Semantic Modelling of Dependency Relations between Life Cycle Analysis Processes

Benjamin Bertin, Marian Scuturici, Jean-Marie Pinon, and Emmanuel Risler

Université de Lyon, CNRS,
INSA-Lyon, LIRIS, UMR5205
20 Avenue Albert Einstein, F-69621 Villeurbanne Cedex
{benjamin.bertin,marian.scuturici,jean-marie.pinon}@liris.cnrs.fr
Université de Lyon, CNRS
INSA-Lyon, ICJ, UMR5208
21 avenue Capelle, F-69621 Villeurbanne Cedex
emmanuel.risler@insa-lyon.fr

Abstract. Life Cycle Assessment provides a well-accepted methodology for modelling environmental impacts of human activities. This methodology relies on the decomposition of a studied system into interdependent processes. Several organisations provide processes databases containing several thousands of processes with their interdependency links. The usual work-flow to manage those databases is based on the manipulation of individual processes which turns out to be a very harnessing work. We propose a new work-flow for LCA inventory databases maintenance based on the addition of semantic information to the processes they contained. This method eases considerably the modelling process and also offers a more understandable model of the dependencies links. In this paper, we explain our approach and some key parts of the implementation. We also present a case study based on the U.S. electricity production and an experiment on the scalability of our implementation.

Keywords: Environmental information management, Life Cycle Assessment, Ontology.

1 Introduction

In order to reduce the environmental impact of human activities it is necessary to model and evaluate the environmental effects of those activities. This is the objective of the Life Cycle Assessment (LCA) method[3], which aims at determining the environmental impacts of a product, a service or, generally speaking, any human activity. This method can take into account all the life cycle stages of a product such as manufacture, use and recycling. LCA can assess various environmental impacts like greenhouse gases emissions or chemical products dissemination.

An LCA study is composed of four phases[3]. The first one consists of the definition of the goal and scope of the studied system. In the second one, the studied

A. Auweter et al. (Eds.): ICT-GLOW 2012, LNCS 7453, pp. 109–124, 2012.

system is factorized into interrelated elementary processes associated to environmental impacts in order to achieve a *Life Cycle Inventory* (LCI). Elementary processes are related to specific stages of a life cycle or to any human activities (*e.g.*, energy production, fertilizer spreading, air plane trips, etc.). Those processes can depend on other processes, for instance: car production depends on steel production. In the last two phases we perform the analysis and the interpretation of the environmental impacts.

The accepted methodology for the LCI is to use an Input/Output (I/O) matrix[12] to model inter-process interactions or interactions between processes and the environment. In order to calculate the impacts of processes, we need to solve the linear equation system corresponding to the I/O matrix.

Several agencies and companies provide Life Cycle Inventories databases[7][1] [2] that are used by LCA practitioners to do an LCA study. But those databases can contain thousands of processes linked together. The model is therefore difficult to understand unless the practitioners do an in-depth analysis. Yet, semantic similarities are noticeable in LCI databases. The dependency links between processes in LCI databases contain many network constructions that are both topologically and semantically close. For instance, in order to produce electricity from coal it is necessary to transport it. In an LCI database, we can find several processes corresponding to electricity production from coal for every type of coal that can be used to produce electricity (lignite coal, bituminous coal or sub-bituminous coal). Those processes are all linked to several merchandise transportation processes depending on the transportation systems (the coal can be transported using a truck, a barge, etc.).

But, if this information is scattered into the I/O matrix or if we look at all the processes independently, it is hard to understand that in order to produce electricity from coal it is always necessary to transport it. Meanwhile, maintaining all those relationships can be tedious if we need to update several semantically close relations. For instance, if we want to add a new mode of transport to transport coal, we would have to add a dependency link between this process and all the electricity production from coal processes.

Thus, we propose a new methodology to model an LCI database based on the semantics of the processes. In our approach, we semantically index the processes and we use this semantic indexing to semantically regroup the processes. Then, we use those groups to model the dependency links between multiple processes. With this methodology we address the two problems we identified: we offer a bird's eye view of the database and we ease the database management using the regrouped processes. Our model is based on the coexistence of two digraphs. The first one, called the *macro-graph*, contains dependencies between regrouped processes (or *macro-processes*). The second one, called the *detailed-graph*, is a transposition of the I/O matrix. In our approach, the modelling process consists in creating dependency links between macro-processes instead of creating dependency relations between individual processes. Due to the potential presence of cycles in the digraph, we cannot calculate the impacts of processes contained in the macro-graph. Hence, those relations are translated into inter-process

relations in the detailed-graph (in order to calculate the impacts of processes as it is usually done with the I/O matrix).

The rest of this paper is structured as follows. The second section contains a brief description of the usual LCI model. In the third section, we present a case study based on a data set extracted from the National Renewable Energy Laboratory LCI Database[2]. In this paper, we restricted our study to electricity production processes only. In the fourth section we explain our model. The fifth section contains the translation method of the *macro-relations* contained in the macro-graph into relations between processes in the detailed-graph. The last section contains the results of this method applied to the NREL data set and experimental results that prove the scalability of the translation algorithm. Some parts of this work have already been presented in [4]. In this paper we focus on the methodology and we propose a new formalization and a new implementation for the translation method.

2 Life Cycle Inventory Model

The goal of Life cycle assessment methodology is to evaluate the environmental impacts of, for instance, a product or a company's activity. This can be achieved by creating an inventory of elementary flows from and to the environment, for every step of a product's production process or for every activity of a company[9][10]. In the LCA terminology, these steps or activities are called *processes*.

A process is associated with environmental impacts (*e.g.*, the greenhouse gases emitted or the resulting water pollution) and with other processes. For instance, it is necessary to extract and transport coal in order to run a coal power plant. The total amount of environmental impacts of a process is the sum of its own impacts (*i.e.*, that are not from its predecessors) and the impacts of its predecessors multiplied by a scalar dependency coefficient. The environmental impacts of a process can be expressed as a linear combination of other processes environmental impacts. Let p be a process and $I(p)$ its cumulated environmental impacts. We denote by $p_0, \ldots, p_n$ the preceding processes of p (often called *upstream processes* of p), and by $c_0, \ldots, c_n$ the dependencies coefficients between upstream processes. We denote by $I(p_0), \ldots, I(p_n)$ the cumulated environmental impacts of p's upstream processes and by $I_{\text{direct}}(p)$ its own impacts. We have:

$$I(p) = I_{\text{direct}}(p) + \sum_{i=0}^{n} (I(p_i) * c_i)$$

The usual model is based on an Input/Output matrix A. For n processes, this is a $n \times n$ matrix, where a_{ij} is the dependency coefficient between process i and process j. This matrix depicts flows to and from the technosphere[1]. These flows are named elementary flows. The m different impacts produced by n processes

[1] The McGraw-Hill Dictionary of Environmental Science defines this term as: The part of the physical environment affected through building or modification by humans.

are modelled in a $n \times m$ matrix B, where b_{ij} is the j^{th} impact of the i^{th} process. This matrix depicts flows from the technosphere to the ecosphere[2].

The I/O matrix can also be represented using a digraph. Let $G(V, E)$ be the digraph representing dependency links between processes. The vertices set V is the set of processes. We also denote this set of processes by P. The edges set E contain the relations between processes and the set of weights associated to the edges is the set of coefficients. We also denote this set of coefficients by C. Let p and $p_0, \ldots, p_n$ be vertices then an edge between p_i and p means that process p depends on p_i.

For instance, in order to produce electricity from coal we need to extract the coal. We can model the dependency relations between the electricity production from coal process and the extraction process by creating an edge between the vertices corresponding to those processes (see Figure 1)[3].

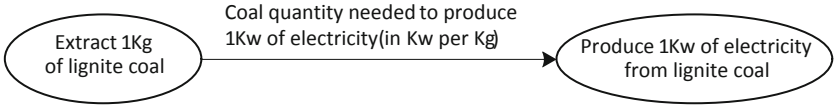

Fig. 1. Detailed-graph for the electricity production from coal process showing its dependency relation with the coal extraction process. This relation is weighted with the quantity of coal required to produce 1Kw of electricity.

Determining the impacts for a specific process requires to recursively calculate the impacts of its predecessors. As explained in [15], the I/O matrix can be considered as a basic system of linear equations. Thus it is possible to calculate the impacts of the processes using iterative methods[4] or any direct method (like the Gaussian elimination)[19].

Interestingly for our proposition, there is one condition for the matrix to be computable: the linear equations system must converge. For instance, if we say that in order to produce one gallon of oil, we need to consume more than one gallon of oil, then an iterative algorithm to solve the linear equations system will

[2] The U.S. Environmental Agency defines this term as: The "bio-bubble" that contains life on earth, in surface waters, and in the air.

[3] Processes and dependency coefficients are associated to a functional unit. The value of a coefficient depends on its unit. In our example, if the impacts of the electricity production from coal process are expressed in produced Kw, and the impacts of the extraction process are expressed in Kg of extracted coal. Then, the unit of the coefficient is expressed in Kw per Kg and can be determined, for instance, considering the average quantity of coal necessary to produce one Kw of electricity. In order to keep our explanation simple, we do not consider the units of the processes and the coefficients in this paper.

[4] Life Cycle Inventory data is subject to uncertainty. It is therefore possible to use an iterative method and stop the algorithm if the delta obtained between two iterations is smaller than the uncertainty. This could significantly lower the computation time compared to a direct method.

not converge. The system converges when the spectral radius of the matrix A is less than 1 [18]. The potential presence of cycles forbids us to directly calculate the impacts of processes in the macro-graph. Therefore, we need to convert the latter into a detailed-graph and apply a linear equations system solving method.

3 Case Study: Electricity Production in the U.S.

The National Renewable Energy Laboratory (NREL) provides an LCI Database containing inventory data for electricity production for every 27 U.S. subregions[2].

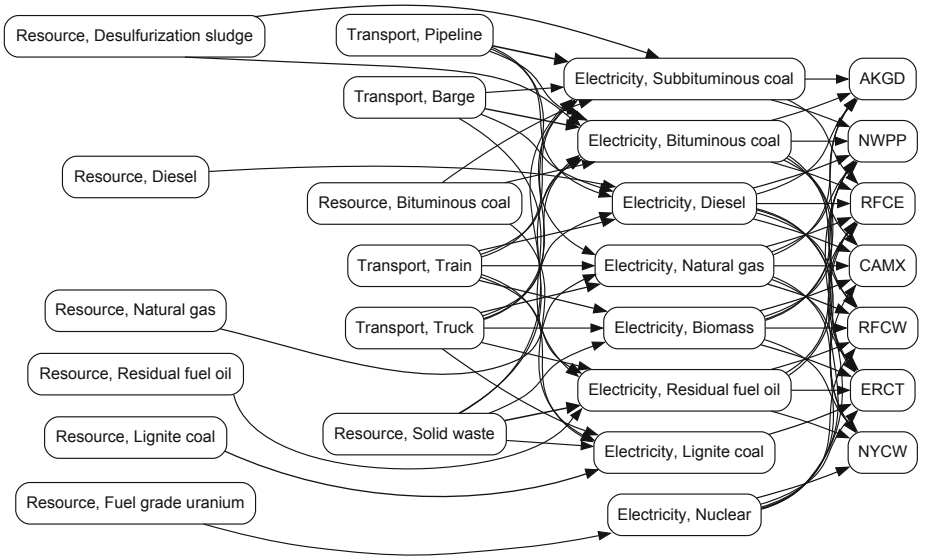

Fig. 2. Sequence from the studied data set extracted from the NREL's LCI Database

These subregions are defined by the U.S. EPA's Emissions and Generation Resource Integrated Database (eGRID)[6]. An eGRID subregion represents a portion of the US power grid that is contained within a single North America Electric Reliability Council (NERC) region, and generally represents sections of the power grid, which have similar emissions and resource mix characteristics, and may be partially isolated by transmission constraints.

The NREL's LCI data can be exported in the *ecospold* format[17] as Excel spreadsheets or XML documents. This is a common data exchange format widely used in the LCA community. Besides including some meta-data for each process (provenance, comments, etc.), it includes dependencies between processes with the dependency coefficients. Some processes in this database are not detailed and flagged as "dummy" processes. So we do not have a complete data set in

terms of dependency relations. But, even after pruning these dummy processes, the data set is still complex enough to illustrate our proposition. Figure 2 shows the detailed-graph corresponding to this data set limited to 7 subregions. This data set restricted to those subregions contains 27 processes and 72 inter-process relations (the whole database contains 593 processes).

As mentioned in the introduction, even on this simple data set, it is hard to understand the dependency links for semantically close processes. Meanwhile, managing this data set to add a new process or a new dependency link can be tedious. For instance, if we want to add a new mode of transport to transport coal, we have to add a dependency link between this process and all the electricity production from coal processes. In this article we use this case study to exemplify the different parts of our proposition.

4 A Methodology Based on Three Graphs

Our approach is based on the existence of two layers of directed graphs. The first graph, called the *detailed-graph*, contains the dependency relations between processes. The second graph, called the *macro-graph*, contains the relations between groups of processes (or *macro-processes*). The macro-graph offers a simplified view of the data contained in the detailed-graph and eases the expression of new dependency relations between semantically close processes. In order to semantically regroup processes, we choose to index them with a set of keywords that are stored in an ontology[8][14] (here is the third graph). The vocabulary of this simple ontology is composed of keywords and predicates to create binary relations between those keywords.

Using this ontology, we can regroup processes and dependency coefficients into semantic groups. A macro-process is similar to a multidimensional matrix, where each dimension is a set of keywords. Those dimensions are described using a query over the ontology. We can create dependency relations between those macro-process using *macro-coefficients*, in the same way that we create dependency relations between processes using individual coefficients.

The methodology based on those three graphs is summarized in Figure 3. With this methodology, if we want to create a new LCI database, we have to follow six steps. During the first one we need to create the keywords ontology. In the second step we create some macro-processes using the ontology. Those macro-processes reference several processes depending on the keywords coordinates of the cells in the macro-processes. Because the database is empty, in the third step, individual processes are automatically created according to the coordinates of the macro-processes cells. Then, in the fourth step, we can create dependency relations between the macro-processes. Finally, the macro-graph is automatically translated into a detailed-graph in the fifth step and, in the sixth step, the coefficients are extracted and the impacts are calculated solving the corresponding linear equation system.

If we add a new macro-process to an already existing LCI database, the processes it references can already exists. Hence, we would enrich the dependency

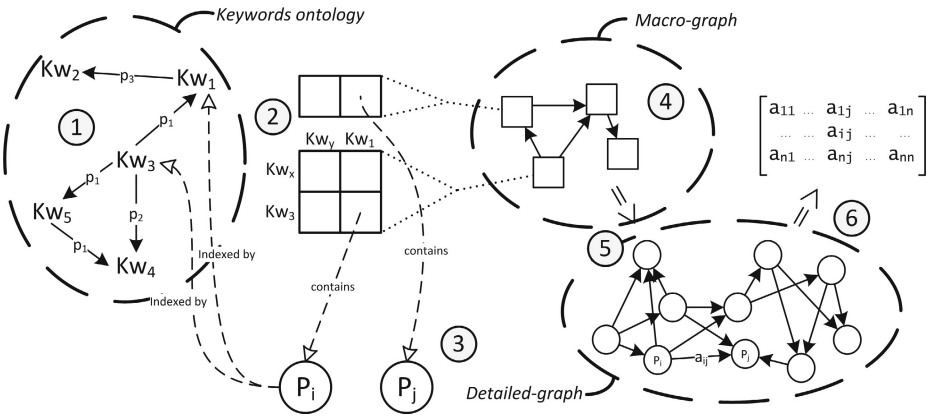

Fig. 3. The proposed methodology. The first part contains the keyword ontology. The second part contains two macro-nodes referencing some processes shown in the third part (those processes are indexed with keywords). The fourth part contains the macro-graph and is translated into the detailed-graph in the fifth part. The sixth part contains the I/O matrix extracted from the detailed-graph.

network of those processes. Furthermore, as the macro-processes are defined using the ontology, we can edit macro-processes by just editing the ontology.

The following subsections explain our keywords ontology with the semantic indexing and the macro-graph.

4.1 The Keywords Ontology

Processes and coefficients are indexed and identified using keywords: there is only one process or coefficient associated to a specific set of keywords, *e.g.*, the process corresponding to using a truck is indexed using the keywords *Transport* and *Truck*. We define the notion of indexed process as follows:

Definition 1. *Let $k_1, \ldots, k_n$ be keywords. An indexed process p^i is a pair composed of impacts and a set of keywords denoted by $p^i = (I(p), K_p)$, where $K_p = \{k_1, \ldots, k_n\}$.*

Similarly, we define the notion of indexed coefficient as follows:

Definition 2. *Let $k_1, \ldots, k_n$ be keywords. An indexed coefficient c^i is a pair composed of a scalar value and a set of keywords denoted by $c^i = (V(c), K_c)$, where $K_c = \{k_1, \ldots, k_n\}$.*

A macro-process or a macro-coefficient form a multidimensional matrix where dimensions are distinct sets of keywords. Therefore, there is only one process or coefficient associated to each coordinate[5].

[5] The coordinates correspond to the Cartesian product of the dimensions.

The keywords are organized in an ontology which is used to dynamically define groups. A set of keywords, called *dimension*, is the result of a query over the ontology. Considering the ontology in Figure 4, we can build a dimension containing all transportation systems with a query retrieving all the keywords linked to the keyword *Transportation system*, considering only the predicate *is a*. If we want to create a dimension containing only transportation systems running on oil, we can use a query to retrieve the intersection of the keywords linked to the keyword *Transportation system* with the predicate *is a* and the keywords linked to the keyword *Oil* with the predicate *uses*.

The vocabulary of our ontology is easily represented using RDF[11], with the following triples expressed in the turtle syntax[5], assuming that we have an XML namespace *ex* for our ontology:

```
ex:Keyword rdf:type rdfs:class;
ex:Predicate [
  rdf:type rdf:Property;
  rdfs:range ex:Keyword;
  rdfs:domain ex:Keyword
] .
```

The ontology shown in Figure 4 corresponds to the following RDF statements:

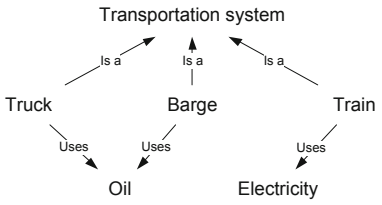

Fig. 4. Ontology example for transportation systems. Nodes are keywords and edges' labels are predicates.

```
ex:transportationSystem
   rdf:type ex:Keyword;
ex:is_a rdf:type ex:Predicate;
ex:truck ex:is_a
   ex:transportationSystem;
ex:barge ex:is_a
   ex:transportationSystem;
ex:train ex:is_a
   ex:transportationSystem;
ex:train ex:uses ex:oil;
ex:barge ex:uses ex:oil;
ex:train ex:uses
   ex:electricity;
```

Using RDF, a dimension is expressed via a SPARQL[16] query. For instance, the query to get all the keywords describing a transportation system is:

```
SELECT ?keyword
WHERE { ?keyword es:is_a ex:transportationSystem. }
```

Any change made to the keywords ontology can trigger an update to previously defined dimensions, therefore it will update already defined groups. For instance: if a new transportation system is added to the ontology shown in Figure 4 (like *air-plane*), every macro-process or macro-coefficient having a dimension containing transportation system keywords will be updated.

4.2 The Macro-graph Layer

We can create dependency relations between macro-processes using macro-coefficients (those relations are called *macro-relations*). The macro-processes and their dependencies are represented in a weighted digraph $G_M(V, E)$, where the vertices set V is the set containing the macro-processes, the edges set E is the set of dependency relations between groups of processes, and the set of weights associated to the edges is the set of macro-coefficients. Macro-processes and macro-coefficients are called *macro-nodes*. The correspondence between the Input/Output methodology and this approach is obvious: a macro-coefficient contain a block of the I/O matrix.

The macro-graph eases the model comprehension and offers a new way to model the dependency relations between processes based on the manipulation of the keywords ontology. Instead of creating several dependency relations between semantically close processes, we can create one macro-relation between two macro-processes.

Because the LCA modelling can contain cycles between processes, it is not possible to calculate impacts of processes directly on the macro-graph. We need to translate the relations between macro-processes into relations between processes. In other words, we need to translate the edges of the macro-graph into edges in the detailed-graph. Then, we can extract the coefficients matrix in order to compute impacts as it is usually performed in the LCA methodology. We will explain this procedure in the next section. First, we need to formalize our approach.

Macro-nodes Definitions

Let S be an ontology. We denote by $\mathscr{P}(S)$ the set of subsets of S without the empty set. We call **dimensions** the elements of $\mathscr{P}(S)$ and we denote by $\mathscr{P}(\mathscr{P}(S))$ the set of subsets of $\mathscr{P}(S)$. A macro-node is valid only if it has distinct dimensions, *i.e.*, if its dimensions have no keywords in common. We define the concept of dimension set consistency as follows:

Definition 3. *An element $\mathscr{D} \in \mathscr{P}(\mathscr{P}(S)), \mathscr{D} = (D_0, \ldots, D_n)$, is called consistent if $\forall (i, j) \in \mathbb{N}^2, i \neq j, D_i \cap D_j = \emptyset$.*

In order not to have any ambiguity in the choice of processes to link together when a macro-relation is translated into a set of relations in the detailed-graph, dimensions of the macro-nodes involved in a macro-relation must not match more than one dimension of the other macro-nodes. A dimension can match another dimension if their intersection is not empty. We define the concept of dimensions sets compatibility as follows:

Definition 4. *Two elements $\mathscr{D}$ and $\mathscr{D}'$ of $\mathscr{P}(\mathscr{P}(S))$ are compatible if the following properties are true:*

$$\forall D \in \mathscr{D},\ Card\{D' \in \mathscr{D}' | D \cap D' \neq \emptyset\} \leq 1$$
$$\forall D' \in \mathscr{D}',\ Card\{D \in \mathscr{D} | D' \cap D \neq \emptyset\} \leq 1$$

Example: let $\mathscr{D} = \{D_1, D_2\}$, $\mathscr{D}' = \{D_3\}$, $\mathscr{D}'' = \{D_4\}$ be three dimensions such that $D_1 = \{A, B\}$, $D_2 = \{C, E\}$, $D_3 = \{A, F\}$ and $D_4 = \{B, C\}$. We say that $\mathscr{D}$ and $\mathscr{D}'$ are compatible, while $\mathscr{D}$ and $\mathscr{D}''$ are incompatible because $D_1 \cap D_4 = \{B\}$ and $D_2 \cap D_4 = \{C\}$.

Having properly defined dimensions and restrictions to the notion of dimension, we can now define what is a macro-process as follows:

Definition 5. *Let P^i be the set of indexed processes. A macro-process is an ordered pair $(\mathscr{D}, \mathscr{P})$, where $\mathscr{D} = (D_0, \ldots, D_n)$ is a consistent set of dimensions and $\mathscr{P}$ is an application $\mathscr{P} : D_0 \times \cdots \times D_n \to P^i$.*

Another notation for a macro-process is based on the enumeration of its processes: $G_p = (\mathscr{D}, P_p) = (\{D_1, \ldots, D_n\}, \{p_1^i, \ldots, p_m^i\}) = (\{D_1, \ldots, D_n\}, \{(I(p_1), K_{p_1}), \ldots, (I(p_n), K_{p_m})\})$.

Similarly we can define what is a macro-coefficient as follows:

Definition 6. *Let C^i be the set of indexed coefficients. A macro-coefficient is an ordered pair $(\mathscr{D}, \mathscr{C})$, where $\mathscr{D} = (D_0, \ldots, D_n)$ is a consistent set of dimensions and $\mathscr{C}$ is an application $\mathscr{C} : D_0 \times \cdots \times D_n \to C^i$.*

We can also express a macro-coefficient as an enumeration: $G_c = (\mathscr{D}, \mathscr{C}) = (\{D_1, \ldots, D_n\}, \{c_1^i, \ldots, c_n^i\}) = (\{D_1, \ldots, D_n\}, \{(V(c_1), K_{c_1}), \ldots, (V(c_n), K_{c_n})\})$.

By extending the definition on dimensions set compatibility, two macro-processes $G_{P1} = (\mathscr{D}_1, \mathscr{P}_1)$ and $G_{P2} = (\mathscr{D}_2, \mathscr{P}_2)$ are compatible only if $\mathscr{D}_1$ and $\mathscr{D}_2$ are compatible. A macro-relation between an upstream macro-process G_{P1}, a macro-coefficient G_c and a downstream macro-process G_{P2} is valid only if those three macro-nodes are compatible. That is to say that G_{P1} must be compatible with G_c and G_{P2}, G_c must be compatible with G_{P2}.

Example: Let G_{P1} be a macro-process containing transportation system processes, such that $G_{P1} = (\{\{$ *Truck,Barge* $\}\}, \mathscr{P}_1)$. Let G_C be a macro-coefficient containing dependency coefficients between transportation processes and some electricity production processes such that $G_C = (\{$ $\{$*Truck,Barge*$\}, \{$*Electri-city*$\}, \{$*Coal,Oil*$\}\}, c)$. We can state that G_{P1} and G_C are compatible. Figure 5 shows a simplified graphical representation of those macro-nodes. The notation p_{Truck} used in this representation conveys that this process, referenced in G_{P1}, is indexed by the keyword *Truck* and the coefficient $c_{Truck,Electricity,Coal}$, referenced in G_C, is indexed by the keywords *Truck*, *Electricity* and *Coal*.

5 Translation from the Macro-graph to the Detailed-Graph

In order to calculate the impacts of the processes involved in a macro-relations between two macro-processes, we have to convert the macro-relation into a set of detailed-relations. A macro-relation is an edge in the macro-graph, weighted with a macro-coefficient. We denote such a relation by $((G_{P1}, G_{P2}), G_c)$, where G_{P1} and G_{P2} are macro-processes and G_c is a macro-coefficient. This relation is

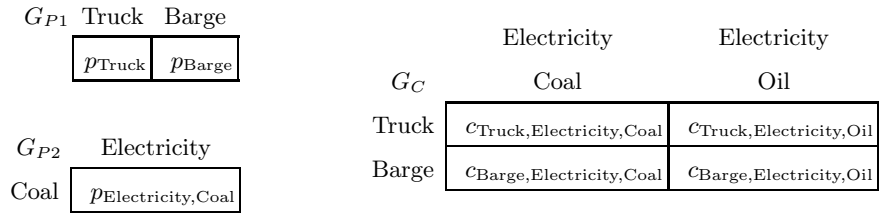

Fig. 5. Graphical representations of two macro-processes G_{P1} and G_{P2} and a macro-coefficient G_C. G_{P1} contains transportation processes and is composed of one dimension. G_{P2} contains the electricity production from coal process and is composed of two dimensions. G_C contains coefficients between G_{P1} and a macro-process containing electricity production processes (such as G_{P2}) and is composed of three dimensions.

translated into a set of detailed-relations, *i.e.*, a set of edges in the detailed-graph, denoted by $\{((p_i, p_j), c_{ij})\}$, where p_i and p_j are processes and c_{ij} is a coefficient. We only create a detailed-relation between two processes and a coefficient sharing a common indexation. Figure 6 shows the macro-graph corresponding to the macro-relation $((G_{P1}, G_{P2}), G_c)$ and its conversion into a detailed-graph. G_{P1}, G_{P2} and G_c are the groups shown in Figure 5. This translation procedure requires that we introduce two notions: the union of two dimensions sets and the matching number between two dimensions sets.

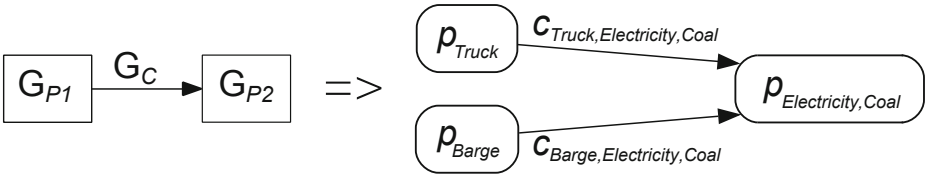

Fig. 6. The macro-graph on the left side contains a macro-relation between G_{P1} (containing transportation system processes) and G_{P2} (containing the electricity production from coal process) and is converted into the detailed-graph shown on the right side

The union of two dimensions sets is the union of the following three sets: 1) the intersection of the dimensions of the two sets that have a not empty intersection; 2) all the dimensions of the first set that do not intersect any dimension of the second set; 3) all the dimensions of the second set that do not intersect any dimension of the first set[6].

[6] This operation is different than the union of two sets or the union of two indexed family of sets. For instance, let $\mathscr{D}_1 = \{\{A, B, C\}\}$ and $\mathscr{D}_2 = \{\{A, B, D\}, \{E, F\}\}$ be two dimensions. The union of those two dimensions set is: $\mathscr{D}_1 \cup_D \mathscr{D}_2 = \{\{A, B\}, \{E, F\}\}$, because $\{A, B, C\} \cap \{A, B, D\} = \{A, B\}$ and $\{E, F\}$ does not intersect any dimension of $\mathscr{D}_1$.

Definition 7. *We denote by* $\cup_D$ *the union operator between two dimensions sets such that, for two dimensions sets* $\mathscr{D}_1$ *and* $\mathscr{D}_2$*:*

$$\begin{aligned}\mathscr{D}_1 \cup_D \mathscr{D}_2 =& \{D_1 \cap D_2 \mid D_1 \in \mathscr{D}_1 \wedge D_2 \in \mathscr{D}_2 \wedge D_1 \cap D_2 \neq \emptyset\} \\ & \cup \{D_1 \mid D_1 \in \mathscr{D}_1 \wedge \forall D_2 \in \mathscr{D}_2, D_1 \cap D_2 = \emptyset\} \\ & \cup \{D_2 \mid D_2 \in \mathscr{D}_2 \wedge \forall D_1 \in \mathscr{D}_1, D_2 \cap D_1 = \emptyset\}\end{aligned}$$

The matching number between two dimensions sets is the number of pairs of dimensions belonging to the two sets that have a not empty intersection.

Definition 8. *We denote by* $\alpha(\mathscr{D}_1, \mathscr{D}_2)$ *the matching number between two dimensions sets* $\mathscr{D}_1$ *and* $\mathscr{D}_2$*, such that:*

$$\alpha(\mathscr{D}_1, \mathscr{D}_2) = card(\{(D_1, D_2) | D_1 \in \mathscr{D}_1 \wedge D_2 \in \mathscr{D}_2 \wedge D_1 \cap D_2 \neq \emptyset\})$$

With those two definitions, we can define the translation rule for a macro-relation $((G_{P1}, G_{P2}), G_c)$. Let G_{P1} and G_{P2} be two macro-processes, G_c be a macro-coefficient and $((G_{P1}, G_{P2}), G_c)$ be an edge in the macro-graph. The translation rule for this edge into the detailed-graph is described as follows:

$$\begin{aligned}((G_1, G_2), G_c) \rightarrow \{((p_1, p_2), c) \mid & p_1 \in P^i \wedge p_2 \in P^i \wedge c \in C^i \\ & \wedge \text{card}(K_{p_1} \cap K_c) = \alpha(\mathscr{D}_1, \mathscr{D}_c) \\ & \wedge \text{card}((K_{p_1} \cup K_c) \cap K_{p_2}) = \alpha(\mathscr{D}_1 \cup_D \mathscr{D}_c, \mathscr{D}_2)\}\end{aligned}$$

When we want to translate a macro-relation, we need to try to associate every process of the upstream macro-process G_{P1} with a coefficient from the macro-coefficient G_c and with a process of the downstream macro-process G_{P2}. Because the three involved macro-nodes must be compatible, if the cardinal of the intersection between the keywords of an upstream process p_u and the keywords of a coefficient c equals the matching number $\alpha(\mathscr{D}_1, \mathscr{D}_c)$, both elements have a common indexation and could be part of a detailed-relation. The result of the combination of p_u and c would be indexed by the union of their keywords. The condition $\text{card}((K_{p_1} \cup K_c) \cap K_{p_2}) = \alpha(\mathscr{D}_1 \cup_D \mathscr{D}_c, \mathscr{D}_2)$ acts in the same way for the association of this combination with a downstream process p_d.

Implementation

The matching number between two dimensions sets calculation requires to count the number of pairs of dimensions that have a not empty intersection. The two dimensions sets are consistent because the macro-nodes in a macro-relation are consistent between each other. So, we can store every keyword of the second dimensions set into a hash table. Then, we can test if any of the keywords of each dimension of the first dimensions set is in this hash table. Hence, we only have to iterate on only one of the two dimensions.

The calculation of the union of two dimensions sets $\mathscr{D}_1$ and $\mathscr{D}_2$ is also done using a hash table to determine if a keyword in $\mathscr{D}_1$ can be found in a dimension

of $\mathscr{D}_2$. We explore all the dimensions of $\mathscr{D}_1$ and, if one dimension has a not null intersection with a dimension of $\mathscr{D}_2$, we store the result of the intersection and we mark both the dimensions of $\mathscr{D}_1$ and $\mathscr{D}_2$ as already used. Then, we just have to add all the unused dimensions of the two dimensions sets to the result. This algorithm is presented in listing 1. We consider that we have a data structure to store a dimension set and another one to store a dimension, and that both have a '+=' operator.

Algorithm 1. Calculation of the union of two dimensions sets

```
Input: 𝒟1,𝒟2
hashTable ← hash table containing the keywords of 𝒟2
union ← ∅ // union is a dimensions set
forall the dimension ∈ 𝒟1 do
    newDimension ← ∅
    forall the keyword ∈ dimension do
        if keyword ∈ hashTable then
            newDimension += keyword
            mark dimension as already used
    if newDimension = ∅ then
        union += dimension
    else
        union += newDimension
forall the unused dimension ∈ 𝒟2 do
    union += dimension
return union
```

The algorithm detailed in the listing 2 is the implementation of the translation rule for a macro-relation $((G_{P1}, G_{P2}), G_c)$. We consider that we have a data structure to store a process or a coefficient offering a property to access their indexation, this property is named *keyword*). We also have a data structure to store detailed-relations offering a '+=' operator.

Algorithm 2. Translation of a macro-relation into a set of detailed-relations

```
Input: GP1,GP2,Gc
detailedRelations ← ∅
forall the p1 ∈ GP1.processes do
    forall the c ∈ Gc.coefficients do
        if p1.keyword ∩ c.keyword = alpha(𝒟1,𝒟c) then
            forall the p2 ∈ GP2.processes do
                if (p1.keyword ∪D c.keyword) ∩ p2.keyword =
                alpha(𝒟1 ∪D 𝒟c,𝒟c) then
                    detailedRelations += ((p1,p2),c)
return detailedRelations
```

6 Experiments

As explained in the second section, the detailed-graph shown in Figure 2 contains the detailed-graph for the electricity production processes in the NREL LCI Database. In this dataset, we have 27 processes and 72 inter-process relations. After applying our methodology on this data set, we obtained the macro-graph shown in Figure 7. In this macro-graph, we have 13 macro-processes and 17 macro-relations, therefore we have 17 macro-coefficients. But we can reduce the number of macro-coefficients to 12 if we use macro-relations that do not link all the elements contained in the macro-nodes together, as in the example shown in Figure 6. For instance, the *Transports* macro-process can be linked to every electricity macro-process using a macro-coefficient. This macro-coefficient would contain all the dependency coefficients between transportation systems processes and the electricity production processes. We can even use only 8 macro-coefficients if we store all the dependency coefficients between every electricity production macro-process to the eGRID macro-process.

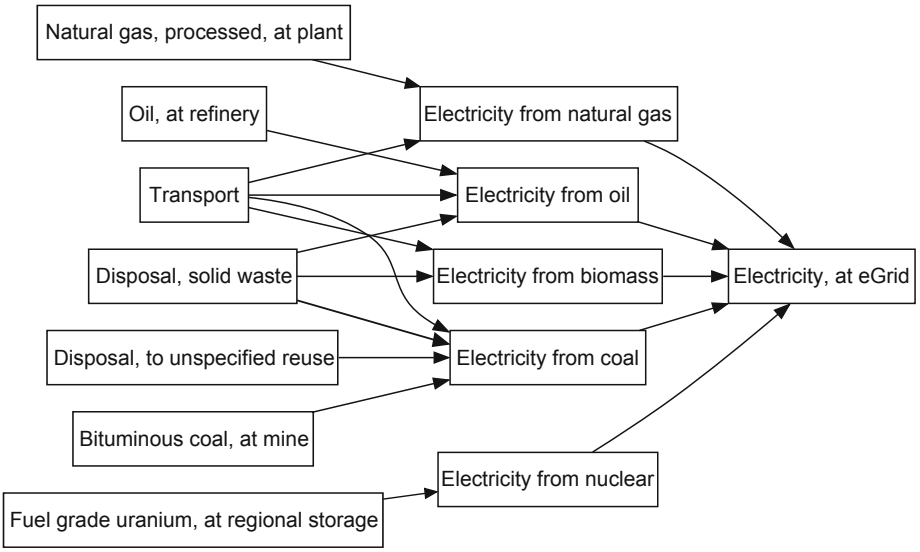

Fig. 7. The macro-graph of the studied data set extracted from the NREL's LCI Database

We also conducted some scalability tests on the translation algorithm to make sure that we can convert an important amount of macro-relations in a reasonable time[7]. This experiment tested the conversion of a macro-relation involving

[7] This methodology is intended to be used in a Software as a Service application. Hence, it is necessary to have a sufficiently optimized algorithm to translate the macro-relations into detailed-relations in a reasonable amount of time.

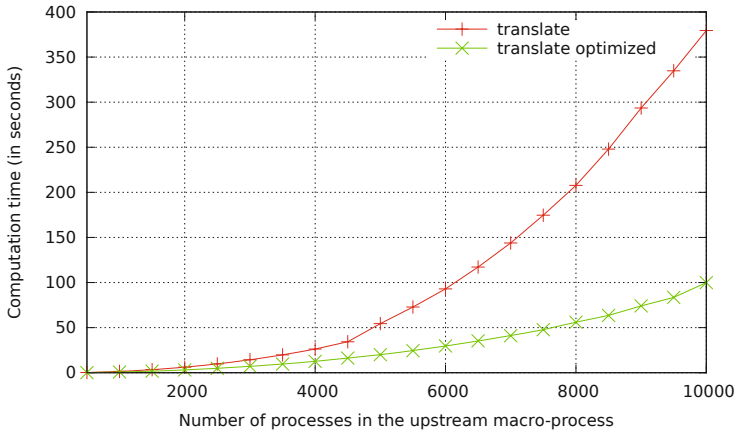

Fig. 8. Calculation time for the conversion algorithm of macro-relations depending on the number of processes included in the upstream macro-process

macro-nodes with a growing number of elements. Therefore, we generated an upstream macro-process with only one dimension and a growing number of keywords, so a growing number of processes. Then, we generated a downstream macro-process and a macro-coefficient related to the generated upstream macro-process. We also tested an optimization of this algorithm: we added a hash table to store all the coefficients in the macro-coefficients and we index this hash table with hash keys created with the common keywords between the dimensions of the upstream macro-process and the dimensions of the macro-coefficient. We conducted this experiment on a PC equipped with a Core i5-750 and 4GB of memory and the algorithms were implemented in PHP. We plan to use this language for the SaaS application as well. The results shown in Figure 8 contain a discontinuity around 5000 processes. This is due to a reallocation of the data structure used to store the detailed-relations. We limited our experiment to macro-nodes containing 10000 processes because, to the best of our knowledge, the most important LCI database contains 4000 processes[7].

7 Conclusions and Future Work

In this papaer, we proposed a new methodology to model LCI using an ontology and relations between semantic groups of processes. The key benefits of this approach is to offer a more understandable model of LCI databases and provide an ontology driven way to create relations between processes.

We plan to implement our model using OWL[13] with description logic rules and a semantic reasoner and study the impact performance of the system on existing LCI data sets. Moreover, processes indexation using keywords stored in an ontology can also be used to answer queries like: what is the impact of processes indexed with a keyword (or a set of keywords) on a specific process.

For instance, we can get the impacts of transport processes on the electricity production for a specific eGRID subregion. This can be done by restricting the calculation to upstream processes indexed by a specific keyword.

References

1. GABI Life Cycle Inventory Databases PE International, `http://www.gabi-software.com/` (last accessed: December 16, 2011)
2. U.S. Life Cycle Inventory Database. National Renewable Energy Laboratory, `http://www.nrel.gov/lci/` (last accessed: September 1, 2011)
3. ISO 14044 (2006): Environmental Management – Life Cycle Assessment – requirements and guidelines. International standard, International Organisation for Standardisation (ISO) (2006)
4. Bertin, B., Scuturici, M., Pinon, J.M., Risler, E.: A Semantic Approach to Life Cycle Assessment Applied on Energy Environmental Impact Data Management. In: Workshop on Energy Data Management in Conjunction with EDBT (2012)
5. Beckett, D.: Turtle Terse RDF Triple Language. W3C Recommendation (2011)
6. Pechan, E.H., Associates., Inc.: The Emissions & Generation Resource Integrated Database for 2010 Technical Support Document. U.S. Environmental Protection Agency, Washington, D.C (2010)
7. Frischknecht, R.: G Rebitzer. The ecoinvent database system: a comprehensive web-based LCA database. Journal of Cleaner Production (2005)
8. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition (1993)
9. Guinée, J.B.: Handbook on Life Cycle Assessment: Operational Guide to the ISO Standards. Springer, New York (2002)
10. Heijungs, R., Suh, S.: The computational structure of life cycle assessment. Kluwer Academic Publishers, Dordrecht (2002)
11. Klyne, G., Carroll, J.J. (eds.): Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation (2004)
12. Leontief, W.: Input-Output Analysis, pp. 53–83. University of British Columbia Press (1986)
13. McGuinness, D., van Harmelen, F. (eds.): OWL 2 Web Ontology Language Document Overview. W3C Recommendation (2009)
14. McGuinness, D.L.: Ontologies come of age. In: The Semantic Web: Why, What, and How, pp. 171–192. MIT Press (2002)
15. Peters, G.P.: Efficient Algorithms for Life Cycle Assessment, Input-Output Analysis, and Monte-Carlo Analysis. The International Journal of Life Cycle Assessment (2007)
16. Prud'hommeaux, E., Seaborne, A. (eds.): SPARQL Query Language for RDF. W3C Recommendation (2008)
17. Frischknecht, R., Jungbluth., N., Althaus, H., et al.: Introduction The ecoinvent Database: Overview and Methodological Framework. The International Journal of Life Cycle Assessment (2005)
18. Varga, R.: Matrix Iterative Analysis. Series in Computational Mathematics. Springer (2010)
19. Nicholson, W.K.: Elementary linear algebra with applications. PWS-Kent Publishing Company (1990)