Measuring Inconsistency in a General Information Space

John Grant[1] Francesco Parisi[2]

[1]Department of Computer Science and UMIACS,
University of Maryland, College Park, USA,
grant@cs.umd.edu
[2]Department of Informatics, Modeling, Electronics and System Engineering (DIMES),
University of Calabria, Italy,
fparisi@dimes.unical.it

11th International Symposium on
Foundations of Information and Knowledge Systems (FoIKS)

February 17–20, 2020

Dortmund, Germany

Inconsistency in real-world information systems (1/3)

- Real-world applications often need to deal with inconsistent information
- E.g., relational databases are often inconsistent

Relation *Asset*

SN	*DateLoaned*	*Employee*	*DateReturned*
999	2015-02-01	123456789	2016-03-15
999	2015-02-01	123456789	2018-12-31
999	2013-06-15	222222222	2017-12-31
888	2016-12-01	222222222	2013-12-01
555	2014-07-01	333333333	2013-06-20
666	2014-07-01	333333333	2015-09-10
777	2014-07-01	333333333	2014-05-21

- c_1 : For every asset, the loan date must predate the return date
- c_2 : The serial number is a key for *Asset*
- c_3 : For every date and employee there can be at most 2 assets loaned

Inconsistency in real-world information systems (1/3)

- Real-world applications often need to deal with inconsistent information
- E.g., relational databases are often inconsistent

Relation *Asset*

SN	***DateLoaned***	***Employee***	***DateReturned***
999	2015-02-01	123456789	2016-03-15
999	2015-02-01	123456789	2018-12-31
999	2013-06-15	222222222	2017-12-31
888	2016-12-01	222222222	2013-12-01
555	2014-07-01	333333333	2013-06-20
666	2014-07-01	333333333	2015-09-10
777	2014-07-01	333333333	2014-05-21

- c_1 : For every asset, the loan date must predate the return date
- c_2 : The serial number is a key for *Asset*
- c_3 : For every date and employee there can be at most 2 assets loaned

Inconsistency in real-world information systems (1/3)

- Real-world applications often need to deal with inconsistent information
- E.g., relational databases are often inconsistent

Relation *Asset*

SN	***DateLoaned***	***Employee***	***DateReturned***
999	2015-02-01	123456789	2016-03-15
999	2015-02-01	123456789	2018-12-31
999	2013-06-15	222222222	2017-12-31
888	2016-12-01	222222222	2013-12-01
555	2014-07-01	333333333	2013-06-20
666	2014-07-01	333333333	2015-09-10
777	2014-07-01	333333333	2014-05-21

- c_1 : For every asset, the loan date must predate the return date
- c_2 : The serial number is a key for *Asset*
- c_3 : For every date and employee there can be at most 2 assets loaned

Inconsistency in real-world information systems (1/3)

- Real-world applications often need to deal with inconsistent information
- E.g., relational databases are often inconsistent

Relation *Asset*

SN	***DateLoaned***	***Employee***	***DateReturned***
999	2015-02-01	123456789	2016-03-15
999	2015-02-01	123456789	2018-12-31
999	2013-06-15	222222222	2017-12-31
888	2016-12-01	222222222	2013-12-01
555	2014-07-01	333333333	2013-06-20
666	2014-07-01	333333333	2015-09-10
777	2014-07-01	333333333	2014-05-21

- c_1 : For every asset, the loan date must predate the return date
- c_2 : The serial number is a key for *Asset*
- c_3 : For every date and employee there can be at most 2 assets loaned

Inconsistency in real-world information systems (1/3)

- Real-world applications often need to deal with inconsistent information
- E.g., relational databases are often inconsistent

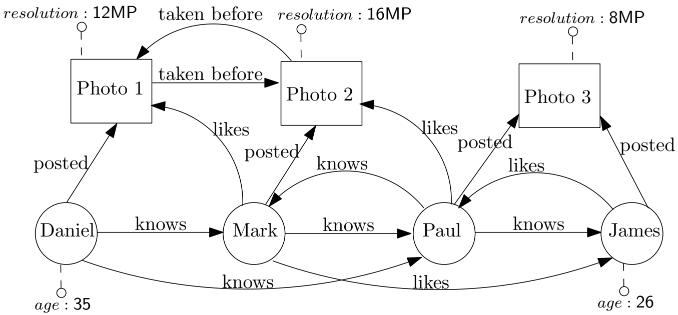
Relation *Asset*

SN	***DateLoaned***	***Employee***	***DateReturned***
999	2015-02-01	123456789	2016-03-15
999	2015-02-01	123456789	2018-12-31
999	2013-06-15	222222222	2017-12-31
888	2016-12-01	222222222	2013-12-01
555	2014-07-01	333333333	2013-06-20
666	2014-07-01	333333333	2015-09-10
777	2014-07-01	333333333	2014-05-21

- c_1 : For every asset, the loan date must predate the return date
- c_2 : The serial number is a key for *Asset*
- c_3 : For every date and employee there can be at most 2 assets loaned

Inconsistency in real-world information systems (2/3)

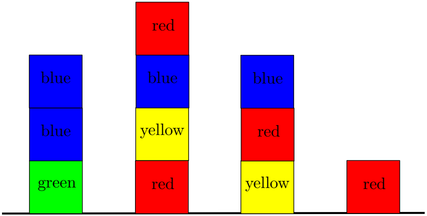
- As another example, graph databases may be inconsistent too



- c_1: Every *person* must have an associated *age* value
- c_2: Every *photo* must have an associated *resolution*
- c_3: The *taken before* relationship cannot be cyclic
- c_4: A given *photo* cannot be posted by different *persons*

Inconsistency in real-world information systems (3/3)

- An inconsistent Blocks-world



- c_1: No blue block can be on top of another blue block.
- c_2: There cannot be a yellow block that has a red block below it and a red block above it.
- c_3: There cannot be a red block on the table (i.e. at the bottom of a stack).
- c_4: No stack has both a green block and a blue block.
- c_5: At least one of the blocks is purple.
- c_6: There must be a blue block in at least 3 stacks.

Living with inconsistency (and measuring it)

- Inconsistency in real-world information systems can not be easily avoided
- Many inconsistency-tolerant approaches have been developed to live with inconsistency
- There are several proposals for mechanisms to handle inconsistent data
- A key issue in such situations is **measuring the amount of inconsistency**
- Measuring inconsistency allow us to assess its nature and understand the degree of the dirtiness of data
- Data quality is more and more important nowadays, the global market of data quality tools is expected to grow from USD 610.2 Million in 2017 to USD 1,376.7 Million by 2022 [MarketsandMarkets, 2019].

How to measure inconsistency of real word data?

- There are several ways to measure the amount of inconsistency in a knowledge base
- but most of this work applies only to knowledge bases formulated as sets of formulas in propositional logic
- Not really applicable to the way that information is actually stored (e.g., the 3 examples discussed before)
- We aim at extending inconsistency measuring to real world information

Dealing with general information spaces

- We define the concept of *general information space* which encompasses various types of databases and scenarios in AI systems
- We show how to transform any general information space to an *inconsistency equivalent* propositional knowledge base
- Apply propositional inconsistency measures to find the inconsistency of the general information space
- We demonstrate the transformation on 3 general information spaces (a relational database, a graph database, and a Blocks world scenario), where we apply several inconsistency measures after performing the transformation
- Our approach lifts the idea of inconsistency measure from propositional knowledge bases to a *range of different frameworks* used for storing real world data

Outline

General idea of an inconsistency measure

- The idea of an inconsistency measure is to assign a nonnegative number to a knowledge base (KB) that measures its inconsistency
- Propositional language of formulas, e.g.,
  $K_{ex} = \{a_1,\ a_2,\ a_3,\ a_4,\ \neg a_1 \vee \neg a_2,\ \neg a_2 \vee \neg a_3,\ a_4 \wedge a_5\}$

Definition (Inconsistency Measure)

Let $\mathcal{K}$ be the of all propositional knowledge bases. A function $I : \mathcal{K} \rightarrow \mathbb{R}^{\geq 0}_{\infty}$ is an *inconsistency measure* if the following conditions hold for all $K, K' \in \mathcal{K}$:

1. *Consistency*. $I(K) = 0$ iff K is consistent.
2. *Monotony*. If $K \subseteq K'$, then $I(K) \leq I(K')$.

- Consistency and Monotony are called (rationality) postulates
- Many other desirable properties for inconsistency measures have been investigated
- Consistency and Monotony are a minimal set for absolute measures (Monotony does not hold for relative measures)

Notation needed to define some concrete measures

- For a knowledge base K,
- $\mathrm{MI}(K)$ is the set of Minimal Inconsistent Subsets (MISs) of K
- If $\mathrm{MI}(K) = \{M_1, ..., M_n\}$, then Problematic$(K)$ = $M_1 \cup ... \cup M_n$ it the set of *problematic* formulas (involved in at least one inconsistency)
- Free(K) = $K \setminus$ Problematic(K) is the set of *free* formulas (not involved in an essential way in any inconsistency)

Example

For $K_{ex} = \{a_1,\ a_2,\ a_3,\ a_4,\ \neg a_1 \vee \neg a_2,\ \neg a_2 \vee \neg a_3,\ a_4 \wedge a_5\}$,
$\mathrm{MI}(K_{ex}) = \{\{a_1,\ a_2,\ \neg a_1 \vee \neg a_2\}, \{a_2,\ a_3,\ \neg a_2 \vee \neg a_3\}\}$, and
Problematic$(K_{ex}) = \{a_1,\ a_2,\ a_3,\ \neg a_1 \vee \neg a_2,\ \neg a_2 \vee \neg a_3\}$,
while a_4 and $a_4 \wedge a_5$ are free formulas

The first 2 measures: I_B and I_M

Definition (Propositional Inconsistency Measures)

For a knowledge base K, the inconsistency measures I_B and I_M are such that:

- $I_B(K) = 1$ if K is inconsistent and $I_B(K) = 0$ if K is consistent.
- $I_M(K) = |\mathrm{MI}(K)|$.

- I_B is also called the *drastic measure* [Hunter and Konieczny, 2008]: it simply distinguishes between consistent and inconsistent KBs.
- I_M counts the number of minimal inconsistent subsets [Hunter and Konieczny, 2008].

Example

For $K_{ex} = \{a_1, a_2, a_3, a_4, \neg a_1 \vee \neg a_2, \neg a_2 \vee \neg a_3, a_4 \wedge a_5\}$, we have that:

- $I_B(K_{ex}) = 1$ as it is inconsistent
- $I_M(K_{ex}) = 2$ as there are 2 minimal inconsistent subsets ($\mathrm{MI}(K_{ex}) = \{\{a_1, a_2, \neg a_1 \vee \neg a_2\}, \{a_2, a_3, \neg a_2 \vee \neg a_3\}\}$)

The next 2 measures: $I_\#$ and I_P

Definition (Propositional Inconsistency Measures)

For a knowledge base K, the inconsistency measures $I_\#$ and I_P are such that:

- $I_\#(K) = \begin{cases} 0 & \text{if K is consistent,} \\ \sum_{X \in \mathsf{MI}(K)} \frac{1}{|X|} & \text{otherwise.} \end{cases}$
- $I_P(K) = |\text{Problematic}(K)|$.

- $I_\#$ also counts the number of minimal inconsistent subsets, but it gives larger sets a smaller weight [Hunter and Konieczny, 2008]
- I_P counts the number of formulas that contribute essentially to one or more inconsistencies [Grant and Hunter, 2011].

Example

For $K_{ex} = \{a_1, a_2, a_3, a_4, \neg a_1 \vee \neg a_2, \neg a_2 \vee \neg a_3, a_4 \wedge a_5\}$, we have that:

- $I_\#(K_{ex}) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$ as both MISs consist of 3 formulas
- $I_P(K_{ex}) = 5$ as there are 5 problematic formulas

Other 2 measures: I_H and I_{nc}

Definition (Propositional Inconsistency Measures)

For a knowledge base K, the inconsistency measures I_H and I_{nc} are such that:

- $I_H(K) = \min\{|X| \mid X \subseteq K \text{ and } \forall M \in \mathsf{MI}(K)(X \cap M \neq \varnothing)\}$.
- $I_{nc}(K) = |K| - \max\{n \mid \forall K' \subseteq K : |K'| = n \text{ implies that } K' \text{ is consistent }\}$.

- I_H counts the minimal number of formulas whose deletion makes the set consistent [Grant and Hunter, 2013]
- I_{nc} uses the largest number such that all sets with that many formulas are consistent [Doder et al., 2010].

Example

For $K_{ex} = \{a_1, a_2, a_3, a_4, \neg a_1 \vee \neg a_2, \neg a_2 \vee \neg a_3, a_4 \wedge a_5\}$, we have that:

- $I_H(K_{ex}) = 1$ as deleting a_2 suffices to make K_{ex} consistent;
- $I_{nc}(K_{ex}) = 7 - 2 = 5$ as 2 is the largest number such that all subsets of size 2 are consistent;

A measure based on 3-valued logic: I_C (1/2)

- A classical interpretation i for K assigns each atom a that appears in a formula of K the truth value T or F, that is, $i : \text{Atoms}(K) \rightarrow \{T, F\}$
- I_C uses Priest's 3-valued logic (3VL), 3 truth values: T (*True*), F (*False*), and B (*Both*), where B indicates inconsistency

Formula	Truth value								
ϕ	T	T	T	B	B	B	F	F	F
ψ	T	B	F	T	B	F	T	B	F
$\phi \vee \psi$	T	T	T	T	B	B	T	B	F
$\phi \wedge \psi$	T	B	F	B	B	F	F	F	F
$\neg\phi$	F	F	F	B	B	B	T	T	T

Truth values on columns 1, 3, 7, and 9, give the classical semantics, and the other columns give the extended semantics.

- An interpretation i satisfies a formula iff the truth-value of the formula for i is T or B.

A measure based on 3-valued logic: I_C (2/2)

Definition (Propositional Inconsistency Measure I_C)

For a knowledge base K, the inconsistency measure I_C is such that:

- $I_C(K) = min\{|i^{-1}(B)|$ such that i satisfies every formula in $K\}$.

- I_C counts the minimal number of atoms that must be assigned the truth-value B in the three-valued logic by an interpretation that satisfies every formula in the KB [Grant and Hunter, 2011].

Example

For $K_{ex} = \{a_1, a_2, a_3, a_4, \neg a_1 \vee \neg a_2, \neg a_2 \vee \neg a_3, a_4 \wedge a_5\}$, we have that:

$I_C(K_{ex}) = 1$ as the following interpretation satisfies all the formulas:

$i(a_1) = i(a_3) = i(a_4) = i(a_5) = T,\ i(a_2) = B.$

Many other measures

- Many other propositional inconsistency measures have been defined
- A survey on the topic of inconsistency measurement can be found in the book [Grant and Martinez, 2018]
- The second chapter of the book provides a comprehensive survey of the inconsistency measures defined for propositional knowledge bases
- Our approach could use any of the inconsistency measures that have been formulated for propositional knowledge bases
- We will mainly focus on measures involving in some way the minimal inconsistent subsets, as they are particularly relevant in view of the transformation
- As an example of measure not directly defined using minimal inconsistent subset, we consider I_C (using 3VL logic)

Outline

Concept of General Information Space

- Lift the idea of inconsistency measure from propositional KBs to more complex cases that are useful in AI and databases

Definition (General Information Space)

A *general information space* $S = \langle F, U, C \rangle$ is a triple where

- F is the *framework* for the information,
- U is a *set of information units*, and
- C is a set of *requirements* that U must satisfy,

where the following hold:

A1 (*Consistency of individual information units*). The set of information units, U, simply gives some information and each unit is itself consistent

A2 (*Consistency of requirements*). There are no inconsistencies among requirements. All inconsistencies arise from the interaction of U and C

A3 (*Procedure for finding violations of the requirements*). For every requirement, there has to be a procedure that finds all violations of that requirement

Concept of General Information Space

- Lift the idea of inconsistency measure from propositional KBs to more complex cases that are useful in AI and databases

Definition (General Information Space)

A *general information space* $S = \langle F, U, C \rangle$ is a triple where

- F is the *framework* for the information,
- U is a *set of information units*, and
- C is a set of *requirements* that U must satisfy,

where the following hold:

A1 (*Consistency of individual information units*). The set of information units, U, simply gives some information and each unit is itself consistent

A2 (*Consistency of requirements*). There are no inconsistencies among requirements. All inconsistencies arise from the interaction of U and C

A3 (*Procedure for finding violations of the requirements*). For every requirement, there has to be a procedure that finds all violations of that requirement

Concept of General Information Space

- Lift the idea of inconsistency measure from propositional KBs to more complex cases that are useful in AI and databases

Definition (General Information Space)

A *general information space* $S = \langle F, U, C \rangle$ is a triple where

- F is the *framework* for the information,
- U is a *set of information units*, and
- C is a set of *requirements* that U must satisfy,

where the following hold:

A1 (*Consistency of individual information units*). The set of information units, U, simply gives some information and each unit is itself consistent

A2 (*Consistency of requirements*). There are no inconsistencies among requirements. All inconsistencies arise from the interaction of U and C

A3 (*Procedure for finding violations of the requirements*). For every requirement, there has to be a procedure that finds all violations of that requirement

Concept of General Information Space

- Lift the idea of inconsistency measure from propositional KBs to more complex cases that are useful in AI and databases

Definition (General Information Space)

A *general information space* $S = \langle F, U, C \rangle$ is a triple where

- F is the *framework* for the information,
- U is a *set of information units*, and
- C is a set of *requirements* that U must satisfy,

where the following hold:

A1 (*Consistency of individual information units*). The set of information units, U, simply gives some information and each unit is itself consistent

A2 (*Consistency of requirements*). There are no inconsistencies among requirements. All inconsistencies arise from the interaction of U and C

A3 (*Procedure for finding violations of the requirements*). For every requirement, there has to be a procedure that finds all violations of that requirement

A first example of General Information Space

- A relational database is an example of a general information space
- The framework is the database schema as well as the language used to describe the database
- The information units are the tuples
- The set of requirements is the set of integrity constraints
- Assumption A1, A2, and A3 hold in many real world scenarios
- A1 for databases: tuples are units of information that are usually assumed to be consistent when considered alone (without interacting with the integrity constraints)
- A2 for databases: integrity constraints are usually satisfiable (there exists a database instance that satisfies them)
- A3 for databases: procedures for checking inconsistency are well-known for large classes of integrity constraints

A first example of General Information Space

- A relational database is an example of a general information space
- The framework is the database schema as well as the language used to describe the database
- The information units are the tuples
- The set of requirements is the set of integrity constraints
- Assumption A1, A2, and A3 hold in many real world scenarios
- A1 for databases: tuples are units of information that are usually assumed to be consistent when considered alone (without interacting with the integrity constraints)
- A2 for databases: integrity constraints are usually satisfiable (there exists a database instance that satisfies them)
- A3 for databases: procedures for checking inconsistency are well-known for large classes of integrity constraints

Arity of a requirement (1/2)

- In many cases a *positive* number, called the *arity*, can be associated with each requirement
- It indicates *the minimal number of information units that together violate the requirement and thereby cause an inconsistency*
- In some cases, where the constraint is inconsistent with respect to the set of information units, the arity is set to 0

Example

Consider a relational database with a binary relation R_1.

- The arity of the constraint $\neg R_1(1,2)$ (saying that tuple $(1,2)$ cannot belong to R_1) is 1
- The arity of the functional dependency $\forall x_1 x_2 x_3 [R_1(x_1, x_2) \wedge R_1(x_1, x_3) \rightarrow x_2 = x_3]$ is 2
  It would be violated, for instance, by the two tuples: $(1,2)$ and $(1,3)$ in R_1 causing an inconsistency.

Arity of a requirement (1/2)

- In many cases a *positive* number, called the *arity*, can be associated with each requirement
- It indicates *the minimal number of information units that together violate the requirement and thereby cause an inconsistency*
- In some cases, where the constraint is inconsistent with respect to the set of information units, the arity is set to 0

Example

Consider a relational database with a binary relation R_1.

- The arity of the constraint $\neg R_1(1,2)$ (saying that tuple $(1,2)$ cannot belong to R_1) is 1
- The arity of the functional dependency $\forall x_1 x_2 x_3 [R_1(x_1,x_2) \wedge R_1(x_1,x_3) \rightarrow x_2 = x_3]$ is 2
  It would be violated, for instance, by the two tuples: $(1,2)$ and $(1,3)$ in R_1 causing an inconsistency.

Arity of a requirement (2/2)

Example

Consider a database a binary relation R_1 and a ternary relation R_2.

- Inclusion dependency: $\forall x_1 x_2 [R_1(x_1, x_2) \rightarrow \exists x_3 x_4 (R_2(x_2, x_3, x_4))]$.
  A violation is caused by a single tuple in R_1 whose second element is not in the first column of R_2. This means that the arity is 1
- $\exists x_1 x_2 R_2(1, x_1, x_2)$ states that *there must be* a tuple in R_2 relation whose first element is 1
  This requirement is *purely existential*: no deletion from the database would negate the violation.
  Hence the arity of such a constraint is 0.

Arity of a requirement (2/2)

Example

Consider a database a binary relation R_1 and a ternary relation R_2.

- Inclusion dependency: $\forall x_1 x_2 [R_1(x_1, x_2) \rightarrow \exists x_3 x_4 (R_2(x_2, x_3, x_4))]$.
  A violation is caused by a single tuple in R_1 whose second element is not in the first column of R_2. This means that the arity is 1
- $\exists x_1 x_2 R_2(1, x_1, x_2)$ states that *there must be* a tuple in R_2 relation whose first element is 1
  This requirement is *purely existential*: no deletion from the database would negate the violation.
  Hence the arity of such a constraint is 0.

Inconsistency of a general information space

- A requirement violation causes an inconsistency for S
- An *inconsistency* of S consists of one of two cases:

(1) The arity of the requirement c is a positive number k.
In this case an inconsistency of S is a set of k information units, $\{u_1, \ldots, u_k\}$, that violates c.
We write such an inconsistency as $\{u_1, \ldots, u_k, c\}$

(2) The arity of the requirement c is 0.
If c is violated by S, there is an inconsistency written as $\{c, \neg c\}$

- $\text{Inc}(S)$ is the set of inconsistencies of S

From a General Information Space to a Propositional KB

- Any general information space $S = \langle F, U, C \rangle$ can be transformed to a propositional KB K_S in such a way that *all the violations of the requirements are inconsistencies in the KB*
- The transformation loses some information: there is no way to go back from KB K_S to the original general information space S
- But the transformation is appropriate if we are interested in measuring inconsistency
- To measure the inconsistency of $S = \langle F, U, C \rangle$ according to an inconsistency measure I_x, apply I_x to the transformed space, i.e., $I_x(S) = I_x(K_S)$

Transformation (1/2)

Definition (Transformation (1/2))

The transformation from a general information space $S = \langle F, U, C \rangle$ to a propositional KB K_S is as follows.

- Let $A_U = \{a_1, \ldots, a_{|U|}\}$ be a set of $|U|$ propositional atoms.
- Define a bijective function $f : U \rightarrow A_U$ that assigns a distinct propositional atom to each information unit in U.
- Let $B_C = \{b_1, \ldots, b_{|C|}\}$ be another set of $|C|$ propositional atoms.
- Define a bijective function $h : C \rightarrow B_C$ that assigns a distinct propositional atom to each requirement in C.
- Let $\mathcal{F}_S$ be the set of propositional formulas using $A_U \cup B_C$.

...to be continued...

Transformation (1/2)

Definition (Transformation (1/2))

The transformation from a general information space $S = \langle F, U, C \rangle$ to a propositional KB K_S is as follows.

- Let $A_U = \{a_1, \ldots, a_{|U|}\}$ be a set of $|U|$ propositional atoms.
- Define a bijective function $f : U \rightarrow A_U$ that assigns a distinct propositional atom to each information unit in U.
- Let $B_C = \{b_1, \ldots, b_{|C|}\}$ be another set of $|C|$ propositional atoms.
- Define a bijective function $h : C \rightarrow B_C$ that assigns a distinct propositional atom to each requirement in C.
- Let $\mathcal{F}_S$ be the set of propositional formulas using $A_U \cup B_C$.

...to be continued...

Applying the (first part of the) transformation

Relation *Asset*

Atom	***SN***	***DateLoaned***	***Employee***	***DateReturned***	Tuple
a_1	999	2015-02-01	123456789	2016-03-15	t_1
a_2	999	2015-02-01	123456789	2018-12-31	t_2
a_3	999	2013-06-15	222222222	2017-12-31	t_3
a_4	888	2016-12-01	222222222	2013-12-01	t_4
a_5	555	2014-07-01	333333333	2013-06-20	t_5
a_6	666	2014-07-01	333333333	2015-09-10	t_6
a_7	777	2014-07-01	333333333	2014-05-21	t_7

- $A_U = \{a_1, \ldots, a_7\}$ is a set of atoms corresponding to the 7 tuples
- $f(t_i) = a_i$ for all i, $1 \leq i \leq 7$, assigns a distinct propositional atom to each information unit
- $B_C = \{b_1, \ldots, b_3\}$ is a set of atoms corresponding to the 3 constraints
- $h(c_i) = b_i$ for all i, $1 \leq i \leq 3$, assigns a distinct atom to each requirement
- $\mathcal{F}_S$ is the set of propositional logic formulas using $A_U \cup B_C$

Transformation (2/2)

Definition (Transformation (2/2))

...continued

- Define a function $g : C \to \mathcal{F}_S$ as follows:
  For each requirement $c \in C$ do as follows.
  1) If there is no violation of the requirement, then set $g(c) = h(c)$.
  2) If the arity of c is greater than 0, then a minimal inconsistency is formed by one or more information units together with c. Find all such sets, say $M_c = \{U_1, \ldots, U_k\}$ and suppose that $|U_i| = n$. Let $U_i = \{u_i^1, \ldots, u_i^n\}$ (where each u_i^j is an information unit). Define $\rho(U_i) = \neg f(u_i^1) \vee \ldots \vee \neg f(u_i^n)$ which is a propositional logic formula. Then, define

$$g(c) = (\bigwedge_{U_i \in M_c} \rho(U_i)) \wedge h(c).$$

  3) When the arity of c is 0, define $g(c) = \neg h(c) \wedge h(c)$.
- Define $K_S = \{f(u) \mid u \in U\} \cup \{g(c) \mid c \in C\}$.

Transformation (2/2)

Definition (Transformation (2/2))

...continued

- Define a function $g : C \to \mathcal{F}_S$ as follows:
  For each requirement $c \in C$ do as follows.
  1) If there is no violation of the requirement, then set $g(c) = h(c)$.
  2) If the arity of c is greater than 0, then a minimal inconsistency is formed by one or more information units together with c. Find all such sets, say $M_c = \{U_1, \ldots, U_k\}$ and suppose that $|U_i| = n$. Let $U_i = \{u_i^1, \ldots, u_i^n\}$ (where each u_i^j is an information unit). Define $\rho(U_i) = \neg f(u_i^1) \vee \ldots \vee \neg f(u_i^n)$ which is a propositional logic formula. Then, define

$$g(c) = (\bigwedge_{U_i \in M_c} \rho(U_i)) \wedge h(c).$$

  3) When the arity of c is 0, define $g(c) = \neg h(c) \wedge h(c)$.
- Define $K_S = \{f(u) \mid u \in U\} \cup \{g(c) \mid c \in C\}$.

Transformation (2/2)

Definition (Transformation (2/2))

...continued

- Define a function $g : C \rightarrow \mathcal{F}_S$ as follows:
  For each requirement $c \in C$ do as follows.
  1) If there is no violation of the requirement, then set $g(c) = h(c)$.
  2) If the arity of c is greater than 0, then a minimal inconsistency is formed by one or more information units together with c. Find all such sets, say $M_c = \{U_1, \ldots, U_k\}$ and suppose that $|U_i| = n$. Let $U_i = \{u_i^1, \ldots, u_i^n\}$ (where each u_i^j is an information unit). Define $\rho(U_i) = \neg f(u_i^1) \vee \ldots \vee \neg f(u_i^n)$ which is a propositional logic formula. Then, define

$$g(c) = \Big(\bigwedge_{U_i \in M_c} \rho(U_i)\Big) \wedge h(c).$$

  3) When the arity of c is 0, define $g(c) = \neg h(c) \wedge h(c)$.
- Define $K_S = \{f(u) \mid u \in U\} \cup \{g(c) \mid c \in C\}$.

Example of the requirements mapping (1/3)

Asset

Atom	*SN*	*DateLoaned*	*Employee*	*DateReturned*	Tuple
a_1	999	2015-02-01	123456789	2016-03-15	t_1
a_2	999	2015-02-01	123456789	2018-12-31	t_2
a_3	999	2013-06-15	222222222	2017-12-31	t_3
a_4	888	2016-12-01	222222222	2013-12-01	t_4
a_5	555	2014-07-01	333333333	2013-06-20	t_5
a_6	666	2014-07-01	333333333	2015-09-10	t_6
a_7	777	2014-07-01	333333333	2014-05-21	t_7

- $c_1 = \forall x_1 \ldots x_4[Asset(x_1, x_2, x_3, x_4) \rightarrow x_2 \leq x_4]$,
  i.e. for every asset, the loan date must predate the return date
- The arity of c_1 is 1
- The 3 tuples t_4, t_5, and t_7 each violate c_1
- Hence, $g(c_1) = \neg a_4 \wedge \neg a_5 \wedge \neg a_7 \wedge b_1$.

Example of the requirements mapping (1/3)

Asset

Atom	***SN***	***DateLoaned***	***Employee***	***DateReturned***	Tuple
a_1	999	2015-02-01	123456789	2016-03-15	t_1
a_2	999	2015-02-01	123456789	2018-12-31	t_2
a_3	999	2013-06-15	222222222	2017-12-31	t_3
a_4	888	2016-12-01	222222222	2013-12-01	t_4
a_5	555	2014-07-01	333333333	2013-06-20	t_5
a_6	666	2014-07-01	333333333	2015-09-10	t_6
a_7	777	2014-07-01	333333333	2014-05-21	t_7

- $c_1 = \forall x_1 \ldots x_4[Asset(x_1, x_2, x_3, x_4) \rightarrow x_2 \leq x_4]$,
  i.e. for every asset, the loan date must predate the return date
- The arity of c_1 is 1
- The 3 tuples t_4, t_5, and t_7 each violate c_1
- Hence, $g(c_1) = \neg a_4 \wedge \neg a_5 \wedge \neg a_7 \wedge b_1$.

Example of the requirements mapping (2/3)

Asset

Atom	***SN***	***DateLoaned***	***Employee***	***DateReturned***	Tuple
a_1	999	2015-02-01	123456789	2016-03-15	t_1
a_2	999	2015-02-01	123456789	2018-12-31	t_2
a_3	999	2013-06-15	222222222	2017-12-31	t_3
a_4	888	2016-12-01	222222222	2013-12-01	t_4
a_5	555	2014-07-01	333333333	2013-06-20	t_5
a_6	666	2014-07-01	333333333	2015-09-10	t_6
a_7	777	2014-07-01	333333333	2014-05-21	t_7

- $c_2 = \forall x_1 \dots x_7[Asset(x_1, x_2, x_3, x_4) \wedge Asset(x_1, x_5, x_6, x_7) \rightarrow (x_2 = x_5 \wedge x_3 = x_6 \wedge x_4 = x_7)]$, i.e. the serial number is a key for *Asset*
- The arity of c_2 is 2
- The 3 tuples t_1, t_2, and t_3 all have the same serial number but are not identical
- Hence, $g(c_2) = (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3) \wedge b_2$

Example of the requirements mapping (2/3)

Asset

Atom	*SN*	*DateLoaned*	*Employee*	*DateReturned*	Tuple
a_1	999	2015-02-01	123456789	2016-03-15	t_1
a_2	999	2015-02-01	123456789	2018-12-31	t_2
a_3	999	2013-06-15	222222222	2017-12-31	t_3
a_4	888	2016-12-01	222222222	2013-12-01	t_4
a_5	555	2014-07-01	333333333	2013-06-20	t_5
a_6	666	2014-07-01	333333333	2015-09-10	t_6
a_7	777	2014-07-01	333333333	2014-05-21	t_7

- $c_2 = \forall x_1 \ldots x_7[Asset(x_1, x_2, x_3, x_4) \wedge Asset(x_1, x_5, x_6, x_7) \rightarrow (x_2 = x_5 \wedge x_3 = x_6 \wedge x_4 = x_7)]$, i.e. the serial number is a key for *Asset*
- The arity of c_2 is 2
- The 3 tuples t_1, t_2, and t_3 all have the same serial number but are not identical
- Hence, $g(c_2) = (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3) \wedge b_2$

Example of the requirements mapping (3/3)

Asset

Atom	*SN*	*DateLoaned*	*Employee*	*DateReturned*	Tuple
a_1	999	2015-02-01	123456789	2016-03-15	t_1
a_2	999	2015-02-01	123456789	2018-12-31	t_2
a_3	999	2013-06-15	222222222	2017-12-31	t_3
a_4	888	2016-12-01	222222222	2013-12-01	t_4
a_5	555	2014-07-01	333333333	2013-06-20	t_5
a_6	666	2014-07-01	333333333	2015-09-10	t_6
a_7	777	2014-07-01	333333333	2014-05-21	t_7

- $c_3 = \forall x_1 \ldots x_8[Asset(x_1, x_2, x_3, x_4) \wedge Asset(x_5, x_2, x_3, x_6) \wedge Asset(x_7, x_2, x_3, x_8) \rightarrow (x_1 = x_5 \vee x_1 = x_7 \vee x_5 = x_7)]$, i.e., the numerical dependency *DateLoaned, Employee* $\rightarrow^2$ *SN*, meaning that for every date and employee there can be at most 2 assets loaned
- The arity of c_3 is 3
- t_5, t_6, and t_7 together violate this constraint
- Hence, $g(c_3) = (\neg a_5 \vee \neg a_6 \vee \neg a_7) \wedge b_3$

Example of the requirements mapping (3/3)

Asset

Atom	*SN*	*DateLoaned*	*Employee*	*DateReturned*	Tuple
a_1	999	2015-02-01	123456789	2016-03-15	t_1
a_2	999	2015-02-01	123456789	2018-12-31	t_2
a_3	999	2013-06-15	222222222	2017-12-31	t_3
a_4	888	2016-12-01	222222222	2013-12-01	t_4
a_5	555	2014-07-01	333333333	2013-06-20	t_5
a_6	666	2014-07-01	333333333	2015-09-10	t_6
a_7	777	2014-07-01	333333333	2014-05-21	t_7

- $c_3 = \forall x_1 \ldots x_8[Asset(x_1, x_2, x_3, x_4) \wedge Asset(x_5, x_2, x_3, x_6) \wedge Asset(x_7, x_2, x_3, x_8) \rightarrow (x_1 = x_5 \vee x_1 = x_7 \vee x_5 = x_7)]$, i.e., the numerical dependency *DateLoaned, Employee* $\rightarrow^2$ *SN*, meaning that for every date and employee there can be at most 2 assets loaned
- The arity of c_3 is 3
- t_5, t_6, and t_7 together violate this constraint
- Hence, $g(c_3) = (\neg a_5 \vee \neg a_6 \vee \neg a_7) \wedge b_3$

Resulting knowledge base

Asset

Atom	**SN**	**DateLoaned**	**Employee**	**DateReturned**	Tuple
a_1	999	2015-02-01	123456789	2016-03-15	t_1
a_2	999	2015-02-01	123456789	2018-12-31	t_2
a_3	999	2013-06-15	222222222	2017-12-31	t_3
a_4	888	2016-12-01	222222222	2013-12-01	t_4
a_5	555	2014-07-01	333333333	2013-06-20	t_5
a_6	666	2014-07-01	333333333	2015-09-10	t_6
a_7	777	2014-07-01	333333333	2014-05-21	t_7

- $c_1 = \forall x_1 \dots x_4 [Asset(x_1, x_2, x_3, x_4) \rightarrow x_2 \leq x_4]$
- $c_2 = \forall x_1 \dots x_7 [Asset(x_1, x_2, x_3, x_4) \wedge Asset(x_1, x_5, x_6, x_7) \rightarrow (x_2 = x_5 \wedge x_3 = x_6 \wedge x_4 = x_7)]$
- $c_3 = \forall x_1 \dots x_8 [Asset(x_1, x_2, x_3, x_4) \wedge Asset(x_5, x_2, x_3, x_6) \wedge Asset(x_7, x_2, x_3, x_8) \rightarrow (x_1 = x_5 \vee x_1 = x_7 \vee x_5 = x_7)]$

$$
\begin{aligned}
K_S = \{ & a_1, \dots, a_{13}, \\
& \neg a_4 \wedge \neg a_5 \wedge \neg a_7 \wedge b_1, \\
& (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3) \wedge b_2, \\
& (\neg a_5 \vee \neg a_6 \vee \neg a_7) \wedge b_3 \}
\end{aligned}
$$

Inconsistency equivalence

- We transform any general information space to an *inconsistency equivalent* propositional knowledge base
- Equivalence between the violation of the requirements C for S and the minimal inconsistent subsets of K_S:

Theorem

A general information space S and its transformation to a propositional knowledge base K_S are equivalent for inconsistencies in the sense that there is a bijection $m : \mathrm{Inc}(S) \to \mathrm{MI}(K_S)$.
Furthermore, for $M \in \mathrm{Inc}(S)$, $|M| = |m(M)|$.

Outline

Database as General Information Space

- For a relational database instance D over the database scheme $\mathcal{DS}$ with a set $\mathcal{C}$ of integrity constraints,
- The components of $S = \langle F, U, C \rangle$ are as follows:
- The framework F is the database scheme $\mathcal{DS}$ and the (function-free) first-order language using a set of uninterpreted constants and predicate symbols for relation names, as well as domains of the attributes for the evaluation of constants
- The set U of information units is the instance D (the set of the tuples in the relation instances), and
- the set C of requirements is the set $\mathcal{C}$ of integrity constraints

A more complex database example

Asset

Atom	***SN***	***DateLoaned***	***Employee***	***DateReturned***	Tuple
a_1	999	2015-02-01	123456789	2016-03-15	t_1
a_2	999	2015-02-01	123456789	2018-12-31	t_2
a_3	999	2013-06-15	222222222	2017-12-31	t_3
a_4	888	2016-12-01	222222222	2013-12-01	t_4
a_5	555	2014-07-01	333333333	2013-06-20	t_5
a_6	666	2014-07-01	333333333	2015-09-10	t_6
a_7	777	2014-07-01	333333333	2014-05-21	t_7

Employee

Atom	***ID***	***Name***	***HiringDate***	Tuple
a_8	333333333	Robert	1980-01-01	t_8
a_9	444444444	William	1975-06-01	t_9
a_{10}	123456789	William	1975-06-01	t_{10}

Family

Atom	***ID***	***Child***	***Project***	Tuple
a_{11}	123456789	Steve	Q1	t_{11}
a_{12}	123456789	Mary	Q2	t_{12}
a_{13}	123456789	Steve	Q2	t_{13}

Additional requirements

- $c_1 = \forall x_1 \ldots x_4[Asset(x_1, x_2, x_3, x_4) \rightarrow x_2 \leq x_4]$,
  i.e. for every asset, the loan date must predate the return date
- $c_2 = \forall x_1 \ldots x_7[\ Asset(x_1, x_2, x_3, x_4) \wedge Asset(x_1, x_5, x_6, x_7) \rightarrow (x_2 = x_5 \wedge x_3 = x_6 \wedge x_4 = x_7)]$,
  i.e. the serial number is a key for *Asset*
- $c_3 = \forall x_1 \ldots x_8[Asset(x_1, x_2, x_3, x_4) \wedge\ Asset(x_5, x_2, x_3, x_6) \wedge\ Asset(x_7, x_2, x_3, x_8) \rightarrow (x_1 = x_5 \vee x_1 = x_7 \vee x_5 = x_7)]$,
  i.e., the numerical dependency *DateLoaned, Employee* $\rightarrow^2$ *SN* whose meaning is that for every date and employee there can be at most 2 assets loaned
- $c_4 = \forall x_1 \ldots x_5[\ Employee(x_1, x_2, x_3) \wedge\ Employee(x_1, x_4, x_5) \rightarrow (x_2 = x_4 \wedge x_3 = x_5)]$,
  i.e., *ID* is a key for *Employee*
- $c_5 = \forall x_1 \ldots x_4[\ Employee(x_1, x_2, x_3) \wedge\ Employee(x_4, x_2, x_3) \rightarrow x_1 = x_4]$,
  i.e., the pair of attributes *Name* and *HiringDate* also form a key for *Employee*
- $c_6 = \forall x_1 \ldots x_6[\ Asset(x_1, x_2, x_3, x_4) \rightarrow \exists\ x_5, x_6\ Employee\ (x_3, x_5, x_6)]$
  i.e., the inclusion dependency $Asset[Employee] \subseteq Employee[ID]$
- $c_7 = \forall x_1 \ldots x_5[\ Family(x_1, x_2, x_3) \wedge\ Family(x_1, x_4, x_5) \rightarrow Family(x_1, x_2, x_5)]$,
  i.e. the multivalued dependency *Family*: $ID \rightarrow\rightarrow$ *Child*.
- $c_8 = \exists x_1 \ldots x_6[\ Family(x_1, x_2, x_3) \wedge\ Family(x_4, x_5, x_6) \wedge x_1 \neq x_4)]$
  i.e., there must be at least two distinct employees referenced in the *Family* relation

Additional requirements

- $c_1 = \forall x_1 \ldots x_4[Asset(x_1, x_2, x_3, x_4) \rightarrow x_2 \leq x_4]$,
  i.e. for every asset, the loan date must predate the return date
- $c_2 = \forall x_1 \ldots x_7[\, Asset(x_1, x_2, x_3, x_4) \wedge Asset(x_1, x_5, x_6, x_7) \rightarrow (x_2 = x_5 \wedge x_3 = x_6 \wedge x_4 = x_7)]$,
  i.e. the serial number is a key for *Asset*
- $c_3 = \forall x_1 \ldots x_8[Asset(x_1, x_2, x_3, x_4) \wedge Asset(x_5, x_2, x_3, x_6) \wedge Asset(x_7, x_2, x_3, x_8) \rightarrow (x_1 = x_5 \vee x_1 = x_7 \vee x_5 = x_7)]$,
  i.e., the numerical dependency *DateLoaned, Employee* $\rightarrow^2$ *SN* whose meaning is that for every date and employee there can be at most 2 assets loaned
- $c_4 = \forall x_1 \ldots x_5[\, Employee(x_1, x_2, x_3) \wedge Employee(x_1, x_4, x_5) \rightarrow (x_2 = x_4 \wedge x_3 = x_5)]$,
  i.e., *ID* is a key for *Employee*
- $c_5 = \forall x_1 \ldots x_4[\, Employee(x_1, x_2, x_3) \wedge Employee(x_4, x_2, x_3) \rightarrow x_1 = x_4]$,
  i.e., the pair of attributes *Name* and *HiringDate* also form a key for *Employee*
- $c_6 = \forall x_1 \ldots x_6[\, Asset(x_1, x_2, x_3, x_4) \rightarrow \exists\, x_5, x_6\ Employee\ (x_3, x_5, x_6)]$
  i.e., the inclusion dependency $Asset[Employee] \subseteq Employee[ID]$
- $c_7 = \forall x_1 \ldots x_5[\, Family(x_1, x_2, x_3) \wedge Family(x_1, x_4, x_5) \rightarrow Family(x_1, x_2, x_5)]$,
  i.e. the multivalued dependency *Family*: $ID \rightarrow\rightarrow Child$.
- $c_8 = \exists x_1 \ldots x_6[\, Family(x_1, x_2, x_3) \wedge Family(x_4, x_5, x_6) \wedge x_1 \neq x_4)]$
  i.e., there must be at least two distinct employees referenced in the *Family* relation

Applying the transformation

- $A_U = \{a_1, \ldots, a_{13}\}$ is a a set of propositional atoms corresponding to the 13 tuples
- $f(t_i) = a_i$ for all i, $1 \leq i \leq 13$; it assigns a distinct propositional atom to each information unit
- $B_C = \{b_1, \ldots, b_8\}$ is a set of propositional atoms corresponding to the 8 constraints
- $h(c_i) = b_i$ for all i, $1 \leq i \leq 8$; it assigns a distinct propositional atom to each requirement
- $\mathcal{F}_S$ is the set of propositional logic formulas using $A_U \cup B_C$

We have already seen the mapping for the first 3 constraints.
Now we show the mapping for the other constraints

Mapping a satisfied requirement

Employee

Atom	***ID***	***Name***	***HiringDate***	Tuple
a_8	333333333	Robert	1980-01-01	t_8
a_9	444444444	William	1975-06-01	t_9
a_{10}	123456789	William	1975-06-01	t_{10}

- $c_4 = \forall x_1 \ldots x_5[\ Employee(x_1, x_2, x_3) \wedge\ Employee(x_1, x_4, x_5) \rightarrow (x_2 = x_4 \wedge x_3 = x_5)]$, stating that *ID* is a key for *Employee*
- This constraint is satisfied
- Hence, $g(c_4) = b_4$

Mapping a requirement encoding a key constraint

Employee

Atom	***ID***	***Name***	***HiringDate***	Tuple
a_8	333333333	Robert	1980-01-01	t_8
a_9	444444444	William	1975-06-01	t_9
a_{10}	123456789	William	1975-06-01	t_{10}

- $c_5 = \forall x_1 \ldots x_4 [\ Employee(x_1, x_2, x_3) \wedge\ Employee(x_4,\ x_2, x_3) \rightarrow x_1 = x_4]$, that is, the pair of attributes *Name* and *HiringDate* also form a key for *Employee*
- The arity of c_5 is 2
- It is violated by the pair t_9 and t_{10}
- Hence, $g(c_5) = (\neg a_9 \vee \neg a_{10}) \wedge b_5$

Mapping an inclusion dependency

Asset

Atom	***SN***	***DateLoaned***	***Employee***	***DateReturned***	Tuple
a_1	999	2015-02-01	123456789	2016-03-15	t_1
a_2	999	2015-02-01	123456789	2018-12-31	t_2
a_3	999	2013-06-15	222222222	2017-12-31	t_3
a_4	888	2016-12-01	222222222	2013-12-01	t_4
a_5	555	2014-07-01	333333333	2013-06-20	t_5
a_6	666	2014-07-01	333333333	2015-09-10	t_6
a_7	777	2014-07-01	333333333	2014-05-21	t_7

Employee

Atom	***ID***	***Name***	***HiringDate***	Tuple
a_8	333333333	Robert	1980-01-01	t_8
a_9	444444444	William	1975-06-01	t_9
a_{10}	123456789	William	1975-06-01	t_{10}

- $c_6 = \forall x_1 \ldots x_6[\, Asset(x_1, x_2, x_3, x_4) \rightarrow \exists\, x_5, x_6\ Employee\ (x_3, x_5, x_6)]$ i.e., the inclusion dependency $Asset[Employee] \subseteq Employee[ID]$.
- The arity of c_6 is 1. It is violated separately by t_3 and t_4
- Hence, $g(c_6) = \neg a_3 \wedge \neg a_4 \wedge b_6$

Mapping a multivalued dependency

Family

Atom	***ID***	***Child***	***Project***	Tuple
a_{11}	123456789	Steve	Q1	t_{11}
a_{12}	123456789	Mary	Q2	t_{12}
a_{13}	123456789	Steve	Q2	t_{13}

- $c_7 = \forall x_1 \ldots x_5[\ Family(x_1, x_2, x_3) \wedge\ Family(x_1, x_4, x_5) \rightarrow Family(x_1, x_2, x_5)]$, i.e. the multivalued dependency [Fagin, 1977] *Family*: $ID \rightarrow\rightarrow Child$.
- The arity of c_7 is 2
- It is violated by the pair t_{11} and t_{12}
- Hence, $g(c_7) = (\neg a_{11} \vee \neg a_{12}) \wedge b_7$

Mapping a purely existential constraint

Family

Atom	***ID***	***Child***	***Project***	Tuple
a_{11}	123456789	Steve	Q1	t_{11}
a_{12}	123456789	Mary	Q2	t_{12}
a_{13}	123456789	Steve	Q2	t_{13}

- $c_8 = \exists x_1 \ldots x_6[\ Family(x_1, x_2, x_3) \wedge\ Family(x_4, x_5, x_6) \wedge x_1 \neq x_4)]$ stating that there must be at least two employees referenced in the *Family* relation
- The arity of c_8 is 0 and it is violated by the set of information units
- Hence, $g(c_8) = \neg b_8 \wedge b_8$

Resulting knowledge base for the database consisting of 13 tuples and with 8 constraints

$K_S = \{a_1, \ldots, a_{13},$ // 13 tuples
$\neg a_4 \wedge \neg a_5 \wedge \neg a_7 \wedge b_1,$ // intra-tuple constraint
$(\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3) \wedge b_2,$ // key constraint
$(\neg a_5 \vee \neg a_6 \vee \neg a_7) \wedge b_3,$ // numerical dependency
$b_4,$ // satisfied constraint
$(\neg a_9 \vee \neg a_{10}) \wedge b_5,$ // key constraint
$\neg a_3 \wedge \neg a_4 \wedge b_6,$ // inclusion dependency
$(\neg a_{11} \vee \neg a_{12}) \wedge b_7,$ // multivalued dependency
$\neg b_8 \wedge b_8\}.$ // purely existential constraint

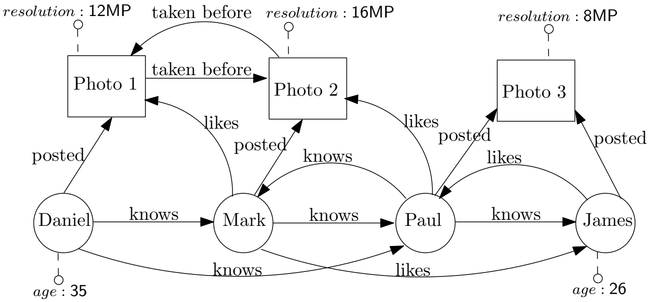
The Calculation of the Inconsistency Measures (1/2)

- Minimal inconsistent subsets for the knowledge base K_S resulting from the transformation
- $$\begin{aligned}\mathrm{MI}(K_S) = \{ & \{a_4, \neg a_4 \wedge \neg a_5 \wedge \neg a_7 \wedge b_1\},\\ & \{a_5, \neg a_4 \wedge \neg a_5 \wedge \neg a_7 \wedge b_1\},\\ & \{a_7, \neg a_4 \wedge \neg a_5 \wedge \neg a_7 \wedge b_1\},\\ & \{a_1, a_2, (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3) \wedge b_2\},\\ & \{a_1, a_3, (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3) \wedge b_2\},\\ & \{a_2, a_3, (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3) \wedge b_2\},\\ & \{a_5, a_6, a_7, (\neg a_5 \vee \neg a_6 \vee \neg a_7) \wedge b_3\},\\ & \{a_9, a_{10}, (\neg a_9 \vee \neg a_{10}) \wedge b_5\},\\ & \{a_3, \neg a_3 \wedge \neg a_4 \wedge b_6\},\\ & \{a_4, \neg a_3 \wedge \neg a_4 \wedge b_6\},\\ & \{a_{11}, a_{12}, (\neg a_{11} \vee \neg a_{12}) \wedge b_7\},\\ & \{\neg b_8 \wedge b_8\}\}\end{aligned}$$

The Calculation of the Inconsistency Measures (2/2)

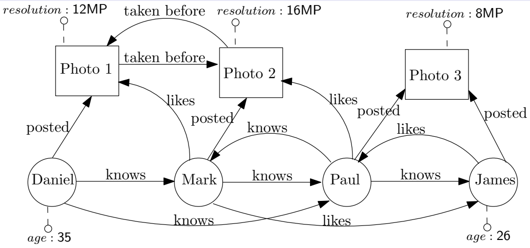
- $I_B(S) = 1$ as K_S is inconsistent.
- $I_M(S) = 12$ as there are 12 minimal inconsistent subsets for K_S.
- $I_\#(S) = 1 + 5 \times \frac{1}{2} + 5 \times \frac{1}{3} + \frac{1}{4} = \frac{65}{12}$ as there is one minimal inconsistent subset of size 1, 5 of size 2, 5 of size 3, and 1 of size 4 in K_S.
- $I_P(S) = 11 + 7 = 18$ as 11 atoms (i.e., tuples) plus 7 propositional formulas (i.e., constraints) are problematic in K_S.
- $I_H(S) = 7$ as the deletion of the 7 formulas of $g(c_i)$ for all i, $1 \leq i \leq 3$ and $5 \leq i \leq 8$ makes K_S consistent and there is no set of smaller cardinality that accomplishes the same.
- $I_{nc}(S) = 21$ as the set $\{\neg b_8 \wedge b_8\}$ has size 1 and is inconsistent.
- $I_C(S) = 8$ as there must be at least 8 atoms, for example a_2, a_3, a_4, a_5, a_7, a_9, a_{11}, and b_8, that must be given the value B for a 3-valued interpretation in order to satisfy all the formulas.

Graph Database as a General Information Space



- Components of $S = \langle F, U, C \rangle$:
- The framework F consists of basic information about the vertices and the edges of the graph, that is, the sets of *vertex names*, *edge labels*, and *vertex properties*
- Each vertex property has an associated domain. For instance, the domain of *type* includes person (circles) and media (rectangles)

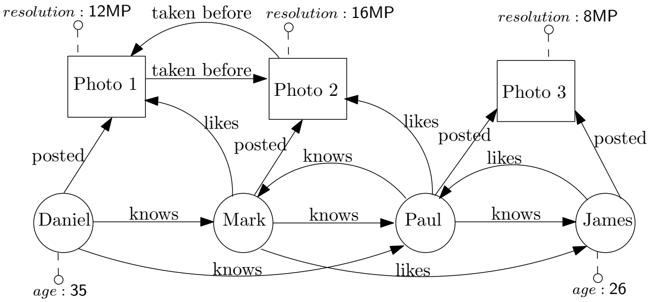
Data units



- The data units are the vertices and the edges

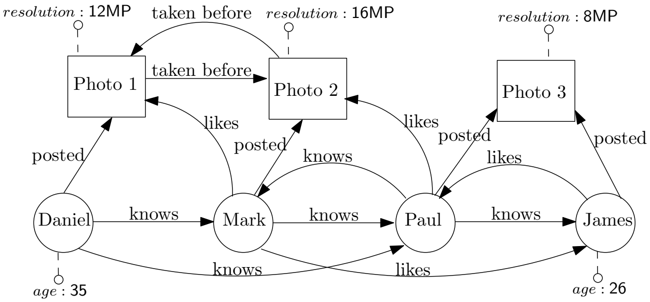
u_1 : (Photo 1, 12MP)
u_2 : (Photo 2, 16MP)
u_3 : (Photo 3, 8MP)
u_4 : (Daniel, 35)
u_5 : (Mark)
u_6 : (Paul)
u_7 : (James, 26)
u_8 : (Daniel, posted, Photo 1)
u_9 : (Daniel, knows, Mark)
u_{10} : (Daniel, knows, Paul)
u_{11} : (Mark, likes, Photo 1)
u_{12} : (Mark, posted, Photo 2)
u_{13} : (Mark, knows, Paul)
u_{14} : (Mark, likes, James)
u_{15} : (Paul, knows, Mark)
u_{16} : (Paul, likes, Photo 2)
u_{17} : (Paul, posted, Photo 3)
u_{18} : (Paul, knows, James)
u_{19} : (James, likes, Paul)
u_{20} : (James, posted, Photo 3)
u_{21} : (Photo 1, taken before, Photo 2)
u_{22} : (Photo 2, taken before, Photo 1)

Requirements



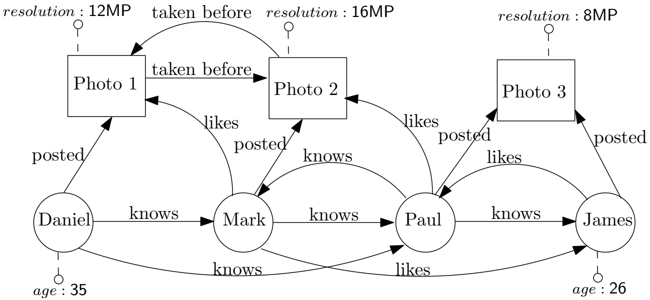
- c_1: Every person (circular vertex) must have an associated age value
- c_2: Every media (rectangular vertex) must have an associated resolution
- c_3: There may not be a cycle on rectangular vertices
- c_4: There cannot be 2 edges with the label “posted” going to the same rectangular vertex
- c_5: For every edge between circular vertices that has the label “likes” there must be another edge with the label “knows”

Transformation to a Propositional Knowledge Base



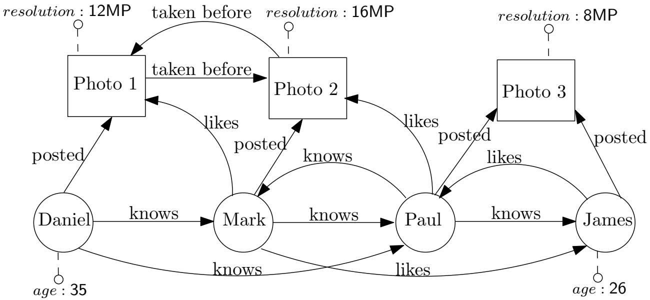
- $A_U = \{a_1, \ldots, a_{22}\}$ corresponding to the 7 vertices and 15 edges
- $f(u_i) = a_i$ for all i, $1 \leq i \leq 22$
- $B_C = \{b_1, \ldots, b_5\}$ corresponding to the 5 constraints
- $h(c_i) = b_i$ for all i, $1 \leq i \leq 5$
- $\mathcal{F}_S$ is the set of propositional formulas using $A_U \cup B_C$

Mapping the constraints



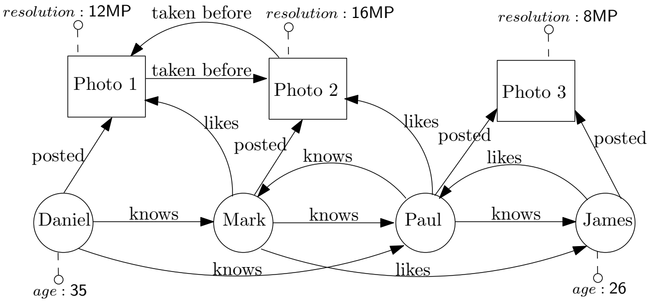
- c_1: Every person (circular vertex) must have an associated age value
  The arity of c_1 is 1.
  The two nodes u_5 (Mark) and u_6 (Paul) each violate c_1.
  Hence, $g(c_1) = \neg a_5 \wedge \neg a_6 \wedge b_1$.
- c_2: Every media (rectangular vertex) must have an associated resolution
  c_2 is satisfied. Hence, $g(c_2) = b_2$.

Mapping the constraints



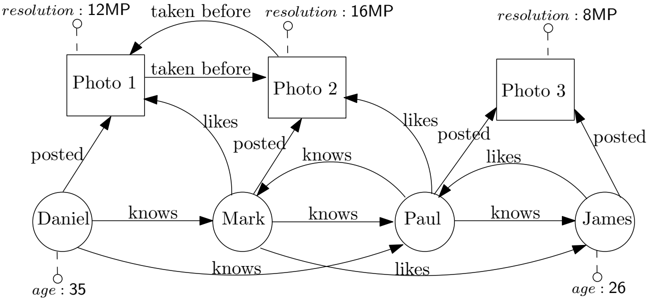
- c_1: Every person (circular vertex) must have an associated age value
  The arity of c_1 is 1.
  The two nodes u_5 (Mark) and u_6 (Paul) each violate c_1.
  Hence, $g(c_1) = \neg a_5 \wedge \neg a_6 \wedge b_1$.
- c_2: Every media (rectangular vertex) must have an associated resolution
  c_2 is satisfied. Hence, $g(c_2) = b_2$.

Mapping a circular path constraint



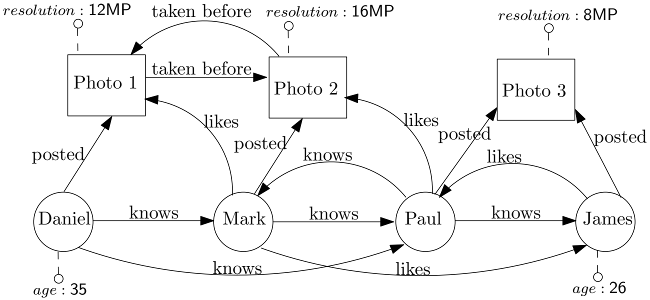
- c_3: There may not be a cycle on rectangular vertices
- This constraint does not have a fixed arity because a cycle does not have a fixed number of elements
- However, if it is violated its arity is greater than zero
- It is violated by the pair of edges u_{21} (Photo 1, taken before, Photo 2) and u_{22} (Photo 2, taken before, Photo 1)
- Hence, $g(c_3) = (\neg a_{21} \vee \neg a_{22}) \wedge b_3$.

Mapping a path denial constraints



- c_4: There cannot be 2 edges with the label “posted” going to the same rectangular vertex
- The arity of c_4 is 2
- It is violated by the pair of edges u_{17} (Paul, posted, Photo 3) and u_{20} (James, posted, Photo 3)
- Hence, $g(c_4) = (\neg a_{17} \vee \neg a_{20}) \wedge b_4$

Mapping an existential path constraints



- c_5: For every edge between circular vertices that has the label "likes" there must be another edge with the label "knows"
- The arity of c_5 is 1
- The two edges u_{14} (Mark, likes, James) and u_{19} (James, likes, Paul) each violate c_5
- Hence, $g(c_5) = \neg a_{14} \wedge \neg a_{19} \wedge b_5$

Resulting knowledge base

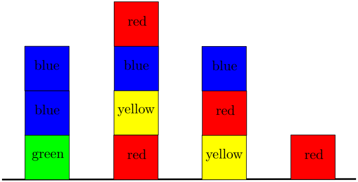
$$
\begin{aligned}
K_S = \{ & a_1, \ldots, a_{22}, && \text{// 22 vertices and edges} \\
& \neg a_5 \wedge \neg a_6 \wedge b_1, && \text{// existential property constraint} \\
& b_2, && \text{// satisfied constraint} \\
& (\neg a_{21} \vee \neg a_{22}) \wedge b_3, && \text{// circular path constraint} \\
& (\neg a_{17} \vee \neg a_{20}) \wedge b_4, && \text{// denial constraint} \\
& \neg a_{14} \wedge \neg a_{19} \wedge b_5 \}. && \text{// existential path constraint}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{MI}(K_S) = \{ & \{a_5, \neg a_5 \wedge \neg a_6 \wedge b_1\}, \\
& \{a_6, \neg a_5 \wedge \neg a_6 \wedge b_1\}, \\
& \{a_{21}, a_{22}, (\neg a_{21} \vee \neg a_{22}) \wedge b_3\}, \\
& \{a_{17}, a_{20}, (\neg a_{17} \vee \neg a_{20}) \wedge b_4\}, \\
& \{a_{14}, \neg a_{14} \wedge \neg a_{19} \wedge b_5\}, \\
& \{a_{19}, \neg a_{14} \wedge \neg a_{19} \wedge b_5\}\}.
\end{aligned}
$$

The Calculation of the Inconsistency Measures

- $I_B(S) = 1$ as K_S is inconsistent.
- $I_M(S) = 6$ as there are 6 minimal inconsistent subsets for K_S.
- $I_\#(S) = 4 \times \frac{1}{2} + 2 \times \frac{1}{3} = \frac{8}{3}$ as there are 4 minimal inconsistent subsets of size 2 and 2 minimal inconsistent subsets of size 3 for K_S.
- $I_P(S) = 8 + 4 = 12$ as 8 atoms (i.e., vertices and edges) plus 4 propositional formulas (i.e., the transformations of the constraints) are problematic in K_S.
- $I_H(S) = 4$ as the deletion of the 4 formulas: $g(c_1)$, $g(c_3)$, $g(c_4)$, and $g(c_5)$ makes K_S consistent and there is no smaller cardinality set that accomplishes the same.
- $I_{nc}(S) = 27 - 1 = 26$ as there is a minimal inconsistent subset of size 2.
- $I_C(S) = 6$ as a 3-valued interpretation must give at least a_5, a_6, a_{14}, a_{19}, one of a_{21} and a_{22}, and one of a_{17} and a_{20} the value B to satisfy all the formulas.

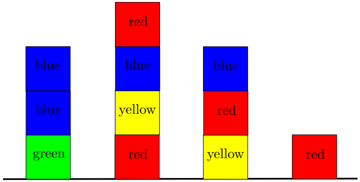
Components of a Blocks World Configuration



- The framework indicates that there is a finite number of colored blocks of the same size in stacks on a table, which is large enough to hold all (i.e., the number of stacks can be equal to number of blocks)
- Data units are the stack and the colors of the block in them
- $st_{i,j}$: *color* means that the block in stack *i* in the j^{th} position has that color

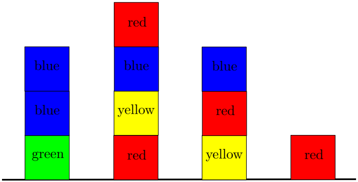
st_{11} : *green*	st_{12} : *blue*	st_{13} : *blue*	
st_{21} : *red*	st_{22} : *yellow*	st_{23} : *blue*	st_{24} : *red*
st_{31} : *yellow*	st_{32} : *red*	st_{33} : *blue*	
st_{41} : *red*			

Requirements for our Blocks World



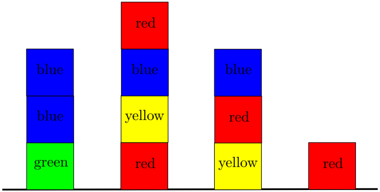
- c_1: No blue block can be on top of another blue block.
- c_2: There cannot be a yellow block that has a red block below it and a red block above it.
- c_3: There cannot be a red block on the table (i.e. at the bottom of a stack).
- c_4: No stack has both a green block and a blue block.
- c_5: At least one of the blocks is purple.
- c_6: There must be a blue block in at least 3 stacks.

Transformation (1/4)



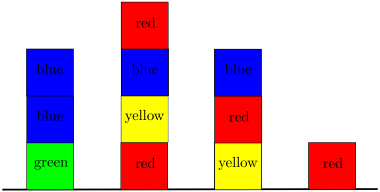
- $A_U = \{a_1, \ldots, a_{11}\}$ corresponding to the 11 blocks
- $f(st_{11}) = a_1$, $f(st_{12}) = a_2$, $f(st_{13}) = a_3$, $f(st_{21}) = a_4$, $f(st_{22}) = a_5$, $f(st_{23}) = a_6$, $f(st_{24}) = a_7$, $f(st_{31}) = a_8$, $f(st_{32}) = a_9$, $f(st_{33}) = a_{10}$, $f(st_{41}) = a_{11}$,
- $B_C = \{b_1, \ldots, b_6\}$ corresponding to the 6 constraints.
- $h(c_i) = b_i$ for all i, $1 \leq i \leq 6$.
- $\mathcal{F}_S$ is the set of propositional formulas using $A_U \cup B_C$.

Transformation (2/4)



- c_1: No blue block can be on top of another blue block
  The arity of c_1 is 2.
  The two blocks st_{12} and st_{13} together violate c_1.
  Hence, $g(c_1) = (\neg a_2 \vee \neg a_3) \wedge b_1$
- c_2: There cannot be a yellow block that has a red block below it and a red block above it
  The arity of c_2 is 3.
  The 3 blocks that together violate this constraint are st_{21}, st_{22}, and st_{24}.
  Hence, $g(c_2) = (\neg a_4 \vee \neg a_5 \vee \neg a_7) \wedge b_2$

Transformation (2/4)



- c_1: No blue block can be on top of another blue block

  The arity of c_1 is 2.

  The two blocks st_{12} and st_{13} together violate c_1.

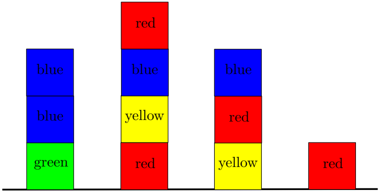
  Hence, $g(c_1) = (\neg a_2 \vee \neg a_3) \wedge b_1$
- c_2: There cannot be a yellow block that has a red block below it and a red block above it

  The arity of c_2 is 3.

  The 3 blocks that together violate this constraint are st_{21}, st_{22}, and st_{24}.

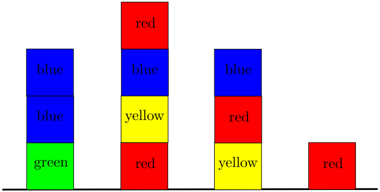
  Hence, $g(c_2) = (\neg a_4 \vee \neg a_5 \vee \neg a_7) \wedge b_2$

Transformation (3/4)



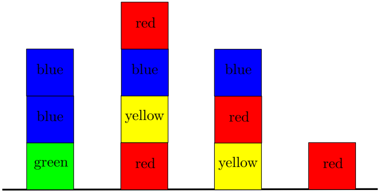
- c_3: There cannot be a red block on the table (i.e. at the bottom of a stack).
  The arity of c_3 is 1.
  The blocks st_{21} and st_{41} both violate this constraint.
  Hence, $g(c_3) = \neg a_4 \wedge \neg a_{11} \wedge b_3$
- c_4: No stack has both a green block and a blue block.
  The arity of c_4 is 2
  The blocks st_{11} and st_{12} as well as the blocks st_{11} and st_{13} violate this constraint
  Hence, $g(c_4) = (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge b_4$

Transformation (3/4)



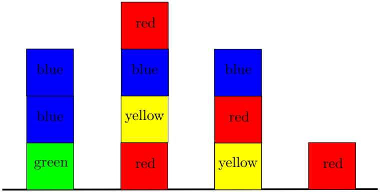
- c_3: There cannot be a red block on the table (i.e. at the bottom of a stack).
  The arity of c_3 is 1.
  The blocks st_{21} and st_{41} both violate this constraint.
  Hence, $g(c_3) = \neg a_4 \wedge \neg a_{11} \wedge b_3$
- c_4: No stack has both a green block and a blue block.
  The arity of c_4 is 2
  The blocks st_{11} and st_{12} as well as the blocks st_{11} and st_{13} violate this constraint
  Hence, $g(c_4) = (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge b_4$

Transformation (4/4)



- c_5: At least one of the blocks is purple
  The arity of c_5 is 0.
  There is no purple block in any stack.
  Hence, $g(c_5) = \neg b_5 \wedge b_5$
- c_6: There must be a blue block in at least 3 stacks
  This constraint is satisfied. Hence, $g(c_6) = b_6$

Transformation (4/4)



- c_5: At least one of the blocks is purple
  The arity of c_5 is 0.
  There is no purple block in any stack.
  Hence, $g(c_5) = \neg b_5 \wedge b_5$
- c_6: There must be a blue block in at least 3 stacks
  This constraint is satisfied. Hence, $g(c_6) = b_6$

Resulting knowledge base

$$
\begin{aligned}
K_S = \{ & a_1, \ldots, a_{11}, \\
& (\neg a_2 \vee \neg a_3) \wedge b_1, \\
& (\neg a_4 \vee \neg a_5 \vee \neg a_7) \wedge b_2, \\
& \neg a_4 \wedge \neg a_{11} \wedge b_3, \\
& (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge b_4, \\
& \neg b_5 \wedge b_5, \\
& b_6 \}.
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{MI}(K_S) = \{ & \{a_2, a_3, (\neg a_2 \vee \neg a_3) \wedge b_1\}, \\
& \{a_4, a_5, a_7, (\neg a_4 \vee \neg a_5 \vee \neg a_7) \wedge b_2\}, \\
& \{a_4, \neg a_4 \wedge \neg a_{11} \wedge b_3\}, \\
& \{a_{11}, \neg a_4 \wedge \neg a_{11} \wedge b_3\}, \\
& \{a_1, a_2, (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge b_4\}, \\
& \{a_1, a_3, (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge b_4\}, \\
& \{\neg b_5 \wedge b_5\}\}
\end{aligned}
$$

The Calculation of the Inconsistency Measures

- $I_B(S) = 1$ as K_S is inconsistent.
- $I_M(S) = 7$ as there are 7 minimal inconsistent subsets for K_S.
- $I_\#(S) = 1 + 2 \times \frac{1}{2} + 3 \times \frac{1}{3} + 1 \times \frac{1}{4} = \frac{13}{4}$ as there is 1 minimal inconsistent subset of size 1, 2 minimal inconsistent subsets of size 2, 3 minimal inconsistent subsets of size 3, and 1 minimal inconsistent subset of size 4 for K_S.
- $I_P(S) = 7 + 5 = 12$ as 7 atoms (i.e., colored block locations) plus 5 propositional formulas (i.e., the transformations of the requirements) are problematic in K_S.
- $I_H(S) = 5$ as the deletion of the 5 formulas: $g(c_1)$, $g(c_2)$, $g(c_3)$, $g(c_4)$, and $g(c_5)$ makes K_S consistent and there is no smaller cardinality set that accomplishes the same.
- $I_{nc}(S) = 17$ as there is a minimal inconsistent subset of size 1.
- $I_C(S) = 5$ as a 3-valued interpretation that satisfies all the formulas must give a_4, a_{11}, b_5, and at least 2 other atoms, for example, a_1 and a_2 the value B.

Outline

Conclusions and future work

- Inconsistency in real-world information systems can not be easily avoided
- We proposed a general approach for measuring inconsistency which encompasses various ways in which information is stored
- Since the transformation creates a propositional KB, all propositional (absolute) inconsistency measures ever proposed are applicable
- We do not deal with more general information spaces, such as those having additional concepts such as probabilities or fuzzyness

FW1 It would be interesting to look into broadening the concept of general information space

FW2 Consider relative inconsistency measures (where the ratio of inconsistency may decrease with the addition of consistent information)

FW3 Investigate the complexity of the transformation (it depends on what we consider as the size of the information space)

FW4 Restricting the information space (e.g. F=relational databases, C=denial constraints only) to get specific measures, postulate analysis and complexity results (some results @ ECAI 2020)

Conclusions and future work

- Inconsistency in real-world information systems can not be easily avoided
- We proposed a general approach for measuring inconsistency which encompasses various ways in which information is stored
- Since the transformation creates a propositional KB, all propositional (absolute) inconsistency measures ever proposed are applicable
- We do not deal with more general information spaces, such as those having additional concepts such as probabilities or fuzzyness

FW1 It would be interesting to look into broadening the concept of general information space

FW2 Consider relative inconsistency measures (where the ratio of inconsistency may decrease with the addition of consistent information)

FW3 Investigate the complexity of the transformation (it depends on what we consider as the size of the information space)

FW4 Restricting the information space (e.g. F=relational databases, C=denial constraints only) to get specific measures, postulate analysis and complexity results (some results @ ECAI 2020)

Conclusions and future work

- Inconsistency in real-world information systems can not be easily avoided
- We proposed a general approach for measuring inconsistency which encompasses various ways in which information is stored
- Since the transformation creates a propositional KB, all propositional (absolute) inconsistency measures ever proposed are applicable
- We do not deal with more general information spaces, such as those having additional concepts such as probabilities or fuzzyness

FW1 It would be interesting to look into broadening the concept of general information space

FW2 Consider relative inconsistency measures (where the ratio of inconsistency may decrease with the addition of consistent information)

FW3 Investigate the complexity of the transformation (it depends on what we consider as the size of the information space)

FW4 Restricting the information space (e.g. F=relational databases, C=denial constraints only) to get specific measures, postulate analysis and complexity results (some results @ ECAI 2020)

ECAI 2020 paper: On Measuring Inconsistency in Relational Databases with Denial Constraints

- Measuring the inconsistency by blaming database tuples only (integrity constraints are assumed to be irrefutable statements)

Measure(s)	$\mathbf{LV}_{\mathcal{I}}(D, v)$	$\mathbf{UV}_{\mathcal{I}}(D, v)$	$\mathbf{EV}_{\mathcal{I}}(D, v)$	$\mathbf{IM}_{\mathcal{I}}(D)$
$\mathcal{I}_B$,$\mathcal{I}_M$,$\mathcal{I}_\#$,$\mathcal{I}_P$	P	P	P	FP
$\mathcal{I}_A$	CP	CP	CP	$\#P$-complete*
$\mathcal{I}_H$, $\mathcal{I}_C$	$coNP$-complete	NP-complete	D^p-complete	$FP^{NP[\log n]}$-complete
$\mathcal{I}_\eta$	$coNP$-complete	NP-complete	D^p	FP^{NP}

Table: Data Complexity of Lower Value (**LV**), Upper Value (**UV**), Exact Value (**EV**), and Inconsistency Measurement (**IM**) problems.

ECAI 2020 paper: On Measuring Inconsistency in Relational Databases with Denial Constraints

- Measuring the inconsistency by blaming database tuples only (integrity constraints are assumed to be irrefutable statements)

Table: Postulates satisfaction for database inconsistency measures.

	Database Inconsistency Measures							
	$\mathcal{I}_B$	$\mathcal{I}_M$	$\mathcal{I}_\#$	$\mathcal{I}_P$	$\mathcal{I}_A$	$\mathcal{I}_H$	$\mathcal{I}_C$	$\mathcal{I}_\eta$
Free-Tuple Independence	✓	✓	✓	✓	✓	✓	✓⃝	✓
Penalty	✗	✓	✓	✓	✗	✗	✗	✗
Super-Additivity	✗	✓	✓	✓	✓⃝	✓	✓⃝	✗
MI-Separability	✗	✓	✓	✗	✗	✗	✗	✗
MI-Normalization	✓	✓	✗	✗	✗	✓	✓⃝	✗
Equal Conflict	✓	✓	✓	✓	✓	✓	✓⃝	✓

✓: satisfied for database measures (and satisfied for the corresponding propositional measure in the knowledge base setting).

✓⃝: satisfied for database measures but not for the corresponding propositional measure in the knowledge base setting.

✗: not satisfied for database measures (and not satisfied for propositional measures).

Thank you!

... any question?

Doder, D., Raskovic, M., Markovic, Z., and Ognjanovic, Z. (2010).
Measures of inconsistency and defaults.
Int. J. Approx. Reasoning, 51(7):832–845.

Fagin, R. (1977).
Multivalued dependencies and a new normal form for relational databases.
ACM Trans. Database Syst., 2(3):262–278.

Grant, J. and Hunter, A. (2011).
Measuring consistency gain and information loss in stepwise inconsistency resolution.
In *Proc. of European Conf. Symbolic and Quant. Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 362–373.

Grant, J. and Hunter, A. (2013).
Distance-based measures of inconsistency.
In *Proc. of ECSQARU*, pages 230–241.

Grant, J. and Martinez, M. V. (2018).
Measuring Inconsistency in Information.
College Publications.

Hunter, A. and Konieczny, S. (2008).
Measuring inconsistency through minimal inconsistent sets.
In *Proc. of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 358–366.

MarketsandMarkets (2019).
Data quality tools market by data type.
https://www.marketsandmarkets.com/Market-Reports/data-quality-tools-market-22437870.html.