Coordination Rules Generation from Coloured Petri Net Models

Adja Ndeye Sylla, Maxime Louvel, François Pacull

Univ. Grenoble Alpes, CEA, LETI, MINATEC Campus, F-38054 Grenoble, France
AdjaNdeye.sylla@cea.fr, maxime.louvel@cea.fr, francois.pacull@cea.fr

Abstract. This paper presents an environment to automatically generate coordination rules from coloured Petri nets models.

1 Introduction

Today's systems such as building automation or industrial control processes are composed of many heterogenous and interacting components. These interactions raise problems such as data sharing and concurrent accesses. Coordination models and languages [4] are essential to provide a simple way to handle these interactions. LINC [3] is a rule based coordination environment. It is used to develop and deploy distributed applications. A LINC application is a set of rules enacted in distributed rule engines. LINC relies on powerful mechanisms such as transactions to ensure the normal execution of a rule. However, this does not prevent from writing conflicting rules. Currently, these conflicts are detected at execution time, when bugs are observed.

2 Contribution

The proposed approach consists in generating LINC rules from coloured Petri Net (CPN) models [2]. An application is first modelled using CPNs. This helps exchanges among all team members. Then the refined CPNs are verified to avoid undesired behaviours. Finally, the corresponding LINC rules are automatically generated and directly executed by LINC.

An application developed using LINC is a set of rules manipulating resources in several bags (bags are a distributed associative memory [1]). A resource is a tuple of `strings` and is manipulated using three operations: rd, get, and put to respectively verify the presence of resources, consume and insert resources. A rule consists of a precondition (i.e. verification of conditions) and a performance (i.e. actions to perform, atomically when the conditions are verified). To enable the automatic generation of LINC rules, we limit the colours. Colours defining states of the system are enumeration. Other tokens are tuples of strings.

To generate the rules, we define a transformation to move from CPN to LINC rules (Table 1a). Tokens in places are mapped to resources in bags. A transition is mapped to a performance. Indeed they both verify conditions and atomically

perform actions. An arc is mapped to an operation: both direction to `rd`, place to transition to `get` and transition to place to `put`. In LINC rule, the precondition explicitly specifies that the performance is triggerable when the specified resources are present. This is implicit in a CPN (i.e. a transition is enabled as soon as the specified tokens are present). Thus precondition is generated using the incoming arcs of a transition. To enable the graphical representation and the verification of LINC applications which were manually developed, we define the reverse transformation to move from LINC rules to CPN (Table 1b).

Coloured Petri net	**LINC rule**
Token	*Resource*
Place	*Bag*
Transition	*Performance*
Arc	*Operation*
orientation	type
Arcs incoming in a transition	*Precondition*

(a) CPN model to LINC model

LINC	**Coloured Petri net**
Resource	*Token*
Bag	*Place*
Operation	*Arc*
type	orientation
Performance	*Transition*
Precondition	*Implicit in CPN*

(b) LINC model to CPN model

Table 1: Defined transformations

Verification of CPN is limited in existing tools. Thus a CPN is first transformed to a PN by unrolling the colours with enumeration. Other places are left as is. Then, the generated PN is verified using existing model checkers. This enables to verify undesired behaviours such as deadclok and livelock.

3 Conclusion

This paper has presented a method to automatically generate rules from validated Coloured Petri Nets. CPNs verification ensures the global behaviour and LINC ensures that each step is actually executed according to the CPN. Hence distributed applications can safely be executed in actual distributed and embedded systems. This has been validated in a smart parking solution including several off-the-shelves hardware and software components.

References

1. N. Carriero and D. Gelernter. Linda in context. *Commun. ACM*, 1989.
2. K. Jensen. *Coloured Petri nets: basic concepts, analysis methods and practical use.* Springer Science & Business Media, 2013.
3. M. Louvel and F. Pacull. Linc: A compact yet powerful coordination environment. In *Coordination Models and Languages*. Springer, 2014.
4. G. Papadopoulos and F. Arbab. Coordination models and languages. *Advances in computers*, 1998.