Better, Faster Topological Relationships in JTS/GEOS with RelateNG

Martin Davis

FOSS4G-NA 2024

September 2024

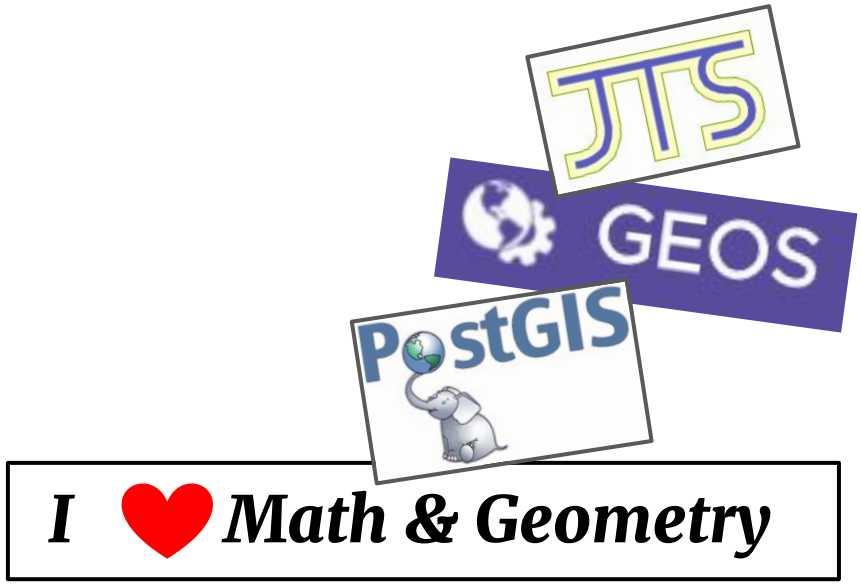
crunchy data

Martin Davis

martin.davis@crunchydata.com

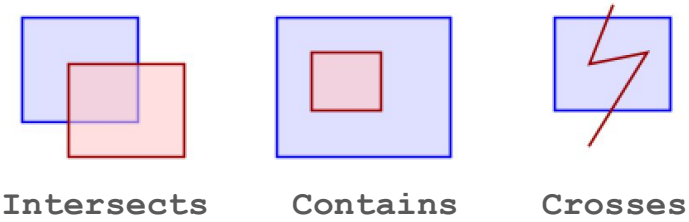
- **Geospatial Engineer** at crunchydata

- Developer on:
  - **JTS Topology Suite**
  - **GEOS**
  - **PostGIS**
  - `pg_featureserv`



Topological Relationships

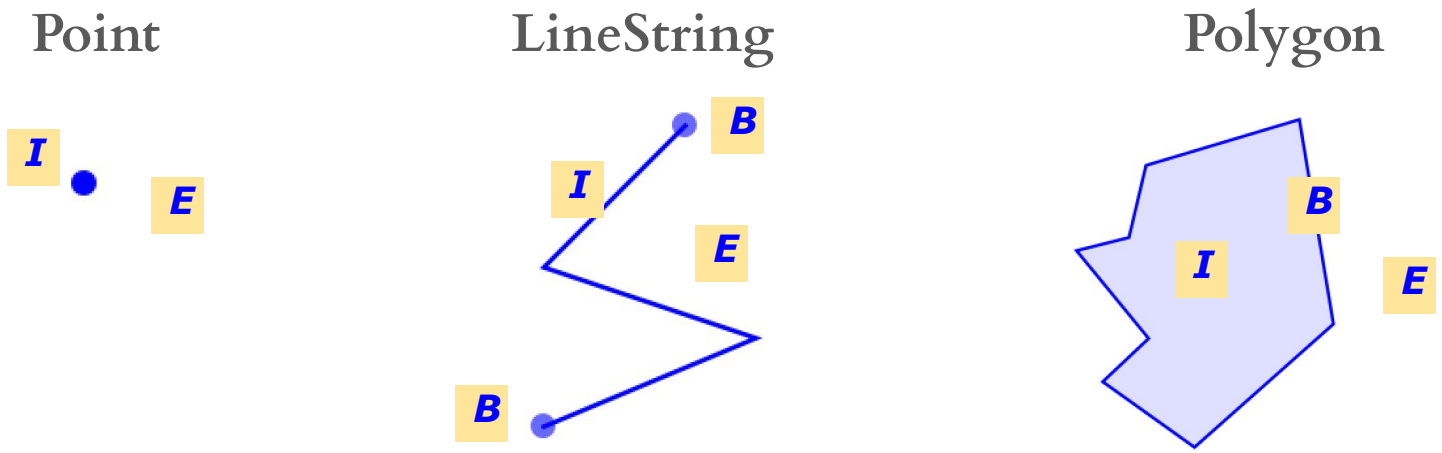
- Fundamental concept in spatial analysis
- Specify the **spatial relationship** between two geometric objects
- OGC Simple Features has **8 named relationships**:
  - `equals, intersects, disjoint`
  - `contains, within`
  - `crosses, touches, overlaps`



- Used in **spatial filters** and **joins**
  - `SELECT fire.* FROM fire WHERE Intersects(fire.geom, area)`
  - `SELECT fire.*, park.id FROM fire JOIN park ON Contains(park.geom, fire.geom)`

Geometry and Point-Set Topology

- A **geometry** is a set of points embedded in the plane (or 2D manifold)
  - OGC Simple Features: **Point, LineString, Polygon,** and **collections**
- In **Point-Set Topology**, a geometry splits the plane into 3 sets:
  - **Interior**, **Boundary** and **Exterior**



Dimensionally Extended 9-Intersection Model

- Theoretical framework for topological relationships: **Dimensionally Extended 9-Intersection Model** (DE-9IM)
  - *Egenhofer (1991), Clementini et al* (1993)
- Relationship between 2 geometries determined by **dimension of intersections** between the sets **Interior**, **Boundary** and **Exterior**
  - 3 x 3 = 9 possible intersections
  - Represent as a 3 x 3 matrix:
  - Or a 9-symbol string:
    - `212101212`

	Int	*Bdy*	*Ext*
Int	*dim*(I ∩ I)	*dim*(I ∩ B)	*dim*(I ∩ E)
Bdy	*dim*(B ∩ I)	*dim*(B ∩ B)	*dim*(B ∩ E)
Ext	*dim*(E ∩ I)	*dim*(E ∩ B)	*dim*(E ∩ E)

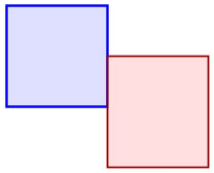
DE-9IM Named Predicates

- Predicate value determined by combinations of matrix entries, and (sometimes) geometry dimensions
- Examples:
  - **Intersects(A, B) ⇔ II ∨ IB ∨ BI ∨ BB**
  - **Contains(A, B) ⇔ II ∧ ~EI ∧ ~EB**
  - **Overlaps(A, B) ⇔ dim(A) = dim(B) ∧ II ∧ IE ∧ EI**
- Other predicates possible (and useful!)
  - `covers/coveredBy, interiorIntersects, adjacent`

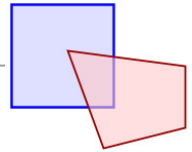
DE-9IM Patterns

- Express relationships by a **DE-9IM pattern**
- Pattern is a 9-symbol string corresponding to matrix entries
- Pattern symbols:
  - `T` - some intersection
  - `F` - no intersection
  - `0, 1, 2` - intersection dimension
  - `*` - Don't Care
- Predicate: `matches(A, B, pattern)` = {T, F}

Adjacent(A,B)	`F***1***`	T
InteriorIntersects(A,B)	`T*******`	F

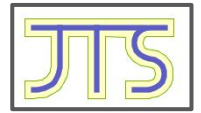


F
T



Implementation in JTS/GEOS - `RelateOp`

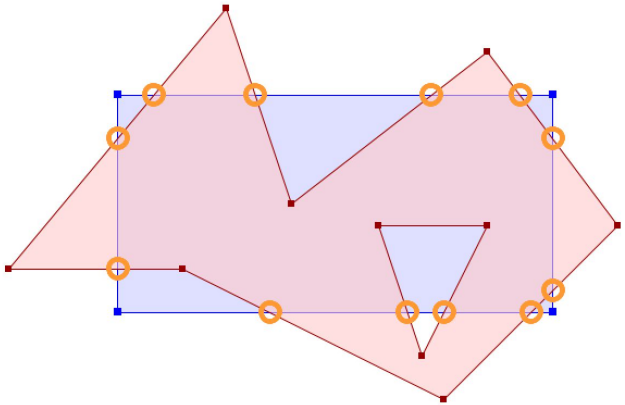
- **`RelateOp`** - first algorithm implemented in **JTS Topology Suite** (2000)
  - OGC named predicates (**`intersects`**, **`contains`**, etc.)
  - DE-9IM calculation, pattern matching
- Ported to GEOS (2003)
- Later improvement: **`PreparedGeometry`** (2008)
  - improves performance & robustness of some predicates
    - **`intersects, contains, covers`**



- Used in **many** spatial systems:
  - PostGIS, QGIS, GDAL, DuckDB, etc. etc.
  - Executed (*probably*) ***billions*** **of times per day** !!

`RelateOp` - Algorithm

- Build **full topology** of **fully-noded** input geometries
  - spatial index for efficient point-in-polygon, noding
- Evaluate topology at **every** node and update Intersection Matrix
- Compute result from **fully-determined** IM
  - **named predicate**: match IM against predicate definition
  - **DE-9IM pattern**: match IM against pattern



	I	B	E
I	2	1	2
B	1	0	1
E	2	1	2

Intersects(A, B) = *T*

Contains(A, B) = *F*

T******** = *T*

Evaluation Modes

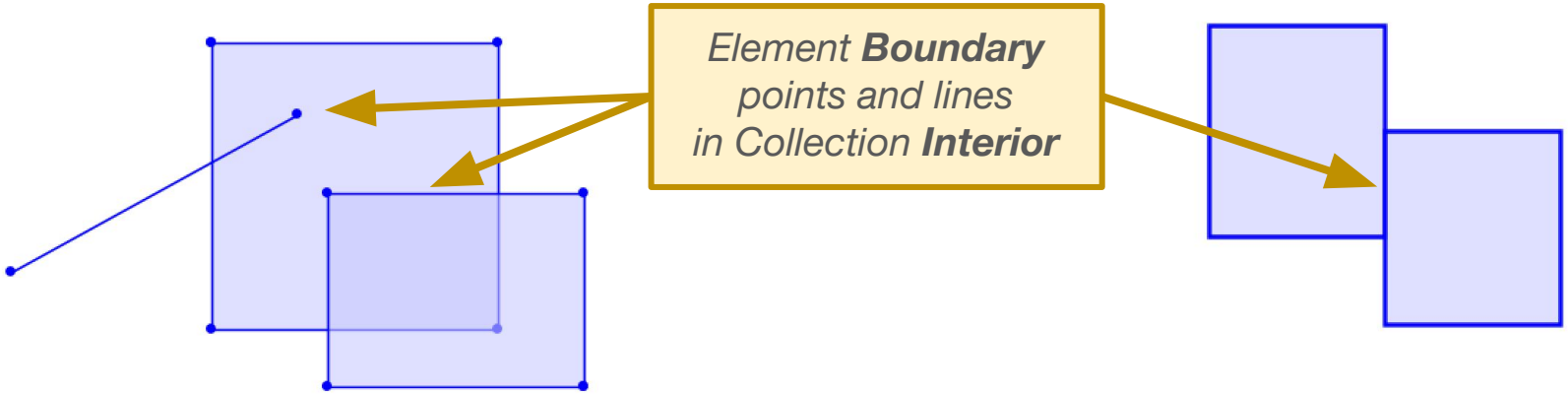
- Two ways to compute a predicate:
  - **Single-shot**
  - **Prepared**
- Single-shot mode (**`RelateOp`**)
  - Simple API: `intersects(A, B)`
  - No state retained - recompute info for every call
- Prepared mode (**`PreparedGeometry`**)
  - More complex API: `rel = Relate(A); rel.intersects(A, B)`
  - Requires state to be maintained
  - Reuse information about A

`RelateOp` Issues

- Built on generalized `GeometryGraph` structure
  - Used by several algorithms: Relate, Overlay, Buffer
  - multiple uses => complex code
  - Hard to maintain and enhance
- Computes **full topology graph** of inputs
  - Robustness problems (`TopologyException` - ugh)
  - Limits performance (no short-circuiting)
- **`GeometryCollection`** not supported
- Works only in **single-shot** mode
- Different code for `RelateOp` and `PreparedGeometry`
  - Requires effort to keep functionality in synch

GeometryCollection “Union Semantics”

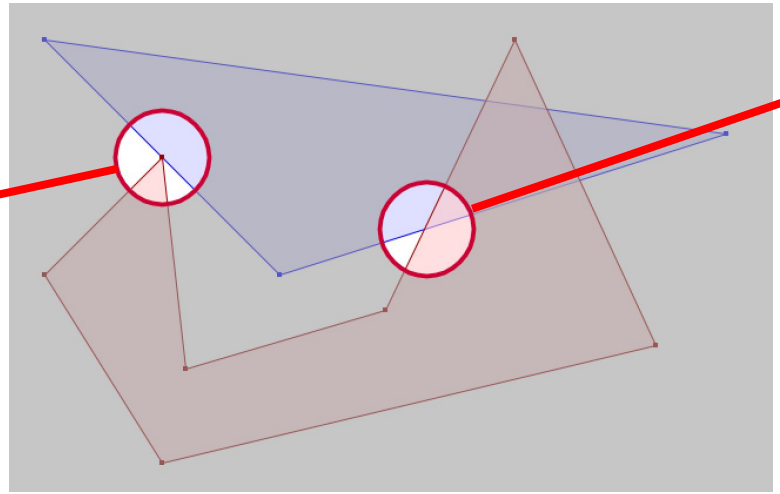
- GeometryCollection elements may overlap
- Points in **Boundary** of elements may be in **Interior** of collection
- Compute topology using *implicit* union of the collection elements
  - Need more sophisticated `PointOnGeometry` algorithm
  - Different short-circuit heuristics

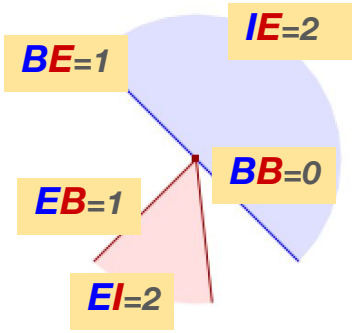


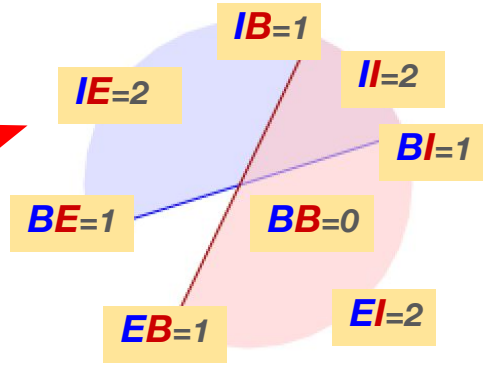
DE-9IM Topology - Key Observations

- Point-set topology is determined by local neighbourhoods of "significant nodes" (intersections and endpoints)
  - **=> *Building full topology graph is not required***
- Most predicates do not require full Intersection Matrix to be computed
  - **=> *Short-circuiting is possible***
- Intersection Matrix computation is **additive only**
  - **=> *Short-circuiting is safe***
- Some predicate cases are determined by dimensions or envelopes
  - **=> *Fast to check***

Example







	I	B	E
I	2	1	2
B	1	0	1
E	2	1	2

	I	B	E
I	-	-	2
B	-	0	1
E	2	1	2

Intersects(A, B) ⇔
II ∨ IB ∨ BI ∨

= T

Contains(A, B) ⇔
II ∧ ~EI ∧ ~EB

= F

Overlaps(A, B) ⇔
dim(A) = dim(B)
∧ II ∧ IE ∧ EI

= T

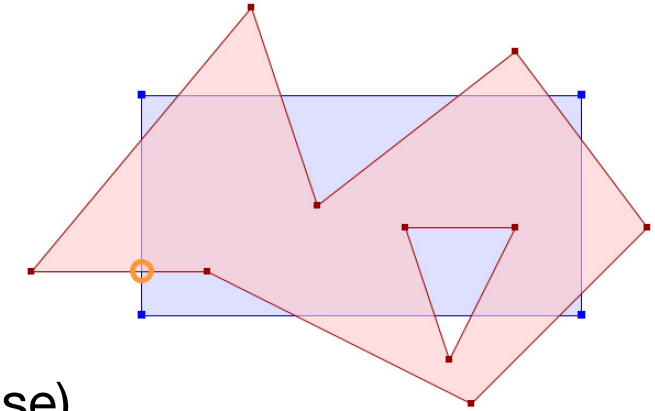
crunchy data

New! - `RelateNG`

- Released in JTS and GEOS in 2024 (soon in PostGIS)
- Key Features:
  - Support for **`GeometryCollection`**
  - Independent code base
  - Fully robust
  - Aggressive short-circuiting
  - Optimizes **any** DE-9IM pattern
  - **Single-shot** and **Prepared mode** for *all* predicates & patterns
  - Faster
    - for “single-shot” mode
    - *most* “prepared” predicates

RelateNG - Algorithm

- Predicate determines short-circuits for dimensions, envelopes, Intersection Matrix entries
  - **Named** - hard-coded
  - **Relate pattern** - dynamic analysis
- Check for **dimension and envelope** short-circuits
- Build **spatial indexes** for point-in-polygon and noding
  - Only when needed (lazily)
  - Cache indexes for prepared mode
- For each “significant point”:
  - Evaluate partial topology at point
  - Update Intersection Matrix
  - Exit if predicate value is known (true or false)

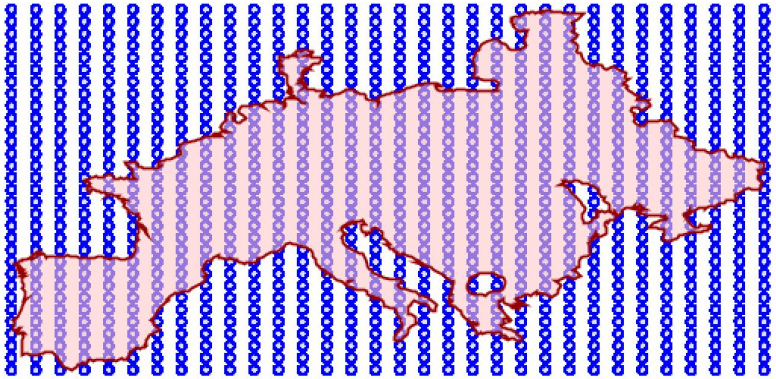


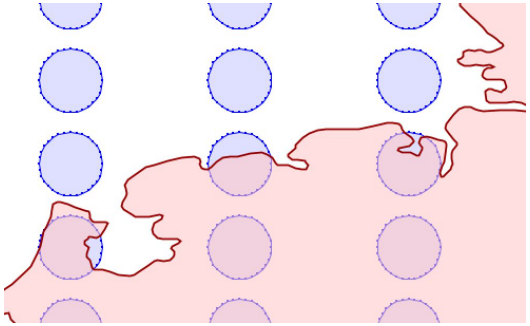
Performance

- Performance varies depending on:
  - geometry type, size
  - spatial relationship
  - predicate used
- Evaluate performance in two modes:
  - Single-shot
  - Prepared mode

Performance - Single-Shot

- Example: Polygon (4,930 vertices) x Polygons (992 x 33 vertices)

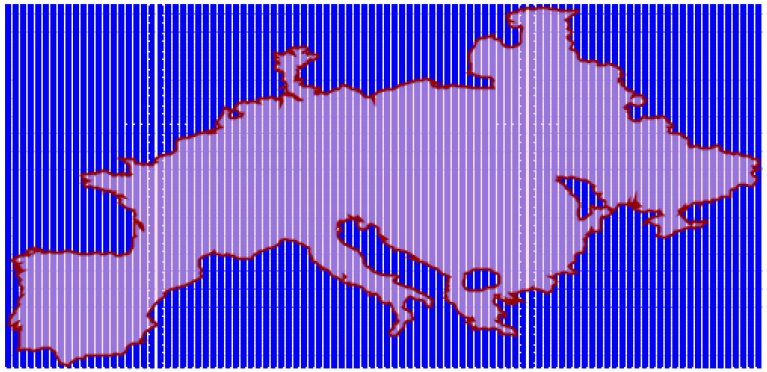


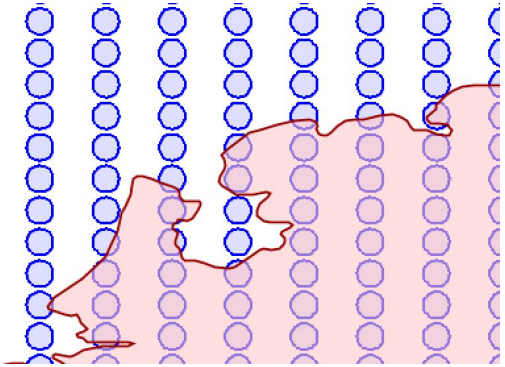


	Intersects	Covers	Touches	T********
RelateOp	469 ms	413 ms	473 ms	473 ms
RelateNG	43 ms	32 ms	40 ms	61 ms

Performance - Prepared Mode

- Example: Polygon (4,930 vertices) x Polygons (10,000 x 33 vertices)





	Intersects	Covers	Touches	T********
PreparedGeometry	23 ms	19 ms	N/A	N/A
RelateNG Prepared	18 ms	24 ms	25 ms	21 ms

Future Work

- Improve spatial indexing?
  - point-in-polygon, line noding
- More optimizations:
  - Additional short-circuit heuristics
  - Specific predicate / geometry cases
    - E.g. **`intersects, contains`** for Point / Polygon
- Extended DE-9IM pattern language?
  - Express **all** named predicates
  - Dimension constraints
  - AND / OR / NOT

Covers(A,B) =

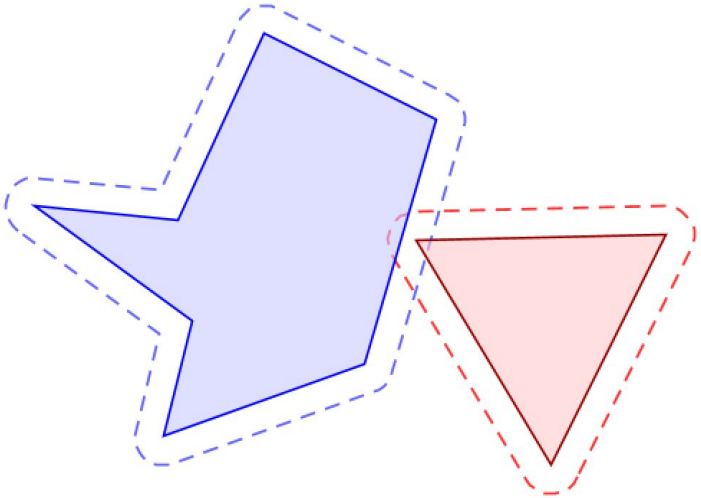
```
T*****FF* | *T****FF* |
***T**FF* | ****T*FF*
```

Overlaps(A,B) =

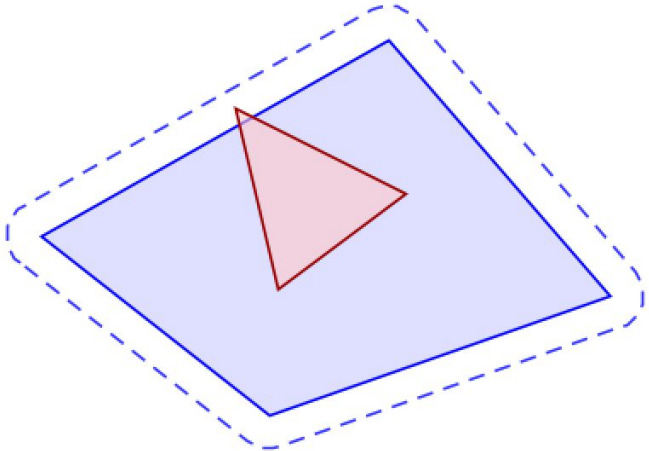
```
[02]T*T***T** | [1]1*T***T**
```

Future Work - Distance Tolerance

- Support **distance tolerance** for topological relationships



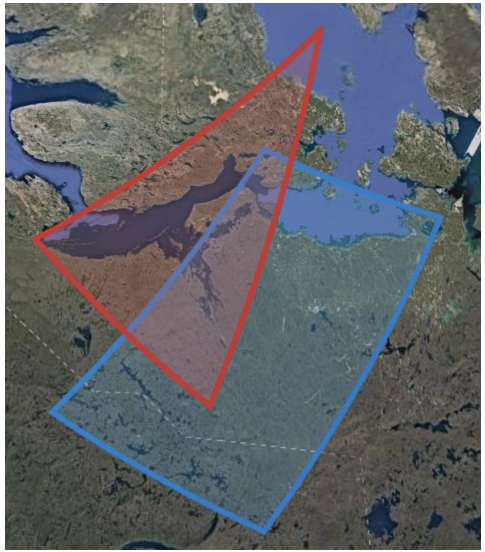
Intersects(A, B, distanceTol)

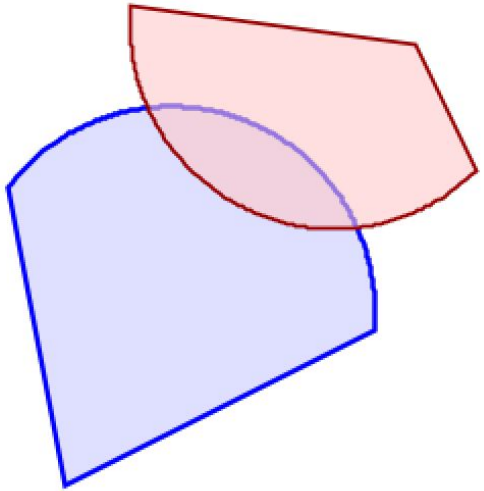


Contains(A, B, distanceTol)

Future Work - Geodetic, Curves

- Support other geometry models:
  - Geodetic geometry (on the spheroid)
  - Curves (OGC circular arcs)





Wrap-up

Questions?

Comments?

Ideas?

martin.davis@crunchydata.com