Foundations of Reasoning over Artifacts

Relational Artifacts

Giuseppe De Giacomo

Joint work with Babak Bagheri Hariri[2], Diego Calvanese[2], Riccardo De Masellis[1], Paolo Felli[1]

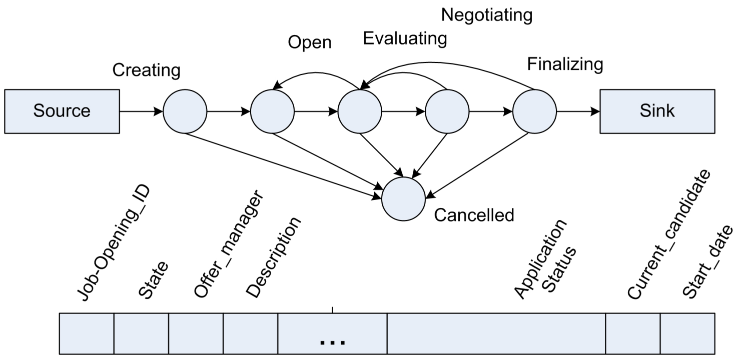
[1] **Sapienza Università di Roma (U. Roma)**
[2] **Free University of Bozen-Bolzano (U. Bolzano)**

IBM Watson, March 7-8, 2011

Artifacts

Artifacts are sort of middle ground between a conceptual formalization of a dynamic system and an actual implementation of the system itself.
Artifacts systems are characterized by:

- Information model: takes into account the structural properties.
- Processes: takes into account the dynamic properties.



The problem: reasoning on artifacts as dynamic entities

We need to decide whether dynamic/temporal properties of interest hold over the life of such systems.

- Verification of temporal formulas
- Dominance/simulation/bisimulation/containment properties
- Automated composition of artifacts-based systems
- Automated process synthesis from dynamic/temporal specification

The problem: reasoning on artifacts as dynamic entities

We need to decide whether dynamic/temporal properties of interest hold over the life of such systems.

- Verification of temporal formulas
- Dominance/simulation/bisimulation/containment properties
- Automated composition of artifacts-based systems
- Automated process synthesis from dynamic/temporal specification

Currently (2010's) the scientific community is quite good at each of these, but only in a finite setting!

The problem: reasoning on artifacts as dynamic entities

- With artifacts the number of different states of the system is affected by the information model.
- Presence of data makes the systems potentially infinite-state.
- The usual techniques, e.g., model checking, used for finite-state systems don't work off-the-shelf.

The problem: reasoning on artifacts as dynamic entities

- With artifacts the number of different states of the system is affected by the information model.
- Presence of data makes the systems potentially infinite-state.
- The usual techniques, e.g., model checking, used for finite-state systems don't work off-the-shelf.

Our research aims at exploring suitable representation formalisms for modeling artifacts that are expressive enough for some real life scenarios, and in the same time admit decidability of reasoning.

A solution for reasoning on artifacts as dynamic entities

Contribution from:

- work on data integration and data exchange that advocate a semantic view of the data ← Databases;
- work data access and update through ontologies and description logics ← KR and Databases;
- work in reasoning about actions formalize dynamic systems using logics ← KR and AI;
- nice results for verification/dominance/composition/synthesis/ available for finite-state systems ← Formal Methods.

A solution for reasoning on artifacts as dynamic entities

Contribution from:

- work on data integration and data exchange that advocate a semantic view of the data ← Databases;
- work data access and update through ontologies and description logics ← KR and Databases;
- work in reasoning about actions formalize dynamic systems using logics ← KR and AI;
- nice results for verification/dominance/composition/synthesis/ available for finite-state systems ← Formal Methods.

Key idea

Work by Fagin & Kolaitis (IBM Almaden) and others on the use of data dependency theory for data exchange (Databases) can be seen as talking about actions effects (KR and AI)

Finite chase ↔ Finite state system

Reduction to reasoning to finite state systems!!!

Relational artifacts

The variant of artifacts (aka BEL) that we consider are characterized by having an information model that is a full-fledged relational database.

Relational artifacts

The variant of artifacts (aka BEL) that we consider are characterized by having an information model that is a full-fledged relational database.

A relational artifact $\mathcal{S} = (T, A_0, R)$ is a stateful device, where:

- T is a fixed DB schema.
- A_0 is a DB conforming to the schema T, which expresses extensional information, and constitutes the initial state of the artifact.
- R is a set of actions, which change the state of the artifact, i.e., the extensional information component.

Actions

An action ρ is constituted by:

- an action signature, i.e., a name, and a list of individual input parameters.
  Parameters are substituted by individual/constants for the execution of the action.
- an effect specification $\{e_1, \ldots, e_n\}$, where each e_i is an effect.
  Effects are assumed to take place simultaneously.

Actions

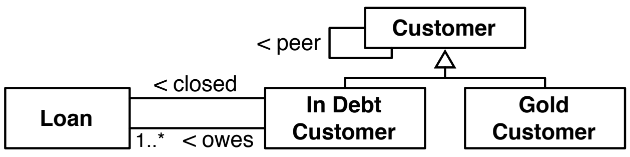
An action ρ is constituted by:

- an action signature, i.e., a name, and a list of individual input parameters.
  Parameters are substituted by individual/constants for the execution of the action.
- an effect specification $\{e_1, \ldots, e_n\}$, where each e_i is an effect.
  Effects are assumed to take place simultaneously.

An effect e_i has the form $q_i^+ \wedge Q_i^- \rightsquigarrow A'_i$ where:

- $q_i^+ \wedge Q_i^-$ is a query over T and constants of A_0:
  - may include some of the input parameters as terms the query
  - q_i^+ is a UCQ over T
  - Q_i^- is an FOL query that acts as a filter
  - free variables of Q_i^- are included in those of q_i^+.
- A'_i is a set of facts over T, which include as terms: constants in $\mathcal{C}_{A_0}$, parameters, free variables of q_i^+, and Skolem functions that form a sort of labeled nulls.

Relational artifact – Example



Initial state: $A_0 = \{\text{Gold}(\text{john}), \text{Cust}(\text{ann}), \text{peer}(\text{mark}, \text{john})\}$.
Actions R:

$$\text{GetLoan}(c) : \{\ [\text{peer}(c,p) \wedge \text{Gold}(p)] \leadsto \{\text{owes}(c, newl(c,p))\},\ CopyAll\ \}$$

$$\text{CloseAllLoans}(c) : \{\ [\text{owes}(c,l)] \leadsto \{\text{closed}(c,l\},\ CopyAll\ \}$$

$$\text{UpdateDebts} : \{\ [\text{owes}(x,l)] \wedge \neg[\text{closed}(x,l)] \leadsto \{\text{owes}(x,l)\}\ CopyAllExceptOwes\ \}$$

Action execution

Consider a state A of $\mathcal{S}$, an action ρ with a substitution σ for it parameters, and an effect e_i: $q_i^+ \wedge Q_i^- \leadsto A_i'$ of ρ:

1. e_i extracts from A the set $ans_{\mathcal{C}_A}((q_i^+ \wedge Q_i^-)\sigma, T, A)$ of tuples of terms (constants and labeled nulls).
2. For each such tuple θ it asserts a set $A_i'\sigma\theta$ of facts obtained from $A_i'\sigma$ by applying the substitution θ for the free variables of q_i^+.

Action execution

Consider a state A of $\mathcal{S}$, an action ρ with a substitution σ for it parameters, and an effect e_i: $q_i^+ \wedge Q_i^- \rightsquigarrow A'_i$ of ρ:

1. e_i extracts from A the set $ans_{\mathcal{C}_A}((q_i^+ \wedge Q_i^-)\sigma, T, A)$ of tuples of terms (constants and labeled nulls).
2. For each such tuple θ it asserts a set $A'_i\sigma\theta$ of facts obtained from $A'_i\sigma$ by applying the substitution θ for the free variables of q_i^+.

Overall set of facts obtained: $e_i\sigma(A) = \bigcup_{\theta \in ans_{\mathcal{C}_A}(Q_i\sigma, T, A)} A'_i\sigma\theta$

Overall effect of the action ρ with parameter substitution σ over A: is a new state $do(\rho\sigma, T, A) = \bigcup_{1 \leq i \leq n} e_i\sigma(A)$.

Action execution – observations

- The effects of an action are a form of update of the previous state, and not of belief revision. That is, we never learn new facts on the state in which an action is executed, but only on the state resulting from the action execution.

- Skolem terms, i.e., labeled nulls, introduced by actions effects have an existential flavor. They are used as witnesses of values chosen by the external user/environment when executing the action. We assume that such a choice depends only on the argument of the Skolem function, which contain information retrieved by the query in the premises of effects.

Processes

Relational artifacts do not say anything about how or when to apply a certain action.

Processes over a relational artifact $\mathcal{S} = (T, A_0, R)$

- Are possibly nondeterministic programs.
- Use the state of $\mathcal{S}$ to store their computation results.
- Use the actions in R as atomic instructions.
- The state A can be arbitrarily queried through the query answering services over T, while it can be updated only through actions in R.

We adopt a rule-based specification for processes.

Condition/action rules for processes

A process is a finite set $\Pi = \{\pi_1, \ldots, \pi_n\}$ of condition/action rules.

Each condition/action rule π for $\mathcal{S}$ has the form:

$$Q \mapsto \rho$$

where:

- ρ is an action in R;
- Q is a FOL query (we assume range restriction) over T and $\mathcal{C}_{A_0}$, whose free variables are exactly the parameters of ρ.

The rule π expresses that, for each tuple θ for which condition Q holds, the action ρ with actual parameters θ can be executed.

Condition/action rules – examples

A customer can get a loan if she does not have one already:

$$[\mathsf{Cust}(x)] \land \neg[\exists y.\mathsf{owes}(x,y)] \mapsto \mathsf{GetLoan}(x)$$

A customer that owes non closed loans, can close them all at once:

$$\exists y.[\mathsf{owes}(x,y)] \land \neg[\mathsf{closed}(x,y)] \mapsto \mathsf{CloseAllLoans}(x)$$

It is always possible to perform UpdateDebts:

$$\mathsf{true} \mapsto \mathsf{UpdateDebts}$$

Process execution

The execution of Π over $\mathcal{S} = (T, A_0, R)$ is defined as follows:

1. We start from the initial state A_0, and continue constructing states by applying rules.
2. For a state A, for each rule $Q \mapsto \rho$ in Π, we evaluate Q, and for each tuple θ returned, we execute $\rho\theta$.
3. If $A' = do(\rho\theta, T, A)$ is consistent wrt T, it becomes a new state.

Process execution

The execution of Π over $\mathcal{S} = (T, A_0, R)$ is defined as follows:

1. We start from the initial state A_0, and continue constructing states by applying rules.
2. For a state A, for each rule $Q \mapsto \rho$ in Π, we evaluate Q, and for each tuple θ returned, we execute $\rho\theta$.
3. If $A' = do(\rho\theta, T, A)$ is consistent wrt T, it becomes a new state.

In this way we obtain a transition system $\Upsilon(\Pi, \mathcal{S})$:

- states represent possible artifact states,
- each transition represents the execution of an instantiated action that is allowed according to Π.

Note: $\Upsilon(\Pi, \mathcal{S})$ is in general infinite state.

Verification formalism

We adopt a variant of μ-calculus.

- Very expressive temporal logic, that subsumes LTL, CTL, CTL*, ...
- We adapt it to our setting, by using as basic predicates ECQ queries over T that is FOL queries obtained from CQs that are restricted to return only constants from A_0.

Verification formalism

We adopt a variant of μ-calculus.

- Very expressive temporal logic, that subsumes LTL, CTL, CTL*, ...
- We adapt it to our setting, by using as basic predicates ECQ queries over T that is FOL queries obtained from CQs that are restricted to return only constants from A_0.

Given $\mathcal{S} = (T, A_0, R)$, formulas of $\mu\mathcal{L}$ over $\mathcal{S}$ have the form:

$$\Phi ::= Q \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \exists x.\Phi \mid \Box\Phi \mid \Diamond\Phi \mid \mu Z.\Phi \mid \nu Z.\Phi \mid Z$$

where Q is an ECQ over T and $\mathcal{C}_{A_0}$, but not labeled nulls.
Z is a predicate variable symbol.

Note: individual variable quantification ranges only over the constants of A_0 only.

Semantic of $\mu\mathcal{L}$ formulae

We assign meaning to μ-calculus formulas by associating to $\Upsilon(\Pi, \mathcal{S})$ and $\mathcal{V}$ an *extension function* $(\cdot)^{\mathfrak{A}}_{\mathcal{V}}$, which maps μ-calculus formulas to subsets of $\Sigma_{\mathcal{T}}$. The extension function $(\cdot)^{\mathfrak{A}}_{\mathcal{V}}$ is defined inductively as follows:

$$\begin{array}{ll}
(Q)^{\mathfrak{A}}_{\mathcal{V}} & = \{A \in \Sigma_{\mathcal{T}} \mid ans_{\mathcal{C}_{A_0}}(Q\mathcal{V}, T, A)\} \\
(Z)^{\mathfrak{A}}_{\mathcal{V}} & = Z\mathcal{V} \subseteq \Sigma_{\mathcal{T}} \\
(\neg\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \Sigma_{\mathcal{T}} - (\Phi)^{\mathfrak{A}}_{\mathcal{V}} \\
(\Phi_1 \wedge \Phi_2)^{\mathfrak{A}}_{\mathcal{V}} & = (\Phi_1)^{\mathfrak{A}}_{\mathcal{V}} \cap (\Phi_2)^{\mathfrak{A}}_{\mathcal{V}} \\
(\exists x.\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \bigcup\{(\Phi)^{\mathfrak{A}}_{\mathcal{V}[x/c]} \mid c \in \mathcal{C}_{A_0}\} \\
(\Diamond\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \{A \in \Sigma_{\mathcal{T}} \mid \exists A'.\ A \Rightarrow_{\mathfrak{A}} A' \text{ and } A' \in (\Phi)^{\mathfrak{A}}_{\mathcal{V}}\} \\
(\Box\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \{A \in \Sigma_{\mathcal{T}} \mid \forall A'.\ A \Rightarrow_{\mathfrak{A}} A' \text{ implies } A' \in (\Phi)^{\mathfrak{A}}_{\mathcal{V}}\} \\
(\mu Z.\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \bigcap\{\mathcal{E} \subseteq \Sigma_{\mathcal{T}} \mid (\Phi)^{\mathfrak{A}}_{\mathcal{V}[Z/\mathcal{E}]} \subseteq \mathcal{E}\} \\
(\nu Z.\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \bigcup\{\mathcal{E} \subseteq \Sigma_{\mathcal{T}} \mid \mathcal{E} \subseteq (\Phi)^{\mathfrak{A}}_{\mathcal{V}[Z/\mathcal{E}]}\}
\end{array}$$

Bisimulation: model theoretic characterization of $\mu\mathcal{L}$

Two artifact transition systems $\mathcal{T}_1 = \Upsilon(\Pi_1, \mathcal{S}_1)$ and $\mathcal{T}_2 = \Upsilon(\Pi_2, \mathcal{S}_2)$ sharing the same initial constants $\mathcal{C} = \mathcal{C}_{A_{10}} = \mathcal{C}_{A_{20}}$, are bisimilar if there exists a bisimulation: i.e., a relation $\mathcal{B} \subseteq \Sigma_{\mathcal{T}_1} \times \Sigma_{\mathcal{T}_2}$ such that:

$(A_1, A_2) \in \mathcal{B}$ implies that:

1. A_1 and A_2 are $\mathcal{C}$-homomorphically equivalent wrt T;
2. if $A_1 \Rightarrow_{\mathcal{T}_1} A_1'$ then exists A_2' such that $A_2 \Rightarrow_{\mathcal{T}_2} A_2'$ and $(A_1', A_2') \in \mathcal{B}$;
3. if $A_2 \Rightarrow_{\mathcal{T}_2} A_2'$ then exists A_1' such that $A_1 \Rightarrow_{\mathcal{T}_1} A_1'$ and $(A_1', A_2') \in \mathcal{B}$.

Theorem

Two states A_1 and A_2 (including the initial ones) are bisimilar iff $A_1 \in (\Phi)^{\mathcal{T}_1}$ iff $A_2 \in (\Phi)^{\mathcal{T}_2}$, for all $\mu\mathcal{L}$ formulas Φ.

Model checking Π and $\mathcal{S}$

Given a relational artifact $\mathcal{S}$ and a process Π, formulas of $\mu\mathcal{L}$ are evaluated over $\mathcal{T} = \Upsilon(\Pi, \mathcal{S})$.

Model checking problem: Given a relational artifact $\mathcal{S} = (T, A_0, R)$, a process Π over $\mathcal{S}$, and a $\mu\mathcal{L}$ formula Φ over $\mathcal{S}$, check whether $A_0 \in \Phi^{\mathfrak{A}}$.

Example

Simple safety property: It is always true that gold customers in A_0 remain so.

$$\forall x.([\mathsf{Gold}(x)] \supset \nu Z.([\mathsf{Gold}(x)] \wedge \Box Z)).$$

Is true, since no action removes individuals from being Gold.

Example

Simple safety property: It is always true that gold customers in A_0 remain so.

$$\forall x.([\mathsf{Gold}(x)] \supset \nu Z.([\mathsf{Gold}(x)] \wedge \Box Z)).$$

Is true, since no action removes individuals from being Gold.

Simple liveness property: It is possible to reach a state in which a gold customer is also an in-debt customer.

$$\mu Z.([\exists x.loan\mathsf{Gold}(x) \wedge \mathsf{owns(x, y)}] \vee \Diamond Z)$$

This formula is true, because we can reach a state where it holds $\exists x, y.\mathsf{Gold}(x) \wedge \mathsf{owes}(x, y)$ by firing the action GetLoan(john), which is allowed by the process.

Undecidability of verification

Theorem

Model checking $\mu\mathcal{L}$ formulas on processes over relational artifacts is undecidable.

Undecidability of verification

Theorem

Model checking $\mu\mathcal{L}$ formulas on processes over relational artifacts is undecidable.

Proof: By reduction from query answering boolean UCQs in relational DBs under a set of TGDs [Beeri and Vardi, 1981].

Key point: TGDs are very close to action effect whose queries are CQs.

Undecidability of verification

Theorem

Model checking $\mu\mathcal{L}$ formulas on processes over relational artifacts is undecidable.

Proof: By reduction from query answering boolean UCQs in relational DBs under a set of TGDs [Beeri and Vardi, 1981].

Key point: TGDs are very close to action effect whose queries are CQs.

Can we exploit the results on TGDs for getting decidability?

Undecidability of verification

Theorem

Model checking $\mu\mathcal{L}$ formulas on processes over relational artifacts is undecidable.

Proof: By reduction from query answering boolean UCQs in relational DBs under a set of TGDs [Beeri and Vardi, 1981].

Key point: TGDs are very close to action effect whose queries are CQs.

Can we exploit the results on TGDs for getting decidability? YES

Bounding the states

To bound the states of $\mathcal{S} = (T, A_0, R)$, we define its inflationary approximate $\mathcal{S}^+ = (T, A_0, R^+)$:

- Each action $\rho^+ \in R^+$ is obtained from an action $\rho \in R$ by:
  - removing all input parameters from the signature
  - substituting each effect $e_i : q_i^+ \wedge Q_i^- \rightsquigarrow A_i'$ with $e_i : q_i^+ \rightsquigarrow A_i'$
  - adding effects to copy all relations.

We consider the generic process $\Pi_\top$, in which all condition/action rules have the trivially true condition.

Bounding the states

To bound the states of $\mathcal{S} = (T, A_0, R)$, we define its inflationary approximate $\mathcal{S}^+ = (T, A_0, R^+)$:

- Each action $\rho^+ \in R^+$ is obtained from an action $\rho \in R$ by:
  - removing all input parameters from the signature
  - substituting each effect $e_i : q_i^+ \wedge Q_i^- \rightsquigarrow A_i'$ with $e_i : q_i^+ \rightsquigarrow A_i'$
  - adding effects to copy all relations.

We consider the generic process $\Pi_\top$, in which all condition/action rules have the trivially true condition.

For the transition system $\Upsilon(\Pi_\top, \mathcal{S}^+)$, we have that $do(\rho^+, T, \cdot)$ is a monotonic operator.
Hence, starting from A_0, we get at the limit (possibly transfinite) a least fixpoint A_{max}.

Lemma

Every state A of the transition system $\Upsilon(\Pi, \mathcal{S})$ is a subset of A_{max}.

Getting finite states

To obtain a finite number of states in A_{max}, we impose a restriction on the form of the effect specifications:

[Fagin et al., 2005] Let $\mathcal{S}$ be a relational artifact and $\mathcal{S}^+ = (T, A_0, \vec{\rho}^+)$ its inflationary approximate. We call dependency graph of $\mathcal{S}^+$ as the directed graph (labeled on edges) defined as follows:

1. *(nodes) for every relation symbol R and every attribute $att \in R$ there is a node (R, att) representing a position and*
2. *(edges) for every effect $e = \phi(\vec{t}) \rightsquigarrow \psi(\vec{t}', f_1(\vec{t}_1), \ldots, f_n(\vec{t}_n))$ (where ϕ, ψ are conjunction of atoms over the artifact system, and $\vec{t}'$, $\vec{t}_1, \ldots, \vec{t}_n \subseteq \vec{t}$ are either constants or variables) and for every variable $x \in \vec{t}$ and for every occurrence of x in ϕ in position p include edges as follows:*
    - *for every occurrence of x in ψ in position p' include $p \rightarrow p'$;*
    - *for every skolem term $f_i(t_i)$ such that $x \in \vec{t}_i$ occurring in ψ in position p'', include a special edge (i.e., labeled by $*$) $p \xrightarrow{*} p''$;*

We say that $\mathcal{S}$ is weakly acyclic if the dependency graph for $\mathcal{S}^+$ has no cycle going through a special edge.

Getting finite states

Lemma

Let $\mathcal{S}$ be a weakly acyclic relational artifact. Then the size of every instance generated by executing the generic process Π_0 over $\mathcal{S}^+$ is polynomial in the size of A_0 (including A_{max}!).

Proof: Follows closely the line of the corresponding theorem for TGDs [Fagin et al., 2005].

Getting finite states

Lemma

Let $\mathcal{S}$ be a weakly acyclic relational artifact. Then the size of every instance generated by executing the generic process Π_0 over $\mathcal{S}^+$ is polynomial in the size of A_0 (including A_{max}!).

Proof: Follows closely the line of the corresponding theorem for TGDs [Fagin et al., 2005].

Lemma

Let $\mathcal{S}$ be a weakly acyclic relational artifact and Π any process over $\mathcal{S}$. Then the number of states that are generated by executing Π over $\mathcal{S}$ is finite and at most exponential in the size of the initial state A_0.

Getting finite states

Lemma

Let $\mathcal{S}$ be a weakly acyclic relational artifact. Then the size of every instance generated by executing the generic process Π_0 over $\mathcal{S}^+$ is polynomial in the size of A_0 (including A_{max}!).

Proof: Follows closely the line of the corresponding theorem for TGDs [Fagin et al., 2005].

Lemma

Let $\mathcal{S}$ be a weakly acyclic relational artifact and Π any process over $\mathcal{S}$. Then the number of states that are generated by executing Π over $\mathcal{S}$ is finite and at most exponential in the size of the initial state A_0.

Theorem (main result)

For weakly-acyclic relational artifacts $\mathcal{S}$, model checking of any closed $\mu\mathcal{L}$ formula on any process over $\mathcal{S}$ is decidable.

Next steps: framework extensions

- Semantic artifacts: $\mathcal{S} = (T, A_0, R)$ where T is a *DL-Lite* / OWL 2 QL TBox, A_0 is an ABox, R as before *(done!)*.
- Multiple artifacts (fixed over time): actions now insist over several artifacts simultaneously; lifecycles become temporal constraints over the use of the artifact information model. *(done!)*.
- Multiple artifacts that can be created and destroyed over time: weak acyclicity on artifact creation is needed; important to talk about nulls/skolem terms representing artifacts identifiers the verification formalism must quantify and treat nulls!
- ACSI Abstract Model: A3M sophisticated models in which artifacts are related to each other and exchange events/messages with the outside world.

Next steps: technical extensions and implementation

- Technical extensions
  - Other temporal logic formalisms: *(straightforward!)*.
  - Relaxing partially weak acyclicity: relationships with theory of logic programming and work on automated verification of (recursive) logic programs – bound on nesting of the skolem functions.
  - Adding constraints in the DB schema:
    - ⋆ Inclusion dependency/foreign key/denial, related to OWL 2 QL *(partially done!)*
    - ⋆ Functional/cardinality restrictions/identification constraints, related to *DL-Lite*$_{\mathcal{A}}$.
- Towards implementation
  - Abstraction, abstraction, abstraction! - *(In ACSI with Alessio Lomuscio of Imperial College London.)*
  - Consider properties of the evolution of some individuals merging the others: homomorphism!.

Thanks!

Questions, Comments, Suggestions ?