Towards a Temporal Logic τLTL for the Verification of Rewriting Theories denoting ECATNets

Hacene Sebih, [•] Mohamed Bettaz, [••] Kamel Barkaoui, Faiza Belala, Zaidi Sahnoun
LIRE Laboratory University of Constantine,
Route de Ain El Bey, Constantine, Algeria
hsebih@hotmail.com

[•] Philadelphia University, Amman Jordan
bettaz@hotmail.com

[••] CEDRIC laboratory CNAM Paris
292 Rue Saint-Martin 75414 Paris Cedex03
barkaoui@cnam.fr

Abstract: The propositional linear temporal logic LTL [**Man89]** is a well studied tool for specifying and reasoning about distributed programs. When an interleaving semantics is adopted, the main problem is the state explosion induced by these many linearizations.
An alternative approach is to use rewriting logic as a truly concurrent model and semantic framework for the specification of languages and systems [**Mes90**]. Some useful models of concurrency, such as ECATNets, can be obtained as special cases of concurrent rewriting by imposing some restrictions on the rewrite theory being used.
The initial model $\tau_{\Re}$ of a rewrite theory $\Re$ is a category with algebraic structure, where the objects correspond to system states, and the arrows, libelled by proof-terms induced by one-step-concurrent rewrites correspond to concurrent system transitions. The behaviour of an ECATNet denoted by a theory $\Re$ is represented in a very compact way.
To make the integration straightforward, both logics must refer to the same mathematical structure. The aim of this paper is defining an action-based temporal logic, namely τLTL devoted to rewriting theories denoting ECATNets where the underlying structure is a category $\tau_{\Re}$.

Key Words: ECATNets, Rewriting Logic, Category Theory, Temporal Logic.

1. Introduction

The verification of programs is a crucial step for the conception of critical systems, especially for concurrent systems. The model checking approach starts with the abstraction of the actual system into some automata based model. Then the specification to be checked is expressed in some suitable temporal logic. Finally, a tool is used to determine whether the system meets its specification.

Petri nets are a widely used model of concurrency. This model is attractive from a theoretical point of view because of its simplicity and intrinsically concurrent nature. Concurrent systems modelled by Petri nets are usually reduced to sequential ones by considering all possible linearizations of concurrent behaviours. That is, interleaving semantics is adopted. The main problem with this approach is the state explosion induced by these many linearizations.

An alternative approach is to use rewriting logic as a truly concurrent model. The rewriting logic is an elegant and expressive semantic framework for the specification of languages and systems [**Mes90**]. Some useful models of concurrency can be obtained as special cases of concurrent rewriting by imposing some restrictions on the rewrite theory being used such as ECATNets model [**Bem93**].

ECATNets are High-level nets devoted to modelling concurrent systems at different level of abstraction. They allow to describe compactly complex systems, using Petri nets to model constraints and abstract data types for specifying the data structure. ECATNets have a well-defined formal semantics, which in an unambiguous way define the behaviour of the specified system.

From a syntactic point of view, an ECATNets places are marked with multi-sets of algebraic terms, input arcs of each transition t ,i.e. (p,t), are labelled by two inscriptions **IC**(p,t) (Input Conditions) and **DT**(p,t) (Destroyed Tokens), output arcs of each transition t , i.e. (t,p'), are labelled by **CT**(p',t) (Created Tokens), and finally each transition t can be labelled by **TC**(t) (Transition Conditions).

The expression of ECATNets in rewriting logic is then obvious. We can view each of the labelled arcs of an ECATNet as a rewrite rule in a rewrite theory having a binary ACI (where ACI stands for associativity, commutativity and identity) operator $\otimes$ (multi-set union) so that rewriting happens modulo ACI. In the order-sorted version of rewriting logic, we can consider the "places" into a type Place and view states of the net, usually called markings, as elements of a super-type marking containing place.

Theories in rewriting logic have initial models. The initial model $\tau_{\Re}$ of a rewrite theory $\Re$ is a mathematical model of the concurrent system axiomatized by $\Re$. Specifically, it is a category with algebraic structure, where the objects correspond to system states, and the arrows correspond to concurrent system transitions. Therefore, $\tau_{\Re}$ can be regarded as a particular Kripke structure whose transitions are labelled by proof-terms induced by rules of deduction. By adopting a "true concurrency" approach the behaviour of an ECATNet denoted by a theory $\Re$ is represented in a very compact way.

Rewriting logic is a logic that we could call internal to the concurrent computation of a system that it axiomatizes. Deduction precisely corresponds to computation, and in fact can be used to implement the system axiomatized by the rules. By contrast, temporal logic can be regarded as taking an external view of the system. It regards it as a mathematical model typically some kind of Kripke structure about which it then makes assertions about its global properties such as safety or liveness [**Den98**]. Both levels of description and analysis are useful in their own right; in fact, they complement each other. We therefore plan to use both logics in combination to prove properties about concurrent systems.

To make this integration straightforward, both logics must talk about the same mathematical structure. The aim of this paper is defining an action-based temporal logic, namely τLTL, devoted to rewriting theories denoting ECATNets and where the underlying structure is a category $C_{\aleph}$. The usual temporal operators are replaced by operators relativised by morphisms and the satisfaction relation is then formalized over the category $C_{\aleph}$ instead of a classical Kripke structure.

The remainder of this paper is organized as follows. In section 2, we give a review of rewriting logic. Section 3, recall some notions on the ECATNet model. In section 4, we explain how a transcription of an ECATNet as a rewriting theory is straightforward and how logical deduction, based on *one-step concurrent rewrites*, may represent a behavioural aspect of an ECATNet in a very compact way. In section 5, we introduce τLTL an action-based temporal logic whose models are categories where the objects represent global states (equivalence classes of Σ-terms) where the morphisms correspond to concurrent system transitions (equivalence classes of proof-terms). In section 6, we introduce a simple example, on which we express and verify some τLTL formulae. The paper ends with some concluding remarks.

2. The Rewriting Logic Review

In this section, we recall the definition and the model theory of rewriting logic. Details may be found in [**Mes96**]. A signature in rewriting logic is a pair (Σ,E) with Σ a ranked alphabet of function symbols and E of Σ-equations (called structural equations). Rewriting will operate on equivalent classes of terms

modulo the set of equations E. Given a signature (Σ,E) sentences of a logic are sequents of the form r : $[t]_E \rightarrow [t']_E$ where t and t' are Σ-terms possibly involving some variables. A theory in this logic, called a rewrite theory is defined as follows:

Definition 1. A labelled rewrite theory ℜ is a 4-tuple ℜ =(Σ, E, L, R) where (Σ,E) is a signature, E a set of Σ-equations, L is a set of labels, and $R \subseteq L \times (T_{\Sigma,E}(X))^2$ is the set of labelled rewrite rules. For a rewrite (r,[t],[t']), we use the notation $r : [t] \rightarrow [t']$

Definition 2. Given a labelled rewrite theory ℜ, we say that ℜ entails a sequent $r : [t] \rightarrow [t']$ and write $\Re \models r : [t] \rightarrow [t']$ iff $[t] \rightarrow [t']$ can be obtained by finite application of the following deduction rules :

1. Reflexivity

$$\forall [t] \in T_{\Sigma,E}(X), \qquad \overline{[t] \rightarrow [t]}$$

2. Congruence

$$\forall f \in \Sigma_n, n \in \mathbb{N} \qquad \frac{[t_1] \rightarrow [t_1'] \quad \ldots [t_n] \rightarrow [t_n']}{[f(t_1,\ldots t_n)] \rightarrow [f(t_1',\ldots t_n')]}$$

3. Replacement

For each rewrite rule $r : [t(x_1,\ldots x_n)] \rightarrow [t'(x_1,\ldots x_n)]$

$$\frac{[w_1] \rightarrow [w_1'] \quad \ldots \quad [w_n] \rightarrow [w_n']}{[t(\overline{w}/\overline{x})] \rightarrow [t'(\overline{w}/\overline{x})]}$$

where $(\overline{w}/\overline{x})$ is a substitution of w_i by x_i, $1 \le i \le n$

4. Transitivity

$$\frac{[t_1] \rightarrow [t_2] \qquad [t_2] \rightarrow [t_3]}{[t_1] \rightarrow [t_3]}$$

Definition 3. A sequent $r : [t] \rightarrow [t']$ is said a one-step concurrent ℵ-rewrite iff it can be derived from ℵ by a finite application of the rules of deduction 1-3 with at least one application of rule 3. When the replacement rule is applied once, the sequent is called a one-step-sequential rewrite.

Rewriting logic is a logic for reasoning correctly about concurrent systems having states, and evolving by means of transitions. The model for a given labelled rewrite theory ℜ=(Σ, E, L, R), is a category $\tau_{\Re}(X)$ whose objects (states) are equivalence classes of terms $[t] \in T_{\Sigma,E}(X)$ and whose morphisms (transitions) are equivalence classes of proof-terms representing proofs in rewriting deduction, i.e., concurrent ℜ-rewrites. The category $\tau_{\Re}(X)$ is one among many other categories that can be assigned to the rewrite theory ℜ. The general notion of model called an ℜ-system is also a category ℜ-sys which is detailed in [**Mar92**][**Mes95**][**Mes96**]. A variety of models of concurrency have been obtained as special cases of concurrent rewriting. A natural way of studying specializations of this kind is to impose restrictions on the rewrite theories being used.

A wide variety of models of computations and languages such as Petri nets, Actors, CCS, event structures can be naturally expressed in a rewriting logic. In particular, ECATNets (Extended

Concurrent Algebraic Term nets) are a special case of concurrent rewriting obtained by the application of rewriting logic to algebraic Petri nets.

3. ECATNets Review

ECATNets [**Bem93**] are a specification framework based on an ad-hoc combination of Petri nets and abstract data types. Their semantics is defined in terms of rewriting logic allowing the analysis of the built models by formal reasoning.

Motivating ECATNets (Extended Concurrent Algebraic Term Nets) leads to motivating Petri nets, abstract data types, as well as their association into a unified framework. Petri nets are used for their foundation in concurrency and dynamics, while abstract data types are used for their data abstraction power and solid theoretical foundation. Their association into a unified framework is motivated by the need to explicitly specify process behaviour and complex data structures in real systems.

3.1 The Syntax

From a syntactic point of view, places and arcs are annotated by multi-sets of algebraic terms. Algebraic terms are distinguishable entities, thus multi-sets including them cannot be represented by natural numbers. By marking a place p, we understand the set of tokens M(p) annotating this place. The input arcs of each transition t ,i.e. (p,t), are labelled by two inscriptions **IC**(p,t) (**I**nput **C**onditions) and **DT**(p,t) (**D**estroyed **T**okens), output arcs of each transition t , i.e. (t,p'), are labelled by **CT**(p',t) (**C**reated **T**okens), and finally each transition t can be labelled by **TC**(t) (**T**ransition **C**onditions). IC(p,t) specifies that the enabling condition of the transition t, DT(p,t) specifies the tokens (a multi-set) which have to be removed from p when t is fired, CT(t,p') specifies the tokens (a multi-set) which have to be added to p' when t is fired. finally, TC(t) represents a boolean term which specifies an additional enabling condition for the transition t. The current ECATNets state is given by the union of terms having the following form (p, M(p)). As an example, the distributed state s of an ECATNet represented by one transition t and one input place p marked by the multi-set a⊕b⊕c, and an empty output place p', is given by the following multi-set: $s = (p, a \oplus b \oplus c) \otimes (p', \phi)$ where $\otimes$ denotes the union on multi-sets.

3.2 The Semantics

It is worthwhile to mention that it is not easy to explain the behaviour of ECATNets merely by giving the equivalent of a firing-like rule [**Bem93**]. This is because of their level of abstraction as well as their concurrent behaviour. We may however, informally comment this behaviour in the following way:

A transition t is fireable when various conditions are simultaneously true. The first condition is that every IC(p,t) for each input place p is satisfied (this conditional part express the fact that IC(p,t) is included or not in M(p)). The second condition is that TC(t) (external Boolean expression) is true. When t is fired DT(p,t) is removed from the input place p and simultaneously CT(p',t) is added to the output place p' with respect to its capacity when this capacity is finite.

Transition firing and its conditions are formally expressed by rewrite rules which are strongly depending on the form of the syntactic notation used for representing IC(p,t). Such rules act in reality as the axioms of the higher mentioned rewrite logic. The basic notions of this logic in relation of its use with ECATNets are summarized as follows:

The axioms are in reality conditional rewriting rules describing transitions effects as elementary types of changes. The deduction rules allow us to draw valid conclusions about the evolution of the ECATNet from these changes. A rewrite rule is a structure of the form " t : u → v if boolexp"; where u and v are respectively the left and the right-hand sides of the rule, t is the transition associated with this rule, and boolexp is a Boolean term. More precisely u and v are multi-sets of pairs of the form (p, $[m]_{\otimes}$), where p is a place of the net and $[m]_{\otimes}$ a multi-set of algebraic terms. The multi-set union on these pairs (p, $[m]_{\otimes}$) considered as singletons will be denoted ⊗. We denote by $[x]_{\otimes}$ the equivalence class of x, i.e. the ACI axioms for ⊗.

An ECATNet state is itself represented by a multi-set of such pairs. Given a set R of rewriting rules (defining all the elementary types of changes), we say that R entails a sequent s → s' (defining a global change from a state s to a state s') iff s → s' can be obtained by finite and concurrent applications of the following rules of deduction: Reflexivity, Congruence, Replacement [Mes96]; Splitting, Recombination, and Identity [**Bem93**].

The Reflexivity rule says that everything may be transformed into itself. The Congruence rule says that elementary changes have to be correctly propagated. The Replacement rule is used when variable instantiations are necessary. The Splitting and Recombination rules allow us, by "judiciously" splitting and recombining different multi-sets of equivalence classes of terms, to detect ECATNet computations exhibiting a maximum of parallelism. The Identity rule allows to relate ϕ_M (i.e., the identity element of ⊗) with ϕ_B (i.e., the identity element of ⊗). The form of the rewrite rule (i.e., the meta-rules) associated to the net transitions will depend on the form of the multi-set IC(p,t) of enabling tokens and on its relation with DT(p,t) the multi-set of destroyed tokens [Bem93].

- $IC(p,t) \subseteq M(p) \neq \phi_M$

  case1 : $[IC(p,t)]_{\oplus} = [DT(p,t)]_{\oplus}$ The associated meta-rule has the following form:

  $$t : (p, [IC(p ,t)])_{\oplus} \rightarrow (p', [CT(p',t)])_{\oplus}$$

  case2 : $[IC(p,t)]_{\oplus} \cap [DT(p,t]_{\oplus} = \phi_M$ The meta-rule has the following form:

  $$t : (p, [IC(p,t)]_{\oplus}) \otimes (p, [DT(p,t)]_{\oplus} \cap [M(p)]) \rightarrow (p, [IC(p,t)]_{\oplus}) \otimes (p',[CT(p',t)]_{\oplus})$$

case3 $[IC(p,t)] \cap [DT(p,t)] \neq \phi_M$ This situation corresponds to the most general case. It may however be solved in an elegant way by noticing that it could be brought to the two previous cases. This is achieved by replacing the transition falling into this case by two transitions which, when fired concurrently, give the same global effect as our transition (see [BEM92] for more detail). In fact, this replacement shows how ECATNets allow to specify a given situation at two levels of abstraction.

- $[IC(p,t)] \varnothing [DT(p,t)] \neq \phi_M$

The form of the rule is given by:

$$t: (p, [DT(p,t)]_{\oplus} \cap [M(p)]_{\oplus}) \rightarrow (p',[CT(p',t)]_{\oplus}) \text{ if } ([IC(p,t)]_{\oplus} \setminus ([IC(p,t)]_{\oplus} \cap [M(p)]_{\oplus}) = \phi_M) \rightarrow [\text{false}]$$

In this situation, the condition consists to check that IC(p,t) is not included (totally) in p.

- $IC(p,t) = \phi_M$

The form of the rule is given by:

$$\text{t: } (p, [DT(p,t)]_{\oplus} \cap [M(p)]_{\oplus}) \rightarrow (p',[CT(p',t)]_{\oplus}) \text{ if } ([M(p)]_{\oplus} \rightarrow \phi_M)$$

When the place capacity C(p) is finite, the conditional part of the rewrite rule will include the following component:

$$([CT(p,t)]_{\oplus} \oplus [M(p)]_{\oplus}) \cap [C(p)]_{\oplus} \rightarrow [CT(p,t)]_{\oplus} \oplus [M(p)]_{\oplus} \quad \text{(Cap)}$$

In the case where there is a transition condition TC(t), the conditional part of our rewrite rule must contain the following component: [TC(t)] $\rightarrow$ true]

4. ECATNets as ACI-Theories

ECATNets have a straightforward and natural expression as labelled theories. Their distributed states correspond to *markings*, that is, to finite multi-sets of *structured data*, which are tuples of the form <place, token>. Tokens are axiomatized by algebraic data types. Algebraic markings are also axiomatized by an *associative* and *commutative* multi-set union operation $\otimes$ with identity the empty multi-set ϕ_B. In an ECATNet, a transition t is simply a labelled rewrite rule $t : M \rightarrow M'$ between two multi-set markings [**Bel02**].

Therefore, we can view an ECATNet *N* as a rewrite theory $\aleph$ with the above algebraic axiomatization for its markings and with one rewrite rule per transition. So that firing of a transition exactly corresponds to a rewriting modulo ACI with the corresponding rewrite rule.

Clearly, an ECATNet labelled rewrite theory is a static description of what an ECATNet can do. The deduction rules allow us to reason correctly about which general concurrent transitions are possible in a system satisfying such a description. The meaning of the theory should be given by a categorical model, noted $C_{\aleph}$ of its actual behaviour. This $C_{\aleph}$ construction is a particular case of the general construction of $\tau_{\Re}(X)$ in [**Mes95**][**Bel02**]. In addition, in this model, objects are represented by states (terms of sort markings), and the finite concurrent computations are described by proof-terms.

Definition 4. (ECATNet rewrite theory): Given the sets P (of places) and T (of transitions), an ECATNet rewrite theory is a labelled rewrite theory $\aleph = (\sum_{\aleph}, E_{\aleph}, L_{\aleph}, R_{\aleph})$, where $\sum_{0,\aleph} = P \cup \sum_{0,TOKENS}$, $\sum_{2,\aleph} = \{_ \otimes _, \langle _, _ \rangle\}$ with all other $\sum_{i,\aleph}$ empty, $E_{\aleph} = ACI_{\otimes} \cup ACI_{\oplus}$, $L_{\aleph} = T$ and $R_{\aleph}$ is the set of pairs whose first component is a transition (label) and the second component is a pair of terms of sort markings.

Let us consider the following parameterized example describing a circular railway divided into n sections from 0 to n-1. Two trains namely $train_1$ and $train_2$ can run on this circular railway, with the constraint that it never happens that both are running on the same or consecutive track sections. The ECATNet specification of the system is described below.

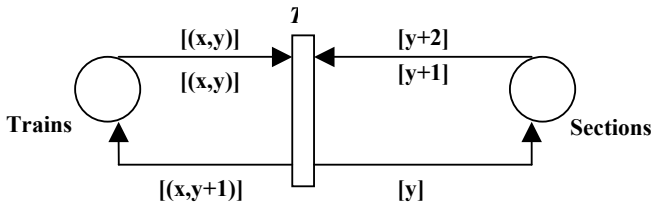


Figure 1. The ECATNet specification of a Circular Railway

The expression of the ECATNet in Figure 1 as a rewriting theory is then obvious. We can view each of the labelled arcs of the Petri net as a rewrite rule having a binary ACI operator $\otimes$ (multi-set union) so that rewriting happens modulo ACI. In the order-sorted version of rewriting logic, we can consider the "places" Trains and Sections as a type Place and view states of the net, usually called markings, as elements of a super-type marking containing place.

The rewrite theory corresponding to the ECATNet of Figure1 is given below in a like-Maude notation

Mod RAILWAY is

Extending TOKENS

Subsorts :

place, marking $\prec$ marking

ops :

Trains, Sections : $\rightarrow$ Place

ϕ : $\rightarrow$ marking

$\langle , \rangle$: place, tokens $\rightarrow$ marking

$\otimes$: marking marking $\rightarrow$ marking [ACI]

rule :

$t(x,y)$: $\langle Trains, [(x,y)] \rangle \otimes \langle Sections, \left[(y+1)_{\mathrm{mod}(n)} \right] \rangle \rightarrow \langle Trains, \left[(x,(y+1)_{\mathrm{mod}(n)}) \right] \rangle \otimes \langle Sections, [y] \rangle$

if $M(Sections) \cap [y+2]_{\mathrm{mod}(n)} \rightarrow [y+2]_{\mathrm{mod}(n)}$

endm

4.1 Behavioural aspects

The meaning of the theory should be given by computational models. Taking advantage of the correspondence between deductions in rewriting logic and *concurrent* computations in an ECATNet *N*, we naturally define the initial model $C_{\aleph}$ of $\aleph$. A categorical model gives the semantic interpretation to $\aleph$ in which objects are elements of $T_{\Sigma\aleph,E\aleph}$ and arrows are all possible sequents [s]$\rightarrow$[s'] inductively generated by the rewrite rule $R_{\aleph}$.

Definition 5. Given an ECATNet labelled rewrite theory $\aleph = (\sum_{\aleph}, E_{\aleph}, L_{\aleph}, R_{\aleph})$, its behavioural model is a category $C_{\aleph}$ whose objects are the elements of $T_{\sum_{\aleph}, E_{\aleph}}$ (equivalence classes of terms [t]

having the sort marking) and whose morphisms are proofs representing the one-step $\aleph$-concurrent - rewrites.

Traditionally, the construction of the category $C_{\aleph}$ is done through two steps:

(**a**) the construction of the algebraic structure $\wp_{\aleph}$ involving all the proof-terms generated through the following rules of deduction:

1. Identities

$\forall [t] \in T_{\Sigma,E}(X), \qquad \overline{[t]:[t] \to [t] \in C_{\aleph}}$

2. Σ-structure

$$\forall f \in \Sigma_n,\ \mathrm{n} \in \mathbb{N} \qquad \frac{[\alpha_1]:[t_1] \to [t_1'] \quad \ldots [\alpha_n]:[t_n] \to [t_n'] \in \aleph}{f[\alpha_1,\ldots\alpha_n]:[f(t_1,\ldots t_n)] \to [f(t_1',\ldots t_n')] \in C_{\aleph}}$$

3. Replacement

For each rewrite rule r : $[t(x_1,\ldots x_n)] \to [t'(x_1,\ldots x_n)] \in \aleph$

$$\frac{[w_1] \to [w_1'] \quad \ldots \quad [w_n] \to [w_n']}{r(\bar{w}) : [t(\bar{w}/\bar{x})] \to [t'(\bar{w}/\bar{x})] \in C_{\aleph}}$$

where $(\bar{w}/\bar{x})$ is a substitution of w_i by x_i, $1 \le \mathrm{i} \le \mathrm{n}$

(**b**) deduce the category $C_{\aleph}$ as the quotient of $\wp_{\aleph}$ modulo the following equations :

Identity : For each $\alpha : [t] \to [t']$, $\alpha; t' = \alpha$ and $t; \alpha = \alpha$

Associativity : $(\alpha;\beta);\gamma = \alpha;(\beta;\gamma)$

Functoriality : $(\alpha_1;\beta_1) \otimes (\alpha_2;\beta_2) = (\alpha_1 \otimes \alpha_2);(\beta_1 \otimes \beta_2)$

5. The Temporal Logic τLTL

The popular approaches to logical specifications are based either on states or actions. In a state-based approach, an execution of a system is viewed as a sequence of states, where a state is an assignment of values to some set of components. An action-based approach views an execution as a sequence of actions. These two approaches are, in some sense, equivalent. An action can be modelled as a state change and a state can be modelled as an equivalence class of sequences of actions [**Aba93**].

We adopt an action-based approach and use the term "action" which informally means one-step-rewrite. The logic τLTL is based on actions built-up from the set of labels $L_{\aleph}$ of a rewriting theory rather than on states. Given a rewrite theory the initial model has an underlying categorical structure given by the morphisms extending its one-step concurrent rewrites.

To define a logic τLTL, a tiny auxiliary logic of actions is introduced. The collection ACT of action formulae over $L_{\Sigma_\aleph} \cup T_{\Sigma_\aleph}$ is defined by the following grammar:

$$\lambda ::= t \mid \neg\lambda \mid \lambda \otimes \lambda'$$

We introduce a function $F^{\rightarrow} : Act \times T^{\Sigma_\aleph \cup L_\aleph} \rightarrow Bool$, which given an action formula and a proof-term determine if the action formula is a sub-term of a proof-term. The satisfaction of an action formula by a morphism $[\alpha] \in T_{\Sigma_\aleph \cup L_\aleph}$ is defined inductively by:

$$[\alpha] \models t \ \textit{iff} \ F^{\rightarrow}(\alpha, t) = true$$

$$[\alpha] \models \neg\lambda \ \textit{iff} \ not([\alpha] \models \lambda)$$

$$[\alpha] \models \lambda \otimes \lambda' \ \textit{iff} \ [\alpha] \models \lambda \ \text{and} \ [\alpha] \models \lambda'$$

Classical LTL provides in addition to the propositional logic operators the temporal operators defined by the following Maude specification. τLTL is an extension of an action-based temporal logic LTL. The set of basic propositions contains only two elements, namely propositions true and false.

The operators X and U are replaced by a relativised operators $X_{[\lambda]}$ and $U_{[\lambda]}$ where λ is an action formulae.

We define τLTL formulae in the following Maude module:

fmod τ-linear-tempral-logic extending propositional-calculus

$Op \ True : formula$

$Op \ \neg : formula \rightarrow formula$

$Op \ \wedge : formula \ formula \rightarrow formula$

$Op \ \vee : formula \ formula \rightarrow formula$

$Op \ (F _) : formula \rightarrow formula$

$Op \ (X_{[-]} _) : formula \rightarrow formula$

$Op \ (_ U_{[-]} _) : formula \ formula \rightarrow formula$

endfm

Let $\aleph = (\Sigma_\aleph, E_\aleph, L_\aleph, R_\aleph)$ be a rewrite theory denoting an ECATNet, with $(\Sigma_\aleph, E_\aleph)$ its underlying equational theory and let state be a sort in $\Sigma_\aleph$ such that, for each $[t] \in T_{\Sigma_\aleph, E_\aleph}$, *if* $[\alpha] : [t] \rightarrow [t']$ is a one-step $\aleph$-rewrite then $[t'] \in T_{\Sigma_\aleph, E_\aleph}$. We can associate to each rewrite theory $\aleph$ the category $C_\aleph = (T_{\Sigma_\aleph}, T_{\Sigma_\aleph \cup L_\aleph}, src(-), tgt(-))$.

The arrows in $C_\aleph$ are labelled by proof-terms corresponding to a one-step-sequential rewrite or a one step-concurrent rewrite (when many actions may be performed concurrently). A sequence

$[\beta]=[\alpha_1];[\alpha_2];...;[\alpha_n]$, a composition of morphisms in $C_{\aleph}$ where $[\alpha_i]\in T_{\Sigma_{\aleph}\cup L_{\aleph}}$ is called a path and correspond to a classical notion of paths over a Kripke structure. This composition is possible *iff* $tgt([\alpha_i])=src([\alpha_{i+1}])$.

Satisfaction of a formula ϕ by a path in $C_{\aleph}$ is given by the following Maude module:

fmod *proof − semantics is extending* $\tau-linear-temporal-logic$

protecting path

Op $_\models_$: *Path formula* $\rightarrow$ *Bool*

var $\alpha: T_{\Sigma_{\aleph}\cup L_{\aleph}}$ *var* β: *Path var* ϕ ψ: *Formula*

eq $\beta \models true = true$

eq $\beta \models \neg\phi = \neg(\beta \models \phi)$

eq $\beta \models \phi\wedge\psi = \beta \models \phi$ *and* $\beta \models \psi$

eq $\beta \models \phi\vee\psi = \beta \models \phi$ *or* $\beta \models \psi$

eq $\alpha;\beta \models F\phi = \alpha;\beta \models \phi$ *or* $\beta \models F\phi$

eq $\alpha;\beta \models X_{[\lambda]}\phi = [\alpha] \models \lambda$ *and* $\beta \models \phi$

eq $\alpha;\beta \models \phi U_{[\lambda]}\psi = ([\alpha]\models\lambda$ *and* $\beta\models\psi)$ *or* $(\alpha;\beta\models\phi$ and $\beta\models\phi U_{[\lambda]}\psi)$

endfm

The meaning of τLTL operators can be stated as follows : τLTL formula $X_{[\lambda]}$ φ is valid iff there exists a morphism $[\alpha]$ such that src($[\alpha]$) = t_0 (the initial state), λ is sub-term of it and φ is valid in state t= tgt ($[\alpha]$). τLTL formula $\phi U_{[\lambda]}\psi$ is valid in state *t* iff there is a path starting in *t* where ϕ is valid until we perform a morphism γ such that λ is a sub-term of it ($F^{\rightarrow}(\gamma,\lambda)=true$) and ψ is valid in state t= tgt ($[\gamma]$).

5.1 A simple example

Let us reconsider the example of figure1 which describe a circular railway, in which n=6 sections. Two trains namely $train_1$ and $train_2$, can run on this circular railway, with the constraint that it never happens that both are running on the same track section. The rewrite theory is slightly different from the theory of Figure 1, since we permit that train1 and train2 may be in consecutive track sections. The input arc (sections,T) is then labelled by [y+1] and [y+1] which correspond to the case where [IC(p,t)] =[DT(p,t)] and then the rule is unconditional.

Mod *RAILWAY* is
Extending TOKENS
Subsorts :
place, marking $\prec$ marking
ops :
Trains, Sections : $\rightarrow$ Place

ϕ : $\rightarrow$ marking

$\langle , \rangle$: place, tokens $\rightarrow$ marking

$\otimes$: marking marking $\rightarrow$ marking [ACI]

rule :

$$t(x,y):\langle Trains,\lfloor (x,y)\rfloor\rangle \otimes \langle Sections,\lfloor (y+1)_{\mathrm{mod}(n)}\rfloor\rangle \rightarrow \langle Trains,\lfloor (x,(y+1)_{\mathrm{mod}(n)})\rfloor\rangle \otimes \langle Sections,\lfloor y\rfloor\rangle$$

endm

From the rewrite theory Railway and applying the one-step concurrent rewrites, we can built the category $C_{\mathfrak{R}}$,shown in figure 3 seen as a graph.

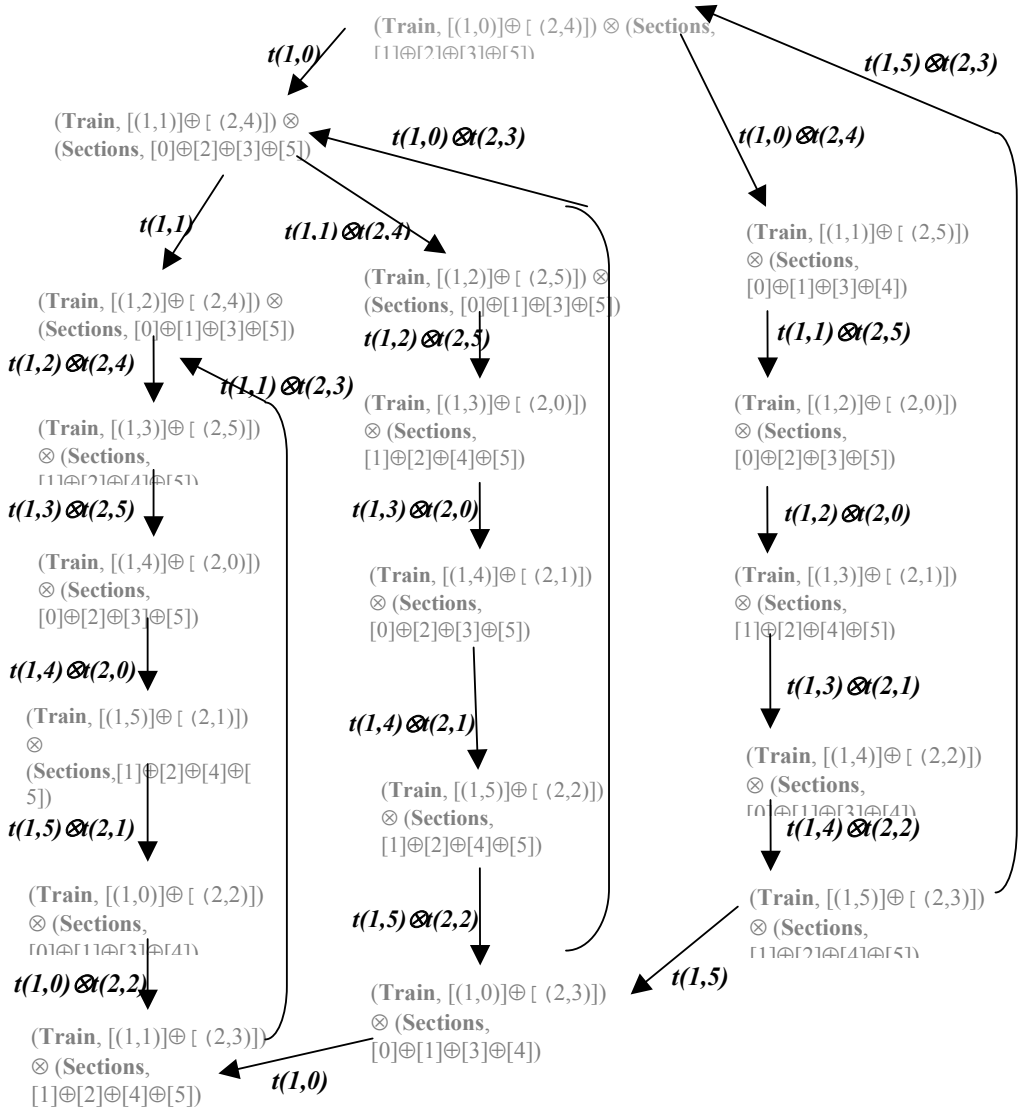


Figure 3. the category $C_{\mathfrak{R}}$

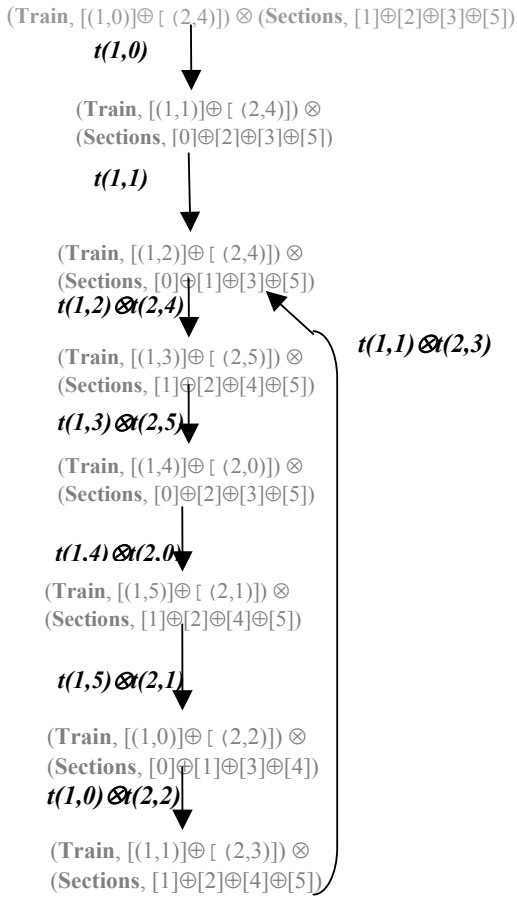
Consider that we have to check that never both trains are running on the same section. For instance, if there is the possibility that $train_1$ moves into $section_1$ and $train_2$ moves into $section_5$, the $train_2$ must wait that $train_1$ moves to $section_2$.

$$X_{[t(1,1)\otimes t(2,5)]}true \Rightarrow (\neg X_{[t(2,0]}true)U_{[t(1,2)]}true)$$

If the situation where $train_1$ is in $section_0$ and $train_2$ is $section_4$ represent the initial state we also may expect, that for any run $train_1$ and $train_2$ must return to the initial situation. The following τLTL formula express the desired requirement :

$$GF(X_{[t(1,1)\otimes t(2,5)]}true)$$

When considering the following path, the formula above is *false.*



5.2 On the Deadlock detection

Usually, the first step in the verification process is to abstract the actual system to some automata based model. Reducing, in a classical way, concurrent computation to all possible linearizations of concurrent behaviours leads us to a well-known Kripke structure.

Let $\aleph=(\sum_{\aleph}, E_{\aleph}, L_{\aleph}, R_{\aleph})$ be a rewrite theory denoting an ECATNet, with $(\sum_{\aleph}, E_{\aleph})$ its underlying equational theory and let state be a sort in $\sum_{\aleph}$ such that, for each $[t] \in T_{\sum_{\aleph}, E_{\aleph}}$, *if* $[\alpha]:[t] \rightarrow [t']$ is a one-step sequential rewrite then $[t'] \in T_{\sum_{\aleph}, E_{\aleph}}$. We can associate to each rewrite theory $\aleph$ a Kripke structure $K(\aleph)=(T_{\sum_{\aleph}, E_{\aleph}}, \rightarrow_{\aleph}, L_{\aleph})$ where $\rightarrow_{\aleph}$ is the one-step sequential $\aleph$-rewrites on $T_{\sum_{\aleph}, E_{\aleph}}$.

From the situation when $train_1$ is in $section_3$ and $train_2$ is in $section_5$, both trains can progress in parallel. This, in the spirit of an "interleaving" semantics, is represented by two sequences of one-step sequential $\aleph$-rewrites. The first one, express that train1 moves to $section_4$ and then $train_2$ moves to $section_0$.

In the partial order techniques[**Wil96**], it is well proved that deadlock detection problem can be handled by considering a limited number of interleaving of independent concurrent actions. In our approach, we show that all deadlock states are also preserved in the category $C_{\aleph}$ generated by a one step-concurrent $\aleph$-rewrite, thus the detection of deadlock states can be checked in an optimized manner since all the states associated to the interleaving sequences are avoided.

Indeed when adopting a "truly concurrent" approach, the one step-concurrent $\aleph$-rewrite, $[t(x,y)] \otimes [t(x',y')]$ is possible because that there is no conflict neither no causality between $[t(x,y)]$ and $[t(x',y')]$. Thus from an intermediate state, after performing $[t(x,y)]$ (resp. $[t(x',y')]$), we always can perform $[t(x',y')]$ (resp. $[t(x,y)]$). The intermediate states cannot be deadlock states. A first result is that, all deadlock states of $K(\aleph)$ also appear in the category $C_{\aleph}$.

6. Conclusion

In this paper, we had shown how the use of ECATNet as a formalism of specification having a well-defined formal semantics, which in an unambiguous way define the behaviour of the specified system in a very compact way.

In addition, we introduced an adequate temporal logic to express properties of concurrent systems modelled as theories of rewriting logic denoting ECATNets and show how some generic properties such as deadlock detection can be verified. The definition of a " model-checking " tool is a work under development.

References

Aba93 M. Abadi and L Lamport. *Composing Specification*. ACM Transactions on Programming Languages and Systems, Vol 15, N° 1, January 1993, Pages 73-132.

Bel02 F. Belala : *Un Cadre formel pour l'Analyse de ECATNets*. Thesis, University of Constantine, Algeria, 2002.

Bem93 M. Bettaz, M. Maouche: *How to Specify Non Determinism and True Concurrency with Algebraic Term Nets*, Lecture Notes in Computer Science, 655, pp. 164-180, Springer-Verlag, 1993.

Den98 G. Denker, J. Meseguer and C. Talcott. *Protocol Specification and Analysis in Maude*. In N. Heintze and J.Wing, editors, Proc. Of Workshop on Formal Methods and Security Protocols, 25 June 1998, Indianapolis, 1998.

Man89 Z. Manna, A. pnueli: *The Anchored Version of the Temporal framework, in Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*. Lecture Notes in Computer Science 354, Springer-Verlag, 1989, pp. 201-284.

Mar92 N. Marti-oliet, J. Meseguer : *Rewriting Logic as a Semantic framework*. Technical Report SRI-CSL, Menlo Park, CA 94025 and center for the study of language and Information, Stanford University, Stanford, CA 94035, 1992.

Mes90 J. Meseguer : *Rewriting as a Unified Model of Concurrency*. CONCUR 1990: 384- 400.

Mes95 J. Meseguer, N. Marti-Oliet: *From Abstract Data Types to Logical framework*, in E. Astesiano and G. Reggio (eds), Recent Trends in Data Type Specification, Lecture Notes in Computer Science vol 906, Springer-Verlag, 1995.

Mes96 J. Meseguer: *Rewriting Logic as a Semantic framework of Concurrency: a Progress Report,* Springer-Verlag, Lecture Notes in Computer Science, 1119, pp.. 331-372, 1996.

Rib02 P. Ribet, F. Vernadat, B. Berthomieu : *On Combining the persistent Sets Method with the Covering Steps Graph Method,* in Proc. Formal Techniques for Networked and Distributed Systems FORTE2002, Houston (Texas), November 2002.

Wil96 B. Willems, P. Wolper. *Partial-order methods for model checking: From linear time to branching time*. In Proc. 11th Symp. on Logic in Computer Science, pages 294--303, New Brunswick, July 1996.