MODELLING DATA TRANSFORMATION PROCESSES USING HIGH-LEVEL PETRI NETS

Li Peng
Software School, Hunan University, Changsha, Hunan, China

Keywords: Data heterogeneity, data transformation, process model, coloured Petri net, higher-order Petri net.

Abstract: Data heterogeneity is one of the key problems in integrating multiple data sources, data warehousing, legacy data migration, etc. For integrating databases or information systems, the data need to be transformed from a source representation into a target representation. The foundation for developing efficient data transformation tools and automating data transformation processes is a data transformation process model. In this paper, I propose a CPN based data transformation process model. This model provides rich constructs to represent various data structures, transformation functions and rules; allows parallelization, composition and decomposition of data transformations. Furthermore, as the extension of the model, the CPNs are combined with higher-order Petri nets. The components of the CPNs can be reused. This improves the efficiency of data transformations.

1 INTRODUCTION

Data heterogeneity is one of the key problems in integrating multiple data sources, data warehousing, legacy data migration, etc. This kind of heterogeneities refers to different representations of the same data. For instance, different units of measurement, different abstraction levels, distinct representations of the same data domain, or different data formats would cause data heterogeneities. In order to integrate databases or information systems, we often need to transform data from a source representation into a target representation.

A data transformation process potentially involves a considerable number of transformation functions or operations , which apply to input data and generate output data. This kind of operations appears implicitly in most languages. For instance, we can perform data transformations by executing a sequence of SQL queries against source data. However, query languages were not developed to represent complex transformations for solving data heterogeneities. Many data transformations cannot be expressed by SQL queries.

In many languages and data transformation tools (for example, Data Fusion tool (Carreira and Galhardas, 2004 (a)), data mapper operators are developed for implementing data transformations. In (Carreira and Galhardas, 2004 (b)), the data mapper operators as an extension to the relational algebra are proposed for increasing its expressive power. The mapper operators are based on a tuple semantics, can not specify many useful data transformations.

The foundation for developing efficient data transformation tools and automating data transformation processes is a data transformation process model, which integrates input data, output data, required transformation functions and transformation rules. This model should be a visual notation, describe progressive transformations, and can be optimized.

This paper is organized as follows. Section 2 gives an overview of data structures and transformation functions in data transformation processes. Section 3 presents the data transformation model. Section 4 concludes the paper with a summary of the contributions of this research.

2 DATA AND FUNCTIONS IN DATA TRANSFORMATION PROCESSES

Data transformations deal with large amounts of data and potentially involve a considerable number of transformation functions or operations. The execution of some functions is controlled by rules. Therefore, data, transformation functions and rules are basic elements in data transformation processes.

2.1 Data Structures

The inputs of data transformations are attributes in relational databases, data warehouses, etc. The attributes can have complex data structures and be represented by using constructors. A composite attribute corresponds to utilization of the tuple constructor, whereas a multivalued attribute may be of a list, set, or bag data type, and corresponds to the list, set, or bag constructs. Complex data structures can be created by the nesting of type constructors (tuple, set, list, bag, array, etc.).

2.2 Transformation Functions

Data transformations may involve a number of transformation functions or operations. Transformations can be composed or decomposed. Composition of transformations allows functions to be composed together and executed as a single function, whereas the decomposition of a transformation allows a function to be partitioned into composeable functions. According to this criterion, data transformations can be classified into simple transformations and complex transformations.

A simple transformation involves only a single function, which is not decomposable, whereas a complex transformation can be decomposed into a number of functions. The functions applied in simple transformations can be relational operations, such as join(), selection(), and projection(); aggregate functions, such as sum(), avg(), count(), max() and min(); etc. A complex transformation is composed of a number of functions and allows sequential and parallel execution of functions.

On the other hand, transformation functions can be classified according to the following criteria:

- One-to-one map: a function receives an input and generates an output.
- One-to-many map: a function receives a single input and generates multiple outputs.
- Many-to-one map: a function receives multiple inputs and generates a single output.
- Many-to-many map: a function receives multiple inputs and generates multiple outputs.
- Algorithmic map: a function is performed according to an algorithm, e.g. iteration.

In order to simulate data transformation processes, the data transformation model should be a visual notation, provide constructs to represent various data structures, transformation functions and rules. Moreover, the composition, decomposition and parallelization of data transformations can be expressed in this model.

3 MODELLING DATA TRANSFORMATION PROCESSES

In graphical notations, UML is one of the most popular graphical specification tools. However, it is not suitable to express progressive data transformation processes. Coloured Petri Net (CPN) is an alternative to UML for modelling data transformations.

3.1 Modelling Data Transformation Processes Using CPN

CPNs are one of the enhancements of Petri nets, which are especially suitable for modelling transformation processes of complex data types and expressing progressive transformation processes.

3.1.1 The CPN Model

CPNs are formally defined as follows (Jensen, 1992):
CPN=(S, P, T, A, N, C, G, E, M_0, I), where
S: a finite set of non-empty types, called color sets;
P: a finite set of places;
T: a finite set of transitions;
A: a finite set of arcs $P \cap T = P \cap A = T \cap A = \varnothing$;
N: a node function $A \to P \times T$ □ $T \times P$;
C: a color function $P \to S$;
G: transition inscriptions T →expression;
E: arc inscriptions A → expression;
M_0: initial tokens;
I: an initialization function $P \to M_0$.

CPNs provide the following graphical notations: circles for places, boxes for transitions, arcs for data flow between them, and dots for tokens.

In a CPN based data transformation process model, a color set represents a data type, which may be the input or the output of a transformation function. Various data types occurring in data transformation processes can be represented using different color sets. For example, a composite attribute "Date" can be represented by a color set defined as follow:

color Date = *record* (year, month, day).

A textual inscription can be assigned to each graphical symbol. Arc inscriptions restrict the arc's data flow, which is described with variables. Transition inscriptions define guard conditions and actions. Guard conditions restrict the firing of the transitions. Actions correspond to transformation functions or operations and are executed when the transitions fire.

Simulation is based on the following firing rules: When data tokens (represented by color sets) are available in each input place of a transition, the arc conditions and the guard conditions are evaluated and the output places are checked for space for tokens. When all these conditions are fulfilled (the preconditions of a data transformation are fulfilled), the transition may fire. When the transition fires, all input data tokens are consumed and output tokens are delivered to all output places according to the action. With the moving of tokens in CPN, data transformations are performed step by step.

3.1.2 Execution of Data Transformations in CPNs

CPNs allow sequential and parallel execution of data transformations. Composition and decomposition of data transformations can be also expressed in CPNs.

The data of an input place can be delivered to different transformation functions; the functions can execute in parallel and yield different output results. If the data type of an input place is a composite type, the input data can be partitioned and delivered to different transformation functions. For example, if the data type of an input place is a tuple, the tuple can be partitioned into single attributes, and the attributes would be delivered to different transformation functions; finally, the results of different functions would be merged into a tuple in the output place.

The concept of hierarchy and composition for CPN allows transitions to be replaced by subnets. A hierarchical CPN (HCPN) model can be developed either top-down or bottom-up. Subnets (sub-pages) can be used to describe decomposition of complex transformations. Each subnet provides a detailed description of a complex transformation.

Data transformation rules can be expressed as guard conditions in transition inscriptions. The transformation rules control the execution of transformation functions, e.g. iteration.

3.2 An Extension for Data Transformation Process Model

CPNs are a powerful modelling notation for data transformation processes. They provide rich constructs for representing various data types, transformation functions and rules. The parallelization, composition and decomposition of data transformations can be expressed in CPNs. However, a data transformation process potentially involves some special procedures or algorithms. Sometimes, the procedures or algorithms involved in CPNs need to be reused. For example, in a data transformation process, more than one hierarchical structures need to be restructured. Obviously, we would not like to rewrite procedures for each hierarchical structure. This problem can be solved by applying higher-order Petri nets (Janneck and Esser, 2002). We can use the higher-order facilities to construct a model for the procedure that, when parameterized with a specific hierarchical structure, automatically instantiates the corresponding procedure. The higher-order Petri nets can be combined with CPNs.

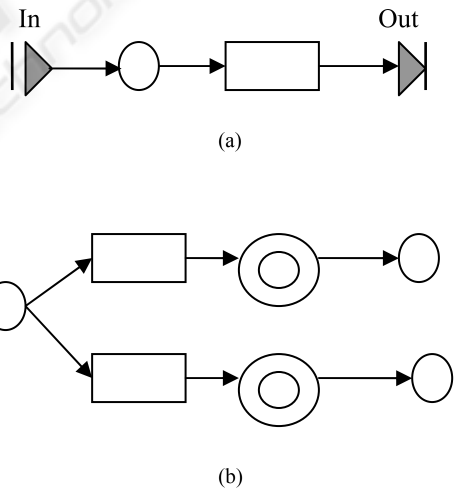

Figure 1: A higher-order Petri net

Figure 1(a) shows a parameterized Petri net which represents the procedure for restructuring hierarchical structures. The parameters are the attributes existing in a hierarchical structure. The net contains an input port and an output port. This model is called a component model. Figure 1(b) is a CPN which describes a data transformation process. The CPN contains two double-rimmed places, called container places. Each container place contains input ports and output ports. The procedure (component) resides as a token on the container places. When a transition that is connected to an input container port produces a token, the token is sent to the input port of the component. Similarly, when the component produces a token at its output port, the token appears at the corresponding output container ports of the container place. By using component models and container places, procedures or algorithms can be reused. This reduces the model size and complexity, and improves the efficiency of data transformations.

4 CONCLUSIONS

In this paper, I proposed a data transformation model for developing efficient data transformation tools and automating data transformation processes in integrating multiple data sources, data warehousing, legacy data migration, etc. The model is based on CPNs (Coloured Petri nets) and provides rich constructs to represent various data structures, transformation functions and rules. The parallelization, composition and decomposition of data transformations can be expressed in this model. As an extension of the model, higher-order Petri nets are combined with CPNs. In this extended model, the components of CPNs can be reused. This improves the efficiency of data transformations.

REFERENCES

Carreira, P., Galhardas, H., 2004(a). Execution of Data Mappers. In *IQIS*. pp.2-9. ACM.

Carreira, P., Galhardas, H., 2004(b). Efficient development of data migration transformations. Demo Paper. In *ACM SIGMOD International Conference on the Managment of Data*. Paris, France.

Esser, R., Janneck, J.W., 2000. Exploratory Performance Evaluation using Dynamic and Parametric Petri Nets. In *Proceedings of the HPC 2000*. pp.357-364, Society for Computer Simulation.

Galhardas, H., Florescu, D., Shasha, D., Simon, E., Saita, C.A., 2001. Declarative data cleaning: Language, model, and algorithms. In *Proceedings of the International Conference o Very Large Data Bases (VLDB'01)*. Rome, Italy.

Hoffmann, K., Mossakowski, T., 2003. **Algebraic Higher Order Nets: Graphs and Petri Nets as Tokens. In** *Recent Trends in Algebraic Development Techniques, 16th International Workshop, WADT 2002.* Frauenchiemsee, Germany, Revised Selected Papers, LNCS Vol. 2755, pp. 253-267, Springer-Verlag.

Huber, P., Jensen, K., Shapiro, R.M., 1989. Hierarchies in Colored Petri Nets. In *10th International Conference on Application and Theory of Petri Nets*. Bonn.

Janneck, J.W., Esser, R., 2002. High-order Petri net Modeling – techniques and applications. In *Conferences in Research and Practice in Information Technology*. Vol. 12. C., pp.17-25.

Jensen, K., 1992. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*, Vol.1, Springer-Verlag.

Lakos, C.A., 1997. On the Abstraction of Coloured Petri Nets. In *Proceedings of the 18th International Conference on the Application and Theory of Petri Nets*. Vol. 1248, pp. 42-61, Lecture Notes in Computer Science, Springer-Verlag.

Lakshmanan, L.V.S., Sadri, F., Subramanian, I. N., 1996. SchemaSQL - a Language for Querying and Restructuring Database Systems. In *Proc. International Conference on Very Large Databases (VLDB'96)*. pp. 239–250, Bombay, India.

Miller, R.J., Haas, L.M., Hernand'ez, M., 2000. Schema Mapping as Query Discovery. In *Proceedings of the International Conference on Very Large Data Bases (VLDB'00)*. pp. 77–78, Cairo, Egypt.

Oswald, H., Esser, R., Mattmann, R., 1990. An Environment for Specifying and Executing Hierarchical Petri Nets. In *Proceedings of the 12th International Conference on Software Engineering*. pp. 164-172.

Raman, V., Hellerstein, J., 2001. Potter's Wheel: An Interactive Data Cleaning System. In *Proceedings of the International Conference on Very Large Data Bases (VLDB'01)*. Roma, Italy.

Schallehn, E., Sattler, K., Saake, G., 2001. Advanced grouping and aggregation for data integration. In *Proceedings 10th International Conference on Information and Knowledge Management, CIKM'01*. Atlanta, GA, USA.

Xu, J., Kuusela, J., 1998. Modeling Execution Architecture of Software System Using Colored Petri Nets. In *WOSP98*. pp. 70-75.