A Short History of Schema Mapping Systems

G. Mecca[1] - P. Papotti[2] - D. Santoro[1]

[1] Università della Basilicata [2] Qatar Computing Research Institute

1975

1980

1975

1977

1980

EXPRESS

1975

1977

1980

EXPRESS

1975

1977

1980

PREHISTORIC

HERO

EXPRESS

1975

1977

1980

PREHISTORIC

HERO

5

2000

200

HISTORIC

HEROIC

SILVER

5

2000

2002

200

HISTORIC

HEROIC

SILVER

Schema Mapping as Query Discovery

Translating Web Data

5

2000

2002

200

HISTORIC

HEROIC

SILVER

Proc. 2002 Very Large Databases Conference (VLDB'02), pp. 598-609

Translating Web Data

Lucian Popa[†] Yannis Velegrakis[‡] Renée J. Miller[‡] Mauricio A. Hernández[†] Ronald Fagin[†]

[†]IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120

[‡]University of Toronto
6 King's College Road
Toronto, ON M5S 3H5

Abstract

We present a novel framework for mapping between any combination of XML and relational schemas, in which a high-level, user-specified mapping is translated into semantically meaningful queries that transform source data into the target representation. Our approach works in two phases. In the first phase, the high-level mapping, expressed as a set of inter-schema correspondences, is converted into a set of mappings that capture the design choices made in the source and target schemas (including their hierarchical organization as well as their nested referential constraints). The second phase translates these mappings into queries over the source schemas that produce data satisfying the constraints and structure of the target schema, and preserving the semantic relationships of the source. Non-null target values may need to be invented in this process. The mapping algorithm is complete in that it produces **all** mappings that are consistent with the schema constraints. We have implemented the translation algorithm in Clio, a schema mapping tool, and present our experience using Clio on several real schemas.

1 Introduction

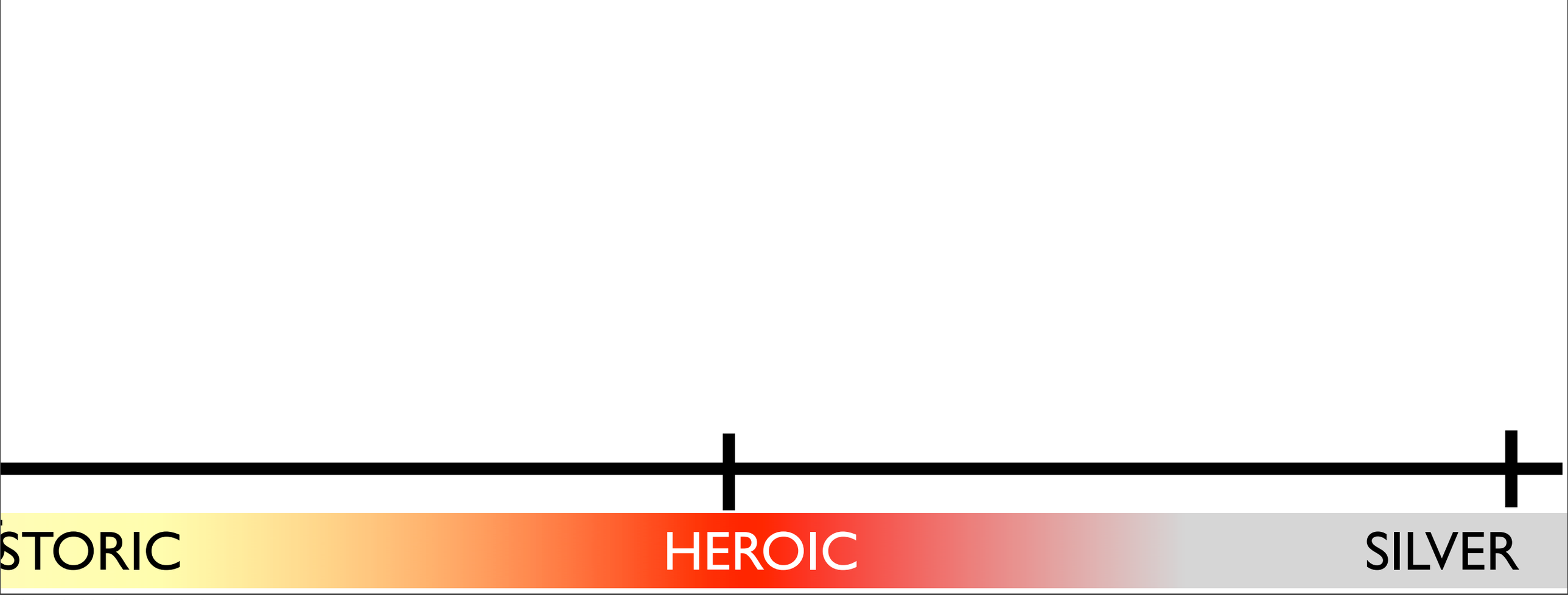
An important issue in modern information systems and e-commerce applications is providing support

semi-structured formats (e.g., DTDs, SGML or XML Schema), scientific formats, etc. Integration of such data is an increasingly important problem. Nonetheless, the effort involved in such integration, in practice, is considerable: translation of data from one format (or schema) to another requires writing and managing complex data transformation programs or queries.

We present a new, comprehensive solution to building, refining and managing transformations between heterogeneous schemas. Given the prevalent use of the Web for data exchange, any data translation tool must handle not only relational data but also data represented in nested data models that are commonly available on the Web. Our solutions are applicable to any structured and semi-structured data that can be described by a schema (a relational schema, a nested XML Schema or DTD). We do not, in this work, consider the exchange of documents or unstructured data (including multimedia and unstructured text). Our approach can be distinguished by its treatment of two fundamental issues. We discuss them below, highlighting our contributions and the main related work.

Heterogeneous Semantics We consider the *schema mapping* problem, where we are given a pair of independently created schemas and asked to translate data from one (the source) to the other (the target). The schemas may have different semantics, and this may be reflected in differences in their logical structures and constraints. In contrast, most work on heterogeneous data focuses on the *schema integration* problem where the target (global) schema is created from one or more source (local) schemas (and designed as a view

Schema Mapping and Data Exchange



Schema Mapping and Data Exchange

Library1

Book1 [0..*]

title
year

Library2

Book2 [0..*]

title
author
year

Library3

Book3 [0..*]

title
authorId

Author3 [0..*]

id
name

Schema Mapping and Data Exchange

Library1

Book1 [0..*]

title
year

title	year
On The Road	1951
Post Office	1971

Library2

Book2 [0..*]

title
author
year

title	author	year
Post Office	Bukowski	1971
Animal Farm	Orwell	1947

Library3

Book3 [0..*]

title
authorId

Author3 [0..*]

id
name

title	authId
Animal Farm	951
On The Road	654

id	name
951	Orwell
654	Kerouac

STORIC HEROIC SILVER

Schema Mapping and Data Exchange

Library1

Book1 [0..*]

title
year

title	year
On The Road	1951
Post Office	1971

Library2

Book2 [0..*]

title
author
year

title	author	year
Post Office	Bukowski	1971
Animal Farm	Orwell	1947

Library3

Book3 [0..*]

title
authorId

Author3 [0..*]

id
name

title	authId
Animal Farm	951
On The Road	654

id	name
951	Orwell
654	Kerouac

Target DB

Book [0..*]

title
year
authorId

Author [0..*]

id
name

STORIC HEROIC SILVER

Schema Mapping and Data Exchange

Library1

Book1 [0..*]

title
year

title	year
On The Road	1951
Post Office	1971

Library2

Book2 [0..*]

title
author
year

title	author	year
Post Office	Bukowski	1971
Animal Farm	Orwell	1947

Library3

Book3 [0..*]

title
authorId

Author3 [0..*]

id
name

title	authId
Animal Farm	951
On The Road	654

id	name
951	Orwell
654	Kerouac

Target DB

Book [0..*]

title
year
authorId

Author [0..*]

id
name

STORIC — HEROIC — SILVER

Schema Mapping and Data Exchange

Library1

Book1 [0..*]

title
year

Library2

Book2 [0..*]

title
author
year

Library3

Book3 [0..*]

title
authorId

Author3 [0..*]

id
name

title	year	authId
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???

id	name
???	???
???	???
???	???

Target DB

Book [0..*]

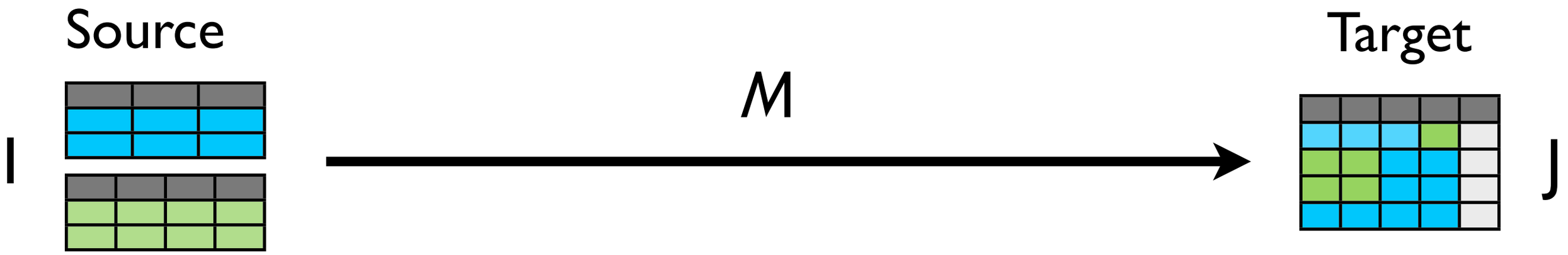
title
year
authorId

Author [0..*]

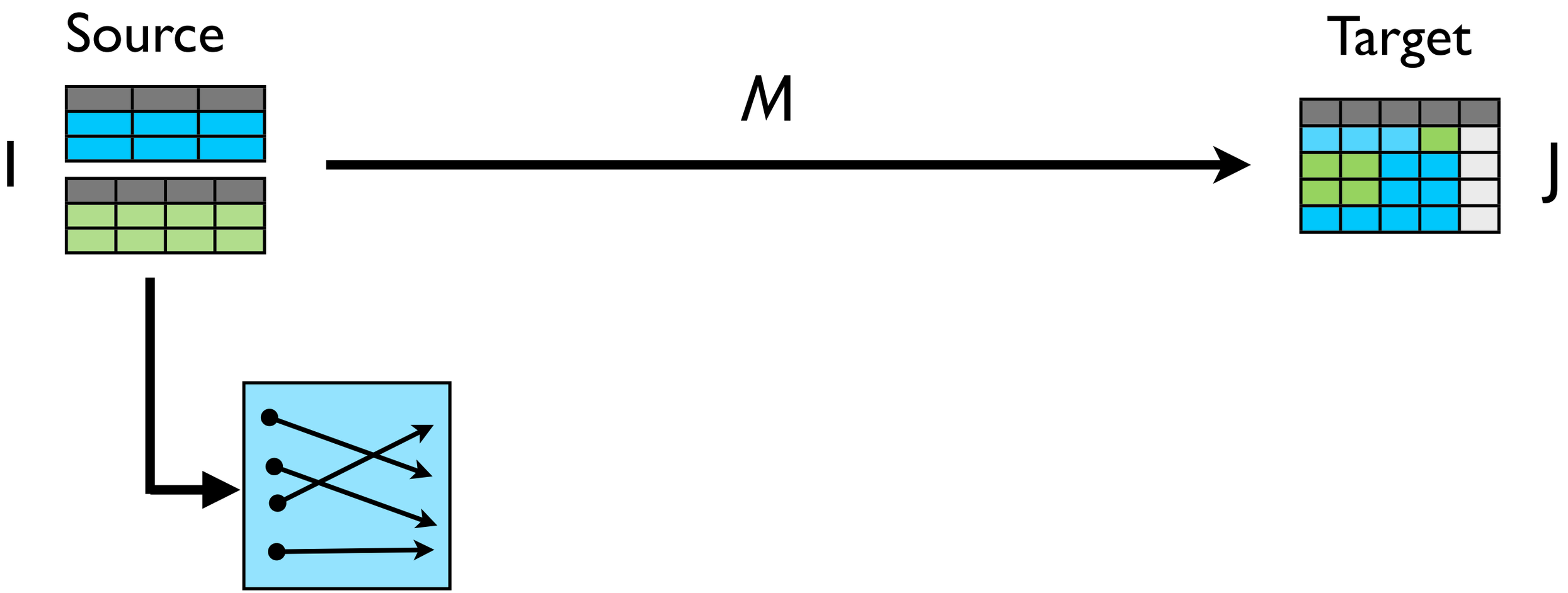
id
name

STORIC HEROIC SILVER

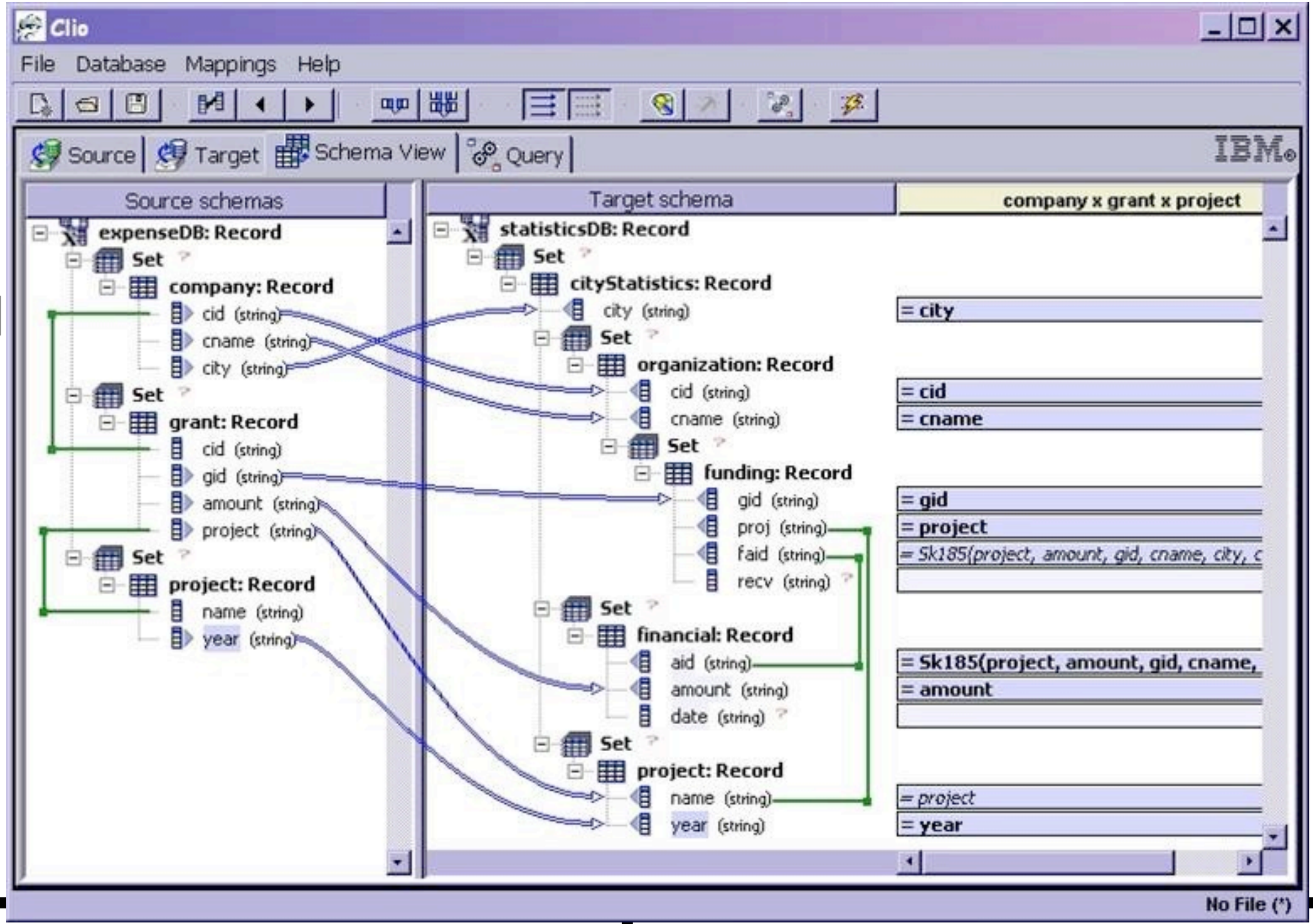
Schema Mapping System



Schema Mapping System

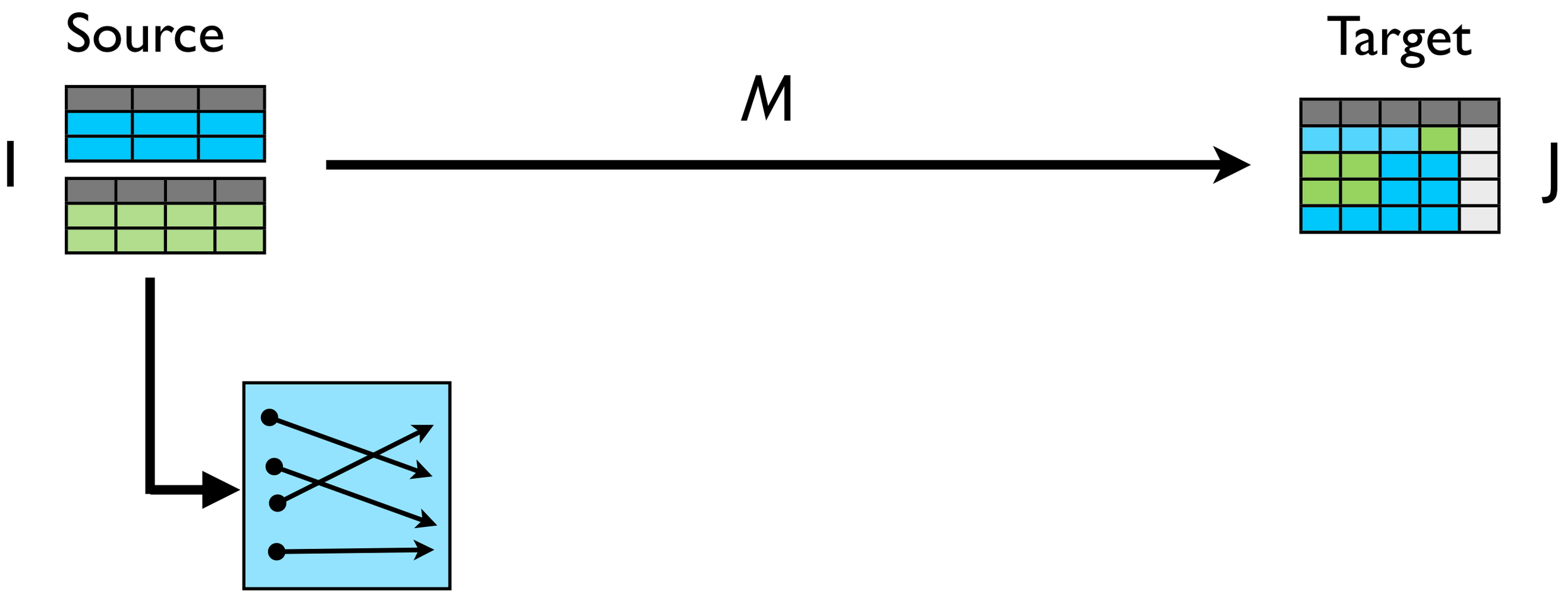


STORIC HEROIC SILVER



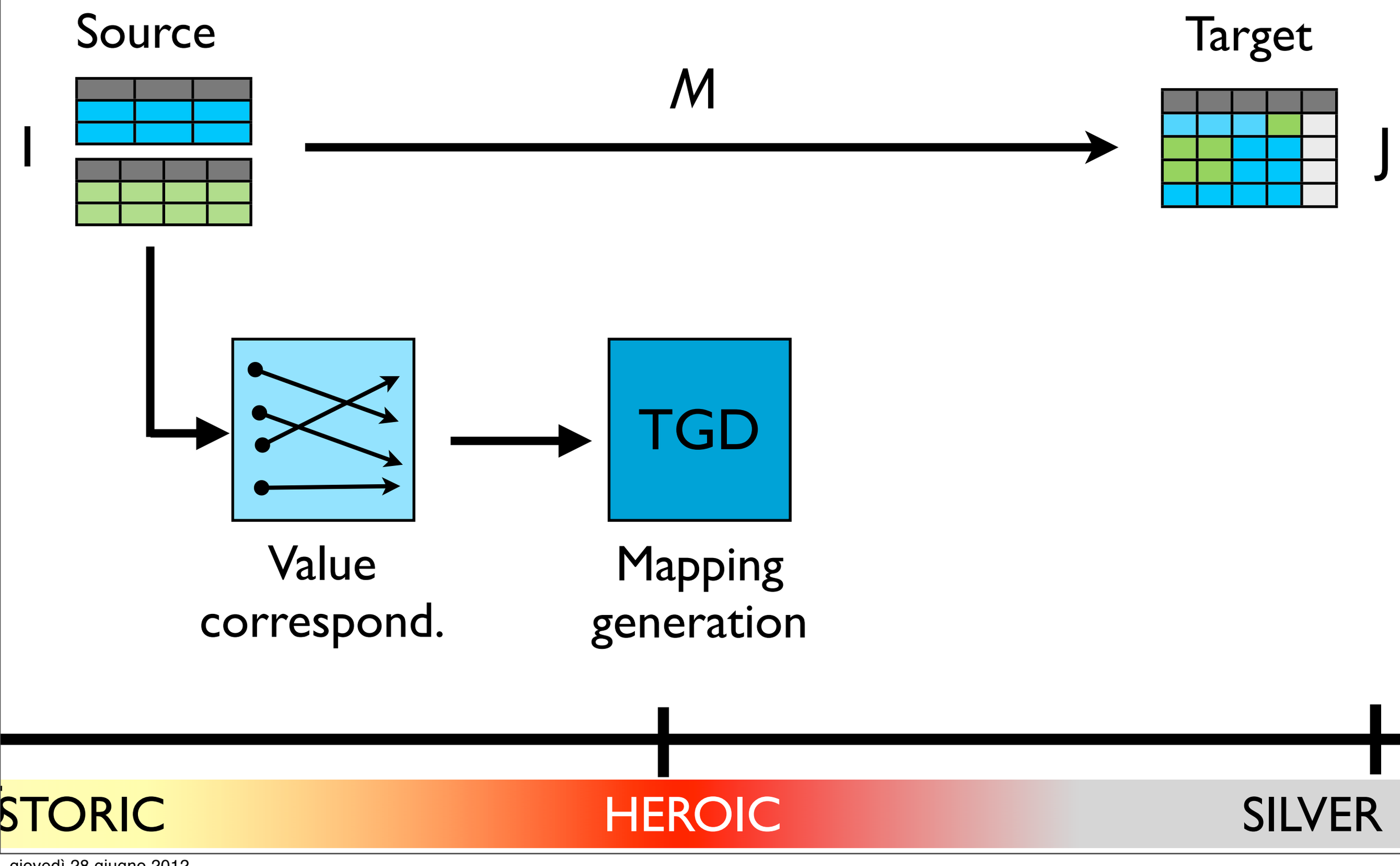
STORIC HEROIC SILVER

Schema Mapping System



STORIC HEROIC SILVER

Schema Mapping System



Schema Mapping System

- Source-to-Target TGD
  - logical formula used to constrain how data should appear in the target based on data in the source

$\forall$ t, y: Book1(t, y) $\rightarrow$ $\exists$N: Book(t, y, N)

$\forall$ t, a, y: Book2(t, a, y) $\rightarrow$ $\exists$N: Book(t, y, N) Author (N,a)

$\forall$ t, i : Book3 (t, i) $\rightarrow$ $\exists$N: Book(t, N, i)

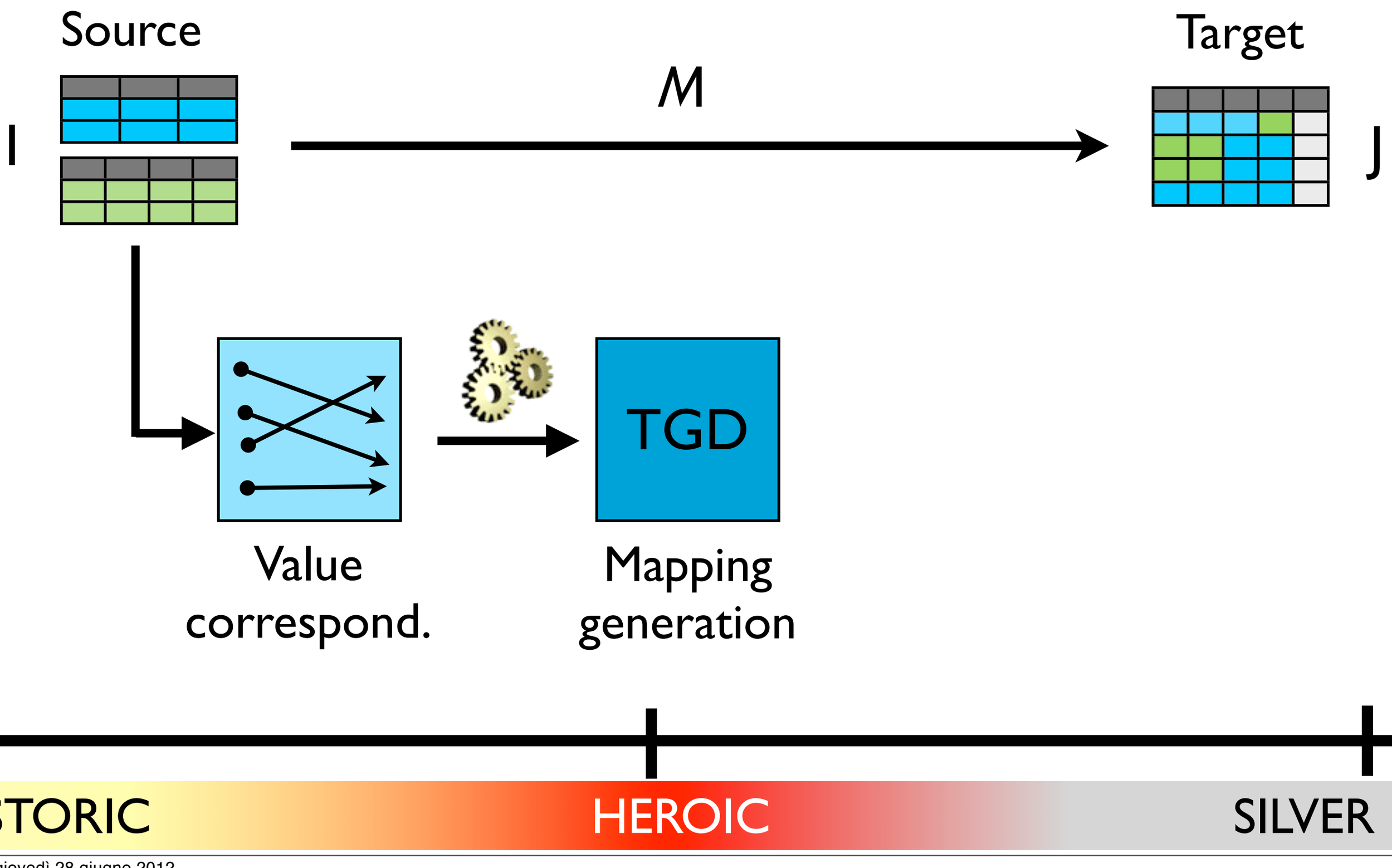
$\forall$ i, a : Author3(i, n) $\rightarrow$ Author (i, n)

- $\forall$ t, y, i : Book (t, y, i) $\rightarrow$ $\exists$N: Publisher(i, N)

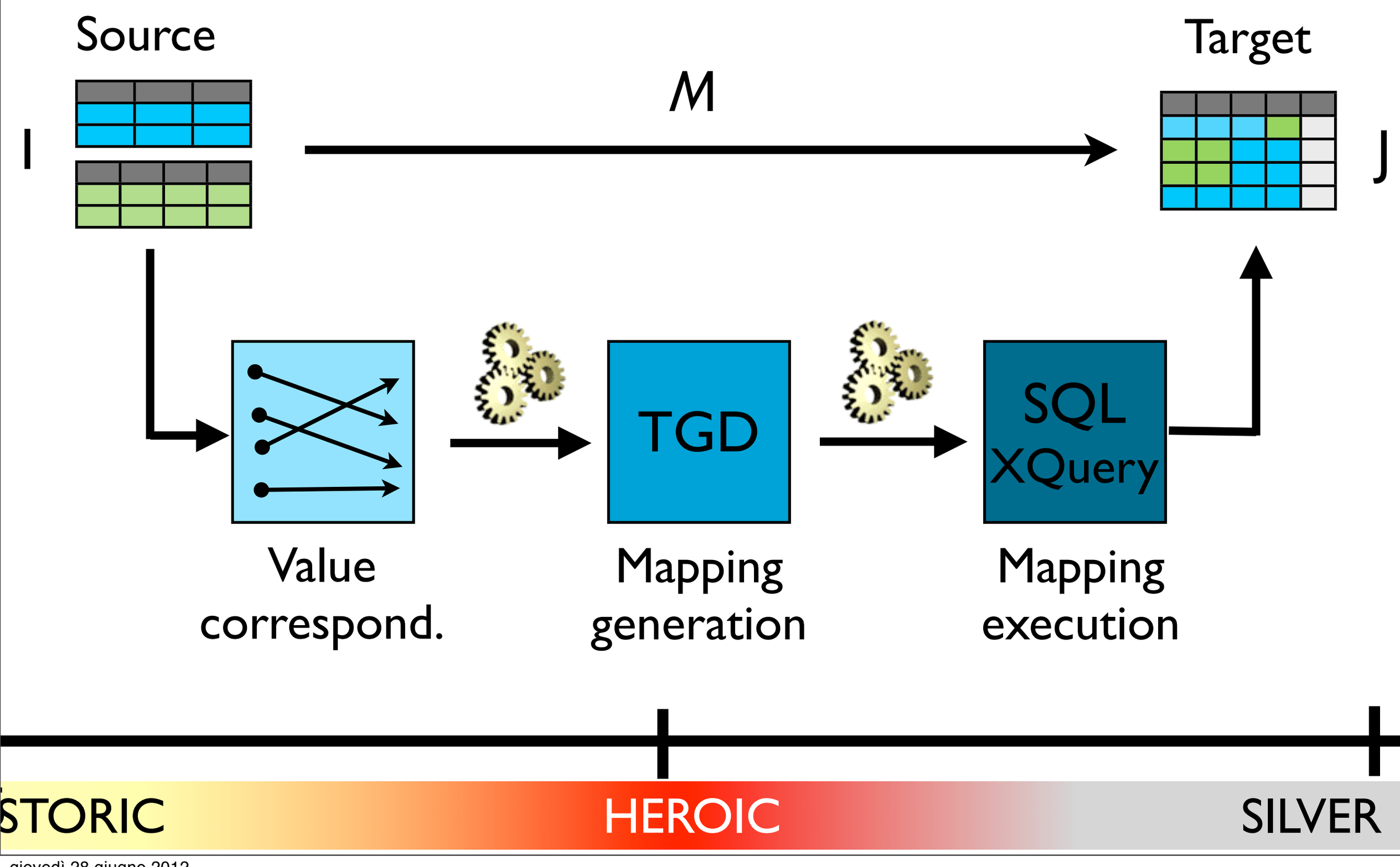
  $\forall$ t, i, y : Book (t, y, i), Book (t, y', i') $\rightarrow$ **(i = i') (y = y')**

STORIC HEROIC SILVER

Schema Mapping System



Schema Mapping System



Sche … stem

Source

Target

Value
correspor…

…lapping
…ecution

SQL
Query

…STORIC

SILVER

Clio

File Database Mappings Help

Source | Target | Schema View | Query | IBM

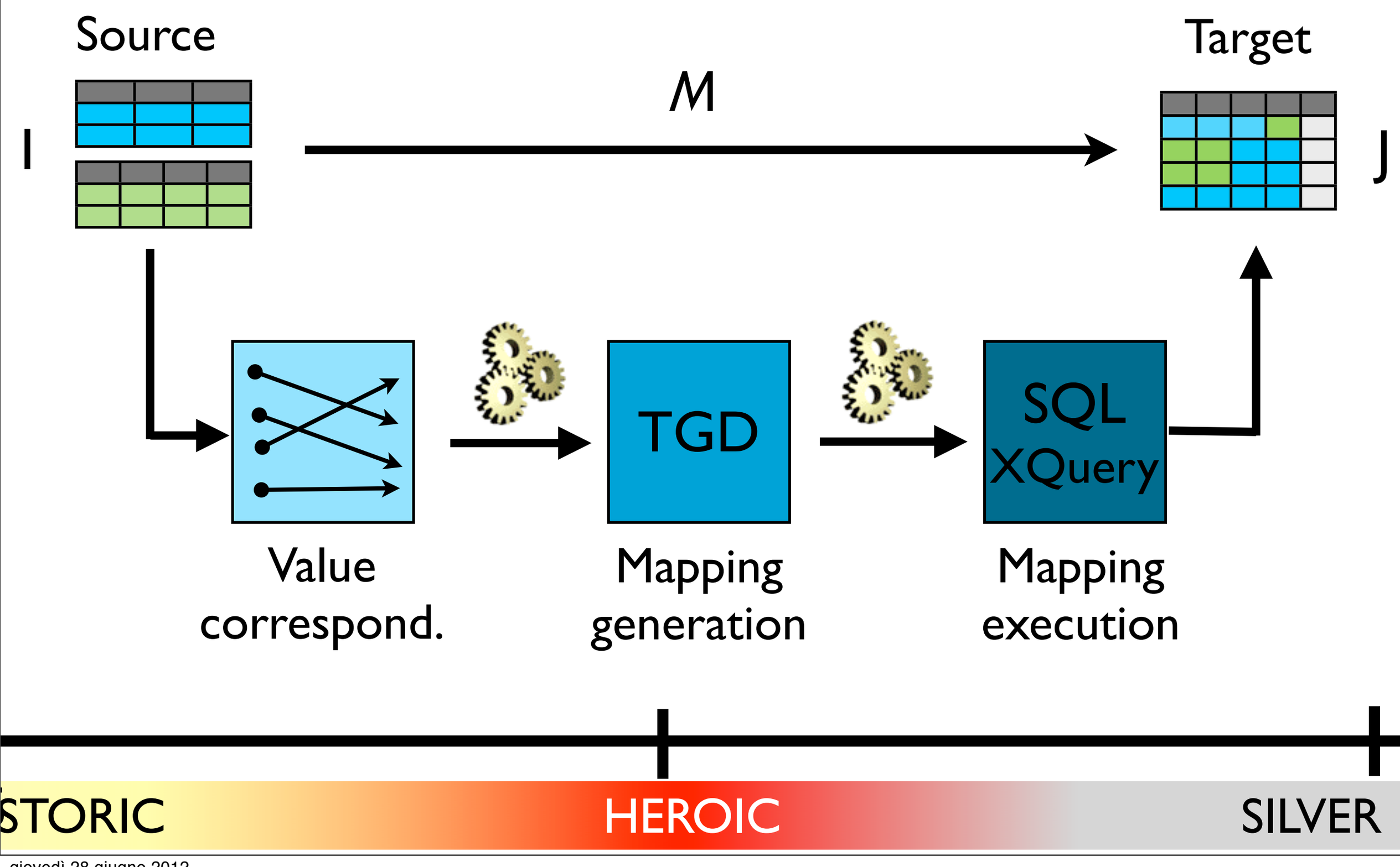
XQuery XSL SQL Query

```
<statisticsDB>
   distinct (
   FOR
     $x0 IN $doc/expenseDB/grant,
     $x1 IN $doc/expenseDB/project,
     $x2 IN $doc/expenseDB/company
   WHERE
      $x0/project/text() = $x1/name/text() AND
      $x2/cid/text() = $x0/cid/text()
   RETURN
      <cityStatistics>
         distinct (
         FOR
           $x0L1 IN $doc/expenseDB/grant,
           $x1L1 IN $doc/expenseDB/project,
           $x2L1 IN $doc/expenseDB/company
         WHERE
            $x0L1/project/text() = $x1L1/name/text() AND
            $x2L1/cid/text() = $x0L1/cid/text() AND
            $x2/city/text() = $x2L1/city/text()
         RETURN
            <organization>
               <cid>   $x0L1/cid/text()   </cid>,
               <cname>   $x2L1/cname/text()   </cname>,
               distinct (
               FOR
                 $x0L2 IN $doc/expenseDB/grant,
```

Preview | Execute Query | Copy to Clipboard

No File (*)

Schema Mapping System



Evolution of schema-mappings and data exchange systems

STORIC HEROIC SILVER

Evolution of schema-mappings and data exchange systems

	First Generation	Intermediate Generation (post-processing)	Intermediate Generation (rewriting)	Second Generation
Scalable Portable	✓			
Quality	✗			
Functional Dependencies	✗			
Nested Relation	✓			

Popa et al., VLDB 2002

STORIC HEROIC SILVER

5

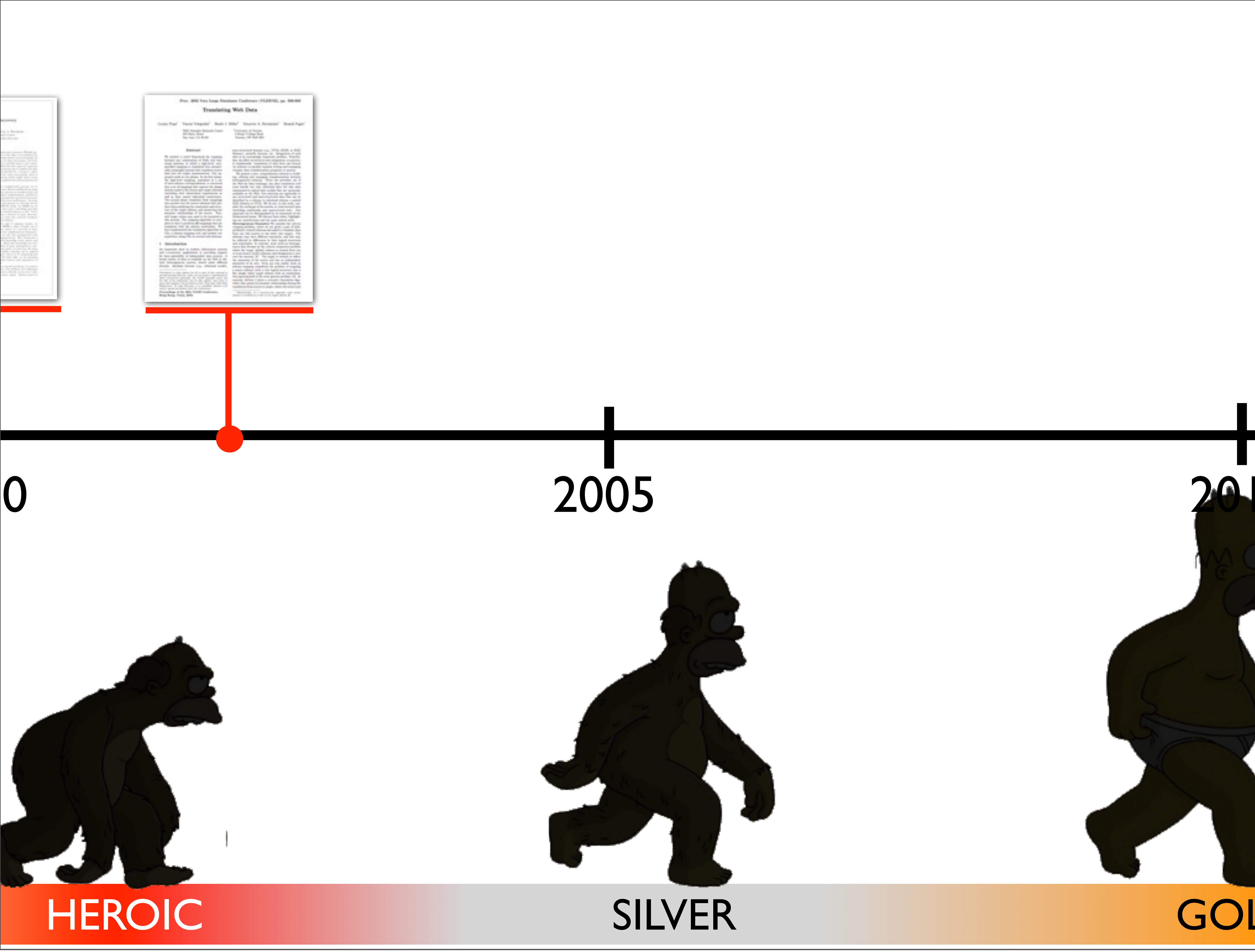
2000

200

HISTORIC

HEROIC

SILVER


Translating Web Data
0
2005
201
HEROIC
SILVER
GOL


Proc. 2002 Very Large Database Conference (VLDB'02), pp. 448-459

Translating Web Data

Abstract

1 Introduction

Proceedings of the 28th VLDB Conference,
Hong Kong, China, 2002

0

2005

20

HEROIC

SILVER

GOL

DE: Semantics and Query Answering

DE: Getting to the Core

0

2005

20

HEROIC

SILVER

GOL

Data Exchange: Semantics and Query Answering ⋆

Ronald Fagin [a] Phokion G. Kolaitis [b,1] Renée J. Miller [c,1]
Lucian Popa [a]

[a] *IBM Almaden Research Center*
{fagin,lucian}@almaden.ibm.com

[b] *University of California at Santa Cruz*
kolaitis@cs.ucsc.edu

[c] *University of Toronto*
miller@cs.toronto.edu

Abstract

Data exchange is the problem of taking data structured under a source schema and creating an instance of a target schema that reflects the source data as accurately as possible. In this paper, we address foundational and algorithmic issues related to the semantics of data exchange and to the query answering problem in the context of data exchange. These issues arise because, given a source instance, there may be many target instances that satisfy the constraints of the data exchange problem.

We give an algebraic specification that selects, among all solutions to the data exchange problem, a special class of solutions that we call *universal*. We show that a universal solution has no more and no less data than required for data exchange and that it represents the entire space of possible solutions. We then identify fairly general, and practical, conditions that guarantee the existence of a universal solution and yield algorithms to compute a canonical universal solution efficiently. We adopt the notion of the "certain answers" in indefinite databases for the semantics for query answering in data exchange. We investigate the computational complexity of computing the certain answers in this context and also address other algorithmic issues that arise in data exchange. In particular, we study the problem of computing the certain answers of target queries by simply evaluating them on a canonical universal solution, and we explore the boundary of what queries can and cannot be answered this way, in a data exchange setting.

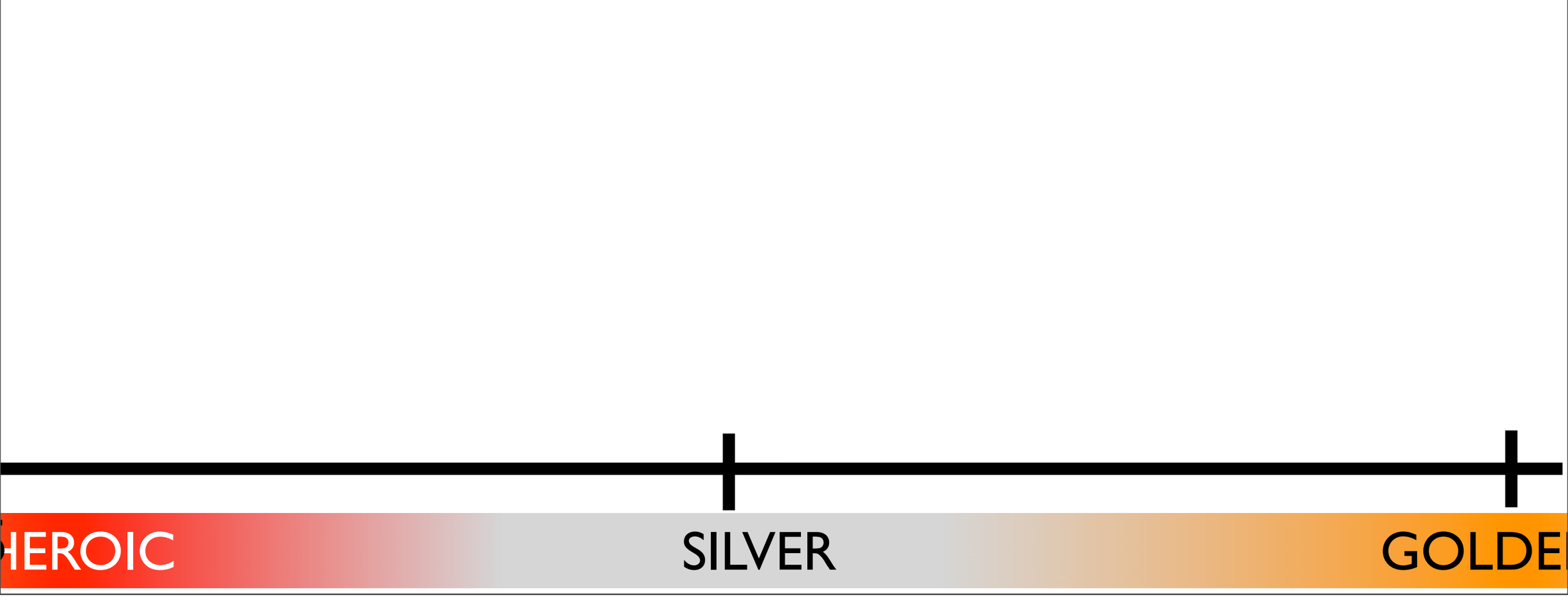
0

HEROIC

20

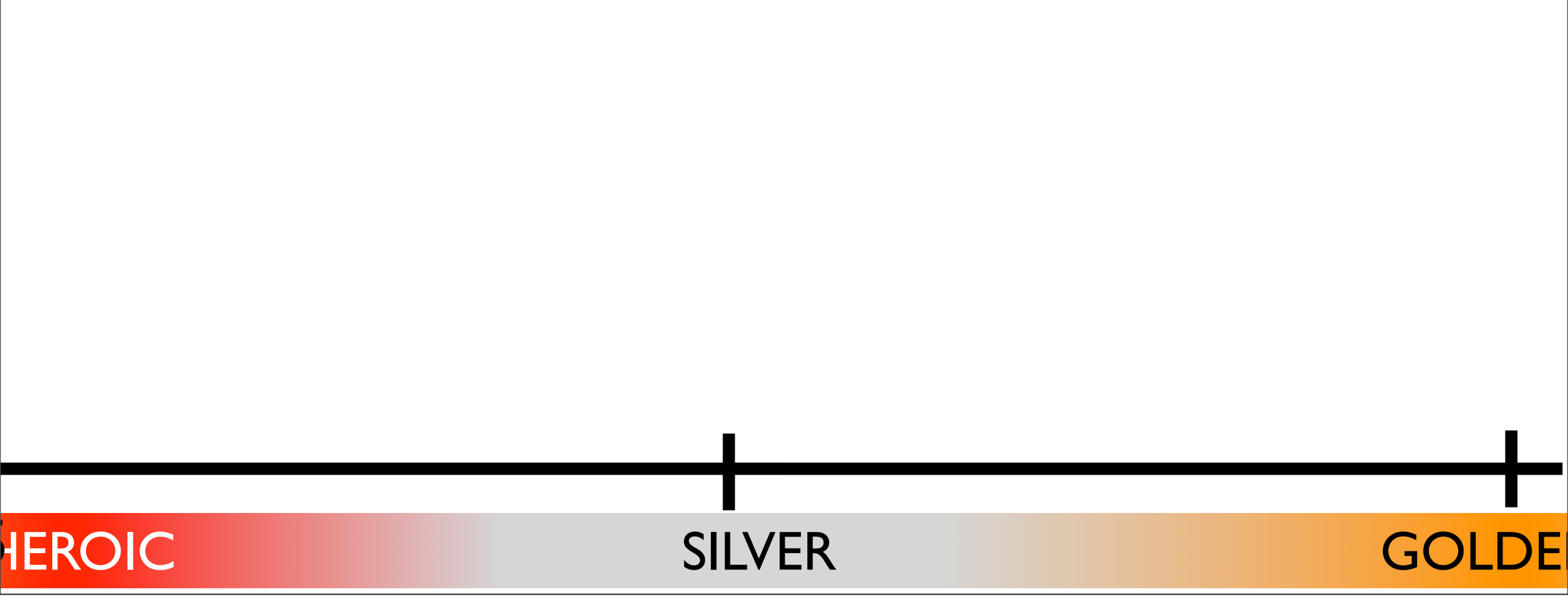
GOL

Data Exchange Setting



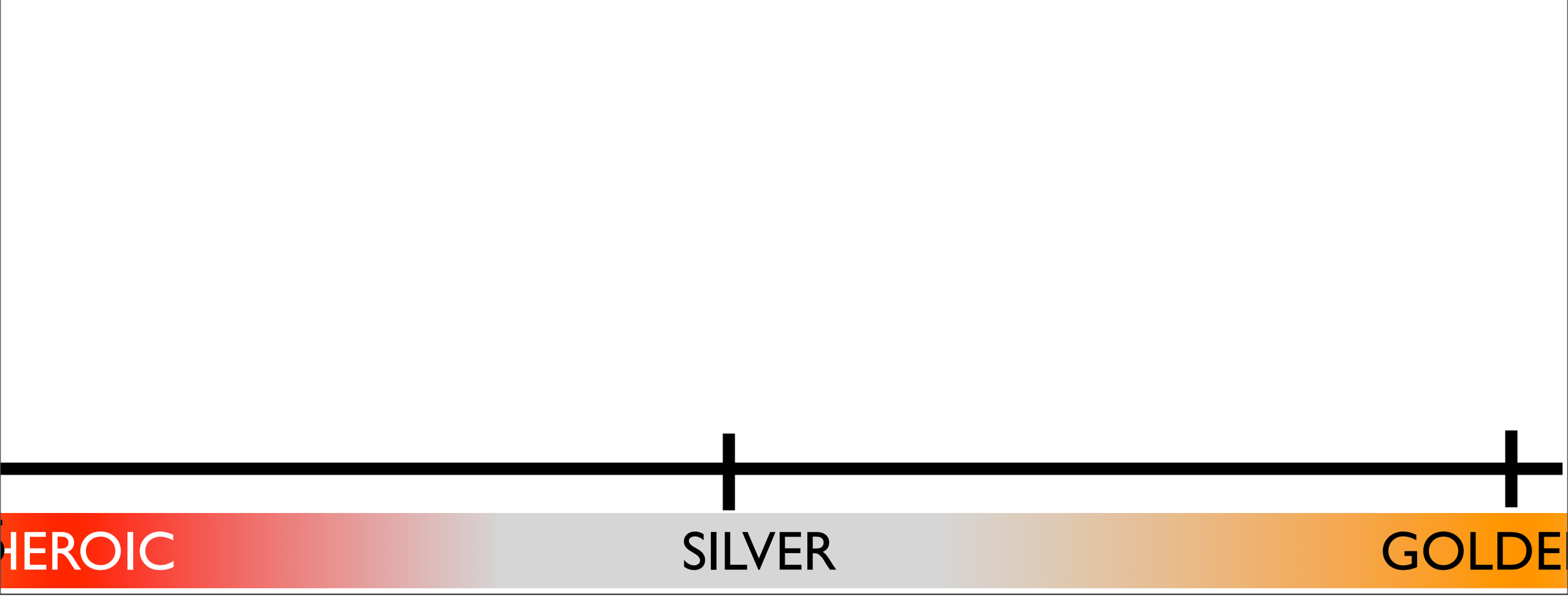
Data Exchange Setting

- M = <S, T, Σst, Σt >



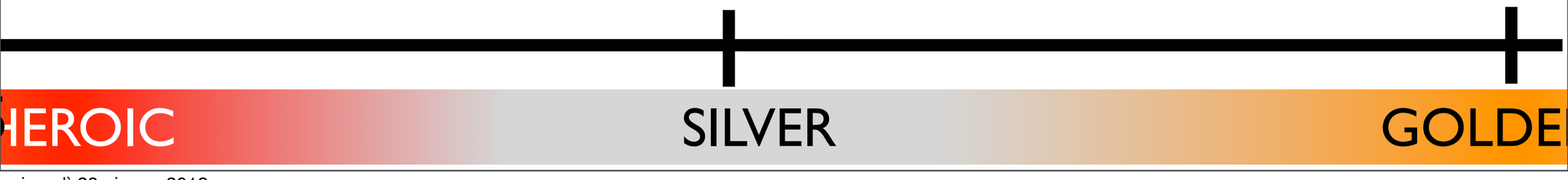
Data Exchange Setting

- M = <S, T, Σst, Σt >
  - S : source schema, T: target schema



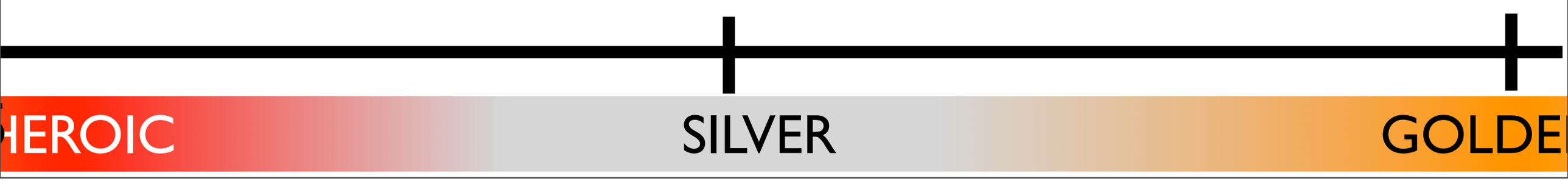
Data Exchange Setting

- M = <S, T, Σst, Σt >
  - S : source schema, T: target schema
  - Σst : source-to-target tgds



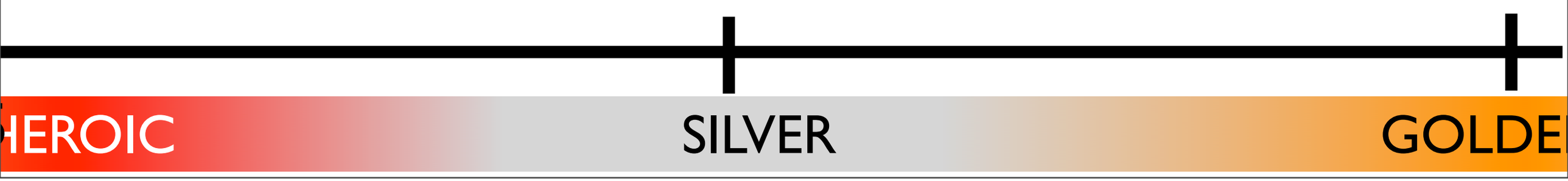
Data Exchange Setting

- M = <S, T, Σst, Σt >
  - S : source schema, T: target schema
  - Σst : source-to-target tgds
  - Σt : target tgds and target egds



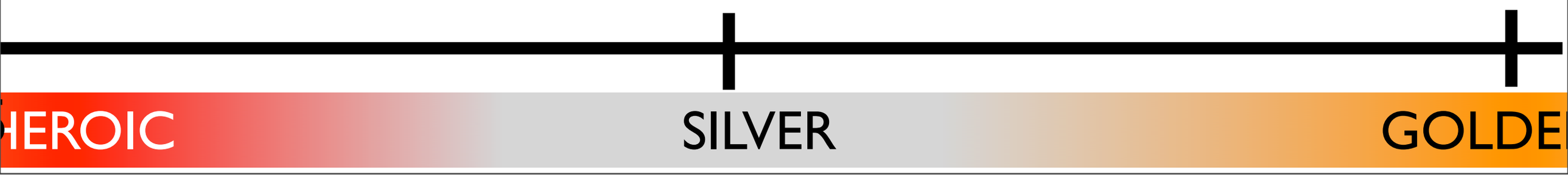
Data Exchange Setting

- M = <S, T, Σst, Σt >
  - S : source schema, T: target schema
  - Σst : source-to-target tgds
  - Σt : target tgds and target egds
- **Solution**



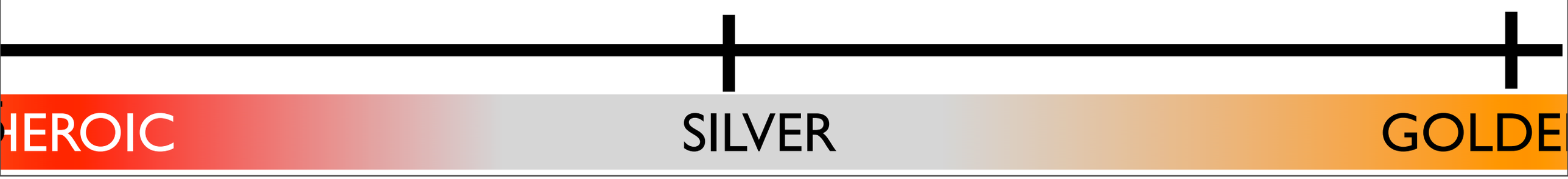
Data Exchange Setting

- M = <S, T, Σst, Σt >
  - S : source schema, T: target schema
  - Σst : source-to-target tgds
  - Σt : target tgds and target egds
- **Solution**
  - an instance of the target, J, such that



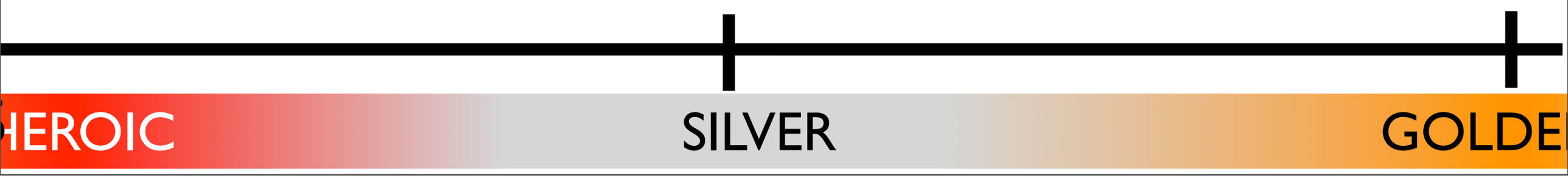
Data Exchange Setting

- M = <S, T, Σst, Σt >
  - S : source schema, T: target schema
  - Σst : source-to-target tgds
  - Σt : target tgds and target egds
- **Solution**
  - an instance of the target, J, such that
    - I and J satisfy the constraints in Σst



Data Exchange Setting

- M = <S, T, Σst, Σt >
  - S : source schema, T: target schema
  - Σst : source-to-target tgds
  - Σt : target tgds and target egds
- **Solution**
  - an instance of the target, J, such that
    - I and J satisfy the constraints in Σst
    - J satisfies the constraints in Σt



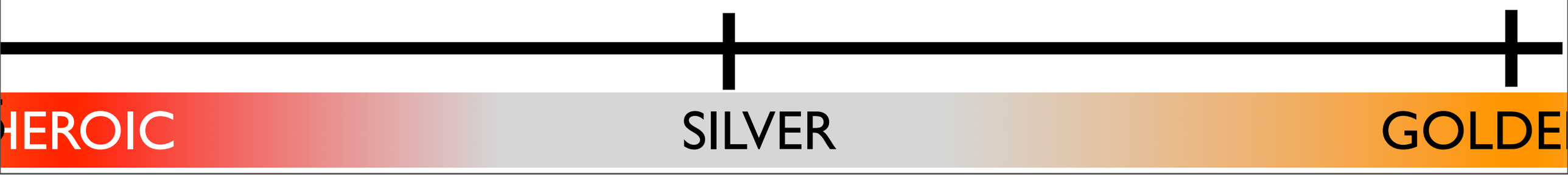
Data Exchange Setting

- M = <S, T, Σst, Σt >
  - S : source schema, T: target schema
  - Σst : source-to-target tgds
  - Σt : target tgds and target egds
- **Solution**
  - an instance of the target, J, such that
    - I and J satisfy the constraints in Σst
    - J satisfies the constraints in Σt

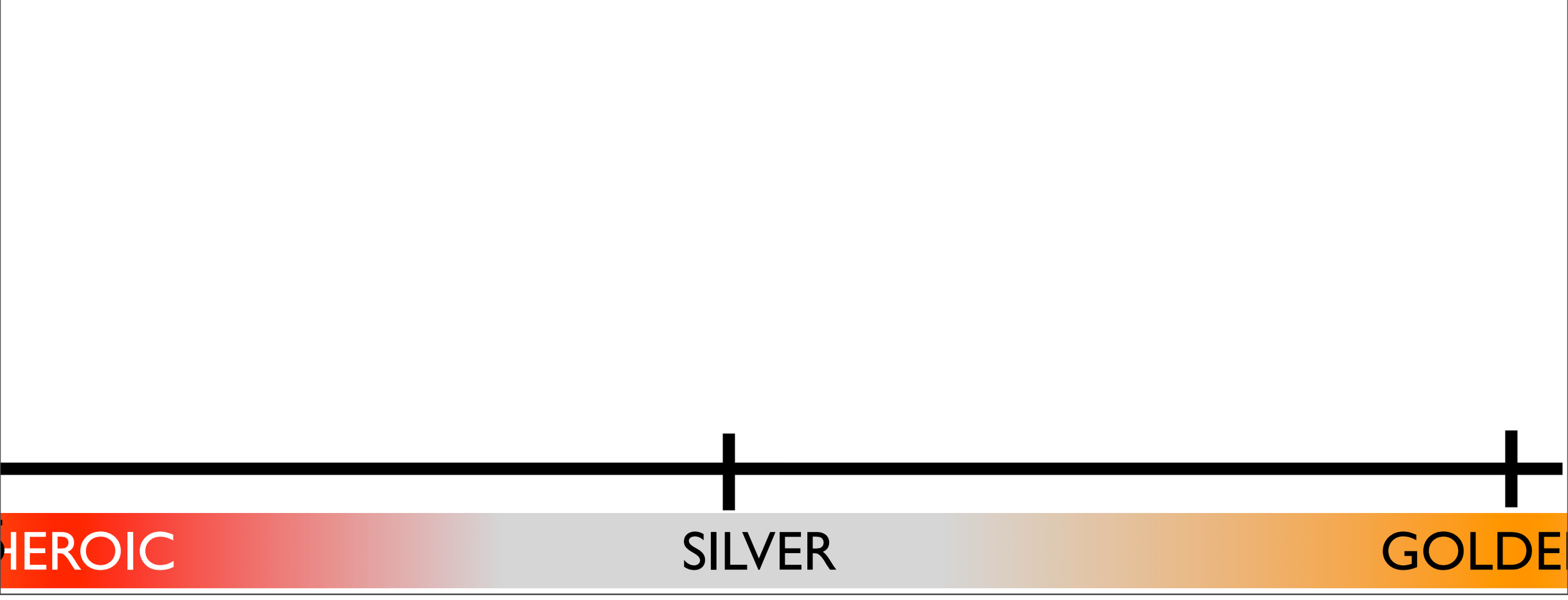
Goal

formalize the semantics of schema mappings

reason about the quality of the solutions



Many solutions...



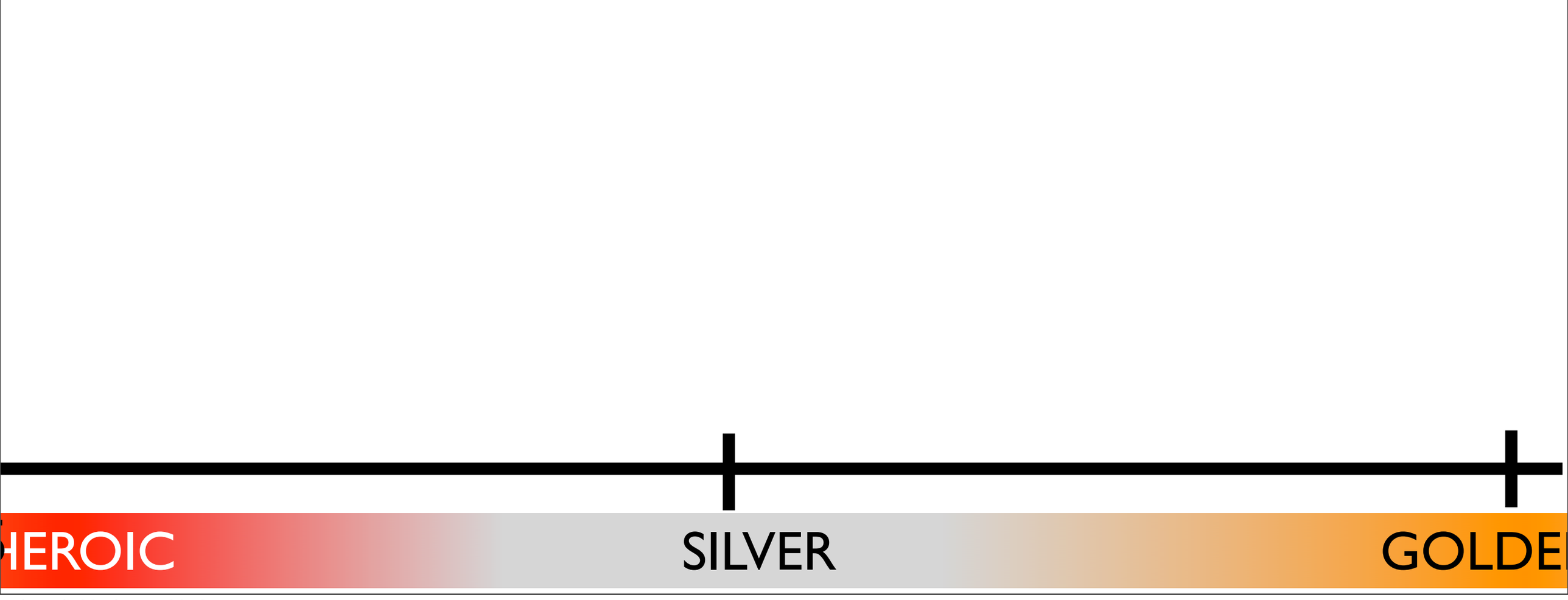
Many solutions...

Book1

title
On The Road
Post Office

Book2

title	year
Post Office	1971
Animal Farm	1947



Many solutions...

Book1

title
On The Road
Post Office

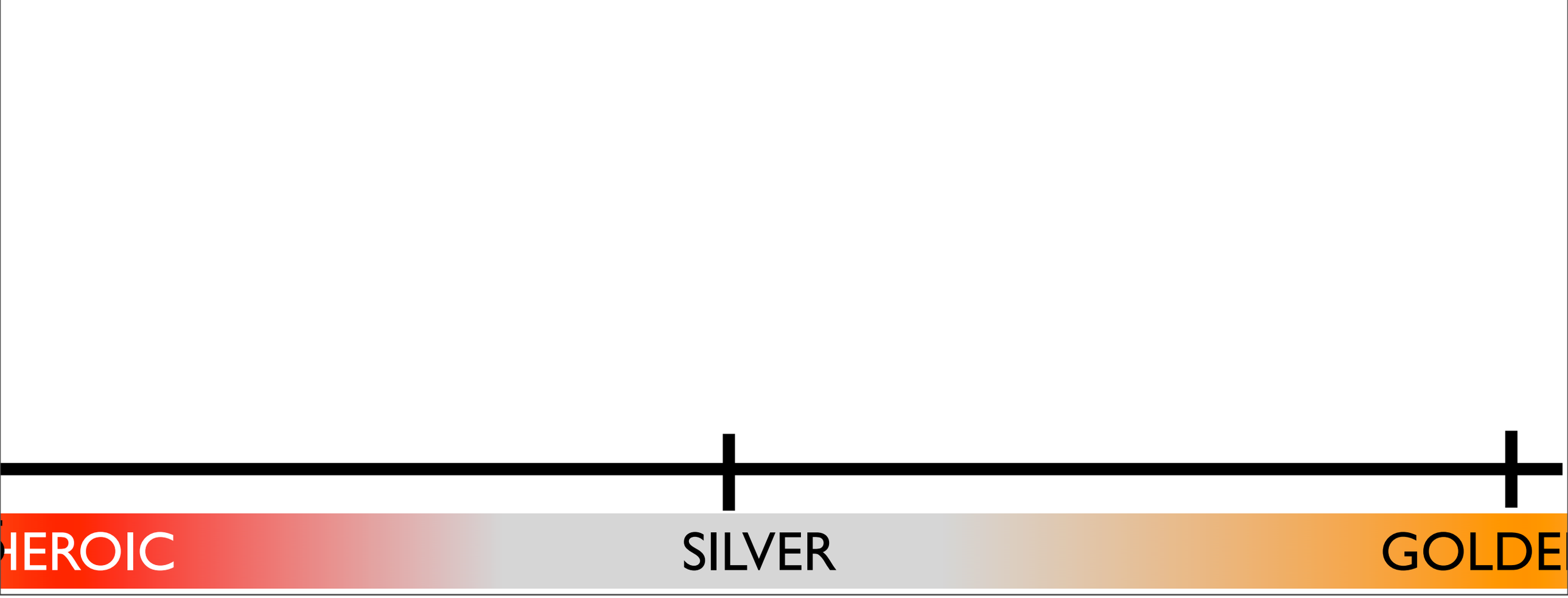
Book2

title	year
Post Office	1971
Animal Farm	1947

ST-TGDs

$\forall$ t: Book1(t) $\rightarrow$ $\exists$N: Book(t, N)

$\forall$ t, y: Book2(t, y) $\rightarrow$ Book(t, y)



Many solutions...

Book1

title
On The Road
Post Office

Book2

title	year
Post Office	1971
Animal Farm	1947

ST-TGDs

$\forall$ t: Book1(t) $\rightarrow$ $\exists$N: Book(t, N)

$\forall$ t, y: Book2(t, y) $\rightarrow$ Book(t, y)

Solution #1

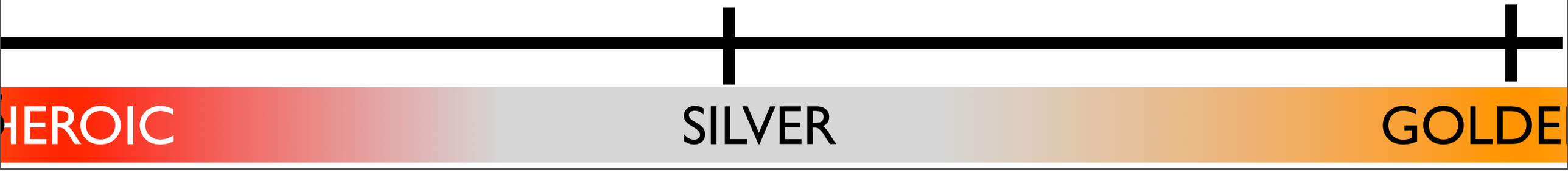
title	year
On The Road	N1
Post Office	N2
Post Office	1971
Animal Farm	1947

Solution #2

title	year
On The Road	2012
Post Office	N2
Post Office	1971
Animal Farm	1947
The Da Vinci Code	2003
Romeo and Juliet	1549

Solution #3

title	year
On The Road	N1
Post Office	1971
Animal Farm	1947



Many solutions...

Book1

title
On The Road
Post Office

Book2

title	year
Post Office	1971
Animal Farm	1947

ST-TGDs

$\forall$ t: Book1(t) $\rightarrow$ $\exists$N: Book(t, N)

$\forall$ t, y: Book2(t, y) $\rightarrow$ Book(t, y)

Solution #1

title	year
On The Road	N1
Post Office	N2
Post Office	1971
Animal Farm	1947

Labeled nulls

generated values used to satisfy existential quantifiers

Solution #2

title	year
On The Road	2012
Post Office	N2
Post Office	1971
Animal Farm	1947
The Da Vinci Code	2003
Romeo and Juliet	1549

Solution #3

title	year
On The Road	N1
Post Office	1971
Animal Farm	1947

HEROIC — SILVER — GOLDE

Many solutions...

Book1

title
On The Road
Post Office

Book2

title	year
Post Office	1971
Animal Farm	1947

ST-TGDs

$\forall$ t: Book1(t) $\rightarrow$ $\exists$N: Book(t, N)

$\forall$ t, y: Book2(t, y) $\rightarrow$ Book(t, y)

Solution #1

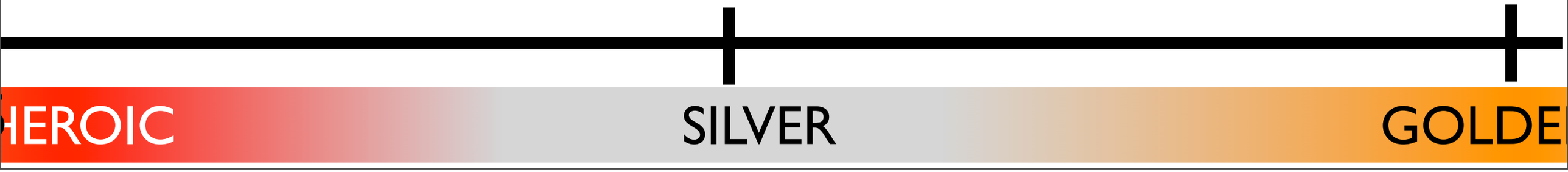
title	year
On The Road	N1
Post Office	N2
Post Office	1971
Animal Farm	1947

Solution #2

title	year
On The Road	2012
Post Office	N2
Post Office	1971
Animal Farm	1947
The Da Vinci Code	2003
Romeo and Juliet	1549

Solution #3

title	year
On The Road	N1
Post Office	1971
Animal Farm	1947



Many solutions...

Book1

title
On The Road
Post Office

Book2

title	year
Post Office	1971
Animal Farm	1947

ST-TGDs

$\forall$ t: Book1(t) $\rightarrow$ $\exists$N: Book(t, N)

$\forall$ t, y: Book2(t, y) $\rightarrow$ Book(t, y)

Solution #1

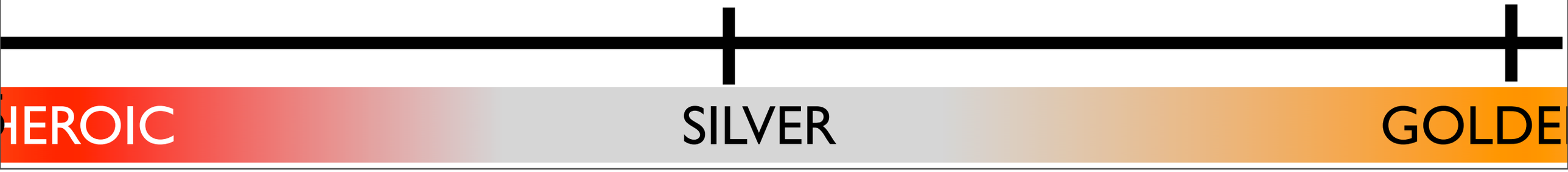
title	year
On The Road	N1
Post Office	N2
Post Office	1971
Animal Farm	1947

Solution #2

title	year
On The Road	2012
Post Office	N2
Post Office	1971
Animal Farm	1947
The Da Vinci Code	2003
Romeo and Juliet	1549

Solution #3

title	year
On The Road	N1
Post Office	1971
Animal Farm	1947



Many solutions...

Book1

title
On The Road
Post Office

Book2

title	year
Post Office	1971
Animal Farm	1947

ST-TGDs

$\forall$ t: Book1(t) $\rightarrow$ $\exists$N: Book(t, N)

$\forall$ t, y: Book2(t, y) $\rightarrow$ Book(t, y)

Solution #1

title	year
On The Road	N1
Post Office	N2
Post Office	1971
Animal Farm	1947

Universal

Solution #2

title	year
On The Road	2012
Post Office	N2
Post Office	1971
Animal Farm	1947
The Da Vinci Code	2003
Romeo and Juliet	1549

Solution #3

title	year
On The Road	N1
Post Office	1971
Animal Farm	1947

Universal

HEROIC — SILVER — GOLDE

Many solutions...

Book1

title
On The Road
Post Office

Book2

title	year
Post Office	1971
Animal Farm	1947

ST-TGDs

$\forall$ t: Book1(t) $\rightarrow$ $\exists$N: Book(t, N)

$\forall$ t, y: Book2(t, y) $\rightarrow$ Book(t, y)

Solution #1

title	year
On The Road	N1
Post Office	N2
Post Office	1971
Animal Farm	1947

Universal

Solution #2

title	year
On The Road	2012
Post Office	N2
Post Office	1971
Animal Farm	1947
The Da Vinci Code	2003
Romeo and Juliet	1549

Non-Universal

Solution #3

title	year
On The Road	N1
Post Office	1971
Animal Farm	1947

Universal

HEROIC — SILVER — GOLDE

Many solutions...

Book1

title
On The Road
Post Office

Book2

title	year
Post Office	1971
Animal Farm	1947

ST-TGDs

$\forall$ t: Book1(t) $\rightarrow$ $\exists$N: Book(t, N)

$\forall$ t, y: Book2(t, y) $\rightarrow$ Book(t, y)

Solution #1

title	year
On The Road	N1
Post Office	N2
Post Office	1971
Animal Farm	1947

Universal

Solution #2

title	year
On The Road	2012
Post Office	N2
Post Office	1971
Animal Farm	1947
The Da Vinci Code	2003
Romeo and Juliet	1549

Non-Universal

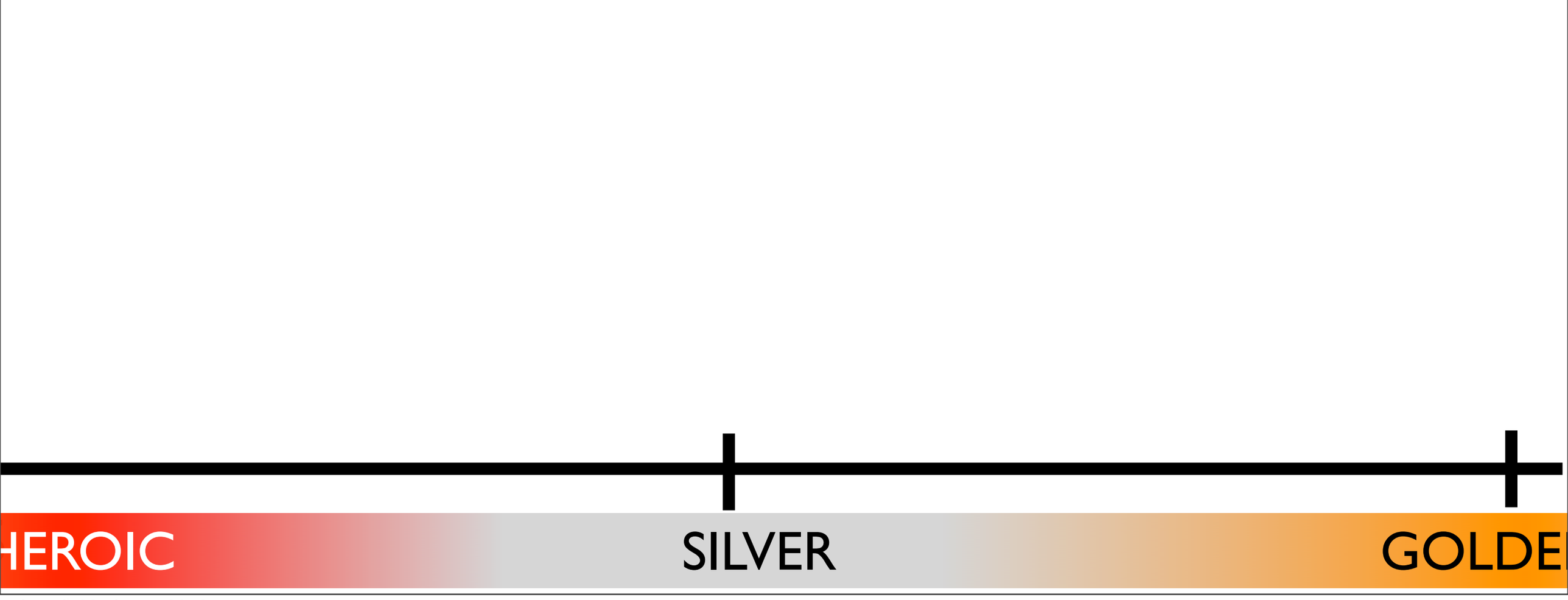
Solution #3

title	year
On The Road	N1
Post Office	1971
Animal Farm	1947

Universal
CORE

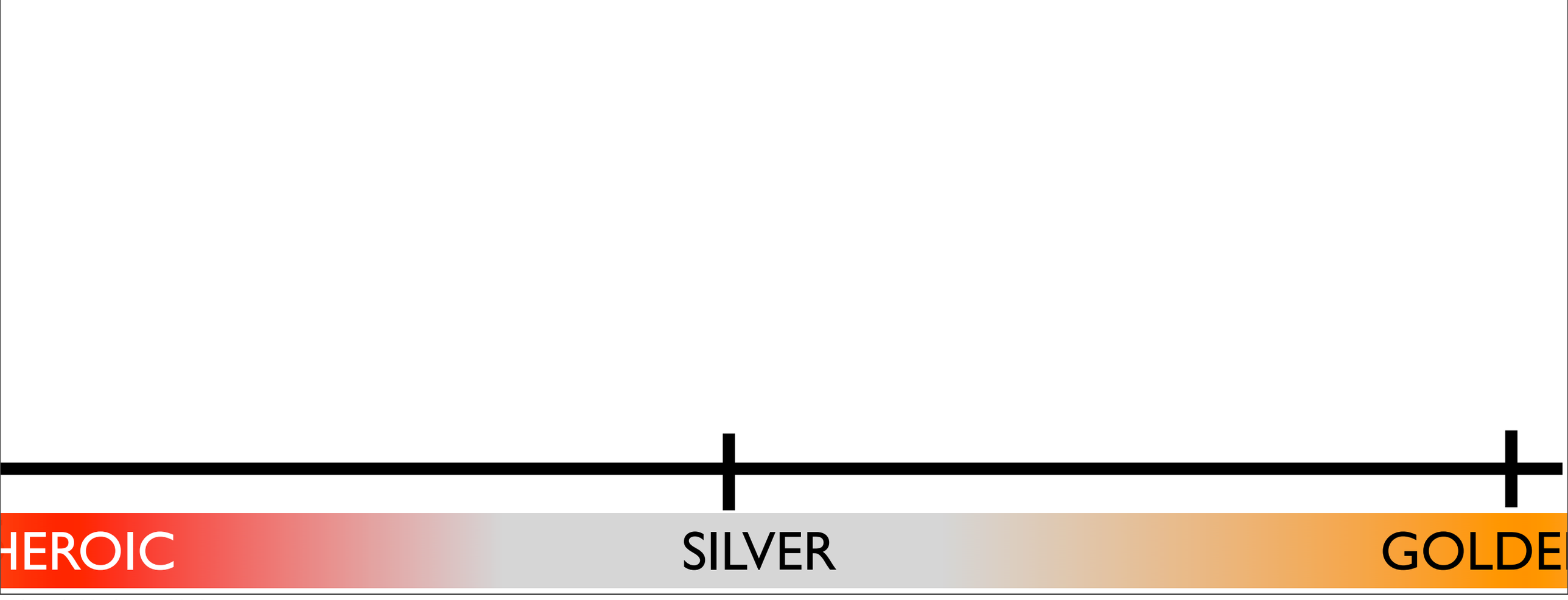
HEROIC — SILVER — GOLDE

Optimal Solution



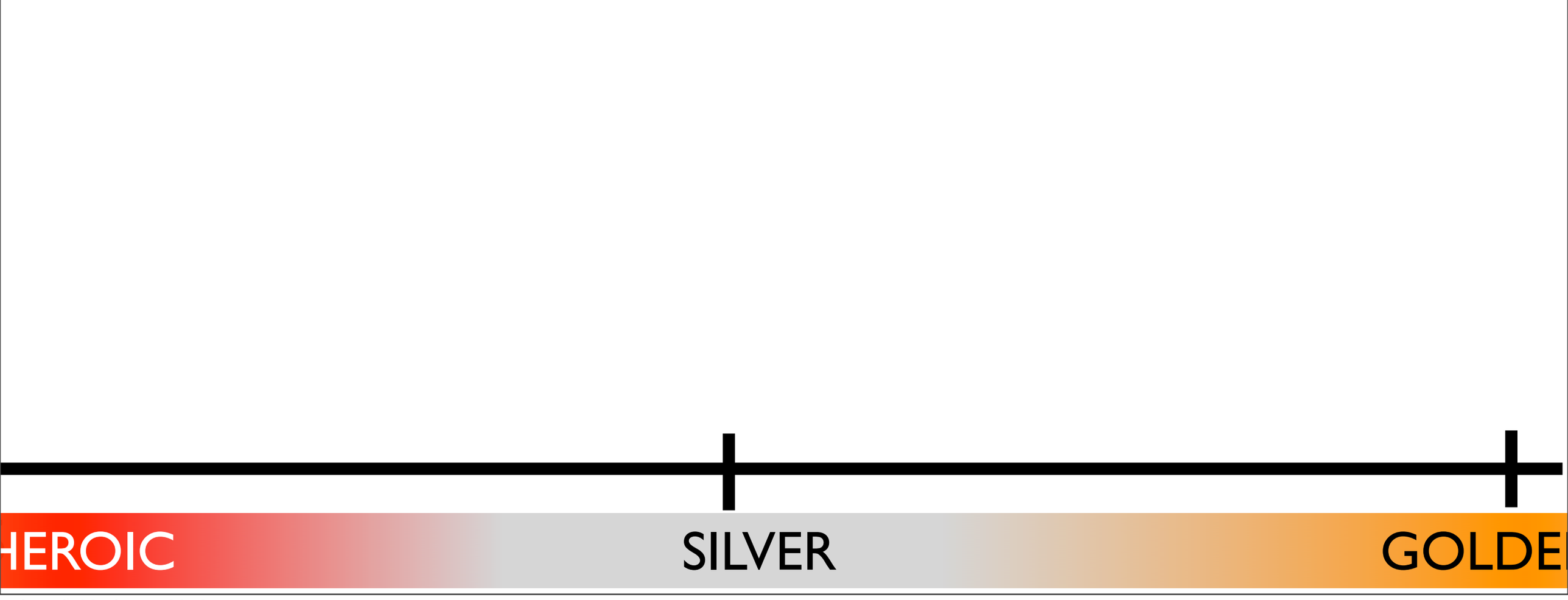
Optimal Solution

- **The Core**



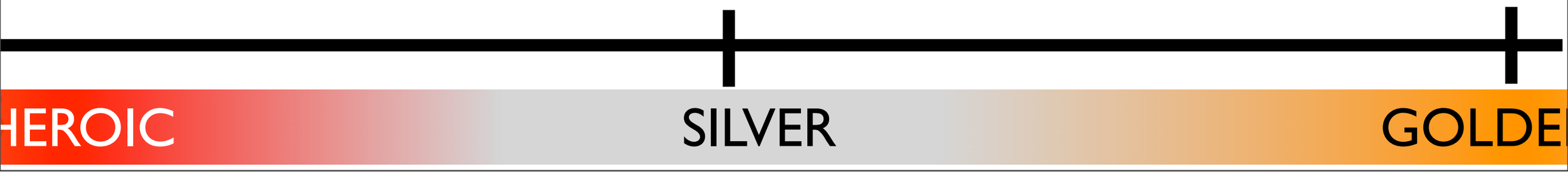
Optimal Solution

- **The Core**
  - the smallest universal solution



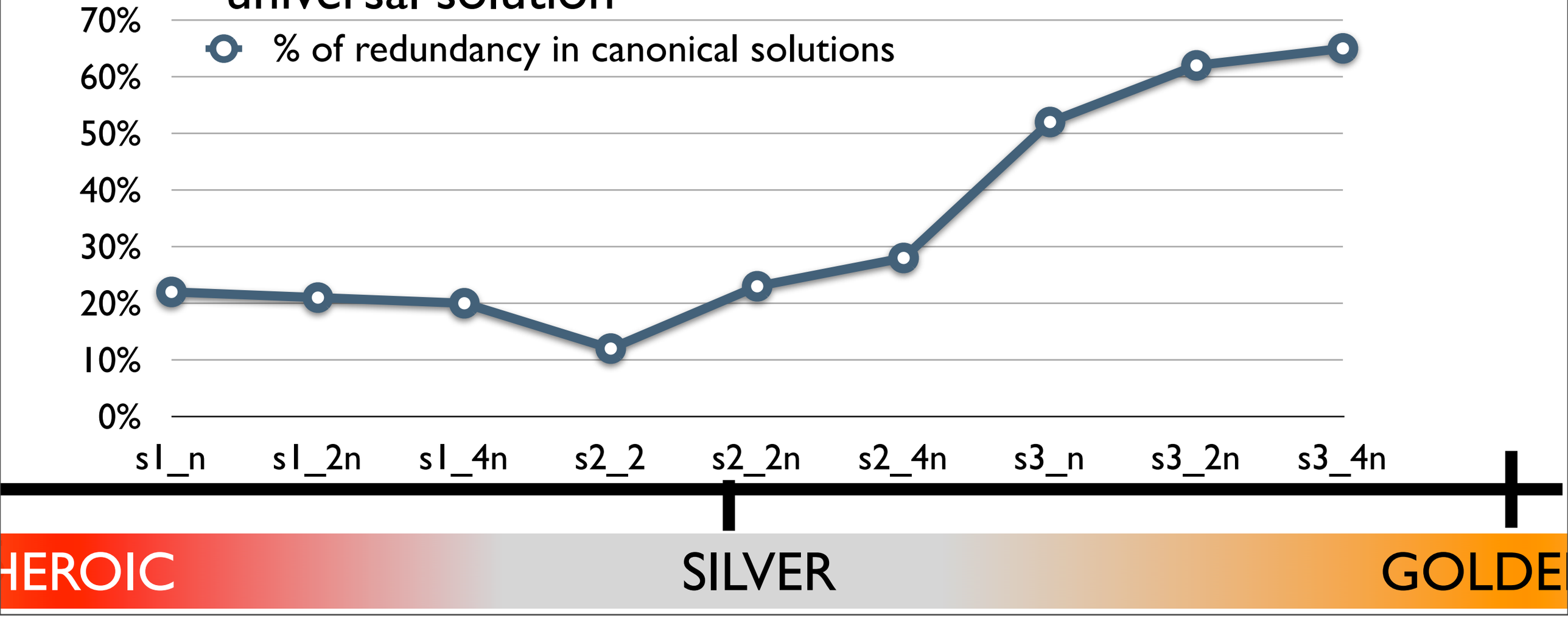
Optimal Solution

- **The Core**
  - the smallest universal solution
  - it does not contain any proper subset that is also a universal solution

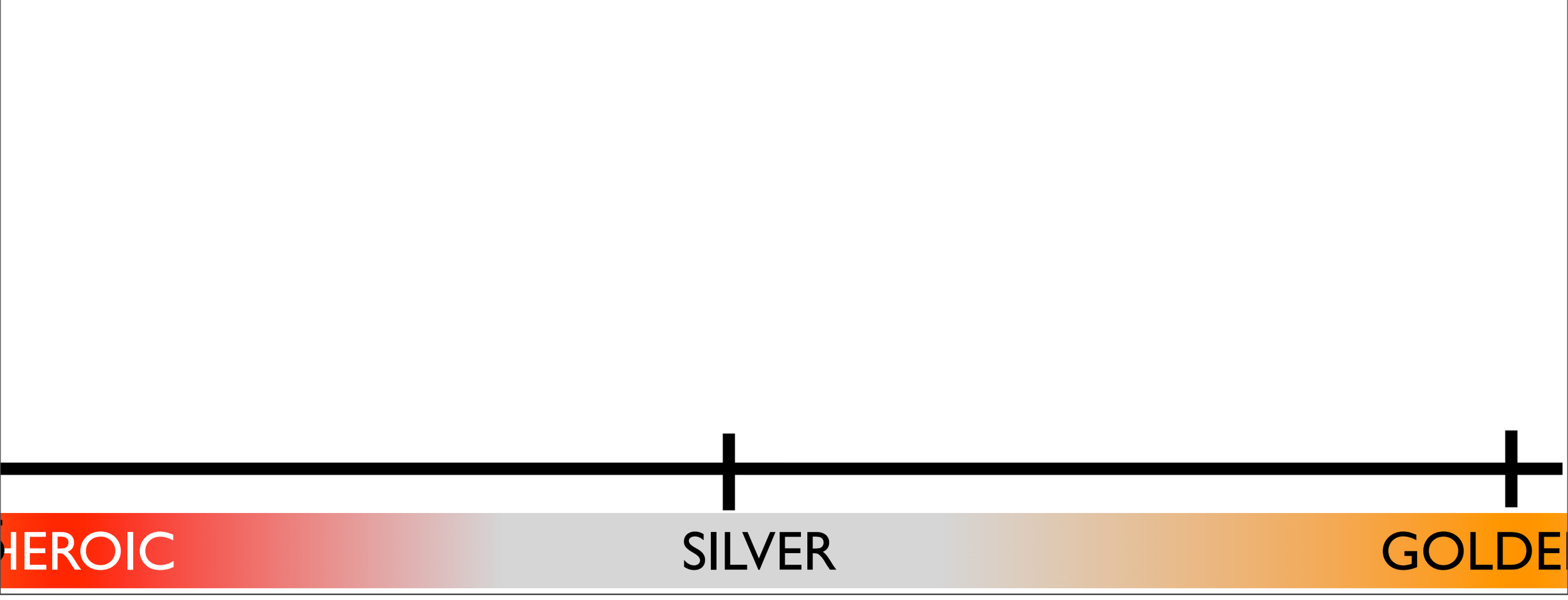


Optimal Solution

- **The Core**
  - the smallest universal solution
  - it does not contain any proper subset that is also a universal solution



Intermediate Generation (Post-Processing)



Intermediate Generation (Post-Processing)

Source

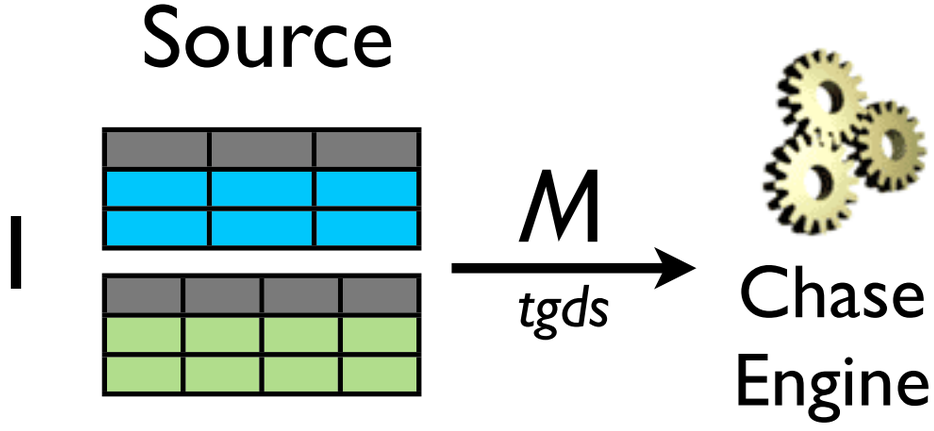
I

HEROIC

SILVER

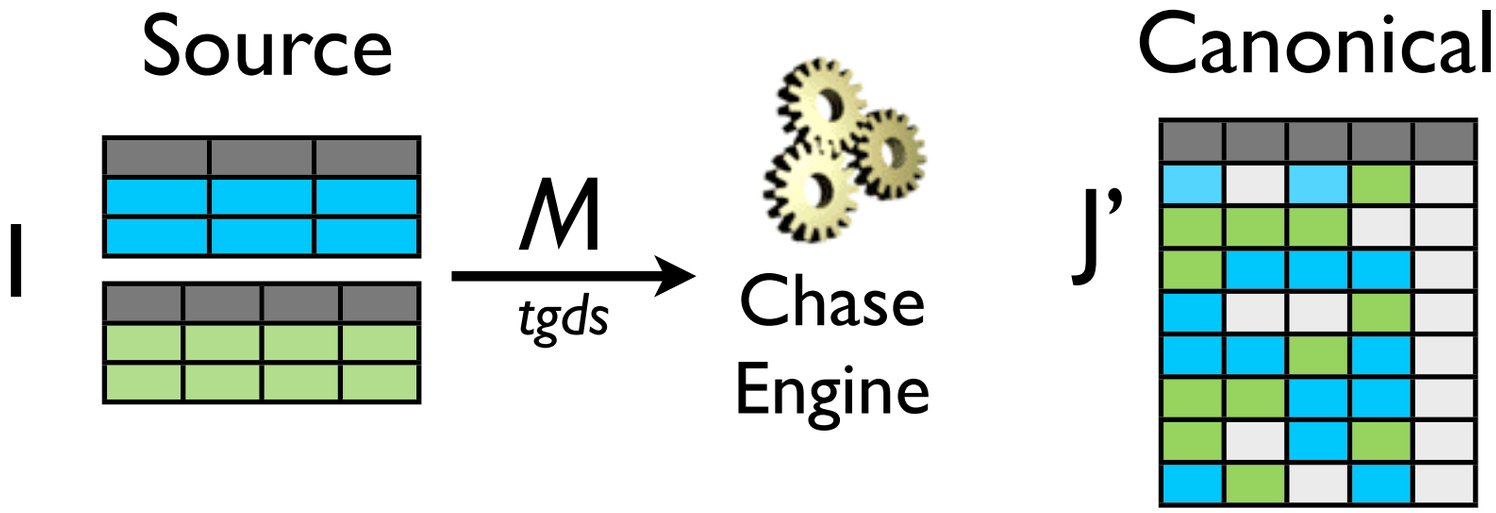
GOLDE

Intermediate Generation (Post-Processing)



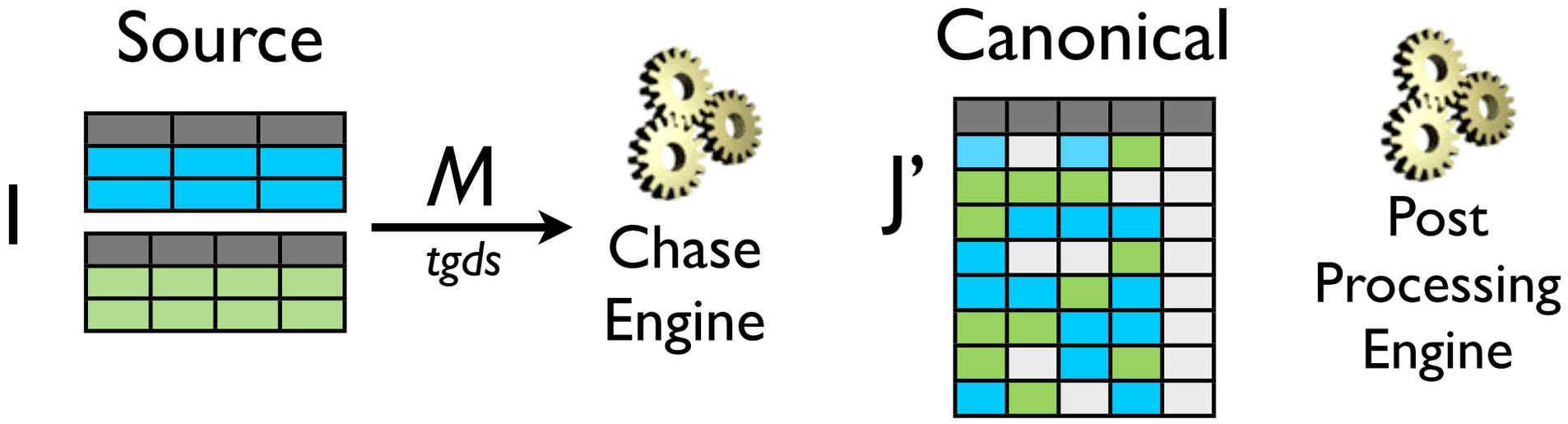
HEROIC SILVER GOLDE

Intermediate Generation (Post-Processing)



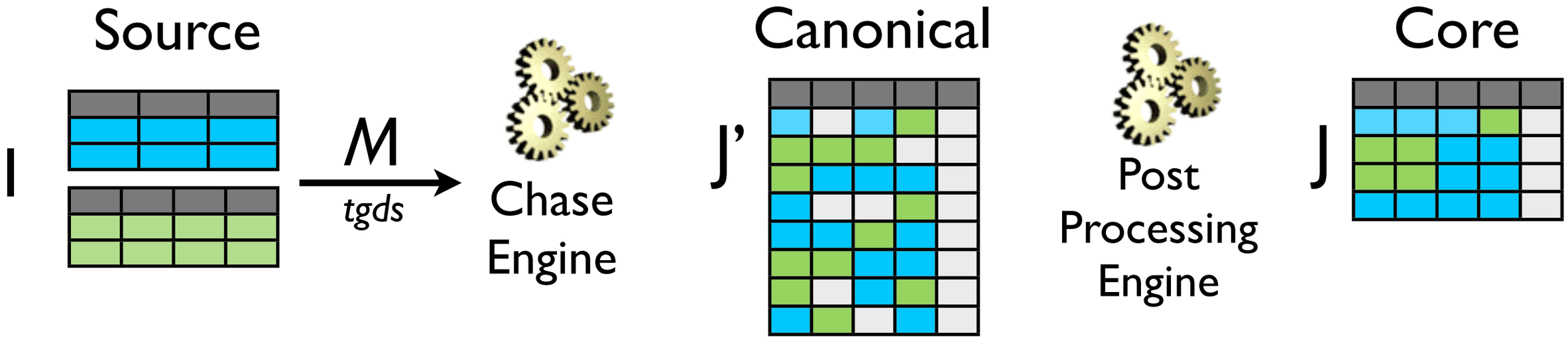
HEROIC SILVER GOLDE

Intermediate Generation (Post-Processing)



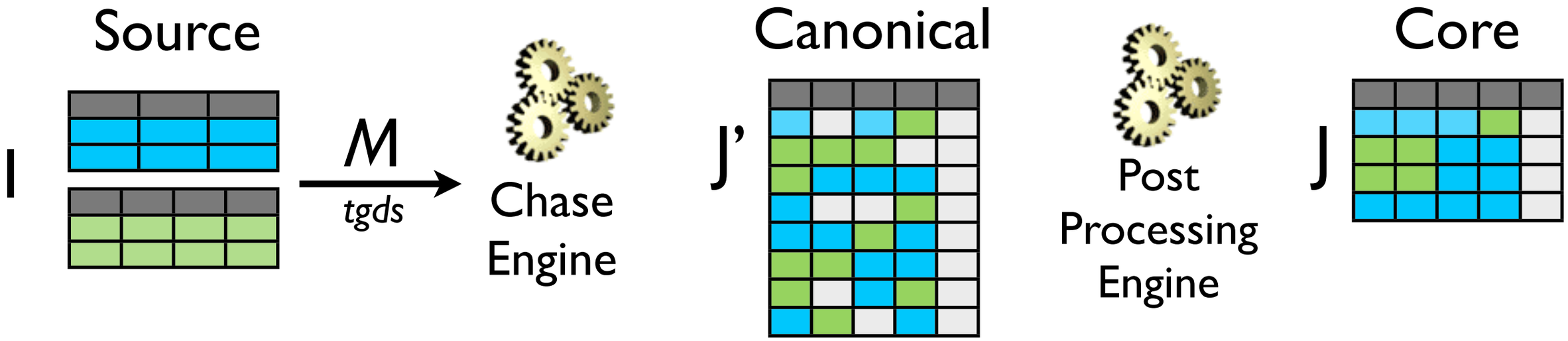
HEROIC SILVER GOLDE

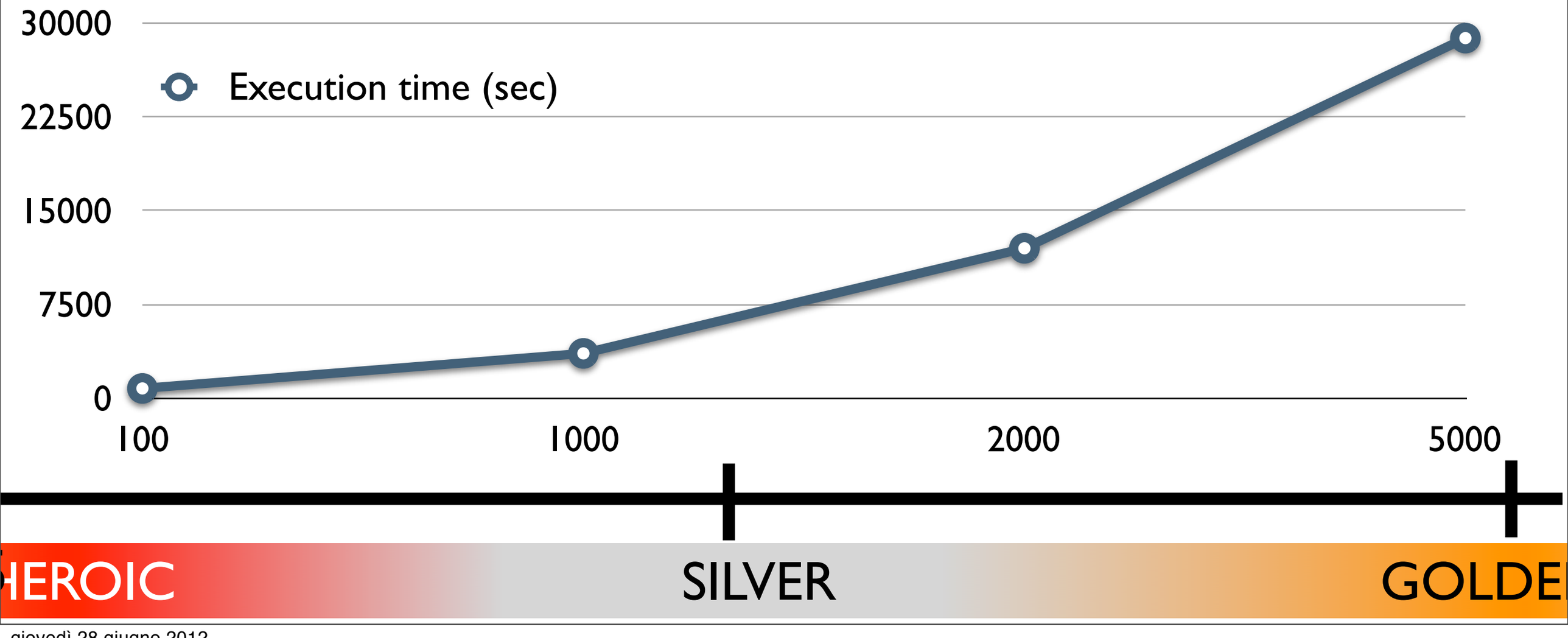
Intermediate Generation (Post-Processing)



HEROIC SILVER GOLDE

Intermediate Generation (Post-Processing)





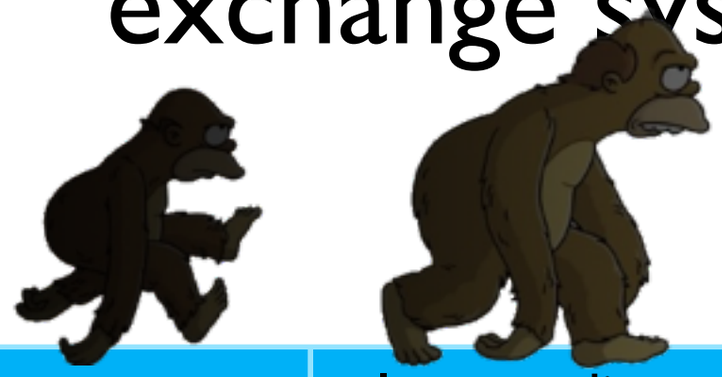
Evolution of schema-mappings and data exchange systems

	First Generation	Intermediate Generation (post-processing)	Intermediate Generation (rewriting)	Second Generation
Scalable Portable	✓	✗		
Quality	✗	✓		
Functional Dependencies	✗	✓		
Nested Relation	✓	✗		

Popa et al., VLDB 2002

Fagin et al., TODS 2005

Evolution of schema-mappings and data exchange systems



	First Generation	Intermediate Generation (post-processing)	Intermediate Generation (rewriting)	Second Generation
Scalable Portable	✓	✗		
Quality	✗	✓		
Functional Dependencies	✗	✓		
Nested Relation	✓	✗		
	Popa et al., VLDB 2002	Fagin et al., TODS 2005		


Translating Web Data


Data Exchange: Semantics and Query Answering *


Schema Mapping as Query Discovery

0

2005

20

HEROIC

SILVER

GOL


Schema Mapping as Query Discovery

2005

2010

HEROIC

SILVER

GOL


Schema Mapping as Query Discovery

2005

2010

HEROIC

SILVER

GOL

Core Schema Mappings

Laconic Schema Mappings


Schema Mapping as Query Discovery


Core Schema Mappings


Laconic schema mappings: computing core universal solutions by means of SQL queries*

2005

2009 2010

HEROIC

SILVER

GOL

Core Schema Mappings

Laconic Schema Mappings

Core Schema Mappings

Giansalvatore Mecca[1] Paolo Papotti[2] Salvatore Raunich[1]
timento di Matematica e Informatica – Università della Basilicata – Potenza, Italy
partimento di Informatica e Automazione – Università Roma Tre – Roma, Italy

ed mappings among data sources un-
On one side, there are studies of prac-
napping generation; these focus on al-
appings based on visual specifications
the other side, we have theoretical re-
change. These study how to generate
et instance – given a set of mappings
le generating dependencies. However,
e notion of a core of a data exchange
ally identified as an optimal solution,
ng systems that support core compu-
we introduce several new algorithms
dge the gap between the practice of
d the theory of data exchange. We
ping scenario, it is possible to gener-
t that computes core solutions for the
hange problem. The algorithms have
tested using common runtime engines
antee very good performances, orders
han those of known algorithms that
post-processing step.

bject Descriptors
gement]: Heterogeneous Databases

a Exchange, Core Computation

TION
ning from disparate sources is a cru-
ications. An essential requirement of
sk is that of manipulating *mappings*

between sources. Mappings are executable tran
– say, SQL or XQuery scripts – that specify how
of the source repository should be translated into
of the target repository. There are several way
such mappings. A popular one consists in usin
erating dependencies (*tgds*) [3]. We may identif
research lines in the literature.

On one side, we have studies on practical t
gorithms for *schema mapping generation*. In th
focus is on the development of systems that ta
an abstract specification of the mapping, usua
a bunch of correspondences between the two s
generate the mappings and the executable scrip
perform the translation. This research topic wa
spired by the seminal papers about the Clio syst
The original algorithm has been subsequently
several ways [12, 4, 2, 19, 7] and various tool
proposed to support users in the mapping gen
cess. More recently, a benchmark has been deve
compare research mapping systems and comme

On the other side, we have theoretical studies
exchange. Several years after the development
Clio algorithm, researchers have realized that
theoretical foundation was needed in order to
practical results obtained on schema mapping sy
consideration has motivated a rich body of resea
the notion of a *data exchange problem* [9] was
and a number of theoretical results were establis
context, a *data exchange setting* is a collection of
usually specified as tgds – that are given as part
therefore, the focus is not on the generation of th
but rather on the characterization of their prop
has brought to an elegant formalization of the
solution for a data exchange problem, and of op
manipulate mappings in order, for example, to
invert them.

However, these two research lines have pro
rather independent way. To give a clear exam
consider the fact that there are many possible
a data exchange problem. A natural question

Laconic schema mappings: computing core universal solutio by means of SQL queries*

Balder ten Cate[1], Laura Chiticariu[2], Phokion Kolaitis[3], and Wang-Chiew Tan[4]

[1] University of Amsterdam
[2] IBM Almaden
[3] UC Santa Cruz and IBM Almaden
[4] UC Santa Cruz

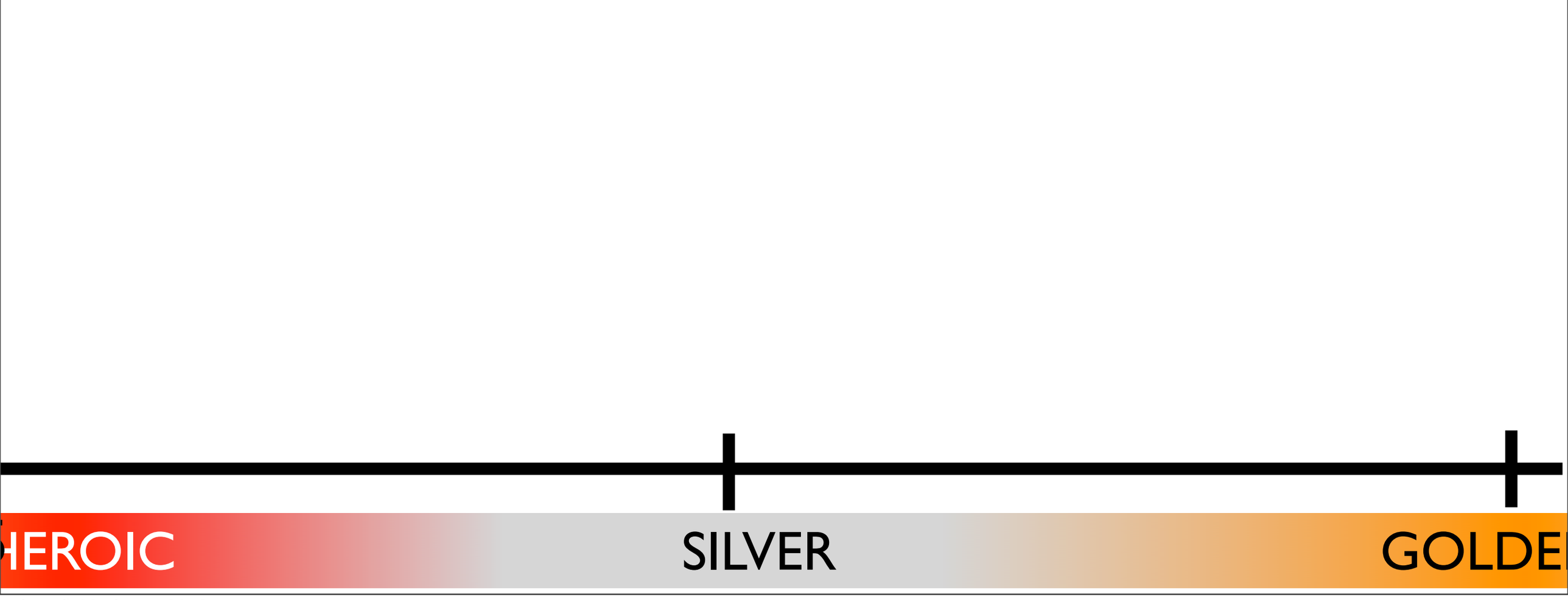
Abstract. We present a new method for computing core universal solutions in data exchange settings specified by source-to-target dependencies, by means of SQL queries. Unlike previously known algorithms, which are recursive in nature, our method can be implemented directly on top of any DBMS. Our method is based on the new notion of a laconic schema mapping. A laconic schema mapping is a schema mapping for which the canonical universal solution is the core universal solution. We give a procedure by which every schema mapping specified by FO s-t tgds can be turned into a laconic schema mapping specified by FO s-t tgds that may refer to a linear order on the domain of the source instance. We show that our results are optimal, in the sense that the linear order is necessary and the method cannot be extended to schema mapping involving target constraints.

1 Introduction

We present a new method for computing core universal solutions in data exchange settings specifie
source-to-target dependencies, by means of SQL queries. Unlike previously known algorithms, w
are recursive in nature, our method can be implemented directly on top of any DBMS. Our meth
based on the new notion of a laconic schema mapping. A laconic schema mapping is a schema map
for which the canonical universal solution is the core universal solution. We give a procedure by w
every schema mapping specified by FO s-t tgds can be turned into a laconic schema mapping spec
by FO s-t tgds that may refer to a linear order on the domain of the source instance.

Outline of the paper: In Section 2, we recall basic notions and facts about schema mappings. Secti
explains what it means to compute a target instance by means of SQL queries, and we state our
result. Section 4 introduces the notion of laconicity, and contains some initial observations In Secti
we present our main result, namely a method for transforming any schema mapping specified by
s-t tgds into a laconic schema mapping specified by FO s-t tgds asssuming a linear order. In Secti
we show that our results cannot be extended to the case with target constraints.

Intermediate Generation (Rewriting)



Intermediate Generation (Rewriting)

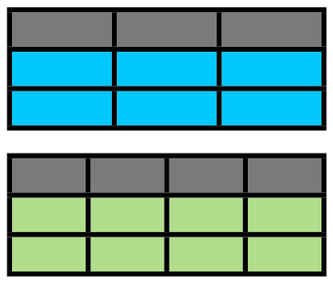
Source

HEROIC SILVER GOLDE

Intermediate Generation (Rewriting)

I

Source



M

st-tgds

Rewriting Engine

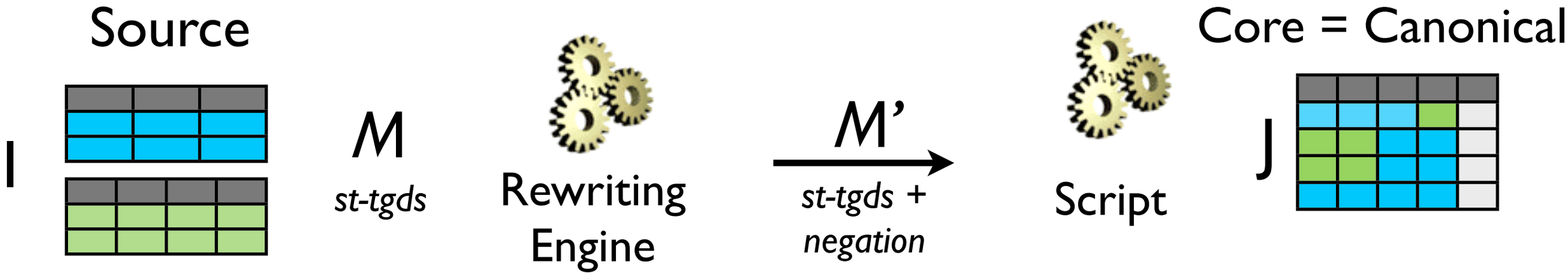
M'

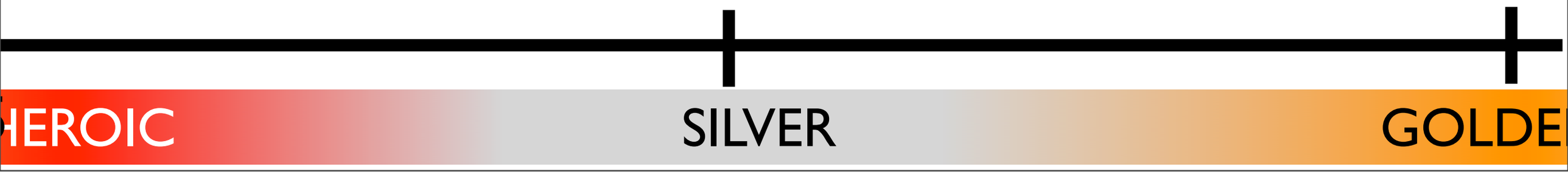
HEROIC

SILVER

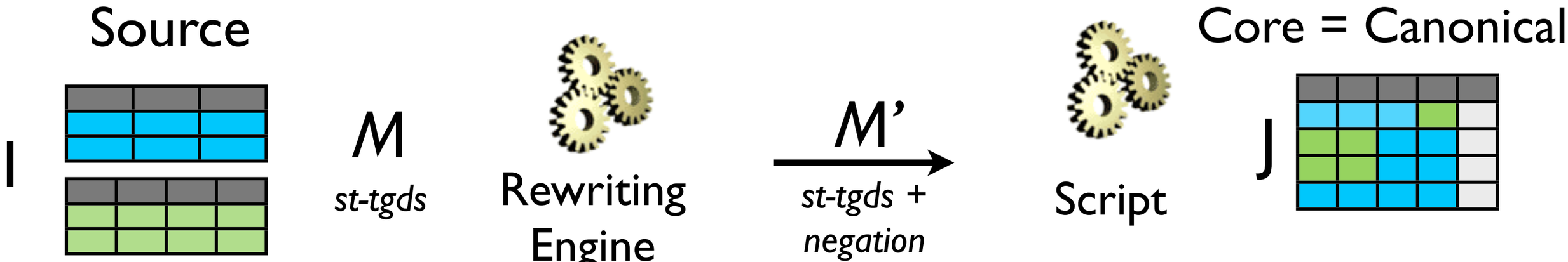
GOLDE

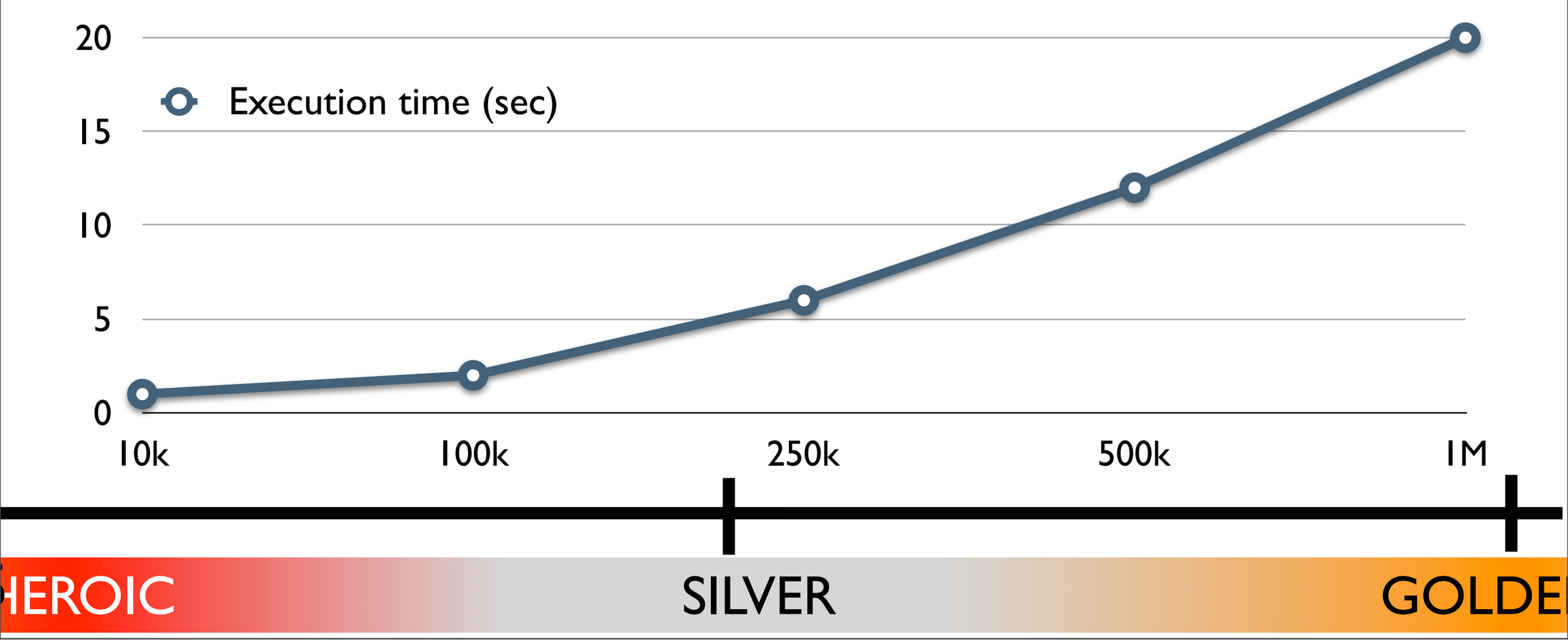
Intermediate Generation (Rewriting)



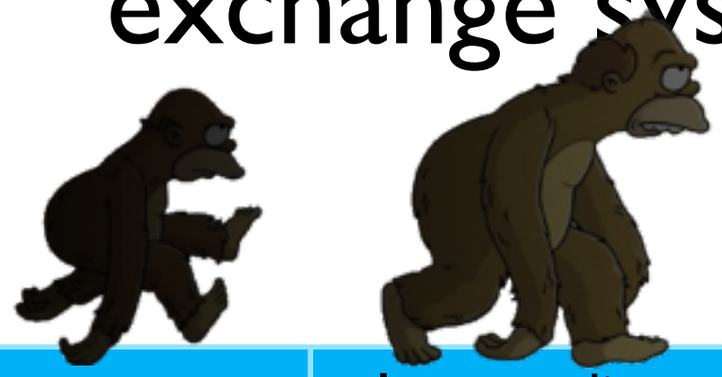


Intermediate Generation (Rewriting)



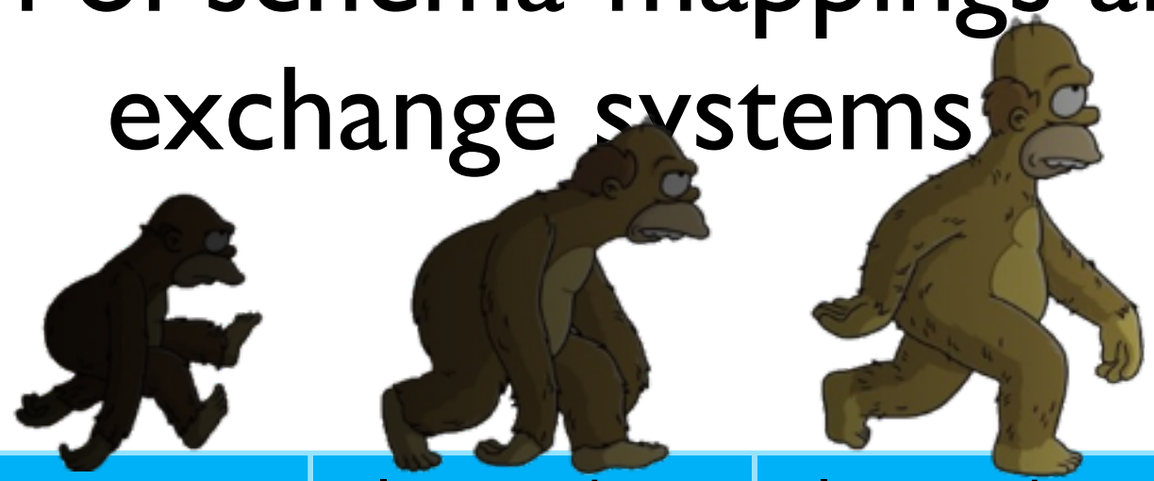


Evolution of schema-mappings and data exchange systems



	First Generation	Intermediate Generation (post-processing)	Intermediate Generation (rewriting)	Second Generation
Scalable Portable	✓	✗	✓	
Quality	✗	✓	✓	
Functional Dependencies	✗	✗	✗	
Nested Relation	✓	✗	✗	
	Popa et al., VLDB 2002	Fagin et al., TODS 2005	Mecca et al., SIGMOD 2009 ten Cate et al., PVLDB 2009	

Evolution of schema-mappings and data exchange systems



	First Generation	Intermediate Generation (post-processing)	Intermediate Generation (rewriting)	Second Generation
Scalable Portable	✓	✗	✓	
Quality	✗	✓	✓	
Functional Dependencies	✗	✗	✗	
Nested Relation	✓	✗	✗	
	Popa et al., VLDB 2002	Fagin et al., TODS 2005	Mecca et al., SIGMOD 2009 ten Cate et al., PVLDB 2009	

2005

2010

HEROIC

SILVER

GOL


Schema Mapping as Query Discovery


Data Exchange: Semantics and Query Answering *

Ronald Fagin † Phokion G. Kolaitis ‡,† Renée J. Miller ‖,†
Lucian Popa †


Schema Mapping as Query Discovery

SILVER

GOLDEN


Schema Mapping as Query Discovery


Data Exchange: Semantics and Query Answering *

Ronald Fagin † Phokion G. Kolaitis ‡,† Renée J. Miller ‖,†
Lucian Popa †


Schema Mapping as Query Discovery

SILVER

GOLDEN





Scalable data exchange with fds

Tractable XML Data Exchange via Relations

2005

2010

2015


SILVER

GOLDEN

05

SILVER

Scalable Data Exchange with Functional Dependencies

Bruno Marnette[1] Giansalvatore Mecca[2] Paolo Papotti[3]

[1] Oxford University Computing Laboratory, UK and INRIA Saclay, France
[2] Dipartimento di Matematica e Informatica – Università della Basilicata – Potenza, Italy
[3] Dipartimento di Informatica e Automazione – Università Roma Tre – Roma, Italy

ABSTRACT

The recent literature has provided a solid theoretical foundation for the use of schema mappings in data-exchange applications. Following this formalization, new algorithms have been developed to generate optimal solutions for mapping scenarios in a highly scalable way, by relying on SQL. However, these algorithms suffer from a serious drawback: they are not able to handle key constraints and functional dependencies on the target, i.e., equality generating dependencies (egds). While egds play a crucial role in the generation of optimal solutions, handling them with first-order languages is a difficult problem. In fact, we start from a negative result: it is not always possible to compute solutions for scenarios with egds using an SQL script. Then, we identify many practical cases in which this is possible, and develop a best-effort algorithm to do this. Experimental results show that our algorithm produces solutions of better quality with respect to those produced by previous algorithms, and scales nicely to large databases.

1. INTRODUCTION

Schema mappings are expressions that specify how an instance of a source database can be transformed into an instance of a target database. In recent years, they have received an increasing attention both from the research community and the tool market.

A *schema-mapping system* is used to support the process of generating and executing mappings in practical scenarios. It typically allows users to provide an abstract specification of the mapping as a set of correspondences among schema elements, specified through a friendly user-interface. Based on such specification, the mapping system will first generate a number of mappings – usually under the form of *tuple-generating dependencies (tgds)* [4] that correlate source tables with target tables; then, based on these mappings, an executable transformation, i.e., a runtime script in SQL or XQuery that can be practically used to run the mappings and generate solutions.

After the seminal Clio papers [20, 21] introduced the key algorithmic techniques needed to generate mappings from correspondences, and executable scripts from mappings, a solid theoretical foundation for the *data-exchange problem* was laid in the framework of data exchange research [10, 12]. More recently, sophisticated algorithms have been proposed [17, 24] to take advantage of the theoretical background of data exchange and adopt a more principled approach to the generation of solutions. In fact, a data-exchange problem may have many different solutions. *Universal solutions* [10] were first identified as preferred solutions, since they contain only information that follows from the source instance and the mapping. Among these, the notion of *core solution* [12] was identified as the "optimal one", since it is the smallest among universal solutions. Second-generation mapping systems [17, 24] made a significant step forward by introducing algorithms that materialize core solutions by using runtime SQL scripts.

Despite their increasing maturity, these techniques still suffer from a significant limitation: they provide very limited support for target dependencies. While target constraints are recognized as an important feature of data exchange, they introduce a number of subtleties in the computation of solutions. Notice that target constraints typically come in two forms: target tgds, and target *equality-generating dependencies (egds)* [4]. These two kinds of constraints have received an unequal share of attention in schema-mappings research. Target tgds corresponding to foreign key constraints – by far the most common form – are handled quite nicely by mapping systems [21, 18]: in fact, the intuition of chasing foreign keys to generate source-to-target tgds is at the core of the original Clio mapping-generation algorithm.

On the contrary, state-of-the-art schema-mapping systems cannot handle target egds, i.e., there is currently no system to efficiently generate optimal solutions for mapping scenarios with key constraints and functional dependencies, as discussed in the next example.

Motivation Suppose we want to solve the mapping problem shown in Figure 1. This is a typical *data-fusion example* [5] which requires to merge together data from three different tables – possibly from three different original data sources – as shown in Figure 1.a: (*i*) a table about students and their birthdates; (*ii*) a table about employees and their salaries; (*iii*) a table about drivers and the cars they drive. The target schema contains two tables, one about persons, the second about cars. On these tables, we have two keys:

20

The Golden Age

SILVER

GOLDEN

The Golden Age

- Manage Functional Dependencies is a key requirement to obtain quality solutions

SILVER GOLDEN

The Golden Age

- Manage Functional Dependencies is a key requirement to obtain quality solutions

Entity Fragmentation

title	year	author
On The Road	1951	NULL
On The Road	NULL	Kerouac
Post Office	1971	Bukowski
Animal Farm	1947	NULL
Animal Farm	NULL	Orwell

SILVER GOLDEN

The Golden Age

- Manage Functional Dependencies is a key requirement to obtain quality solutions

Entity Fragmentation

title	year	author
On The Road	1951	NULL
On The Road	NULL	Kerouac
Post Office	1971	Bukowski
Animal Farm	1947	NULL
Animal Farm	NULL	Orwell

SILVER GOLDEN

The Golden Age

- Manage Functional Dependencies is a key requirement to obtain quality solutions
- Algorithm to rewrites it into a new scenario without egds that can be efficiently implemented using an SQL script

Entity Fragmentation

title	year	author
On The Road	1951	NULL
On The Road	NULL	Kerouac
Post Office	1971	Bukowski
Animal Farm	1947	NULL
Animal Farm	NULL	Orwell

SILVER GOLDEN

The Golden Age

- Manage Functional Dependencies is a key requirement to obtain quality solutions
- Algorithm to rewrites it into a new scenario without egds that can be efficiently implemented using an SQL script

Entity Fragmentation

title	year	author
On The Road	1951	NULL
On The Road	NULL	Kerouac
Post Office	1971	Bukowski
Animal Farm	1947	NULL
Animal Farm	NULL	Orwell

- it is not always possible to enforce EGDs using a first-order language as SQL

The Golden Age

- Manage Functional Dependencies is a key requirement to obtain quality solutions
- Algorithm to rewrites it into a new scenario without egds that can be efficiently implemented using an SQL script

Entity Fragmentation

title	year	author
On The Road	1951	NULL
On The Road	NULL	Kerouac
Post Office	1971	Bukowski
Animal Farm	1947	NULL
Animal Farm	NULL	Orwell

- it is not always possible to enforce EGDs using a first-order language as SQL
  - best-effort algorithm

Tractable XML Data Exchange via Relations

Rada Chirkova #1, Leonid Libkin *2, Juan Reutter *3

#NC State University
1chirkova@csc.ncsu.edu

*University of Edinburgh
2libkin@inf.ed.ac.uk
3juan.reutter@ed.ac.uk

Abstract— **We consider data exchange for XML documents: given source and target schemas, a mapping between them, and a document conforming to the source schema, construct a target document and answer target queries in a way that is consistent with source information. The problem has primarily been studied in the relational context, in which data-exchange systems have also been built.**

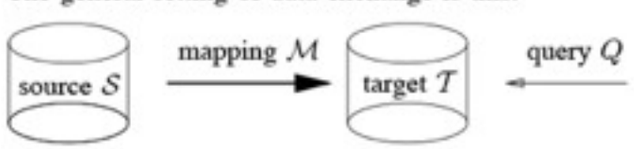
Since many XML documents are stored in relations, it is natural to consider using a relational system for XML data exchange. However, there is a complexity mismatch between query answering in relational and XML data exchange, which indicates that restrictions have to be imposed on XML schemas and mappings, and on XML shredding schemes, to make the use of relational systems possible.

We isolate a set of five requirements that must be fulfilled in order to have a faithful representation of the XML data-exchange problem by a relational translation. We then demonstrate that these requirements naturally suggest the inlining technique for data-exchange tasks. Our key contribution is to provide shredding algorithms for schemas, documents, mappings and queries, and demonstrate that they enable us to correctly perform XML data-exchange tasks using a relational system.

I. Introduction

Data exchange is the problem of finding an instance of a target schema, given an instance of a source schema and a schema mapping, that is, a specification of the relationship between the source and the target. Such a target instance should correctly represent information from the source instance under the constraints imposed by the target schema, and should allow one to evaluate queries on the target instance in a way that is semantically consistent with the source data. The problem has received much attention in the past few years, with several surveys already available [22], [9], [8].

The general setting of data exchange is this:



We have fixed source and target schemas, an instance $\mathcal{S}$ of the source schema, and a mapping $\mathcal{M}$ that specifies the relationship between the source and the target schemas. The

The mappings rarely specify the target instance completely, that is, for each source $\mathcal{S}$ and mapping $\mathcal{M}$, there could be multiple target instances $\mathcal{T}_1, \mathcal{T}_2, \ldots$ that satisfy the conditions of the mapping. Such instances are called *solutions*. The notion of query answering has to account for their non-uniqueness. Typically, one tries to compute *certain answers* $\text{CERTAIN}_{\mathcal{M}}(Q, \mathcal{S}) = \bigcap_i Q(\mathcal{T}_i)$, that is, answers independent of a particular solution chosen. Such an answer must be produced by evaluating some query – not necessarily Q but perhaps its *rewriting* Q_{rewr} over a particular solution $\mathcal{T}$: so that $Q_{\text{rewr}}(\mathcal{T}) = \text{CERTAIN}_{\mathcal{M}}(Q, \mathcal{S})$.

Thus, the key tasks in data exchange are: (a) choosing a particular solution $\mathcal{T}$ among $\{\mathcal{T}_1, \mathcal{T}_2, \ldots\}$ to materialize, and (b) finding a way of producing query answers over that solution by running a rewritten query Q_{rewr} over it. Usually one builds a so-called *universal* solution [13], [8]; these solutions behave particularly nicely with respect to query answering.

These basics of data exchange are independent of a particular model of data. Most research on data exchange, however, occurred in the relational context [13], [14], [22], [8] or slight extensions [33], [19]; the first paper that attempted to extend relational results to the XML context was [6], and a few followups have since appeared [4], [3]. They all concentrate on the algorithmic aspects of query answering and constructing solutions, with the main goal of isolating tractable cases. The problem these papers do not address is *how XML data exchange can be implemented*?

One possibility is to use a native XML DBMS such as [20], but this is not the most common route: XML data are commonly stored in relational DBMSs. In fact, many ETL products claim that they handle XML data simply by producing relational translations (known as *shredding* [23]). This leads to a two-step approach:

- first shred XML data into relations;
- then apply a relational data-exchange engine (and publish the result back as an XML document).

The approach seems very natural, but the key question is whether it will *work correctly*. That is, are we guaranteed to have the same result as we would have gotten had we implemented a native XML data-exchange system?

The Golden Age

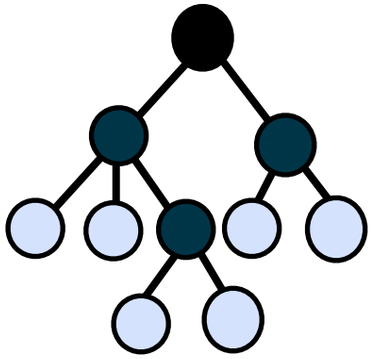
SILVER

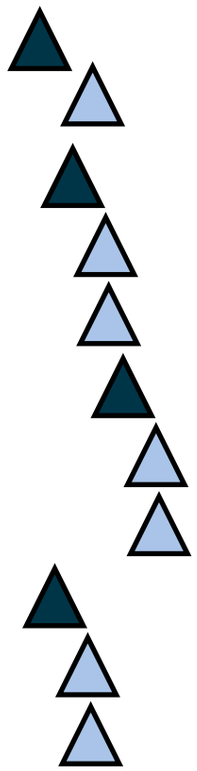
GOLDEN

The Golden Age

Nested Scenario

1





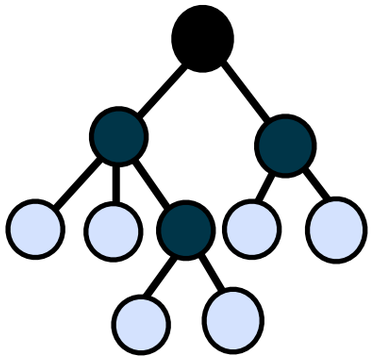
SILVER

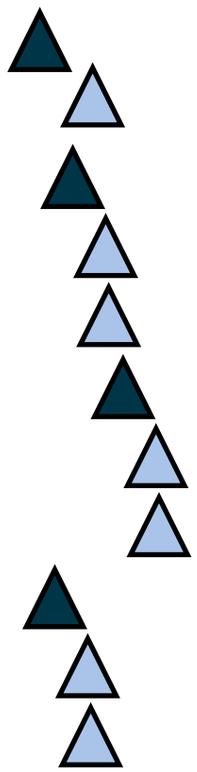
GOLDEN

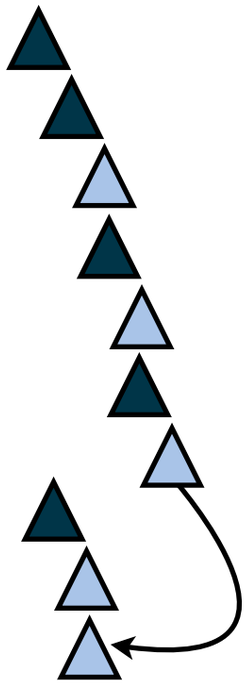
The Golden Age

Nested Scenario

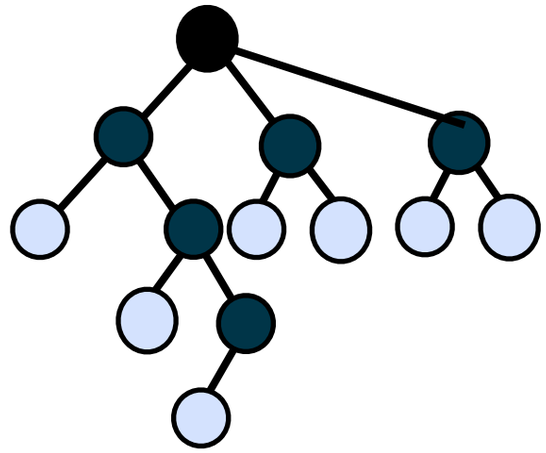
I







J



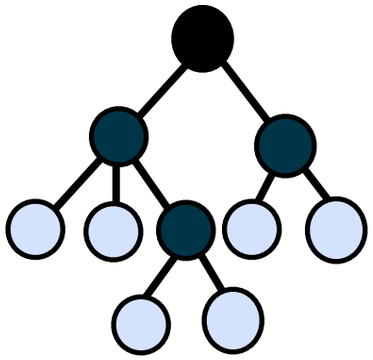
SILVER

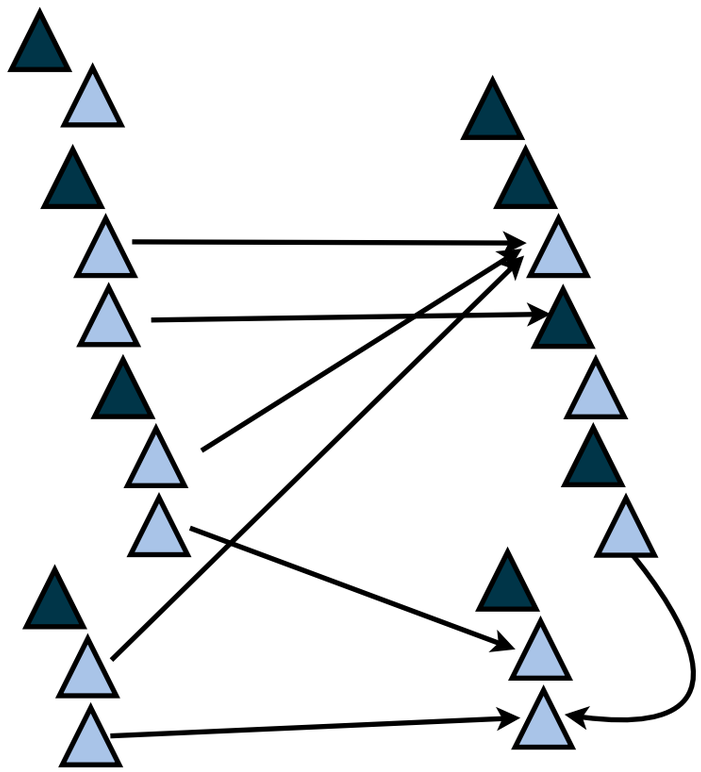
GOLDEN

The Golden Age

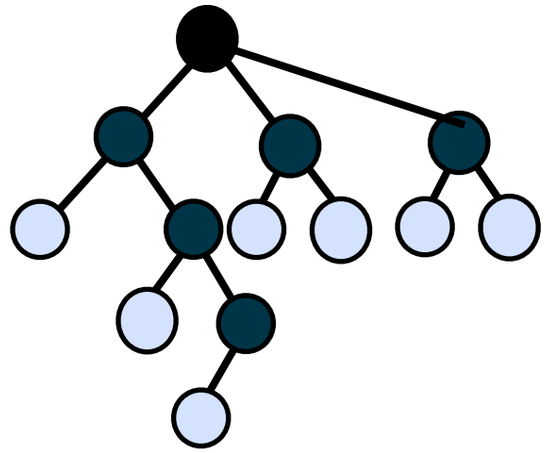
Nested Scenario

I





J

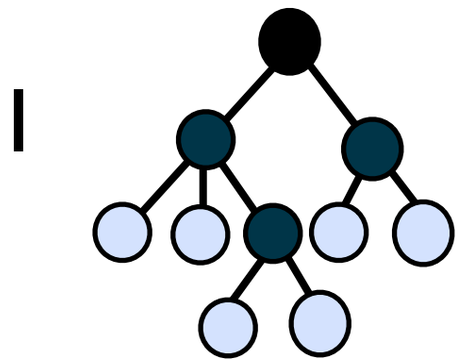


SILVER

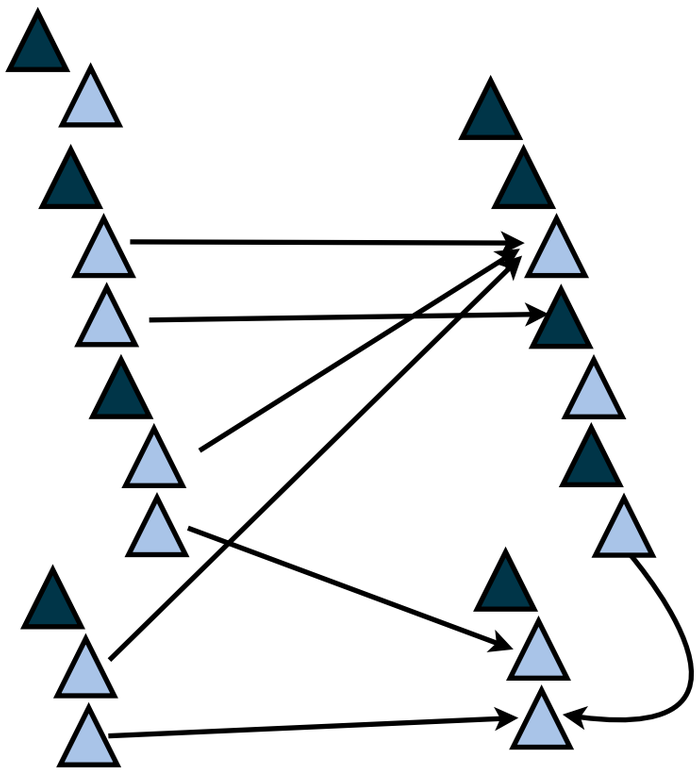
GOLDEN

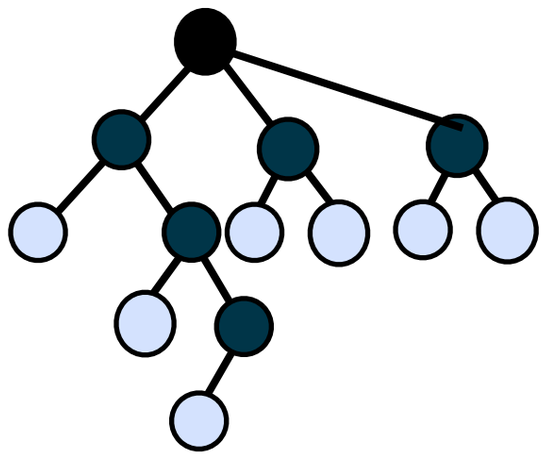
The Golden Age

Nested Scenario

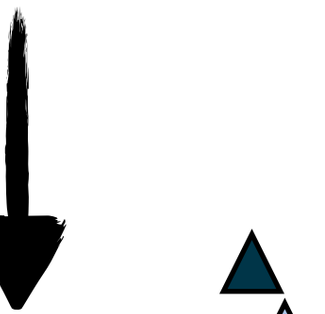


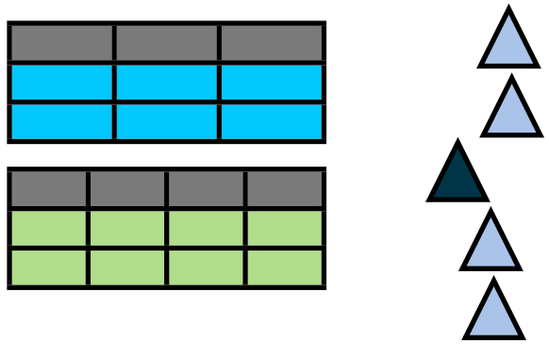
I





J





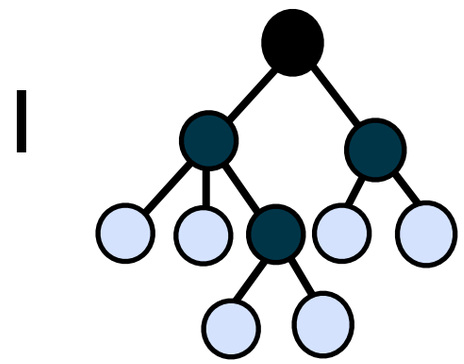
I

SILVER

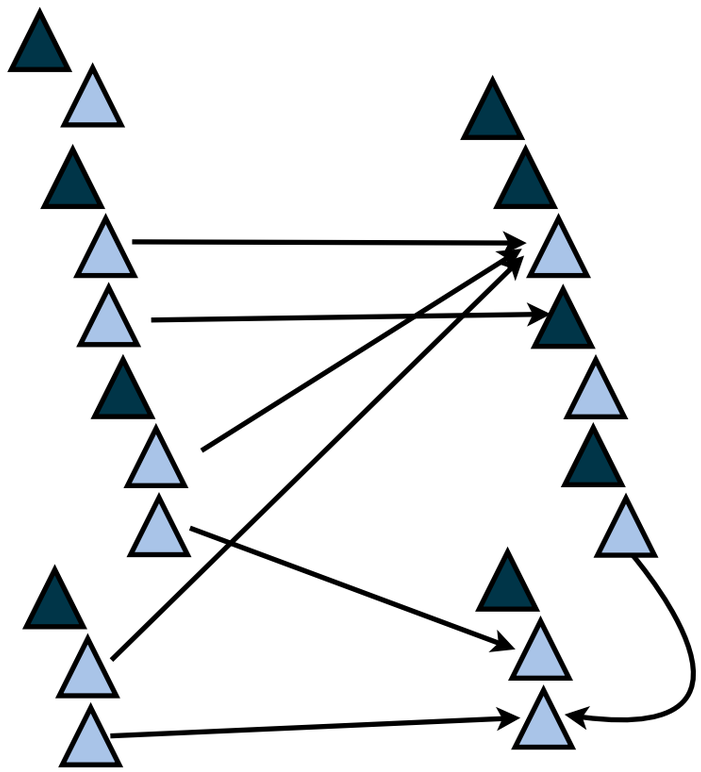
GOLDEN

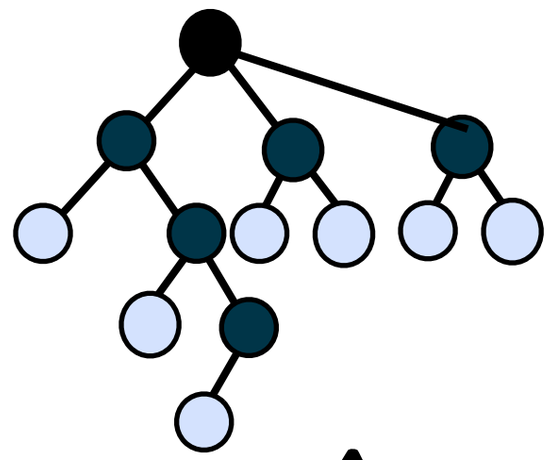
The Golden Age

Nested Scenario



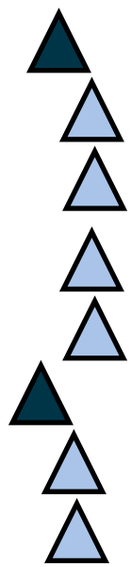
I



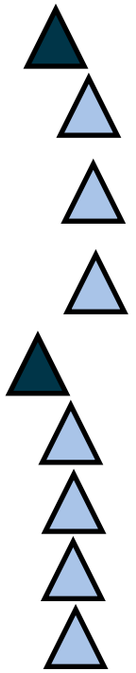


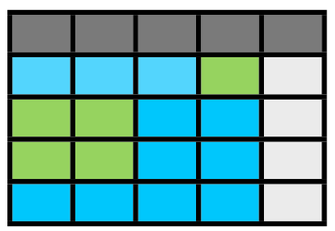
J

I



M





J

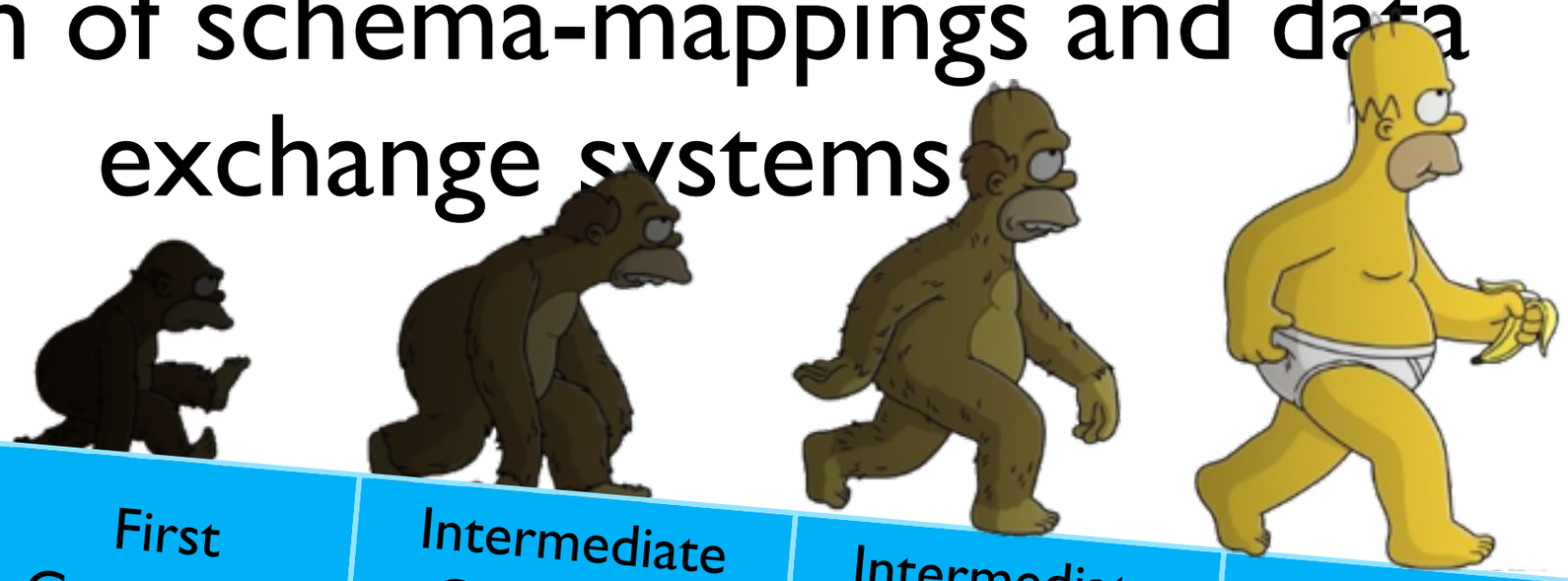
SILVER

GOLDEN

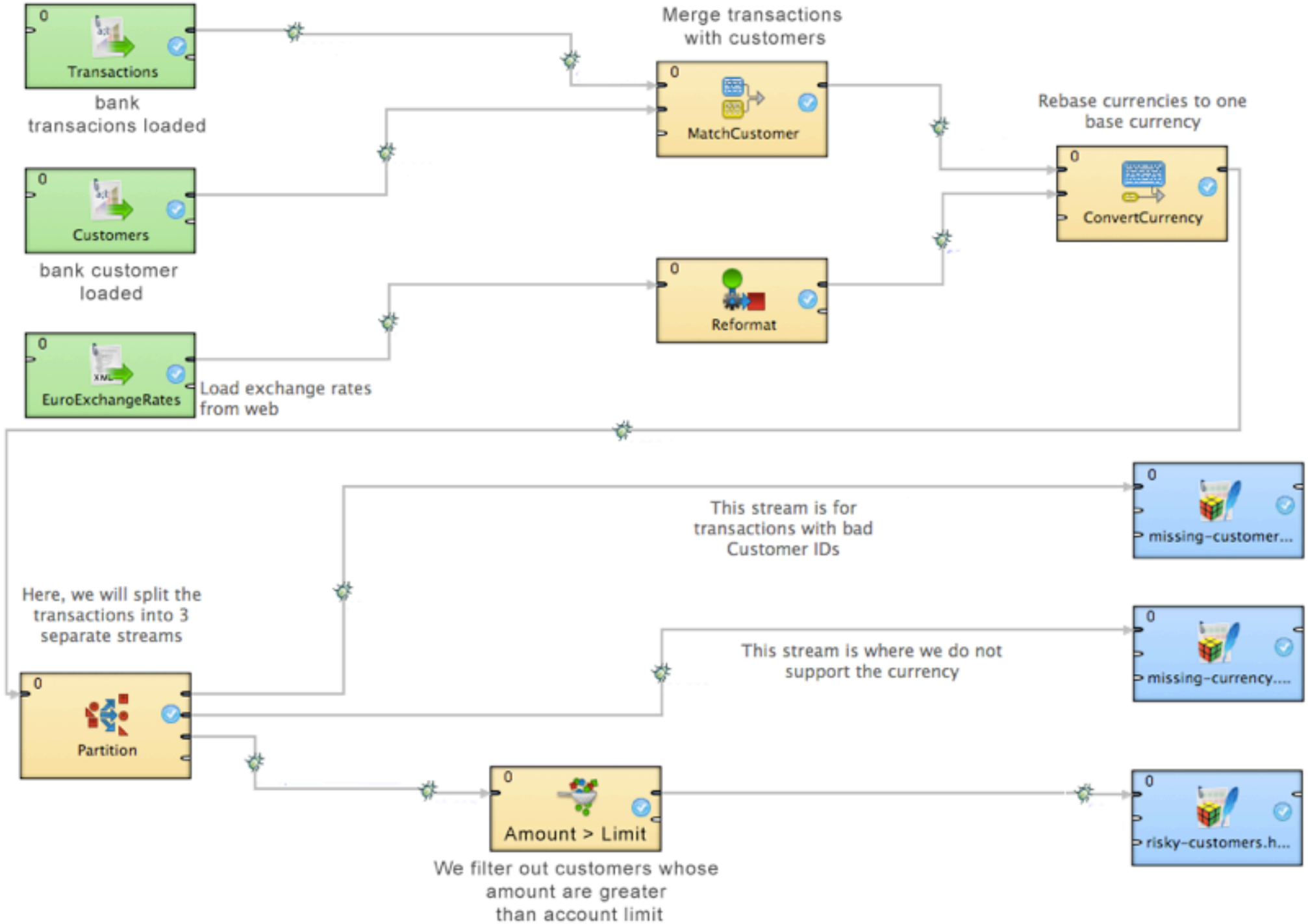
Evolution of schema-mappings and data exchange systems

	First Generation	Intermediate Generation (post-processing)	Intermediate Generation (rewriting)	Second Generation
Scalable Portable	✓	✗	✓	✓
Quality	✗	✓	✓	✓
Functional Dependencies	✗	✗	✗	✓
Nested Relation	✓	✗	✗	✓
	Popa et al., VLDB 2002	Fagin et al., TODS 2005	Mecca et al., SIGMOD 2009 ten Cate et al., PVLDB 2009	Marnette et al., VLDB 2010 Chirkova et al., CIKM 2011

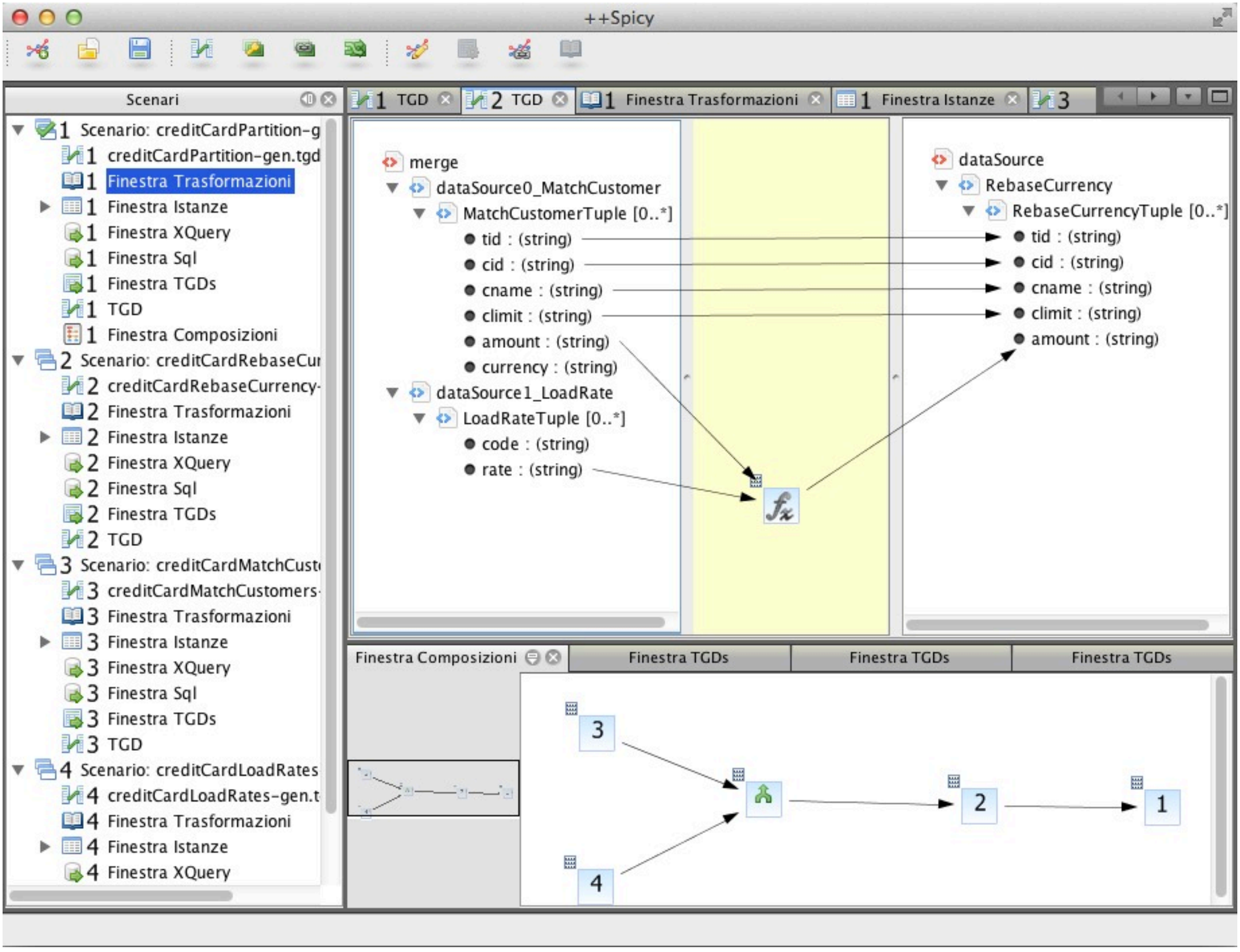
Evolution of schema-mappings and data exchange systems



	First Generation	Intermediate Generation (post-processing)	Intermediate Generation (rewriting)	Second Generation
Scalable Portable	✓	✗	✓	✓
Quality	✗	✓	✓	✓
Functional Dependencies	✗	✗	✗	✓
Nested Relation	✓	✗	✗	✓
	Popa et al., VLDB 2002	Fagin et al., TODS 2005	Mecca et al., SIGMOD 2009 ten Cate et al., PVLDB 2009	Marnette et al., VLDB 2010 Chirkova et al.,



SILVER GOLDEN


++Spicy
Scenari
1 Scenario: creditCardPartition-g
1 creditCardPartition-gen.tgd
1 Finestra Trasformazioni
1 Finestra Istanze
1 Finestra XQuery
1 Finestra Sql
1 Finestra TGDs
1 TGD
1 Finestra Composizioni
2 Scenario: creditCardRebaseCur
2 creditCardRebaseCurrency-
2 Finestra Trasformazioni
2 Finestra Istanze
2 Finestra XQuery
2 Finestra Sql
2 Finestra TGDs
2 TGD
3 Scenario: creditCardMatchCust
3 creditCardMatchCustomers-
3 Finestra Trasformazioni
3 Finestra Istanze
3 Finestra XQuery
3 Finestra Sql
3 Finestra TGDs
3 TGD
4 Scenario: creditCardLoadRates
4 creditCardLoadRates-gen.t
4 Finestra Trasformazioni
4 Finestra Istanze
4 Finestra XQuery
1 TGD
2 TGD
1 Finestra Trasformazioni
1 Finestra Istanze
3
merge
dataSource0_MatchCustomer
MatchCustomerTuple [0..*]
tid : (string)
cid : (string)
cname : (string)
climit : (string)
amount : (string)
currency : (string)
dataSource1_LoadRate
LoadRateTuple [0..*]
code : (string)
rate : (string)
dataSource
RebaseCurrency
RebaseCurrencyTuple [0..*]
tid : (string)
cid : (string)
cname : (string)
climit : (string)
amount : (string)
Finestra Composizioni
Finestra TGDs
Finestra TGDs
Finestra TGDs
3
4
2
1

The Golden Age

SILVER

GOLDEN

The Golden Age

- Due to previous limitations

SILVER GOLDEN

The Golden Age

- Due to previous limitations
  - adoption of schema mapping systems in real-life tasks has been quite slow

SILVER GOLDEN

The Golden Age

- Due to previous limitations
  - adoption of schema mapping systems in real-life tasks has been quite slow
- Recents results open the way to several applications

SILVER GOLDEN

The Golden Age

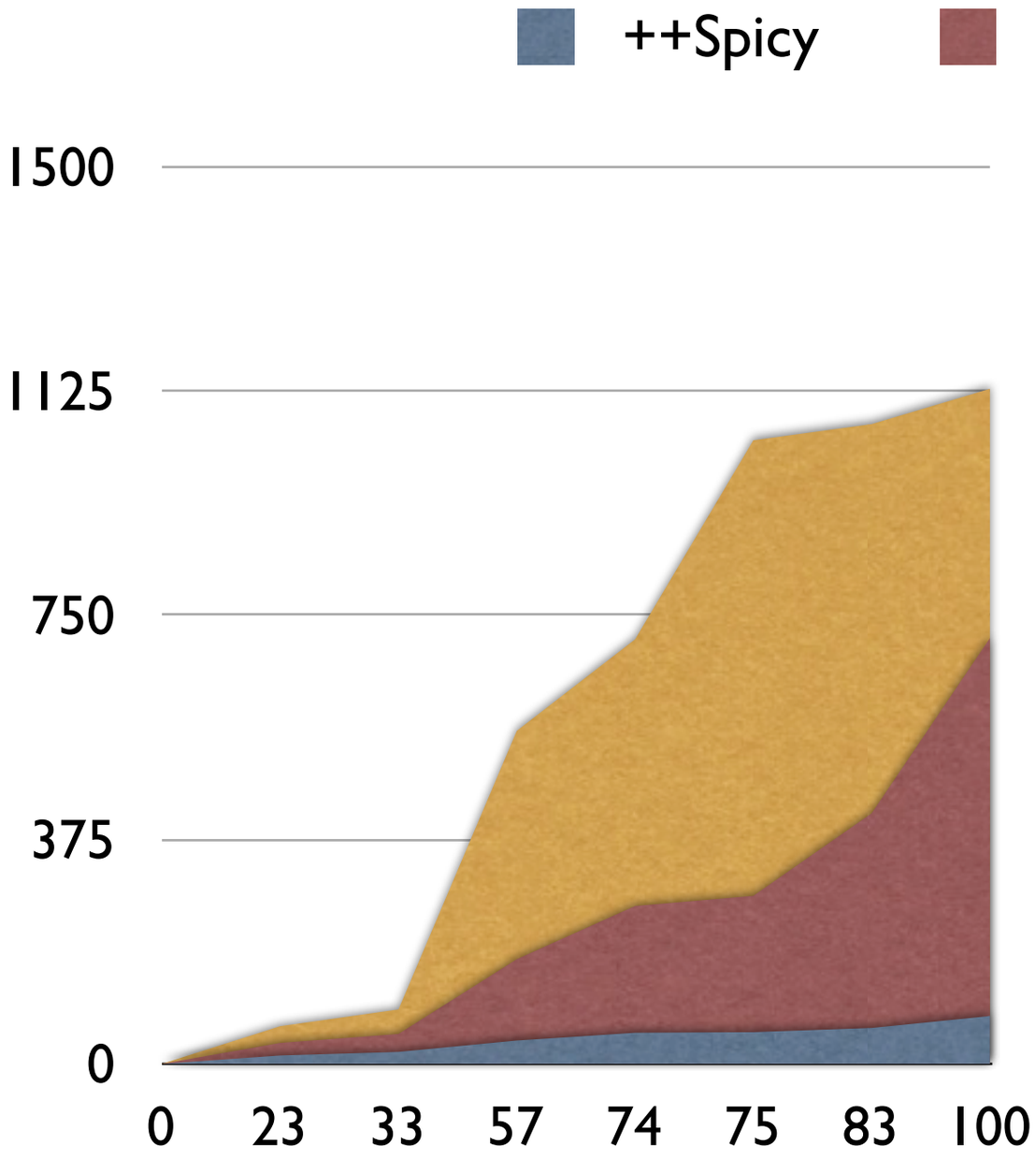
- Due to previous limitations
  - adoption of schema mapping systems in real-life tasks has been quite slow
- Recents results open the way to several applications
  - for the first time they become comparable, in some scenarios, to other Data Transformations System, as ETL

The Golden Age

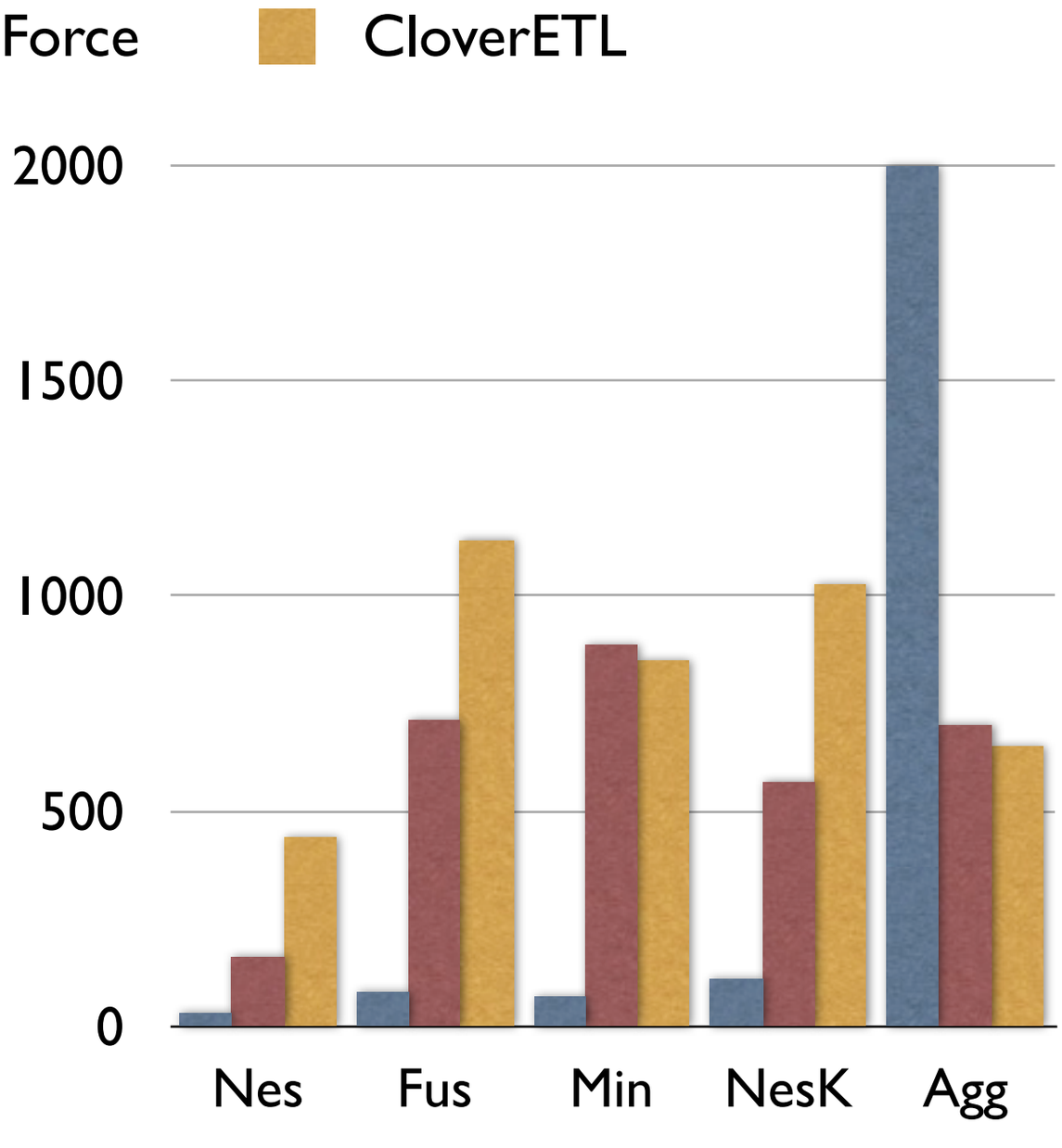
- Due to previous limitations
  - adoption of schema mapping systems in real-life tasks has been quite slow
- Recents results open the way to several applications
  - for the first time they become comparable, in some scenarios, to other Data Transformations System, as ETL
  - in other scenarios ETL remains more expressive

SILVER GOLDEN

The Golden Age



Quality-Effort Graphs - DataFusion Scenario



Effort to obtain 100% quality

SILVER GOLDEN

Thanks!