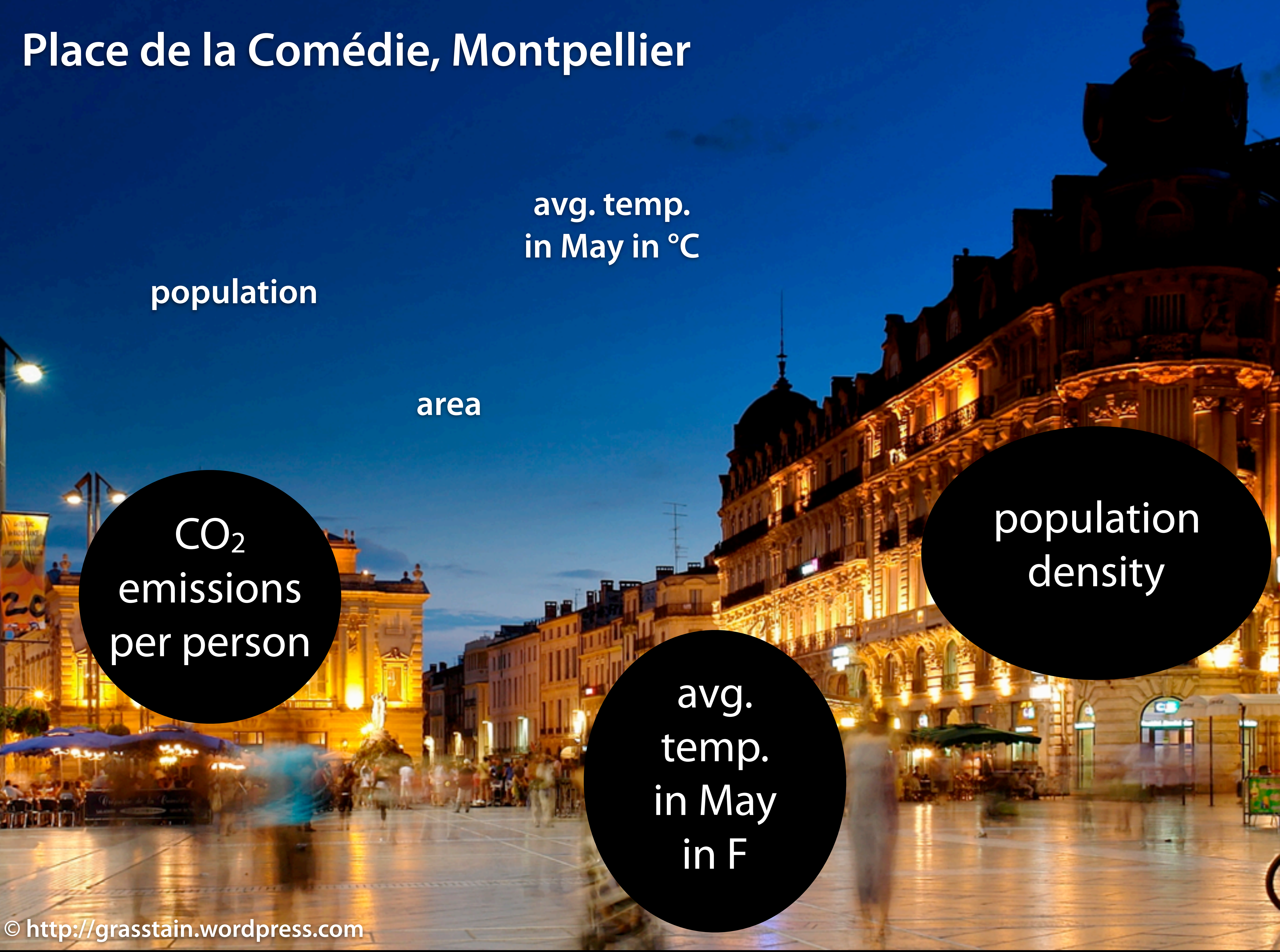
RDFS with Attribute Equations via SPARQL Rewriting

Stefan Bischof and **Axel Polleres**
Vienna University of Technology
Siemens AG Österreich

Place de la Comédie, Montpellier


Place de la Comédie, Montpellier
avg. temp. in May in °C
population
area
CO_2 emissions per person
avg. temp. in May in F
population density
© http://grasstain.wordpress.com

Use equations to infer missing numbers
What is the population density of Montpellier?

- **Montpellier**
  population: 252 998
  area: 56 880 000 m^2
  population density: ??? in people/km^2
- Can we infer population density from given data?
  computations not supported by Semantic Web reasoners
- How can we get area in km^2?
  unit conversion by computation

RDFS with Attribute Equations via SPARQL Rewriting

the big picture

SPARQL
query

?

Results

RDF
data

DBpedia eurostat GeoNames

Triple store

What is the population density of Montpellier?
written in SPARQL

```
SELECT ?dens
WHERE { :Montpellier :populationDensity ?dens .}
```

```
SELECT ?city ?dens
WHERE {
  :Montpellier :populationDensity ?mdens .
  ?city rdf:type :City ;
        :populationDensity ?dens .
  FILTER(?dens > ?mdens)
}
```

RDFS with Attribute Equations via SPARQL Rewriting
the big picture

SPARQL
query

?

RDFS
ontology

Results

RDF
data

DBpedia eurostat GeoNames

Triple store

We need RDFS for integrating different sources

- Unify RDFS properties of different data sources
- Use unified name for population
  ```
  dbp:populationTotal rdfs:subpropertyOf :population
  geo:population rdfs:subpropertyOf :population
  ```
- Use already implemented RDFS reasoners
  which allow SPARQL queries

RDFS with Attribute Equations via SPARQL Rewriting
the big picture

SPARQL query

RDFS ontology

+ equations

?

Results

RDF data

DBpedia eurostat GeoNames

Triple store

Syntax RDFS and equations

converting between RDFS and DL_{RDFS}^{E}

$A_1 \sqsubseteq A_2$	$\mathsf{A_1\ rdfs:subClassOf\ A_2}$
$\exists P \sqsubseteq A$	$\mathsf{P\ rdfs:domain\ A}$
$\exists P^- \sqsubseteq A$	$\mathsf{P\ rdfs:range\ A}$
$\exists U \sqsubseteq A$	$\mathsf{U\ rdfs:domain\ A}$
$P_1 \sqsubseteq P_2$	$\mathsf{P_1\ rdfs:subPropertyOf\ P_2}$
$U_1 \sqsubseteq U_2$	$\mathsf{U_1\ rdfs:subPropertyOf\ U_2}$
$U_0 = f(U_1, \ldots, U_n)$	$\mathsf{U_0\ definedByEquation\ “f(U_1, \ldots, U_n)”}$
$A(x)$	$x\ \mathsf{rdf:type\ A}$
$R(x, y)$	$x\ \mathsf{R}\ y$
$U(x, q)$	$x\ \mathsf{U}\ ”q”\ \hat{}\hat{}\ \mathsf{owl:rational}$

Syntax RDFS and equations
converting between RDFS and DL_{RDFS}^{E}

dbp:population rdfs:domain dbp:populatedPlace.

$A_1 \sqsubseteq A_2$	$\mathsf{A_1\ rdfs:subClassOf\ A_2}$
$\exists P \sqsubseteq A$	$\mathsf{P\ rdfs:domain\ A}$
$\exists P^- \sqsubseteq A$	$\mathsf{P\ rdfs:range\ A}$
$\exists U \sqsubseteq A$	$\mathsf{U\ rdfs:domain\ A}$
$P_1 \sqsubseteq P_2$	$\mathsf{P_1\ rdfs:subPropertyOf\ P_2}$
$U_1 \sqsubseteq U_2$	$\mathsf{U_1\ rdfs:subPropertyOf\ U_2}$
$U_0 = f(U_1, \ldots, U_n)$	$\mathsf{U_0\ definedByEquation\ “f(U_1, \ldots, U_n)”}$

$A(x)$	x rdf:type A
$R(x, y)$	x R y
$U(x, q)$	x U "q"^^owl:rational

Syntax RDFS and equations

converting between RDFS and DL_{RDFS}^{E}

dbp:population rdfs:domain dbp:populatedPlace.

geonames:population rdfs:subPropertyOf dbp:population .

$A_1 \sqsubseteq A_2$	$\mathsf{A_1\ rdfs:subClassOf\ A_2}$
$\exists P \sqsubseteq A$	
$\exists P^- \sqsubseteq A$	$\mathsf{P\ rdfs:range\ A}$
$\exists U \sqsubseteq A$	$\mathsf{U\ rdfs:domain\ A}$
$P_1 \sqsubseteq P_2$	$\mathsf{P_1\ rdfs:subPropertyOf\ P_2}$
$U_1 \sqsubseteq U_2$	$\mathsf{U_1\ rdfs:subPropertyOf\ U_2}$
$U_0 = f(U_1, \ldots, U_n)$	$\mathsf{U_0\ definedByEquation\ "f(U_1, \ldots, U_n)"}$

$A(x)$	x rdf:type A
$R(x, y)$	x R y
$U(x, q)$	x U "q"^^owl:rational

Syntax RDFS and equations

converting between RDFS and $DL_{RDFS}{}^{E}$

dbp:population rdfs:domain dbp:populatedPlace.

geonames:population rdfs:subPropertyOf dbp:population .

$A_1 \sqsubseteq A_2$	$\mathsf{A_1\ rdfs:subClassOf\ A_2}$
$\exists P \sqsubseteq A$	
$\exists P^- \sqsubseteq A$	$\mathsf{P\ rdfs:range\ A}$
$\exists U \sqsubseteq A$	$\mathsf{U\ rdfs:domain\ A}$
$P_1 \sqsubseteq P_2$	$\mathsf{P_1\ rdfs:subPropertyOf\ P_2}$
$U_1 \sqsubseteq U_2$	$\mathsf{U_1\ rdfs:subPropertyOf\ U_2}$
$U_0 = f(U_1, \ldots, U_n)$	$\mathsf{U_0\ definedByEquation\ “f(U_1, \ldots, U_n)''}$
$A(x)$	x rdf : type A
$R(x, y)$	x R y
$U(x, q)$	x U ”q” ^^owl : rational

:Montpellier dbp:population 252998 .

Syntax RDFS and equations

converting between RDFS and DL_{RDFS}^{E}

dbp:population rdfs:domain dbp:populatedPlace.

geonames:population rdfs:subPropertyOf dbp:population .

$A_1 \sqsubseteq A_2$	$\mathsf{A_1}$ rdfs:subClassOf $\mathsf{A_2}$
$\exists P \sqsubseteq A$	
$\exists P^{-} \sqsubseteq A$	P rdfs:range A
$\exists U \sqsubseteq A$	U rdfs:domain A
$P_1 \sqsubseteq P_2$	$\mathsf{P_1}$ rdfs:subPropertyOf $\mathsf{P_2}$
$U_1 \sqsubseteq U_2$	$\mathsf{U_1}$ rdfs:subPropertyOf $\mathsf{U_2}$
$U_0 = f(U_1, \ldots, U_n)$	$\mathsf{U_0}$ definedByEquation "$\mathsf{f(U_1, \ldots, U_n)}$"
$A(x)$	x r
$R(x, y)$	x R
$U(x, q)$	x U "q" ^^owl:rational

dbp:populationDensity :definedByEquation
"dbp:population / dbp:area" .

:Montpellier dbp:population 252998 .

Extend the DL_{RDFS} Semantics by equations

- RDFS + attributes: usual DL model theoretic semantics

- For an equation, infer a new value
  if all other attributes of the equation are given and
  there is no division by zero
  then the computation result is the new attribute value

- $U_0 = f(U_1, \ldots, U_n)$ satisfied in $\mathcal{I}$

$$\text{if } \forall x, y_1, \ldots, y_n (\bigwedge_{i=1}^{n} (x, y_i) \in U_i^{\mathcal{I}}) \wedge \text{defined}(f(U_1/y_1, \ldots, U_n/y_n))$$
$$\Rightarrow (x, \text{eval}(f(U_1/y_1, \ldots, U_n/y_n)) \in U_0^{\mathcal{I}}$$

- Query answers are not necessarily finite
  ABoxes inconsistent with equations

Extend the DL_{RDFS} Semantics by equations

dbp:populationDensity :definedByEquation
"dbp:population / dbp:area" .

- RDFS + attributes: usual DL model theoretic semantics
- For an equation, infer a new value
  if all other attributes of the equation are given and
  there is no division by zero
  then the computation result is the new attribute value
- $U_0 = f(U_1, \ldots, U_n)$ satisfied in $\mathcal{I}$

$$\text{if } \forall x, y_1, \ldots, y_n (\bigwedge_{i=1}^{n} (x, y_i) \in U_i^{\mathcal{I}}) \wedge \text{defined}(f(U_1/y_1, \ldots, U_n/y_n))$$
$$\Rightarrow (x, \text{eval}(f(U_1/y_1, \ldots, U_n/y_n)) \in U_0^{\mathcal{I}}$$

- Query answers are not necessarily finite
  ABoxes inconsistent with equations

Extend the DL_{RDFS} Semantics by equations

dbp:populationDensity :definedByEquation
"dbp:population / dbp:area" .

- RDFS + attributes: usual DL model theoretic semantics
- For an equation, infer a new value
  if all other attributes of the equation
  there is no division by zero
  then the computation result is the n

:Montpellier dbp:population 252998 .
:Montpellier dbp:area 56.88 .

- $U_0 = f(U_1, \ldots, U_n)$ satisfied in $\mathcal{I}$

$$\text{if } \forall x, y_1, \ldots, y_n (\bigwedge_{i=1}^{n} (x, y_i) \in U_i^{\mathcal{I}}) \wedge \text{defined}(f(U_1/y_1, \ldots, U_n/y_n))$$
$$\Rightarrow (x, \text{eval}(f(U_1/y_1, \ldots, U_n/y_n)) \in U_0^{\mathcal{I}}$$

- Query answers are not necessarily finite
  ABoxes inconsistent with equations

Extend the DL_{RDFS} Semantics by equations

dbp:populationDensity :definedByEquation
"dbp:population / dbp:area" .

- RDFS + attributes: usual L model theoretic semantics
- For an equation, infe a new value
  if all other attributes of the equation
  there is no division by zero
  then the computation result is the n

:Montpellier dbp:population 252998 .
:Montpellier dbp:area 56.88 .

- $U_0 = f(U_1, \ldots, U_n)$ satisfied in $\mathcal{I}$

$$\text{if } \forall x, y_1, \ldots, y_n (\bigwedge_{i=1}^{n} (x, y_i) \in U_i^{\mathcal{I}}) \wedge \text{defined}(f(U_1/y_1, \ldots, U_n/y_n))$$
$$\Rightarrow (x, \text{eval}(f(U_1/y_1, \ldots, U_n/y_n)) \in U_0^{\mathcal{I}}$$

- Query answers are not necessarily finite
  ABoxes inconsistent with equations

:Montpellier dbp:populationDensity 4447.93 .

Formulate equations as rules
n rules for equations in *n* variables

- Equation given for population density

$$area_{km2} = \frac{population}{popDensity}$$

- Formulate equation as rule

$$area_{km2}(X, A) \Leftarrow popDensity(X, PD), population(X, P), A = P \div PD.$$

- More rules needed to cover all directions

$$\begin{aligned} popDensity(X, PD) &\Leftarrow population(X, P), area_{km2}(X, A), PD = P \div A. \\ population(X, P) &\Leftarrow area_{km2}(X, A), popDensity(X, PD), P = A \times PD. \end{aligned}$$

Forward chaining often does not terminate
because of rounding errors

$$
\begin{aligned}
popDensity(X, PD) &\Leftarrow population(X, P), area_{km2}(X, A), PD = P \div A. \\
area_{km2}(X, A) &\Leftarrow population(X, P), popDensity(X, PD), A = P \div PD. \\
population(X, P) &\Leftarrow area_{km2}(X, A), popDensity(X, PD), P = A \times PD.
\end{aligned}
$$

- DBpedia: population 252 998, area 56.88 km^2
- Apply rule: population density 4447.925...293
- Apply rules: population 252 997.999...999 and area 56.880...003
- Rules engine computes population density again: 4447.925...275

Naive backward chaining does not terminate

unfolding of recursive rules blows up arbitrarily

$$popDensity(X, PD) \;\Leftarrow\; population(X, P), area_{km2}(X, A), PD = P \div A.$$

$$population(X, P) \;\Leftarrow\; area_{km2}(X, A), popDensity(X, PD), P = A \times PD.$$

$$area_{km2}(X, A) \Leftarrow popDensity(X, PD), population(X, P), A = P \div PD.$$

- To compute the population density query for population density
  makes no sense

Naive backward chaining does not terminate

unfolding of recursive rules blows up arbitrarily

$$popDensity(X, PD) \Leftarrow population(X, P), area_{km2}(X, A), PD = P \div A.$$

$$population(X, P) \Leftarrow area_{km2}(X, A), popDensity(X, PD), P = A \times PD.$$

$$area_{km2}(X, A) \Leftarrow popDensity(X, PD), population(X, P), A = P \div PD.$$

- To compute the population density query for population density makes no sense

Naive backward chaining does not terminate

unfolding of recursive rules blows up arbitrarily

$$popDensity(X, PD) \Leftarrow population(X, P), area_{km2}(X, A), PD = P \div A.$$

$$population(X, P) \Leftarrow area_{km2}(X, A), popDensity(X, PD), P = A \times PD.$$

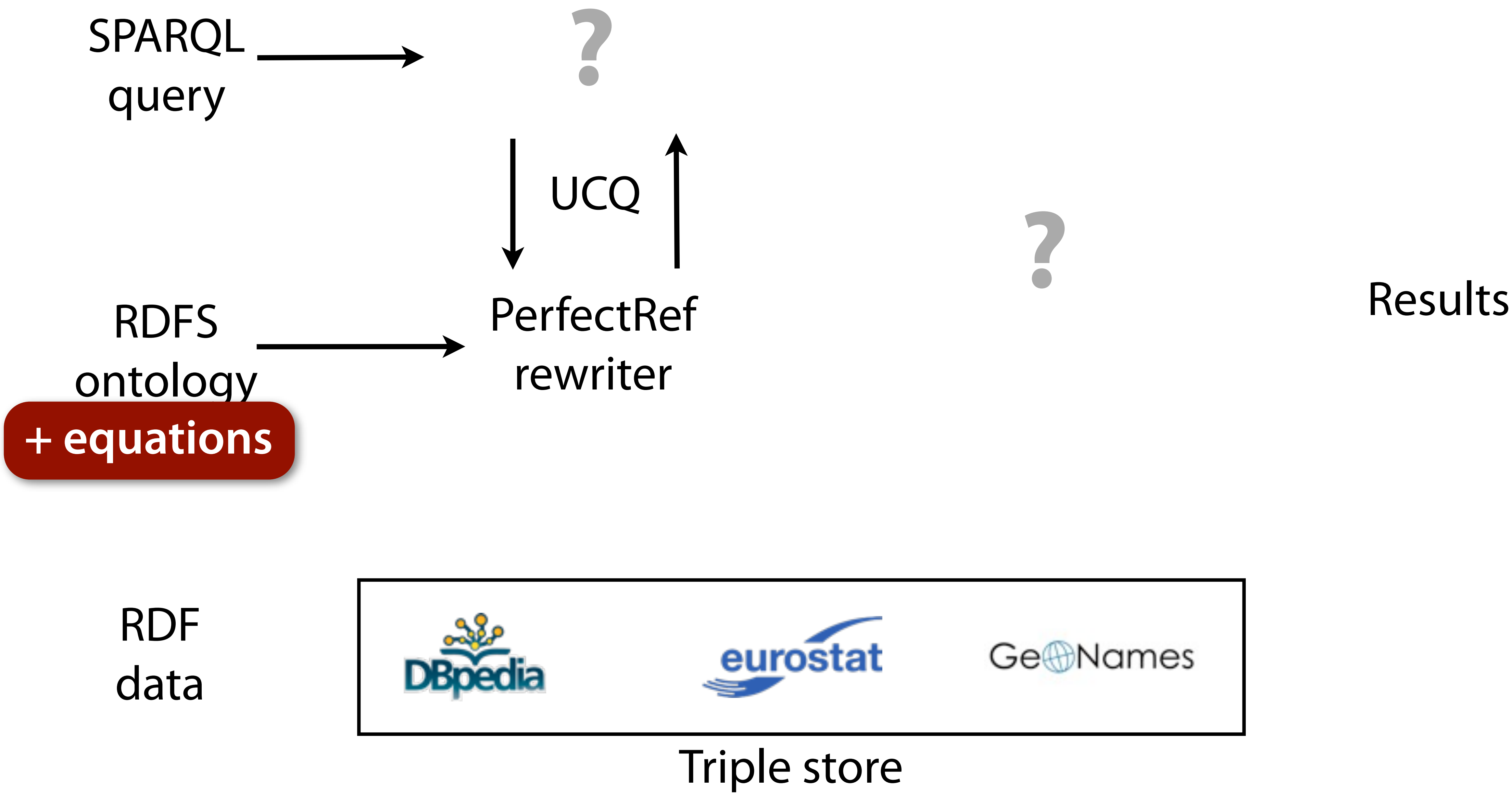
$$area_{km2}(X, A) \Leftarrow popDensity(X, PD), population(X, P), A = P \div PD.$$

- To compute the population density query for population density makes no sense

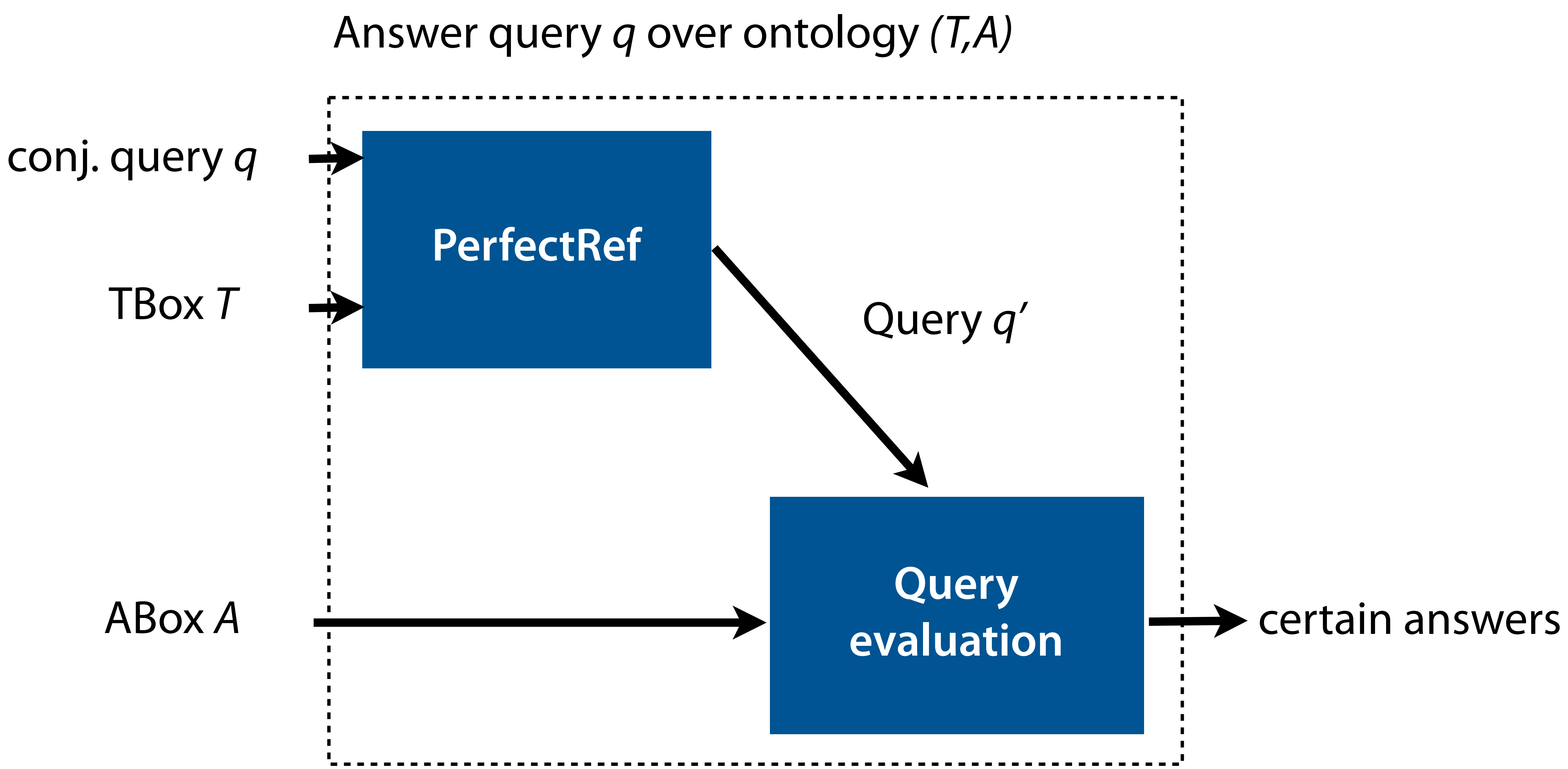
Rules are problematic for applying equations
break the infinite series of rule applications

- Need to specify all directions of the equation
  not as intuitive and short as equations
- Forward chaining often does not terminate
  division or multiplication is often enough for non-termination
  implementation dependent
- Backward chaining does not terminate
  unfolding of recursive rules can blow up arbitrarily
  even for a single equation no termination
- We have to *break* the infinite series of rule applications

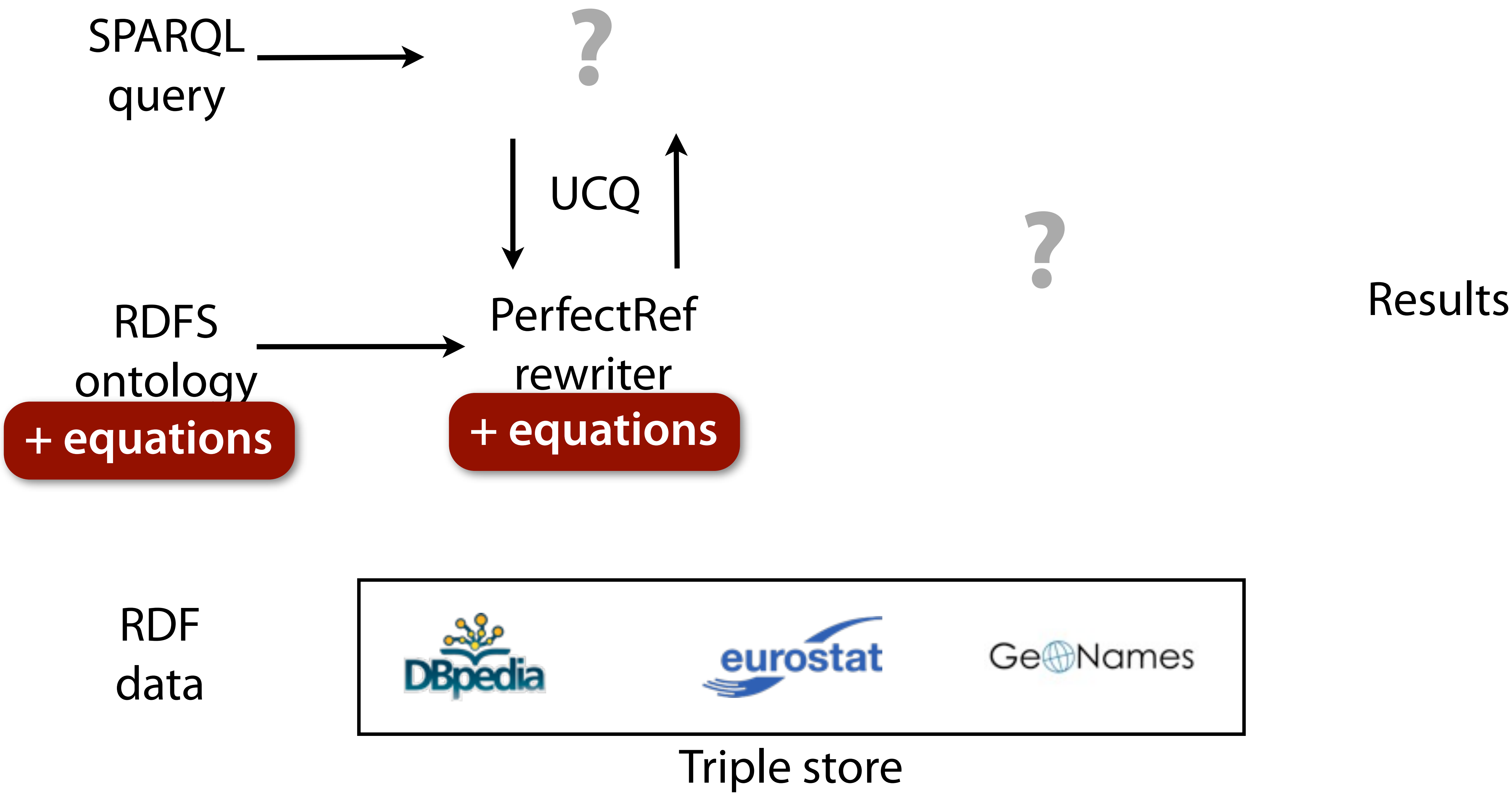
RDFS with Attribute Equations via SPARQL Rewriting
the big picture



Query answering in DL-Lite: PerfectRef
Encode TBox in the query [Calvanese et al., 2009]



RDFS with Attribute Equations via SPARQL Rewriting
the big picture



Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$

$$area_{m2} = area_{km2} \times 1\,000\,000$$

$$area_{mile2} = area_{km2} \times 2.590$$

popDensity

Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$

$$area_{m2} = area_{km2} \times 1\,000\,000$$

$$area_{mile2} = area_{km2} \times 2.590$$

popDensity

↓

$$popDensity = \frac{population}{area_{km2}}$$

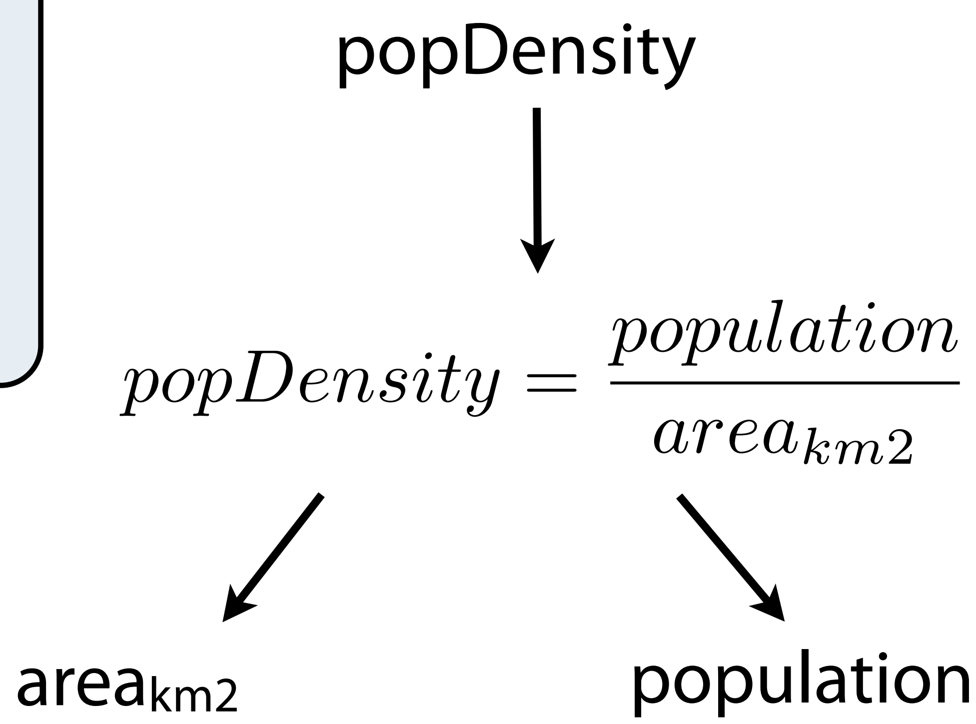
Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$

$$area_{m2} = area_{km2} \times 1\,000\,000$$

$$area_{mile2} = area_{km2} \times 2.590$$



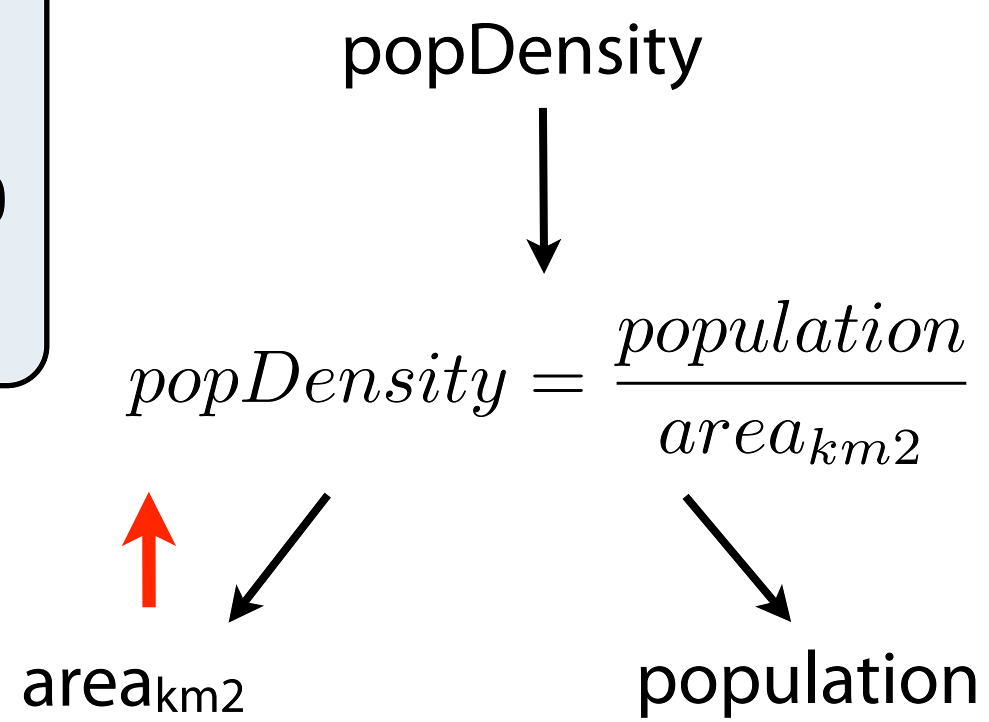
Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$

$$area_{m2} = area_{km2} \times 1\,000\,000$$

$$area_{mile2} = area_{km2} \times 2.590$$



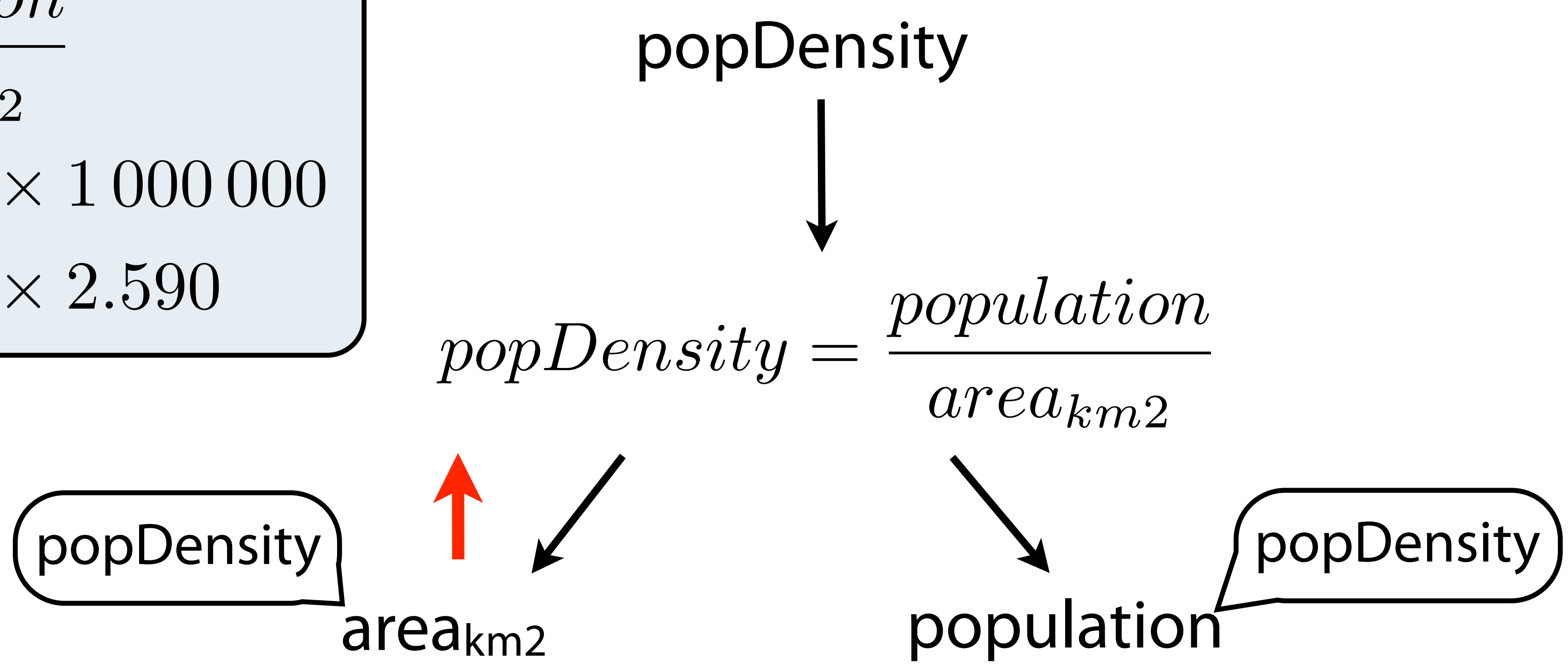
Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$

$$area_{m2} = area_{km2} \times 1\,000\,000$$

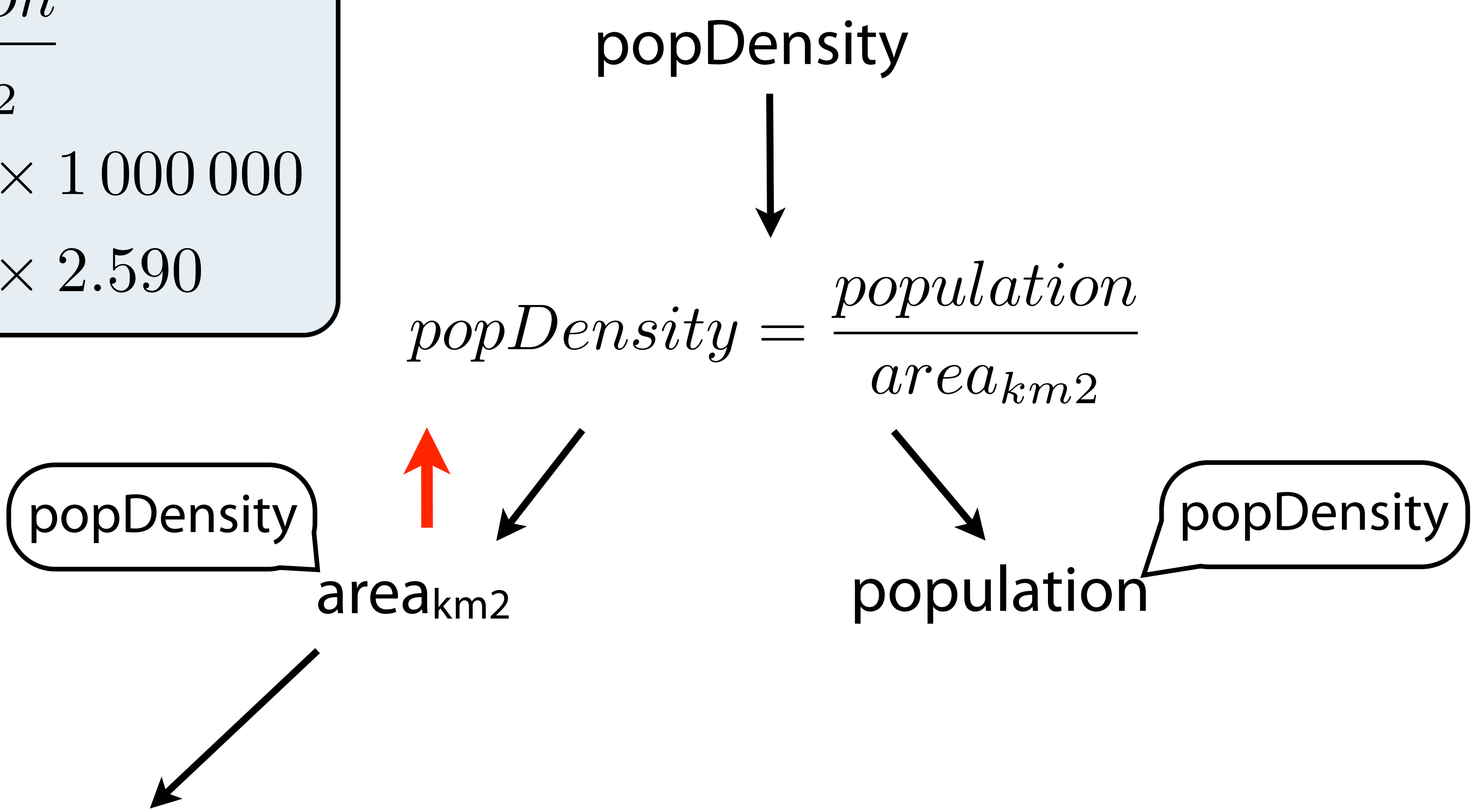
$$area_{mile2} = area_{km2} \times 2.590$$



Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$
$$area_{m2} = area_{km2} \times 1\,000\,000$$
$$area_{mile2} = area_{km2} \times 2.590$$



$$area_{m2} = area_{km2} \times 1\,000\,000$$

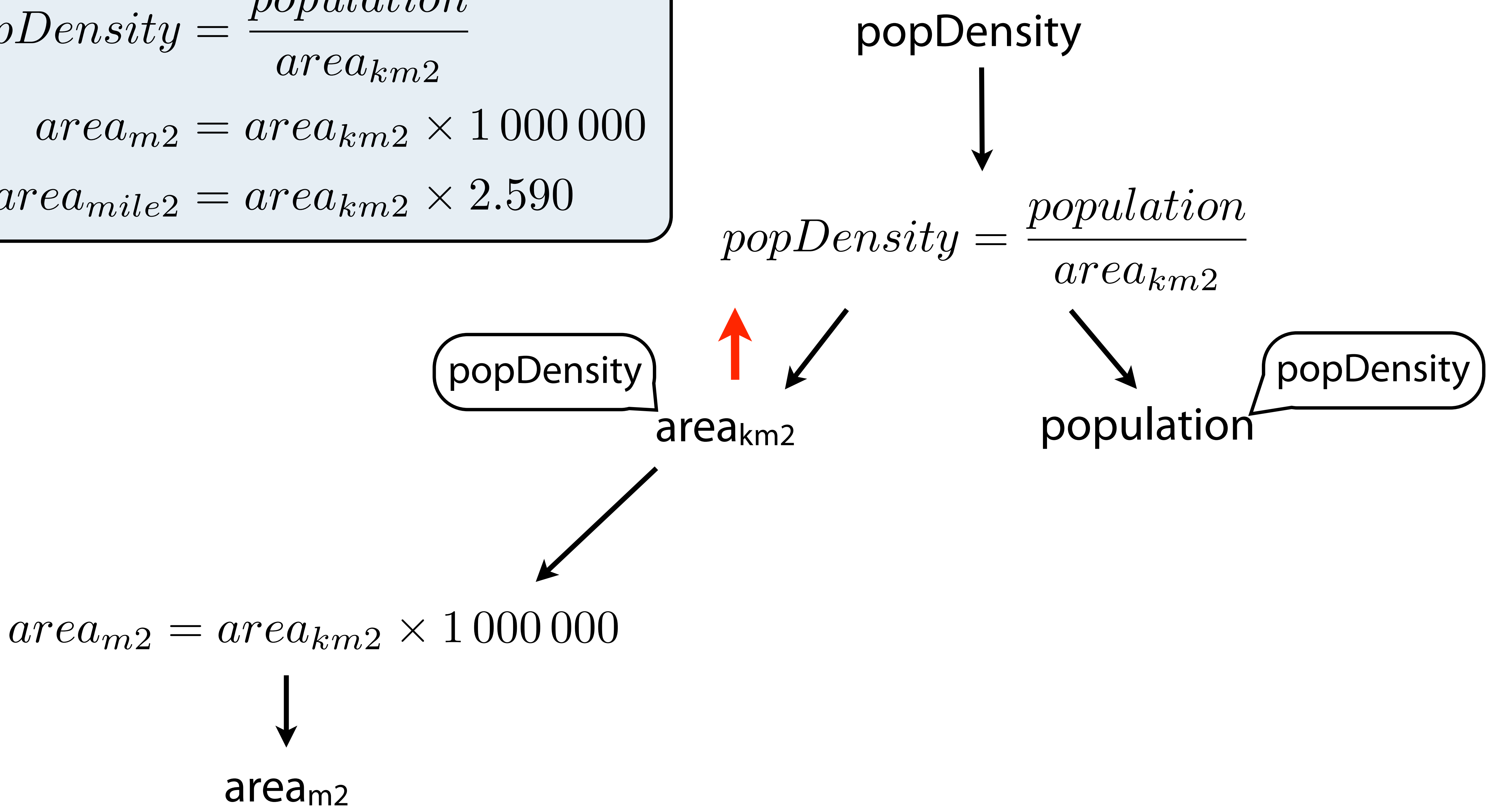
Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$

$$area_{m2} = area_{km2} \times 1\,000\,000$$

$$area_{mile2} = area_{km2} \times 2.590$$



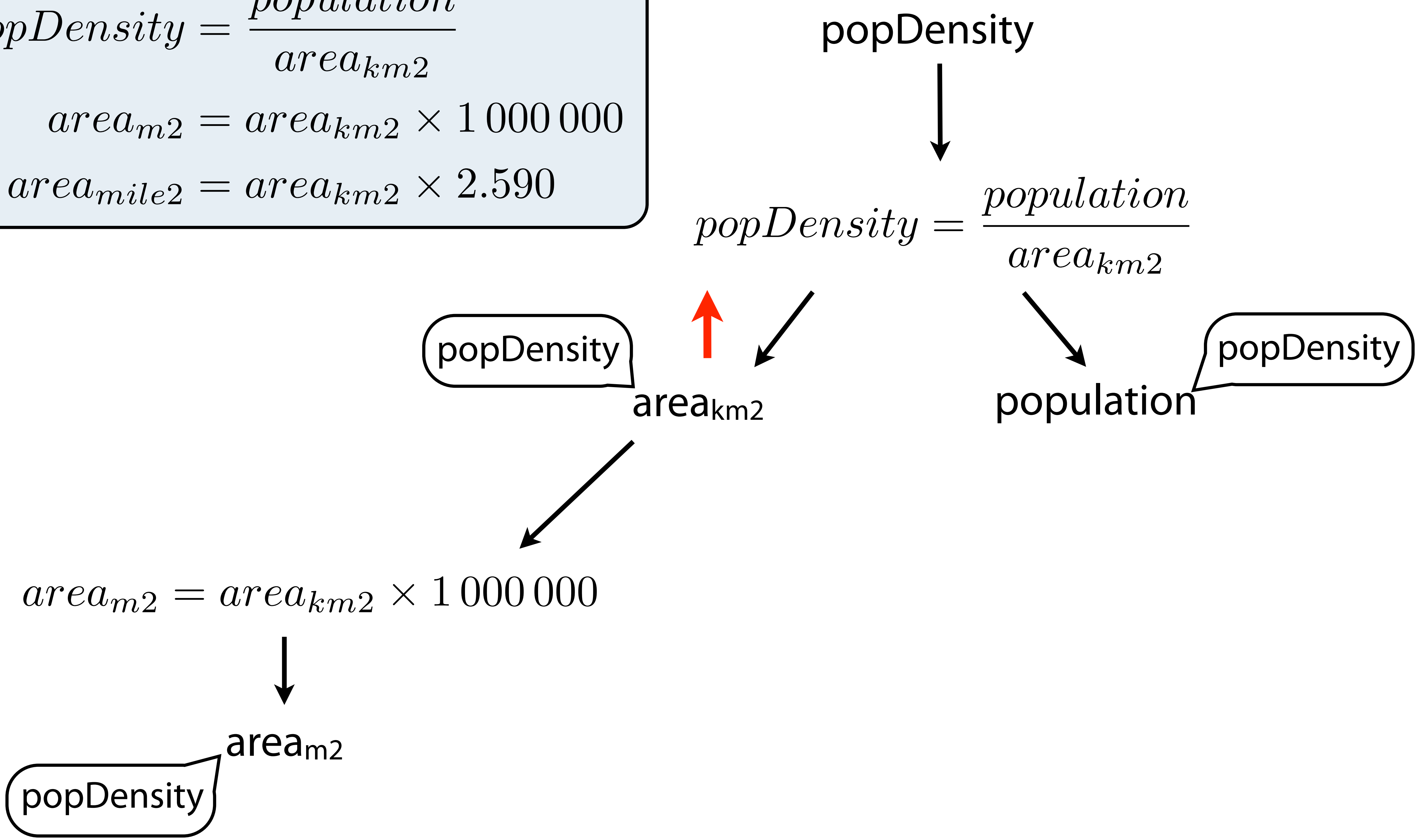
Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$

$$area_{m2} = area_{km2} \times 1\,000\,000$$

$$area_{mile2} = area_{km2} \times 2.590$$



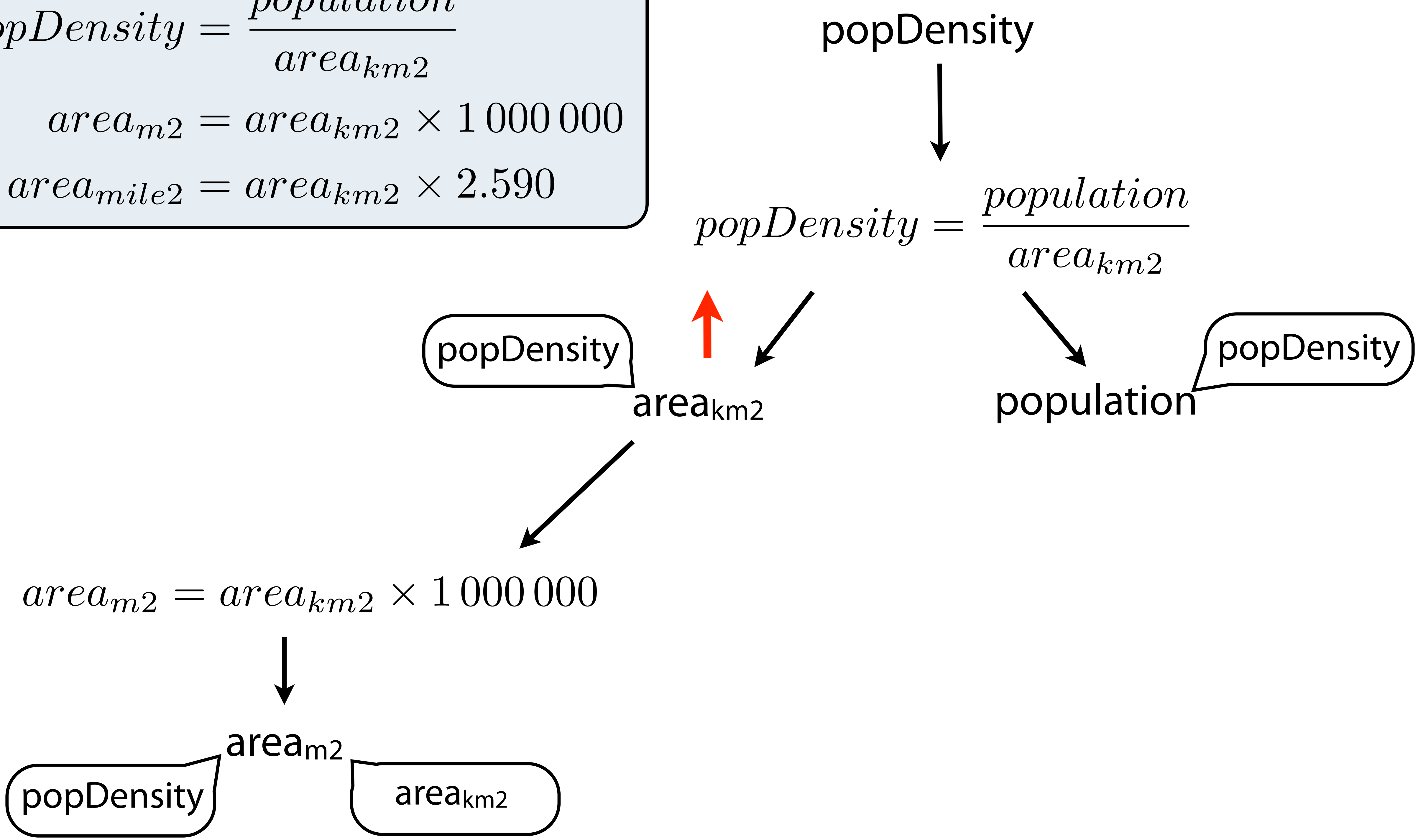
Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$

$$area_{m2} = area_{km2} \times 1\,000\,000$$

$$area_{mile2} = area_{km2} \times 2.590$$



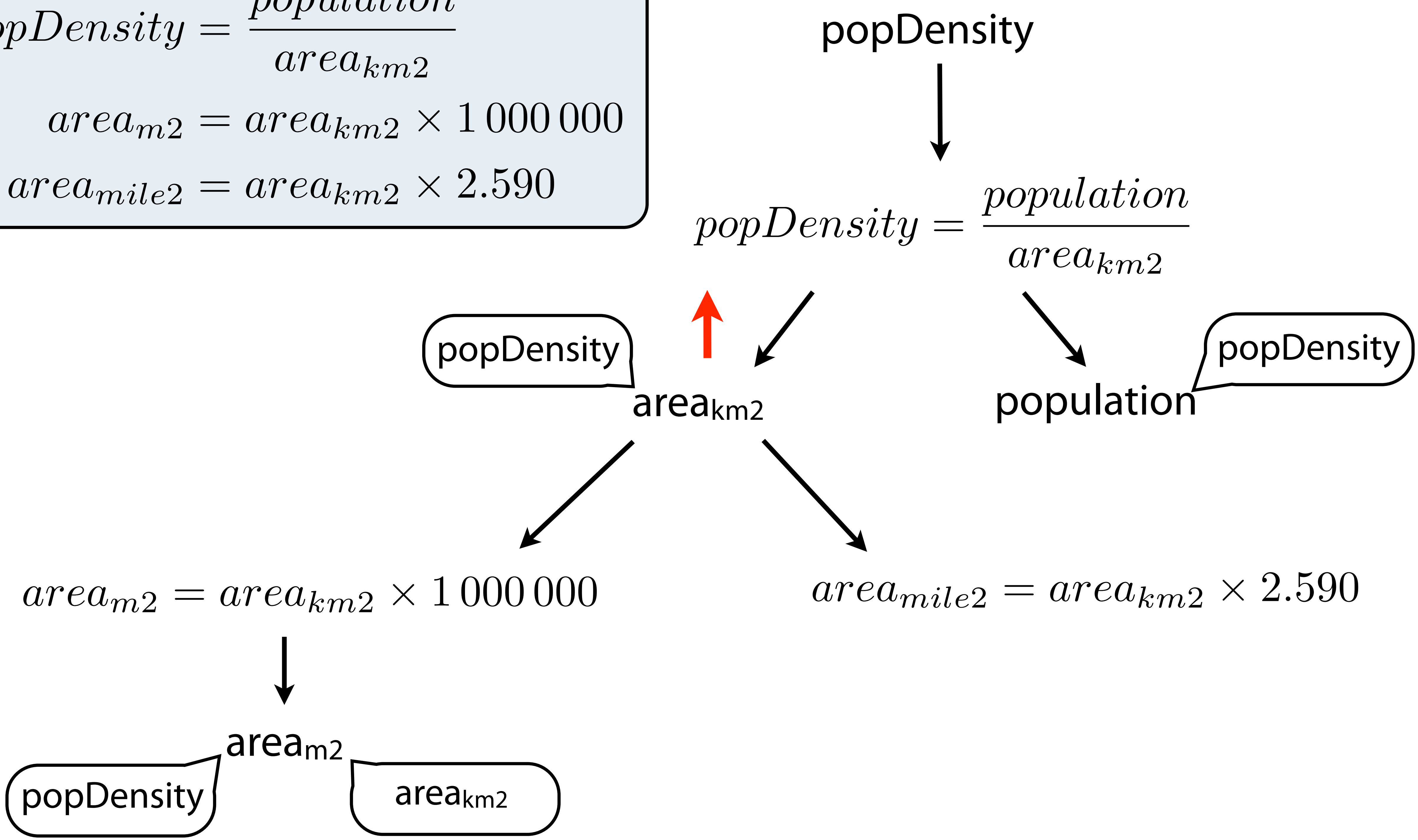
Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$

$$area_{m2} = area_{km2} \times 1\,000\,000$$

$$area_{mile2} = area_{km2} \times 2.590$$



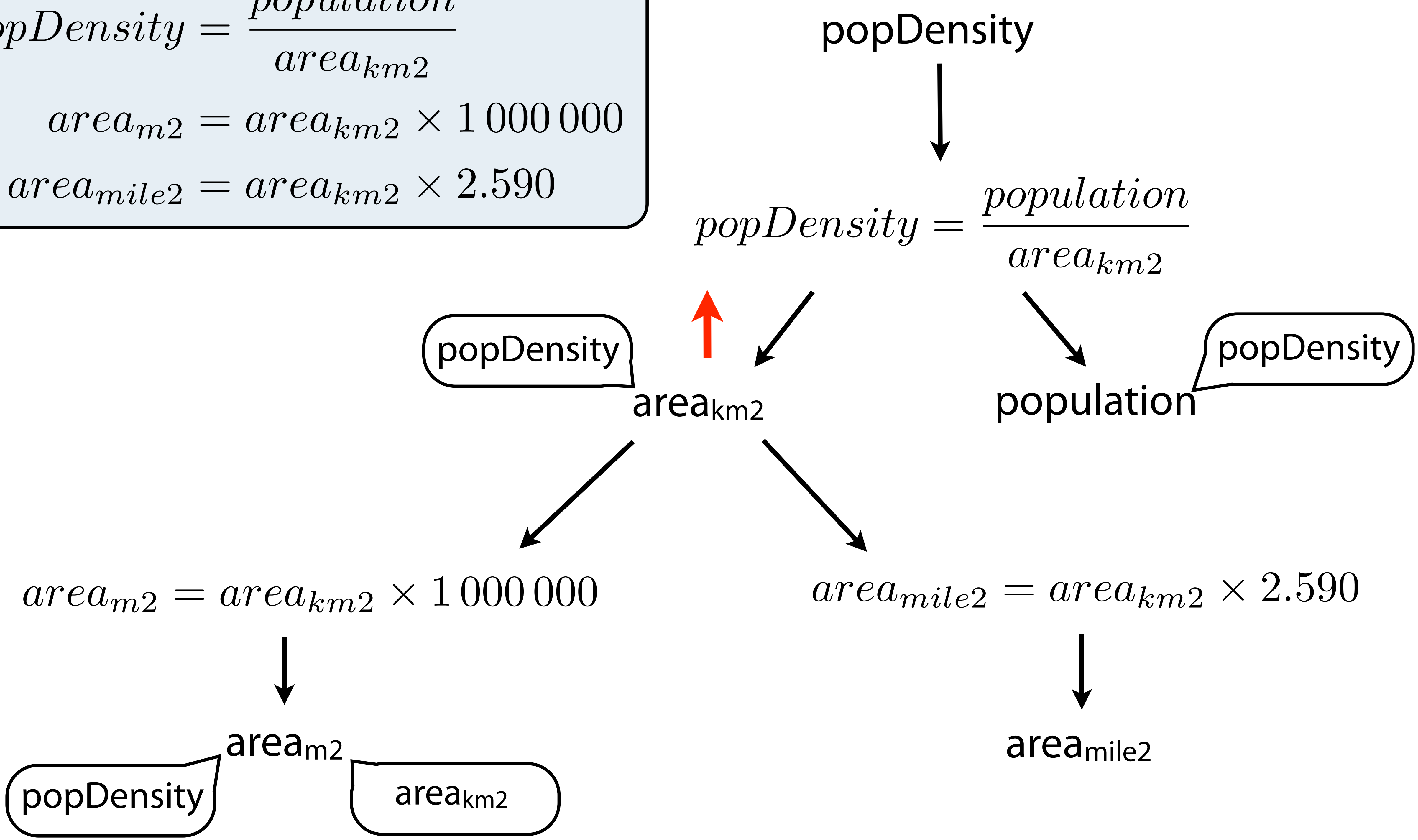
Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$

$$area_{m2} = area_{km2} \times 1\,000\,000$$

$$area_{mile2} = area_{km2} \times 2.590$$



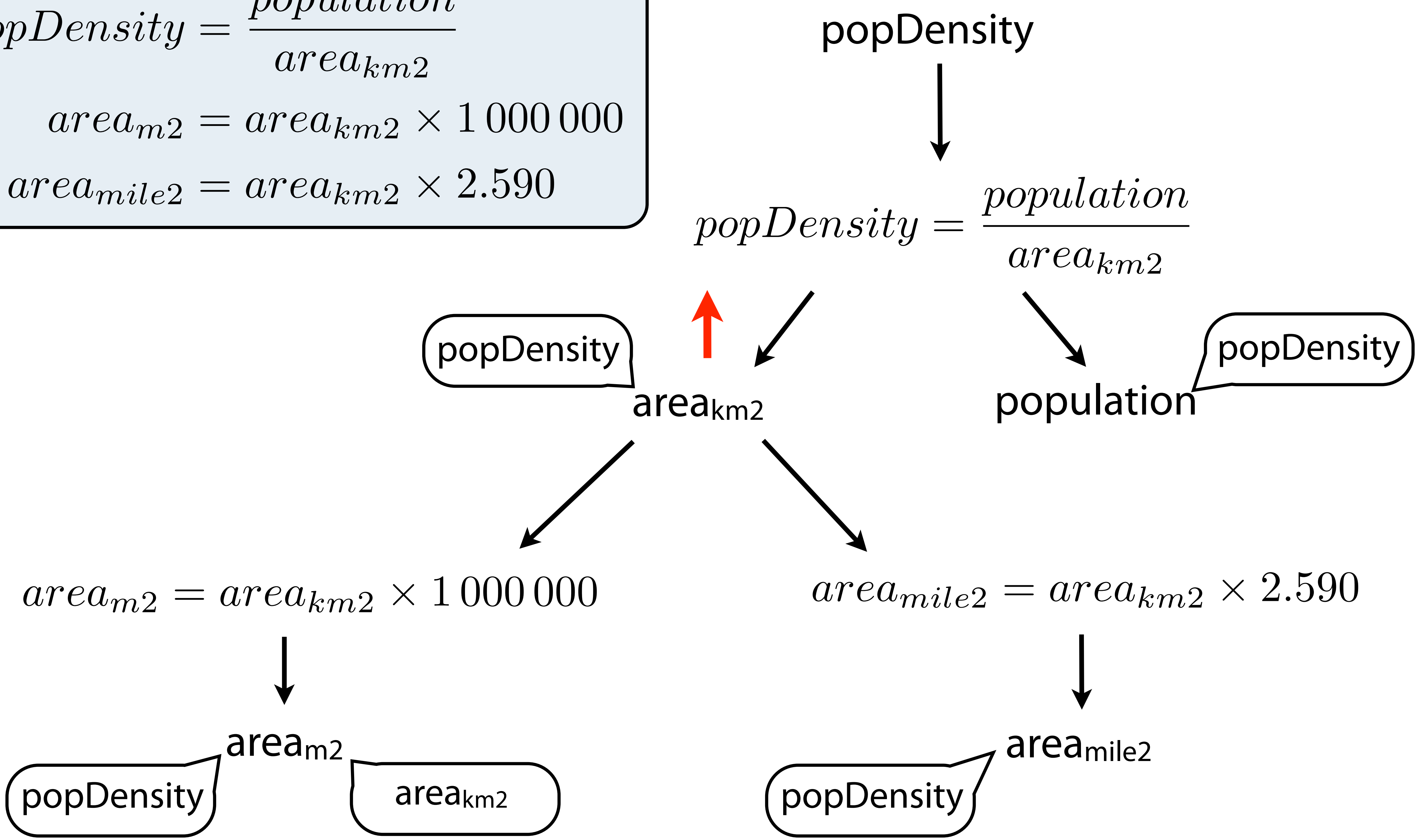
Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$

$$area_{m2} = area_{km2} \times 1\,000\,000$$

$$area_{mile2} = area_{km2} \times 2.590$$



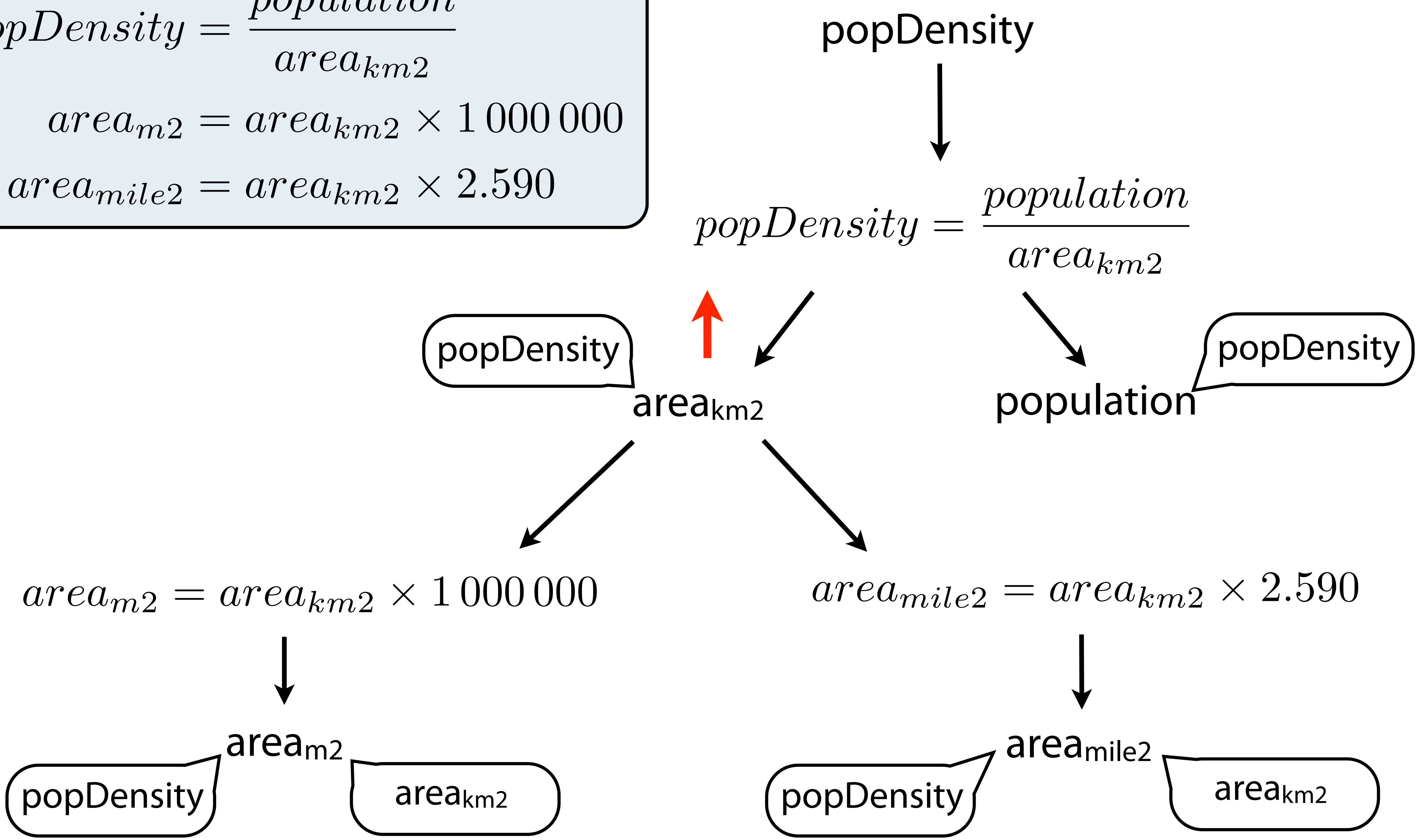
Break the infinite series of equation applications

Adorn attributes by used attributes

$$popDensity = \frac{population}{area_{km2}}$$

$$area_{m2} = area_{km2} \times 1\,000\,000$$

$$area_{mile2} = area_{km2} \times 2.590$$



Extend the DL-Lite PerfectRef algorithm by equations and adorned attributes

```
Input: Conjunctive query q, TBox 𝒯
Output: Union (set) of conjunctive queries
P := {q}
repeat
    P' := P
    foreach q ∈ P' do
        foreach g in q do    // expansion
            foreach inclusion axiom I in 𝒯 do
                if I is applicable to g then
                    P := P ∪ {q[g/ gr(g, I)]}
            foreach equation axiom E in 𝒯 do
                if g = U^adn(g)(x, y) is an (adorned) attribute atom, U ∈ vars(E) and
                vars(E) ∩ adn(g) = ∅ then
                    P := P ∪ {q[g/ expand(g, E)]}
until P' = P
return P
```

PerfectRef with adorned attributes
query rewriting

- Equations

$$E1\colon popDens = \frac{population}{areakm2}$$
$$E2\colon aream2 = areakm2 \times 1\,000\,000$$

- Original query

$popDens(montpellier, X)$

PerfectRef with adorned attributes
query rewriting

- Equations

$$E1\colon popDens = \frac{population}{areakm2}$$
$$E2\colon aream2 = areakm2 \times 1\,000\,000$$

- Original query

$$popDens(montpellier, X)$$

- Step 1: Expand popDens by E1

$$population^{\{popDens\}}(montpellier, P),$$
$$areakm2^{\{popDens\}}(montpellier, A), X = P/A$$

PerfectRef with adorned attributes
query rewriting

- Equations

$$E1\colon popDens = \frac{population}{areakm2}$$
$$E2\colon aream2 = areakm2 \times 1\,000\,000$$

- Original query

$$popDens(montpellier, X)$$

- Step 1: Expand popDens by E1

$$population^{\{popDens\}}(montpellier, P),$$
$$areakm2^{\{popDens\}}(montpellier, A), X = P/A$$

- Step 2: Expand area by E2

$$population^{\{popDens\}}(montpellier, P),$$
$$aream2^{\{popDens,area\}}(montpellier, A1), A = A1 * 1000000, X = P/A$$

PerfectRef^E is sound but incomplete in general

conditions for completeness

- ABox is data-coherent with the TBox
  model of each object has at most one value per attribute
  attribute inclusions must also be considered

- For data-coherent ABoxes wrt. the TBox and
  rewritten SPARQL queries (free of non-distinguished variables)
  PerfectRef^E is sound and complete

PerfectRef^E is sound but incomplete in general

conditions for completeness

```
:M dbp:area_km2  1  .
:M dbp:area_mi2  2.590  .
```

- ABox is data-coherent with the TBox
  model of each object has at most one value per attribute
  attribute inclusions must also be considered

- For data-coherent ABoxes wrt. the TBox and
  rewritten SPARQL queries (free of non-distinguished variables)
  PerfectRef^E is sound and complete

PerfectRef^E is sound but incomplete in general

conditions for completeness

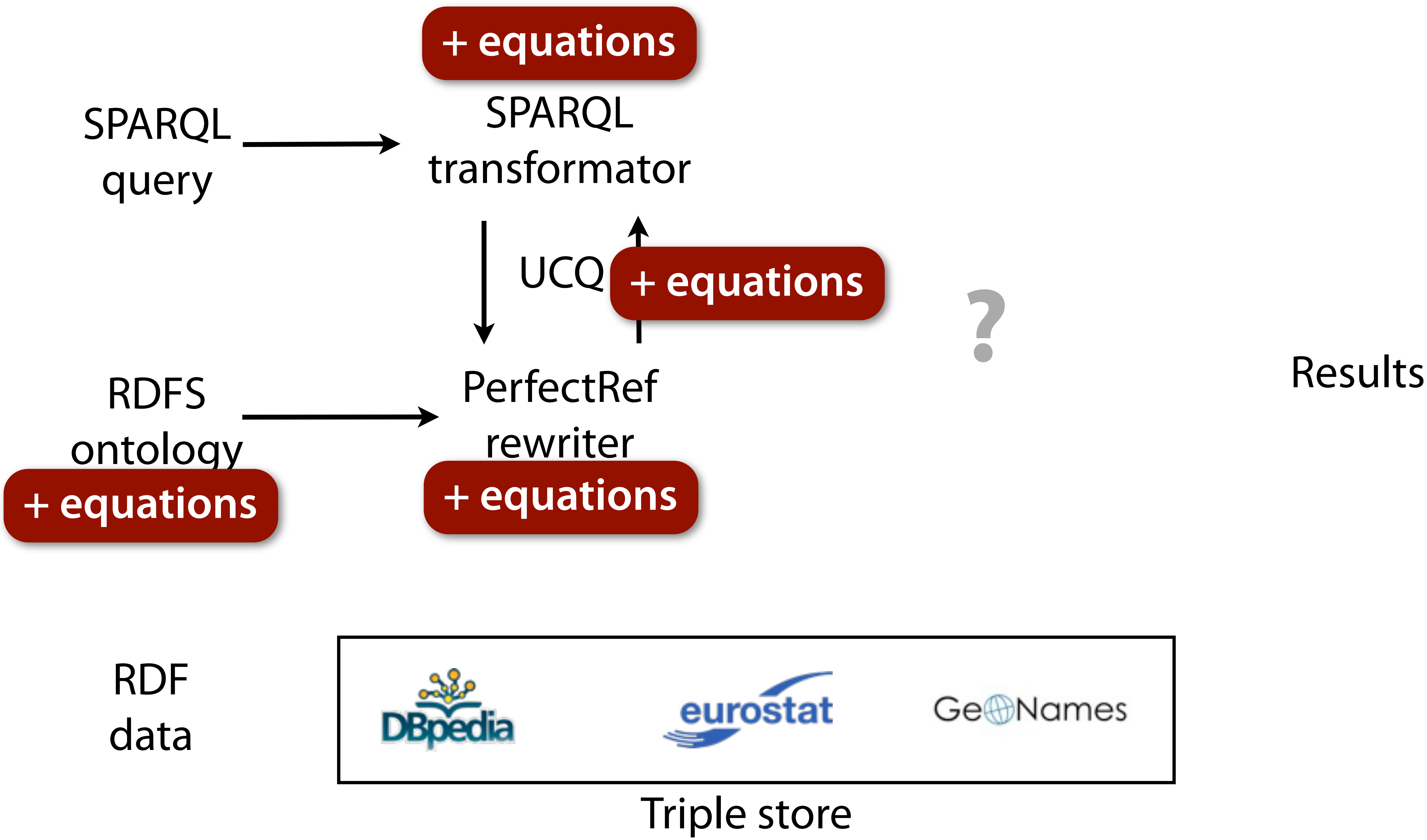
:M dbp:area_km2 1 .
:M dbp:area_mi2 2.590 .

:M dbp:area_km2 1 .
:M dbp:area_mi2 2.6 .

- ABox is data-coherent with the TBox
  model of each object has at most one value per attribute
  attribute inclusions must also be considered

- For data-coherent ABoxes wrt. the TBox and
  rewritten SPARQL queries (free of non-distinguished variables)
  PerfectRef^E is sound and complete

RDFS with Attribute Equations via SPARQL Rewriting
the big picture



Rewrite SPARQL queries by PerfectRef

- SPARQL basic graph patterns (BGPs)
  the fundamental building block for graph pattern matching
- BGPs can be expressed by conjunctive queries [Perez et al., 2009]
  no variables as predicates `:Montpellier ?prop "Montpellier"`
  no variables for classes `:Montpellier rdf:type ?class`
- Convert BGPs to CQs, rewrite CQs to UCQs, convert UCQs to SPARQL
- SPARQL translator
  rewrite each BGP independently by PerfectRef^{E}
- Variable assignments are rewritten to SPARQL 1.1 BIND

Rewrite SPARQL query by PerfectRef

rewritten SPARQL query

- CQ 1: $popDens(montpellier, X)$
- CQ 2: $population^{\{popDens\}}(montpellier, P),$
  $area^{\{popDens\}}(montpellier, A), X = P/A$
- CQ 3: $population^{\{popDens\}}(montpellier, P), A = A1 * 1000000,$
  $aream2^{\{popDens,area\}}(montpellier, A1), X = P/A$

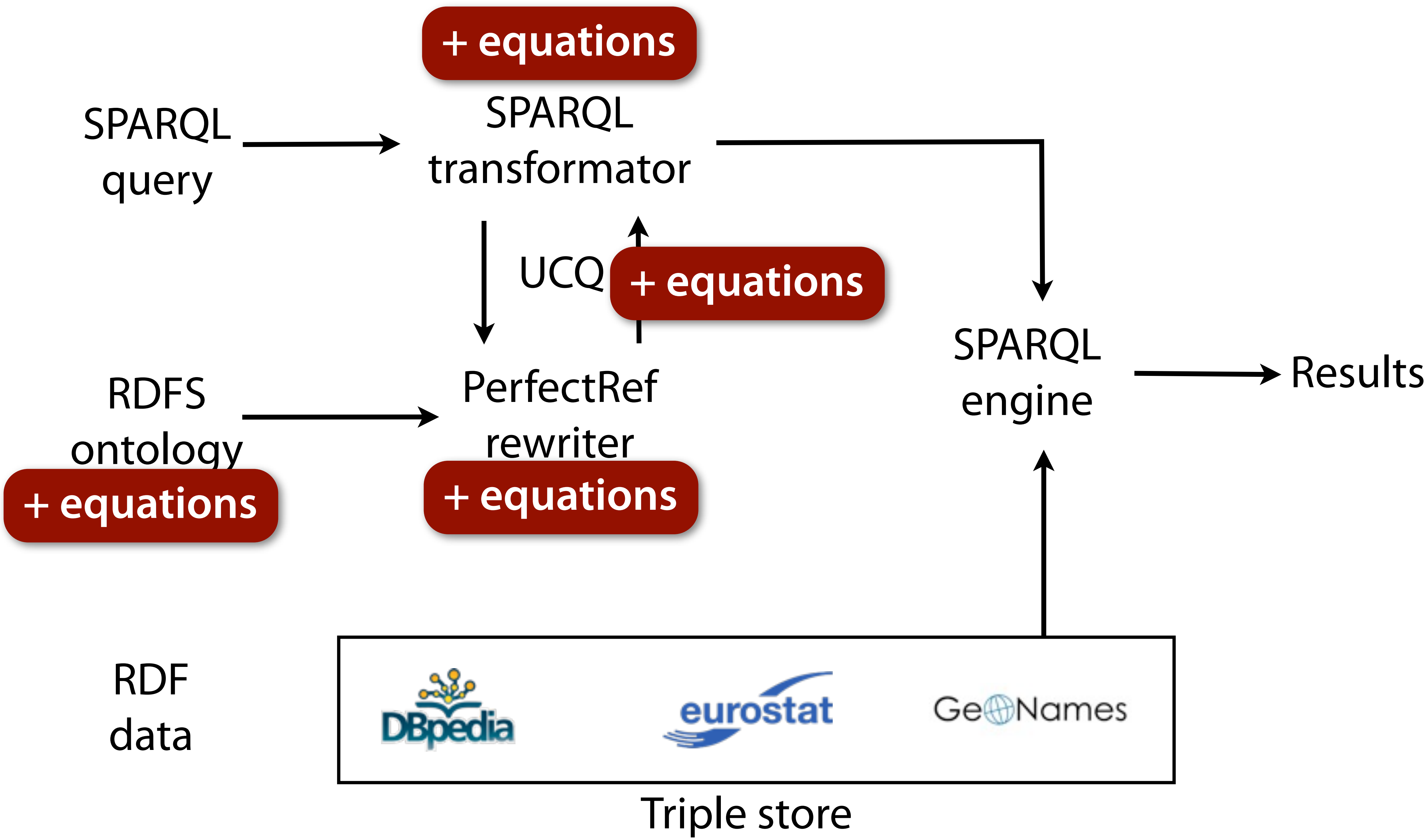
```
SELECT ?X WHERE
  { :Montpellier dbo:populationDensity ?X . }
```

Rewrite SPARQL query by PerfectRef
rewritten SPARQL query

- CQ 1: $popDens(montpellier, X)$
- CQ 2: $population^{\{popDens\}}(montpellier, P),$
  $area^{\{popDens\}}(montpellier, A), X = P/A$
- CQ 3: $population^{\{popDens\}}(montpellier, P), A = A1 * 1000000,$
  $aream2^{\{popDens,area\}}(montpellier, A1), X = P/A$

```
SELECT ?X WHERE {
  { :Montpellier dbo:populationDensity ?X . }
  UNION
  { :Montpellier dbo:populationTotal ?p ; dbp:areaTotalKm ?a .
    BIND (?p/?a as ?X) }
  UNION
  { :Montpellier dbo:populationTotal ?p ; dbo:area ?a2 .
    BIND (?a2/1000000 as ?a) BIND (?p/?a as ?X) }
}
```

RDFS with Attribute Equations via SPARQL Rewriting
the big picture



How does the rewriting algorithm perform on real world data?

- Collected data about cities
  from several sources (e.g., DBpedia, Eurostat)
  254 081 triples for 3161 city contexts
  inconsistent and consistent dataset

- 6 equations, 2 subProperties and 1 subClass axioms

- 4 different queries

- 3 implementations
  Jena forward rules with ARQ
  Jena forward rules with ARQ with noValue
  Rewriting with ARQ

How does the rewriting algorithm perform on real world data?

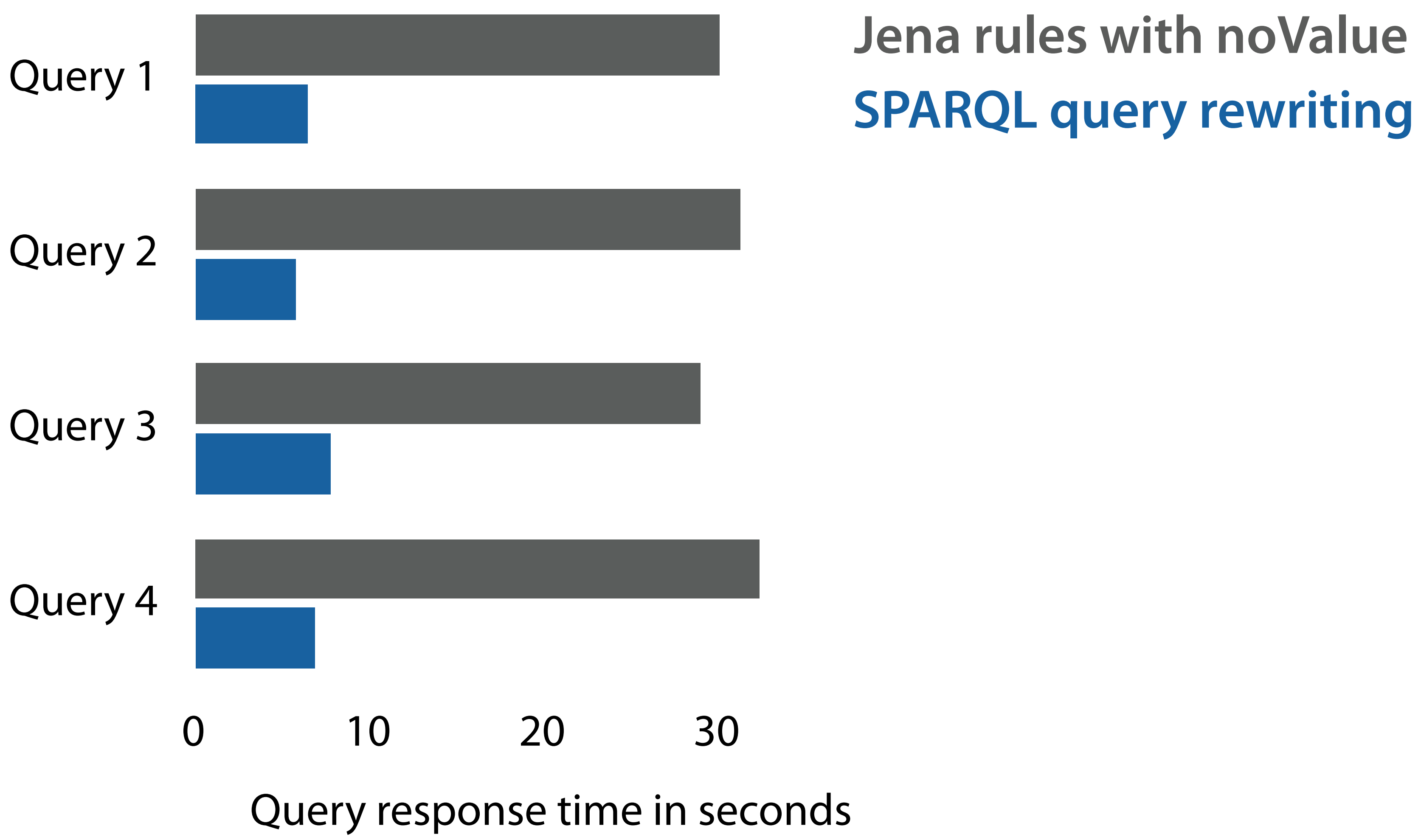
- Collected data about cities
  from several sources (e.g., DBpedia, Eurostat)
  254 081 triples for 3161 city contexts
  inconsistent and con
- 6 equations, 2 subPro
- 4 different queries
- 3 implementations
  Jena forward rules with ARQ
  Jena forward rules with ARQ with noValue
  Rewriting with ARQ

```
(?city :area ?ar)
(?city :populationDensity ?pd)
product(?ar, ?pd, ?p)
noValue (?city, :populationDensity)
-> (?city :population ?p)
```

Jena rules with ARQ gives no results

- *n* rules for an equation in *n* variables
- Forward chaining implementation
- No query returned any result within 10 minutes
- Even for reduced dataset

Rewriting is significantly faster than Jena rules with noValue



Conclusions

- Reasoning about equations on numerical properties is important and feasible
  lots of numeric open data available
- Rule engines are not well suited for such attribute equations
  especially on real world data
- Query rewriting enables such reasoning on top of off-the-shelf SPARQL engines
  also possible on public SPARQL endpoints
- Query rewriting can be significantly faster
  than forward chaining rule engines

Do not query for concrete values

query for ranges instead

- Equations are not constraints; this will not work

```
SELECT ?city WHERE { ?city dbo:populationDensity 4447.93. }
```

- Instead use filter

```
SELECT ?city WHERE {
  ?city dbo:populationDensity ?dens.
  FILTER(?dens = 4447.93 }
```

- Can you guess all the correct digits after the comma?
  solution with filter will hardly return any results

- Instead query for ranges

```
SELECT ?city WHERE {
  ?city dbo:populationDensity ?dens.
  FILTER(?dens > 4440.0 && ?dens < 4450.0 }
```