Efficient Repair of Inconsistent Databases

Luís Cruz-Filipe

Dept. Mathematics and Computer Science, Univ. Southern Denmark (Denmark)
LabMAg (Portugal)

FoIKS 2014
March 5th, 2014

The Problem

Databases typically pose conditions on data ("integrity constraints")...

The Problem

Databases typically pose conditions on data ("integrity constraints")...

...but because of errors sometimes these conditions no longer hold.

The Problem

Databases typically pose conditions on data ("integrity constraints")...
...but because of errors sometimes these conditions no longer hold.

Question

How can we repair a database that no longer satisfies its integrity constraints?

Outline

1. Integrity constraints

Outline

1. Integrity constraints
2. Active integrity constraints

Outline

Outline

Outline

A typical company database

A set of integrity constraints

$$\text{employee}(X), \neg\text{insured}(X,' \text{basic}') \supset$$

$$\text{employee}(X), \text{onLeave}(X), \neg\text{salary}(X,' 0') \supset$$

$$\text{employee}(X), \text{salary}(X,' 0'), \neg\text{onLeave}(X) \supset$$

$$\text{salary}(X, Y), \text{salary}(X, Z), X \neq Z \supset$$

A typical company database

A set of integrity constraints

$$\text{employee}(X), \neg\text{insured}(X, '\text{basic}') \supset$$

$$\text{employee}(X), \text{onLeave}(X), \neg\text{salary}(X, '0') \supset$$

$$\text{employee}(X), \text{salary}(X, '0'), \neg\text{onLeave}(X) \supset$$

$$\text{salary}(X, Y), \text{salary}(X, Z), X \neq Z \supset$$

An inconsistent database

$$\{\text{employee}('\text{john}'), \text{salary}('\text{john}', '500'), \text{onLeave}('\text{john}')\}$$

Can we fix the problem automatically?

An inconsistent database

$$\{\text{employee}('\text{john}'), \text{salary}('\text{john}', '500'), \text{onLeave}('\text{john}')\}$$

Can we fix the problem automatically?

An inconsistent database

$$\{\text{employee}('\text{john}'), \text{salary}('\text{john}', '500'), \text{onLeave}('\text{john}')\}$$

Possible solutions

$$\mathcal{U}_1 = \{-\text{employee}('\text{john}')\}$$

$$\mathcal{U}_2 = \{+\text{insured}('\text{john}', '\text{basic}'), -\text{onLeave}('\text{john}')\}$$

$$\mathcal{U}_3 = \{+\text{insured}('\text{john}', '\text{basic}'), +\text{salary}('\text{john}', '0'), \\ -\text{salary}('\text{john}', '500')\}$$

Can we fix the problem automatically?

An inconsistent database

$$\{\text{employee}('\text{john}'), \text{salary}('\text{john}','500'), \text{onLeave}('\text{john}')\}$$

Possible minimal solutions

$$\mathcal{U}_1 = \{-\text{employee}('\text{john}')\}$$

$$\mathcal{U}_2 = \{+\text{insured}('\text{john}','\text{basic}'), -\text{onLeave}('\text{john}')\}$$

$$\mathcal{U}_3 = \{+\text{insured}('\text{john}','\text{basic}'), +\text{salary}('\text{john}','0'), \\ -\text{salary}('\text{john}','500')\}$$

Can we fix the problem automatically?

An inconsistent database

$$\{\text{employee}('\text{john}'), \text{salary}('\text{john}', '500'), \text{onLeave}('\text{john}')\}$$

Possible minimal solutions

$$\mathcal{U}_1 = \{-\text{employee}('\text{john}')\}$$

$$\mathcal{U}_2 = \{+\text{insured}('\text{john}', '\text{basic}'), -\text{onLeave}('\text{john}')\}$$

$$\mathcal{U}_3 = \{+\text{insured}('\text{john}', '\text{basic}'), +\text{salary}('\text{john}', '0'), \\ -\text{salary}('\text{john}', '500')\}$$

... but how automatic is this?

Historical background

This problem has been around since the 70s.

Historical background

This problem has been around since the 70s.

- Beeri & Vardi, 1981: Classification of integrity constraints into *universal*, *tuple-generating* and *equality-generating* dependencies.

Historical background

This problem has been around since the 70s.

- Beeri & Vardi, 1981: Classification of integrity constraints into *universal*, *tuple-generating* and *equality-generating* dependencies.
- Abiteboul, 1988: Seminal paper clearly identifying the problem and definining it as one of the great challenges in databases. Three main change operations: *addition*, *deletion* and *modification* of facts.

Historical background

This problem has been around since the 70s.

- Beeri & Vardi, 1981: Classification of integrity constraints into *universal*, *tuple-generating* and *equality-generating* dependencies.
- Abiteboul, 1988: Seminal paper clearly identifying the problem and definining it as one of the great challenges in databases. Three main change operations: *addition*, *deletion* and *modification* of facts.
- Eiter1992: Deciding whether a database can be repaired is typically Π_p^2- or co-Σ_P^2-complete.

Criteria for restricting repairs

- Winslett, 1990: Minimality of change – every repair should be a minimal set of changes, i.e. no proper subset of itself suffices.

Criteria for restricting repairs

- Winslett, 1990: Minimality of change – every repair should be a minimal set of changes, i.e. no proper subset of itself suffices.
- Przymusinski & Turner, 1997: Common-sense law of inertia – every repair should change only things that really must be changed, so no *ad-hoc* changes.

Criteria for restricting repairs

- Winslett, 1990: Minimality of change – every repair should be a minimal set of changes, i.e. no proper subset of itself suffices.
- Przymusinski & Turner, 1997: Common-sense law of inertia – every repair should change only things that really must be changed, so no *ad-hoc* changes.
- Flesca et al., 2004: Active integrity constraints – every integrity constraint should specify what actions are allowed to repair it.

Outline

Active integrity constraints

Motivation

Specify a constraint and propose possible solutions.

Active integrity constraints

Motivation

Specify a constraint and propose possible solutions.

Allows one to:

- express preferences among repairs
- eliminate options in the search for repairs

The company database, revisited

Original integrity constraints

$$\text{employee}(X), \neg\text{insured}(X,' \text{basic}') \supset$$

$$\text{employee}(X), \text{onLeave}(X), \neg\text{salary}(X,' 0') \supset$$

$$\text{employee}(X), \text{salary}(X,' 0'), \neg\text{onLeave}(X) \supset$$

$$\text{salary}(X, Y), \text{salary}(X, Z), X \neq Z \supset$$

The company database, revisited

Active integrity constraints

$$\text{employee}(X), \neg\text{insured}(X,'\text{basic}') \supset +\text{insured}(X,'\text{basic}')$$

$$\text{employee}(X), \text{onLeave}(X), \neg\text{salary}(X,'0') \supset +\text{salary}(X,'0')$$

$$\text{employee}(X), \text{salary}(X,'0'), \neg\text{onLeave}(X) \supset -\text{salary}(X,'0')$$

$$\text{salary}(X,Y), \text{salary}(X,Z), X \neq Z \supset -\text{salary}(X,Y)$$

The company database, revisited

Active integrity constraints

$$\text{employee}(X), \neg\text{insured}(X, '\text{basic}') \supset +\text{insured}(X, '\text{basic}')$$

$$\text{employee}(X), \text{onLeave}(X), \neg\text{salary}(X, '0') \supset +\text{salary}(X, '0')$$

$$\text{employee}(X), \text{salary}(X, '0'), \neg\text{onLeave}(X) \supset -\text{salary}(X, '0')$$

$$\text{salary}(X, Y), \text{salary}(X, Z), X \neq Z \supset -\text{salary}(X, Y)$$

No longer a solution

$$\mathcal{U}_1 = \{-\text{employee}('\text{john}')\}$$

$$\mathcal{U}_2 = \{+\text{insured}('\text{john}', '\text{basic}'), -\text{onLeave}('\text{john}')\}$$

The company database, revisited

Active integrity constraints

$$\text{employee}(X), \neg\text{insured}(X, '\text{basic}') \supset +\text{insured}(X, '\text{basic}')$$

$$\text{employee}(X), \text{onLeave}(X), \neg\text{salary}(X, '0') \supset +\text{salary}(X, '0')$$

$$\text{employee}(X), \text{salary}(X, '0'), \neg\text{onLeave}(X) \supset -\text{salary}(X, '0')$$

$$\text{salary}(X, Y), \text{salary}(X, Z), X \neq Z \supset -\text{salary}(X, Y)$$

Possible solution

$$\mathcal{U}_3 = \{+\text{insured}('\text{john}', '\text{basic}'), \\ +\text{salary}('\text{john}', '0'), -\text{salary}('\text{john}', '500')\}$$

Formal definition

Definition (Flesca2004)

An *active integrity constraint* is a formula of the form

$$L_1, \ldots, L_m \supset \alpha_1 \mid \ldots \mid \alpha_k$$

where $\{\alpha_1^D, \ldots, \alpha_k^D\} \subseteq \{L_1, \ldots, L_m\}$.

Formal definition

Definition (Flesca2004)

An *active integrity constraint* is a formula of the form

$$L_1, \ldots, L_m \supset \alpha_1 \mid \ldots \mid \alpha_k$$

where $\{\alpha_1^D, \ldots, \alpha_k^D\} \subseteq \{L_1, \ldots, L_m\}$.

Intuitive semantics:

- conjunction on the left ("body")
- disjunction on the right ("head")
- semantics of (normal) implication
- holds iff one of the L_is fails (but...)
- $\{\alpha_1^D, \ldots, \alpha_k^D\}$ are *updatable* literals

Repairs

Definition (Caroprese et al., 2006)

Let $\mathcal{I}$ be a database and η be a set of AICs. A *weak repair* for *I* and η is a consistent set $\mathcal{U}$ of update actions such that:

- $\mathcal{U}$ consists of essential actions only
- $\mathcal{I} \circ \mathcal{U} \models \eta$

A *repair* is a weak repair that is minimal w.r.t. inclusion.

Founded and justified repairs

Intuitively: if $\mathcal{U}$ is founded, then removing an action α from $\mathcal{U}$ causes some AIC with α in the head to be violated.

Founded and justified repairs

Intuitively: if $\mathcal{U}$ is founded, then removing an action α from $\mathcal{U}$ causes some AIC with α in the head to be violated.

Definition

A set of update actions $\mathcal{U}$ is founded w.r.t. $\mathcal{I}$ and η if, for every $\alpha \in \mathcal{U}$, there is a rule $r \in \eta$ such that $\alpha \in \mathrm{head}(r)$ and

$$\mathcal{I} \circ (\mathcal{U} \setminus \{\alpha\}) \not\models r\,.$$

Founded and justified repairs

Intuitively: if $\mathcal{U}$ is founded, then removing an action α from $\mathcal{U}$ causes some AIC with α in the head to be violated.

Definition

A set of update actions $\mathcal{U}$ is founded w.r.t. $\mathcal{I}$ and η if, for every $\alpha \in \mathcal{U}$, there is a rule $r \in \eta$ such that $\alpha \in \mathrm{head}(r)$ and

$$\mathcal{I} \circ (\mathcal{U} \setminus \{\alpha\}) \not\models r\,.$$

Definition

A (weak) repair that is founded is called a founded (weak) repair.

Founded and justified repairs

Intuitively: if $\mathcal{U}$ is founded, then removing an action α from $\mathcal{U}$ causes some AIC with α in the head to be violated.

Definition

A set of update actions $\mathcal{U}$ is founded w.r.t. $\mathcal{I}$ and η if, for every $\alpha \in \mathcal{U}$, there is a rule $r \in \eta$ such that $\alpha \in \text{head}(r)$ and

$$\mathcal{I} \circ (\mathcal{U} \setminus \{\alpha\}) \not\models r\,.$$

Definition

A (weak) repair that is founded is called a founded (weak) repair.

Founded repairs can exhibit *circularity of support*, so Caroprese et al. introduced *justified* repairs.

Complexity

Deciding whether...

there is a	for a DB is
weak repair	NP-complete
repair	NP-complete
founded weak repair	NP-complete
founded repair	Σ^2_P-complete
justified weak repair	Σ^2_P-complete
justified repair	Σ^2_P-complete

Outline

Motivation

Reduce the size of the problem by splitting the set of AICs into smaller sets.

Motivation

Reduce the size of the problem by splitting the set of AICs into smaller sets.

Goals:

- do not lose repairs;
- efficient combination of results.

Motivation

Reduce the size of the problem by splitting the set of AICs into smaller sets.

Goals:

- do not lose repairs;
- efficient combination of results.

Definition

Two (A)ICs r_1 and r_2 are *independent*, $r_1 \perp\!\!\!\perp r_2$, if the literals in their bodies do not share atoms.
Two sets of (A)ICs η_1 and η_2 are *independent*, $\eta_1 \perp\!\!\!\perp \eta_2$, if $r_1 \perp\!\!\!\perp r_2$ for every $r_1 \in \eta_1$ and $r_2 \in \eta_2$.

The company database, revisited

Active integrity constraints

$$\text{employee}(X), \neg\text{insured}(X, '\text{basic}') \supset +\text{insured}(X, '\text{basic}')$$

$$\text{employee}(X), \text{onLeave}(X), \neg\text{salary}(X, '0') \supset +\text{salary}(X, '0')$$

$$\text{employee}(X), \text{salary}(X, '0'), \neg\text{onLeave}(X) \supset -\text{salary}(X, '0')$$

$$\text{salary}(X, Y), \text{salary}(X, Z), X \neq Z \supset -\text{salary}(X, Y)$$

Independence vs. parallelization (I)

Theorem

Let $\eta = \eta_1 \cup \eta_2$ *with* $\eta_1 \perp\!\!\!\perp \eta_2$; $\mathcal{I}$ *be a database; and* $\mathcal{U}$ *be a weak repair for* $\mathcal{I}$ *and* η.

Define $\mathcal{U}_i$ *as the set of actions in* $\mathcal{U}$ *affecting literals in the bodies of rules in* η_i, *for* $i = 1, 2$.

Then:

- *each* $\mathcal{U}_i$ *is a weak repair for* $\mathcal{I}$ *and* η_i;
- *if* $\mathcal{U}$ *is a *-repair, then so is each* $\mathcal{U}_i$.

Furthermore, if every action in $\mathcal{U}$ *affects a literal in the body of a rule in* η, *then* $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

Independence vs. parallelization (I)

Theorem

Let $\eta = \eta_1 \cup \eta_2$ with $\eta_1 \perp\!\!\!\perp \eta_2$; $\mathcal{I}$ be a database; and $\mathcal{U}$ be a weak repair for $\mathcal{I}$ and η.
Define $\mathcal{U}_i$ as the set of actions in $\mathcal{U}$ affecting literals in the bodies of rules in η_i, for $i = 1, 2$.
Then:

- *each $\mathcal{U}_i$ is a weak repair for $\mathcal{I}$ and η_i;*
- *if $\mathcal{U}$ is a *-repair, then so is each $\mathcal{U}_i$.*

Furthermore, if every action in $\mathcal{U}$ affects a literal in the body of a rule in η, then $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

This last hypothesis is (very) reasonable in practice.

Independence vs. parallelization (I)

Theorem

Let $\eta = \eta_1 \cup \eta_2$ with $\eta_1 \perp\!\!\!\perp \eta_2$; $\mathcal{I}$ be a database; and $\mathcal{U}$ be a weak repair for $\mathcal{I}$ and η.
Define $\mathcal{U}_i$ as the set of actions in $\mathcal{U}$ affecting literals in the bodies of rules in η_i, for $i = 1, 2$.
Then:

- *each $\mathcal{U}_i$ is a weak repair for $\mathcal{I}$ and η_i;*
- *if $\mathcal{U}$ is a *-repair, then so is each $\mathcal{U}_i$.*

Furthermore, if every action in $\mathcal{U}$ affects a literal in the body of a rule in η, then $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

This last hypothesis is (very) reasonable in practice.
This means that we can parallelize the search for repairs without losing solutions.

Independence vs. parallelization (II)

Theorem

Let $\eta = \eta_1 \cup \eta_2$ with $\eta_1 \perp\!\!\!\perp \eta_2$; $\mathcal{I}$ be a database; and $\mathcal{U}_i$ be weak repairs for $\mathcal{I}$ and η_i, for $i = 1, 2$, such that all actions in $\mathcal{U}_i$ affect a literal in the body of a rule in η_i, and define $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

Then:

- *$\mathcal{U}$ is a weak repair for $\mathcal{I}$ and η;*
- *if each $\mathcal{U}_i$ is a *-repair, then so is $\mathcal{U}$.*

Independence vs. parallelization (II)

Theorem

Let $\eta = \eta_1 \cup \eta_2$ with $\eta_1 \perp\!\!\!\perp \eta_2$; $\mathcal{I}$ be a database; and $\mathcal{U}_i$ be weak repairs for $\mathcal{I}$ and η_i, for $i = 1, 2$, such that all actions in $\mathcal{U}_i$ affect a literal in the body of a rule in η_i, and define $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.
Then:

- *$\mathcal{U}$ is a weak repair for $\mathcal{I}$ and η;*
- *if each $\mathcal{U}_i$ is a *-repair, then so is $\mathcal{U}$.*

Thus parallelization does not add “new” (false) solutions.

Independence vs. parallelization (II)

Theorem

Let $\eta = \eta_1 \cup \eta_2$ *with* $\eta_1 \perp\!\!\!\perp \eta_2$; $\mathcal{I}$ *be a database; and* $\mathcal{U}_i$ *be weak repairs for* $\mathcal{I}$ *and* η_i, *for* $i = 1, 2$, *such that all actions in* $\mathcal{U}_i$ *affect a literal in the body of a rule in* η_i, *and define* $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.
Then:

- $\mathcal{U}$ *is a weak repair for* $\mathcal{I}$ *and* η;
- *if each* $\mathcal{U}_i$ *is a *-repair, then so is* $\mathcal{U}$.

Thus parallelization does not add “new” (false) solutions.
These results generalize to several independent sets of AICs.

Independence vs. parallelization (II)

Theorem

Let $\eta = \eta_1 \cup \eta_2$ *with* $\eta_1 \perp\!\!\!\perp \eta_2$; $\mathcal{I}$ *be a database; and* $\mathcal{U}_i$ *be weak repairs for* $\mathcal{I}$ *and* η_i, *for* $i = 1, 2$, *such that all actions in* $\mathcal{U}_i$ *affect a literal in the body of a rule in* η_i, *and define* $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.
Then:

- $\mathcal{U}$ *is a weak repair for* $\mathcal{I}$ *and* η;
- *if each* $\mathcal{U}_i$ *is a *-repair, then so is* $\mathcal{U}$.

Thus parallelization does not add “new” (false) solutions.
These results generalize to several independent sets of AICs.
A stronger notion of independence can be defined if only founded or justified (weak) repairs are sought.

The company database, revisited

Active integrity constraints

$$\text{employee}(X), \neg\text{insured}(X, '\text{basic}') \supset +\text{insured}(X, '\text{basic}')$$

$$\text{employee}(X), \text{onLeave}(X), \neg\text{salary}(X, '0') \supset +\text{salary}(X, '0')$$

$$\text{employee}(X), \text{salary}(X, '0'), \neg\text{onLeave}(X) \supset -\text{salary}(X, '0')$$

$$\text{salary}(X, Y), \text{salary}(X, Z), X \neq Z \supset -\text{salary}(X, Y)$$

The company database, revisited

Active integrity constraints

$$\text{employee}(X), \neg\text{insured}(X, '\text{basic}') \supset +\text{insured}(X, '\text{basic}')$$

$$\text{employee}(X), \text{onLeave}(X), \neg\text{salary}(X, '0') \supset +\text{salary}(X, '0')$$

$$\text{employee}(X), \text{salary}(X, '0'), \neg\text{onLeave}(X) \supset -\text{salary}(X, '0')$$

$$\text{salary}(X, Y), \text{salary}(X, Z), X \neq Z \supset -\text{salary}(X, Y)$$

Possible solution

$$\mathcal{U}_3 = \{+\text{insured}('\text{john}', '\text{basic}')\} \cup \{+\text{salary}('\text{john}', '0'), -\text{salary}('\text{john}', '500')\}$$

Precedence

Definition

AIC r_1 *affects* AIC r_2, $r_1 \prec r_2$, if some action in the head of r_1 affects a literal in the body of r_2.

Precedence

Definition

AIC r_1 *affects* AIC r_2, $r_1 \prec r_2$, if some action in the head of r_1 affects a literal in the body of r_2.

- Reflexive relation.
- $\langle {}^{\eta}/_{\approx}, \preceq \rangle$ is a partial order, where $\preceq$ is the transitive closure of $\prec$ and $\approx$ is the induced equivalence relation.

Precedence

Definition

AIC r_1 *affects* AIC r_2, $r_1 \prec r_2$, if some action in the head of r_1 affects a literal in the body of r_2.

- Reflexive relation.
- $\langle {}^{\eta}/_{\approx}, \preceq \rangle$ is a partial order, where $\preceq$ is the transitive closure of $\prec$ and $\approx$ is the induced equivalence relation.

(Similar to stratified negation in logic programming. . .)

The company database, modified

Active integrity constraints

$$\text{employee}(X), \neg\text{insured}(X, '\text{basic}') \supset +\text{insured}(X, '\text{basic}')$$

$$\text{employee}(X), \text{onLeave}(X), \neg\text{salary}(X, '0') \supset -\text{onLeave}(X)$$

$$\text{employee}(X), \text{salary}(X, '0'), \neg\text{onLeave}(X) \supset +\text{onLeave}(X)$$

$$\text{salary}(X, Y), \text{salary}(X, Z), X \neq Z \supset -\text{salary}(X, Y)$$

Precedence vs. stratification (I)

Theorem

Let $\eta_1, \eta_2 \in {}^{\eta}/_{\approx}$ with $\eta_1 \prec \eta_2$; $\mathcal{I}$ be a database; and $\mathcal{U}$ be a weak repair for $\mathcal{I}$ and $\eta_1 \cup \eta_2$.
Assume that every action in $\mathcal{U}$ occurs in the head of a rule in $\eta_1 \cup \eta_2$.
Define $\mathcal{U}_i$ as the set of actions in $\mathcal{U}$ in the head of a rule in η_i, for $i = 1, 2$.
Then:

- *$\mathcal{U}_1$ is a weak repair for $\mathcal{I}$ and η_1 and $\mathcal{U}_2$ is a weak repair for $\mathcal{I} \circ \mathcal{U}_1$ and η_2;*
- *if $\mathcal{U}$ is founded/justified, then so is each $\mathcal{U}_i$.*

Precedence vs. stratification (I)

Theorem

Let $\eta_1, \eta_2 \in {}^{\eta}/_{\approx}$ with $\eta_1 \prec \eta_2$; $\mathcal{I}$ be a database; and $\mathcal{U}$ be a weak repair for $\mathcal{I}$ and $\eta_1 \cup \eta_2$.
Assume that every action in $\mathcal{U}$ occurs in the head of a rule in $\eta_1 \cup \eta_2$.
Define $\mathcal{U}_i$ as the set of actions in $\mathcal{U}$ in the head of a rule in η_i, for $i = 1, 2$.
Then:

- *$\mathcal{U}_1$ is a weak repair for $\mathcal{I}$ and η_1 and $\mathcal{U}_2$ is a weak repair for $\mathcal{I} \circ \mathcal{U}_1$ and η_2;*
- *if $\mathcal{U}$ is founded/justified, then so is each $\mathcal{U}_i$.*

This allows us to sequentialize the search for repairs.

Precedence vs. stratification (II)

Theorem

Let η_1, η_2 and $\mathcal{I}$ be as before; $\mathcal{U}_1$ be a weak repair for $\mathcal{I}$ and η_1; $\mathcal{U}_2$ be a weak repair for $\mathcal{I} \circ \mathcal{U}_1$ and η_2; such that every action in $\mathcal{U}_i$ occurs in the head of a rule in η_i, and define $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.
Then:

- *$\mathcal{U}$ is a weak repair for $\mathcal{I}$ and η;*
- *if each $\mathcal{U}_i$ is a repair, then so is $\mathcal{U}$;*
- *if each $\mathcal{U}_i$ is founded/justified, then so is $\mathcal{U}$.*

Outline

What we achieved. . .

What we achieved. . .

- Split a large problem in several smaller ones
- Possibility of parallelization
- Stratification relation

. . . and what we still hope to do

. . . and what we still hope to do

- (More) practical evaluation
- Prototype implementation

Thank you.