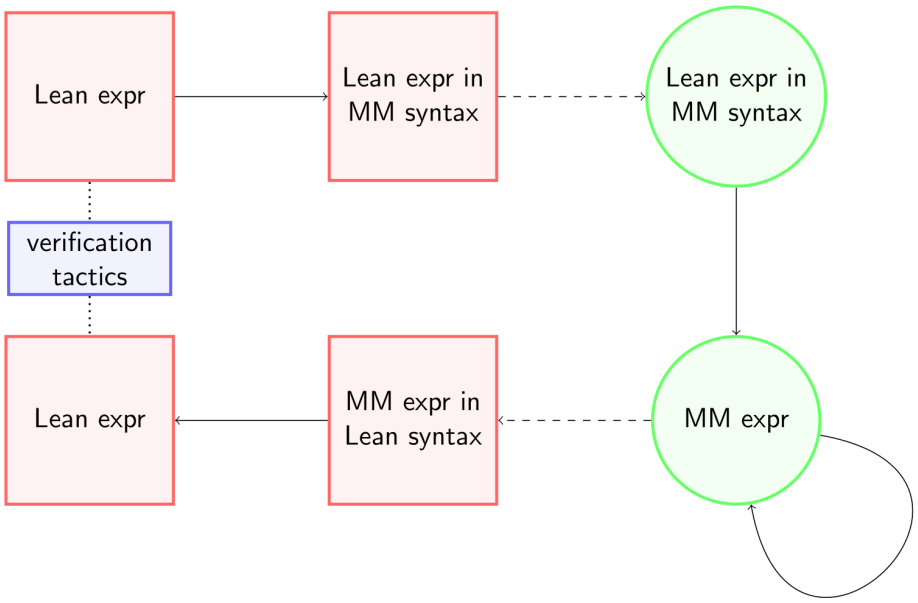
Rather than specify the axioms for `FindEquationalProof` manually, you can use one of our curated axiomatic theories:

```
In[ ]:= AxiomaticTheory[]
Out[ ]= {AbelianGroupAxioms, AbelianHigmanNeumannAxioms, AbelianMcCuneAxioms, BooleanAxioms,
  GroupAxioms, HigmanNeumannAxioms, HillmanAxioms, HuntingtonAxioms, McCuneAxioms,
  MeredithAxioms, MonoidAxioms, RingAxioms, RobbinsAxioms, SemigroupAxioms,
  ShefferAxioms, WolframAlternateAxioms, WolframAxioms, WolframCommutativeAxioms}

In[ ]:= axioms = AxiomaticTheory[
    {"GroupAxioms", <|"Multiplication" → g, "Inverse" → inv, "Identity" → e|>}]
Out[ ]= {∀_{a,b,c} g[a, g[b, c]] == g[g[a, b], c], ∀_a g[a, e] == a, ∀_a g[a, inv[a]] == e}

In[ ]:= FindEquationalProof[
   ForAll[x, Exists[y, g[x, y] == e]],
   axioms
  ]
Out[ ]= ProofObject[ Logic: EquationalLogic  Steps: 11
                     Theorem: ∀x ∃y g[x, y] == e ]
```

Integration with the Lean Theorem Prover

There is an experimental link between Lean and the Wolfram Language developed by Robert Lewis and Minchao Wu.

Using Wolfram Language from Lean



Integration with the Lean Theorem Prover

Example application: prove $x^2 - 2\,x + 1 \geq 0$ for $x \in \mathbb{R}$

- Transform the Lean representation of $x^2 - 2\,x + 1$ into WL syntax
- Interpret this as a WL polynomial
- Use `Factor` to factor the polynomial
- Transform this back into Lean syntax
- Verify that the new expression is equal to the old and use this equality in the proof

Integration with the Lean Theorem Prover

Using Lean from Wolfram Language

In[ ]:=
```
SetDirectory["/home/ianf/Workspace/Lean/mm-lean/src"];
LeanExecutable = "!/usr/bin/lean";
<< "main.m"
```

In[ ]:=
```
LeanProof["iff.symm", LeanExecutable]
```

Out[ ]=

Goal	∀ {a b : Prop}, (a ↔ b) → (b ↔ a)
ID	8
Rule	∀I
Proofs	▼ Arguments

Goal	a : Prop
ID	0
Rule	Assumption

Goal	∀ {b : Prop}, (a ↔ b) → (b ↔ a)
ID	7
Rule	∀I
Proofs	▼ Arguments

Goal	b : Prop
ID	1
Rule	Assumption

Goal	(a ↔ b) → (b ↔ a)
ID	6
Rule	∀I
Proofs	▶ Arguments

Semantic Web

RDF

The *Resource Description Framework* (RDF) is the standardized, graph-based data model of the semantic web. RDF data can be represented symbolically, imported, and exported using the WL.

In[1]:=
```
ClearAll["Global`*"];
Needs["GraphStore`"];
rdf[s_] := IRI["http://www.w3.org/1999/02/22-rdf-syntax-ns#" <> s];
schema[s_] := IRI["http://schema.org/" <> s];
ex[s_] := IRI["http://example.org/" <> s];
```

In[6]:=
```
store = RDFStore[{RDFTriple[ex["anover"], rdf["type"], schema["Book"]],
    RDFTriple[ex["anover"], schema["name"], RDFString["A nővér", "hu"]],
    RDFTriple[ex["anover"], schema["author"], "Sándor Márai"],
    RDFTriple[ex["anover"], schema["dateCreated"], "1946"],
    RDFTriple[ex["anover"], schema["isbn"], "978-84-9838-089-7"]}]
```

Out[6]=
```
RDFStore[ Triples: 5
          Entities: 1
          Classes: 1
          Properties: 5 ]
```

In[7]:=
```
ExportString[store, "Turtle", "Prefixes" → <|"schema" → "http://schema.org/"|>]
```

Out[7]=
```
PREFIX schema: <http://schema.org/>

<http://example.org/anover>
  a schema:Book ;
  schema:author "SÃ¡ndor MÃ¡rai" ;
  schema:dateCreated "1946" ;
  schema:isbn "978-84-9838-089-7" ;
  schema:name "A nÅ□vÃ©r"@hu .
```

Semantic Web

SPARQL

The *SPARQL Protocol and RDF Query Language* (SPAQRL) is the query and update language for the semantic web. We can query a SPARQL endpoint:

In[8]:=
```
endpoint = "https://query.wikidata.org/sparql";
query = "
  select distinct ?name ?elevation where {
    ?mountain wdt:P31 wd:Q8502 .
    ?mountain p:P2044 / psn:P2044 / wikibase:quantityAmount ?elevation .
    ?mountain rdfs:label ?name . filter(lang(?name) = \"en\")
  }
  order by desc(?elevation)
  limit 1
  ";
```

Symbolic SPARQL queries are also supported:

In[10]:=
```
fruitData =
  GraphStore`RDFStore[{GraphStore`RDFTriple[GraphStore`IRI["http://example.org/banana"],
     GraphStore`IRI["http://example.org/color"], "yellow"],
    GraphStore`RDFTriple[GraphStore`IRI["http://example.org/strawberry"],
     GraphStore`IRI["http://example.org/color"], "red"],
    GraphStore`RDFTriple[GraphStore`IRI["http://example.org/strawberry"],
     GraphStore`IRI["http://example.org/shape"],
     GraphStore`IRI["http://example.org/round"]]}, Association[]];
```

In[11]:=
```
query = SPARQLSelect[{
    RDFTriple[SPARQLVariable["fruit"], ex["color"], SPARQLVariable["color"]],
    SPARQLOptional[
     RDFTriple[SPARQLVariable["fruit"], ex["shape"], SPARQLVariable["shape"]]]
   }];
```

In[12]:=
```
query[fruitData]
```

Out[12]=
```
{<|fruit → IRI[http://example.org/strawberry], color → red, shape → IRI[http://example.org/round]|>,
 <|fruit → IRI[http://example.org/banana], color → yellow|>}
```

Semantic Web

OWL

The *Web Ontology Language* (OWL) is a Semantic Web language used to describe relations between entities, entity classes, and their properties. Future versions of the WL language will include functionality to do automated reasoning on OWL ontologies.

Ontology of topological spaces that are both T1 and T3:

```
In[ ]:= store = RDFStore[ ⊞ Triples: 1071
                          Entities: 154
                          Classes: 5
                          Properties: 8 ];
```

```
In[ ]:= store // SPARQLQuery[
          SPARQLSelect[RDFTriple[ex["space"], rdf["type"], SPARQLVariable["a"]]],
          SPARQLEntailmentRegime → "OWLDirect"
         ]
```

```
Out[ ]= {⟨|a → IRI[http://www.w3.org/2002/07/owl#Thing »]|⟩,
         ⟨|a → IRI[http://example.org/IsCompletelyHausdorff »]|⟩,
         ⟨|a → IRI[http://example.org/IsSemiregular »]|⟩, ⟨|a → IRI[http://example.org/IsLocallyHausdorff »]|⟩,
         ⟨|a → IRI[http://example.org/IsT3 »]|⟩, ⟨|a → IRI[http://example.org/IsRegular »]|⟩,
         ⟨|a → IRI[http://example.org/IsHausdorff »]|⟩, ⟨|a → IRI[http://example.org/IsT0 »]|⟩,
         ⟨|a → IRI[http://example.org/TopologicalSpace »]|⟩, ⟨|a → IRI[http://example.org/IsT1 »]|⟩}
```

Semantic Web

https://www.wolfram.com/language/12/rdf-and-sparql/

Query a SPARQL Endpoint »

Represent and Export RDF Data »

Query In-Memory RDF Data »

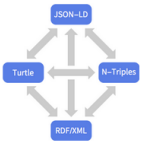
Import RDF Data »

Symbolic SPARQL »

SPARQL Update »

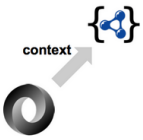
Request Linked Data from a Website »

Standard Compliance: Run Tests »



Convert Files between RDF Formats »

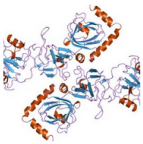
RDF Export Option: Prefixes »



Upgrade JSON to Linked Data »

Symbolic SPARQL: Import and Export »

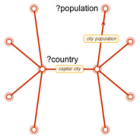
Find Theaters around a Location »



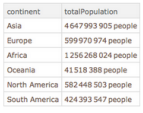
Retrieve Gene Transcript Positions »

Build a Tango Explorer »

Find Public and Alive SPARQL Endpoints »

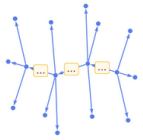


Property Paths for Entities »



continent	totalPopulation
Asia	4 647 993 905 people
Europe	599 970 974 people
Africa	1 256 268 024 people
Oceania	41 518 388 people
North America	582 448 503 people
South America	424 393 547 people

Aggregates for Entities »



Variable-Length Entity Property

Statistics of Pokémon Entities »

Plane Geometry

Analytic Geometry

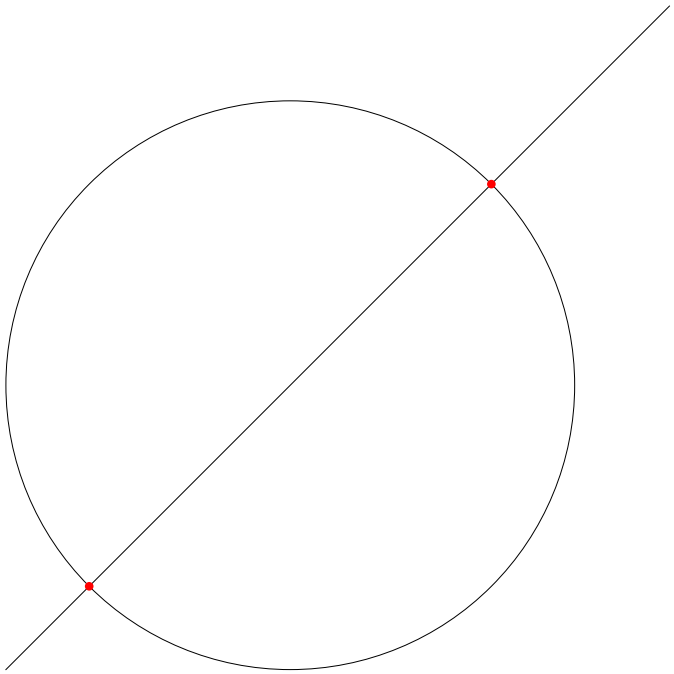
The Wolfram Language has long had powerful functionality for analytic geometry. For example, find the intersection of a line and a circle:

In[15]:=
```
pts = RegionIntersection[Line[{{-2, -1}, {5, 6}}], Circle[{1, 2}, 3]]
```

Out[15]=
$$\text{Point}\left[\left\{\left\{\frac{1}{2}\left(2-3\sqrt{2}\right),\ \frac{1}{2}\left(4-3\sqrt{2}\right)\right\},\ \left\{\frac{1}{2}\left(2+3\sqrt{2}\right),\ \frac{1}{2}\left(4+3\sqrt{2}\right)\right\}\right\}\right]$$

In[16]:=
```
Graphics[{Line[{{-2, -1}, {5, 6}}], Circle[{1, 2}, 3], {Red, PointSize[Medium], pts}}]
```

Out[16]=



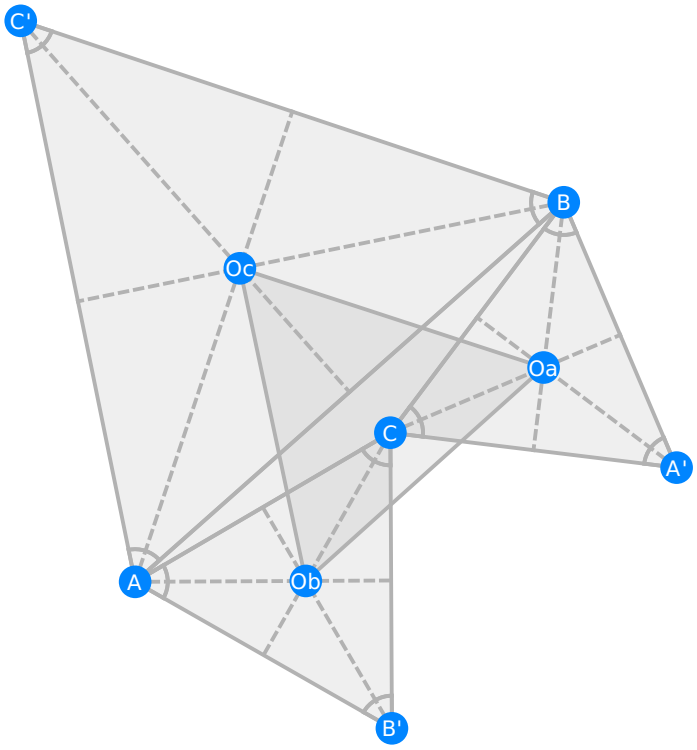
Plane Geometry

Synthetic Geometry

Now you can write synthetic descriptions of geometric scenes in the plane using `GeometricScene`:

In[17]:=
```
scene = RandomInstance[GeometricScene[{"C", "B", "A", "C'", "B'", "A'", "Oc", "Ob", "Oa"},
   {Triangle[{"C", "B", "A"}], TC == Triangle[{"A", "B", "C'"}],
    TB == Triangle[{"C", "A", "B'"}], TA == Triangle[{"B", "C", "A'"}],
    GeometricAssertion[{TC, TB, TA}, "Regular"], "Oc" == TriangleCenter[TC, "Centroid"],
    "Ob" == TriangleCenter[TB, "Centroid"], "Oa" == TriangleCenter[TA, "Centroid"],
    Triangle[{"Oc", "Ob", "Oa"}]}], RandomSeeding → 1234]
```

Out[17]=



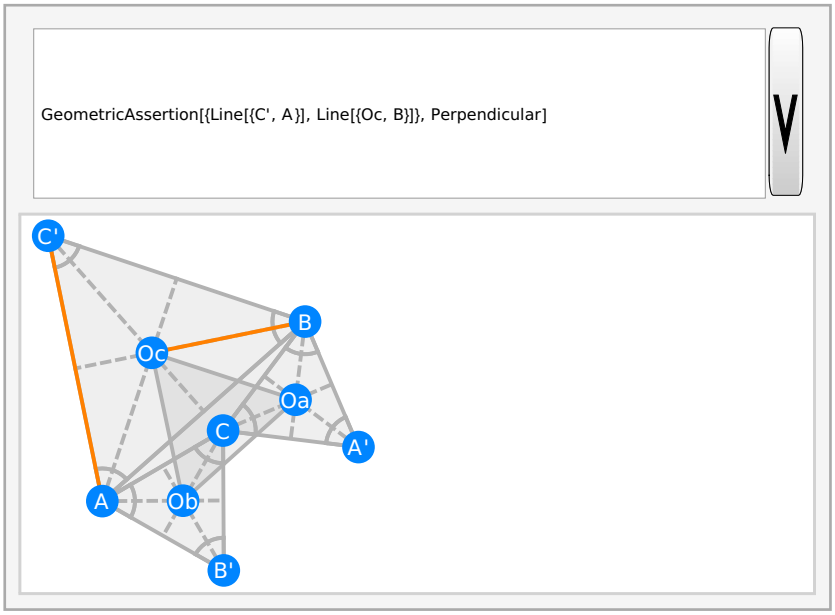
Plane Geometry

Find Conjectures

Find conjectures for the scene:

In[18]:=
```
FindGeometricConjectures[scene]
```

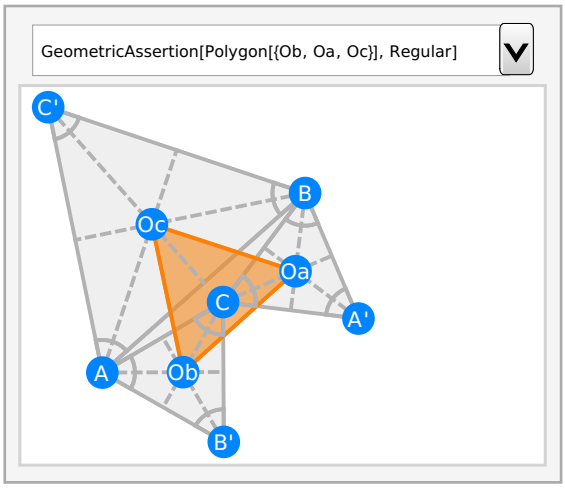
Out[18]=



Find Napoleon's Theorem:

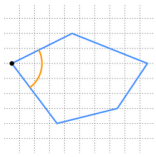
In[19]:=
```
FindGeometricConjectures[scene, GeometricAssertion[_, "Regular"]]
```

Out[19]=

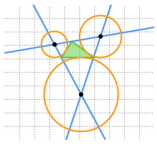


Plane Geometry

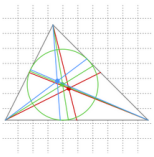
https://www.wolfram.com/language/12/plane-geometry/



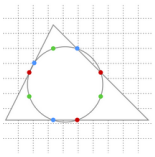
Angles »



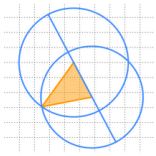
Bisectors »



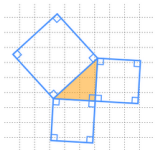
Triangle Centers »



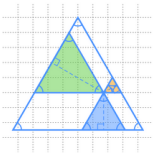
Nine-Point Circle »



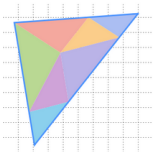
Euclid's *Elements* »



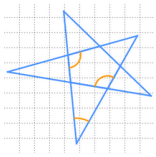
Pythagorean Theorem »



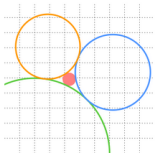
Viviani's Theorem »



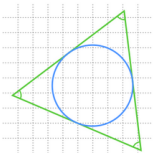
Perfect Triangle Dissections »



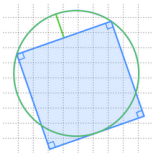
Solve for Unknown Angles »



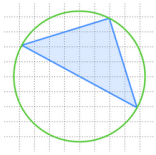
Solve for an Unknown Radius »



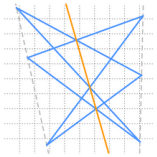
Solve for Unknown Side Lengths »



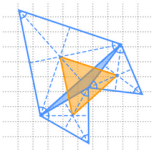
Find the Area of a Square »



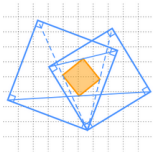
Find Simple Conjectures: Thales's Theorem »



Formulate Theorems in Multiple Ways: Pappus's Hexagon Theorem »



Filter Conjectures: Napoleon's Theorem »



Refine Theorem Statements: Finsler–Hadwiger Theorem »