Abstract

CATNets\cite{3} are a general-purpose semi-graphical formalism for specifying concurrent objects, which has already been defined in \cite{3}. In this paper we propose an extension to this formalism (ECATNets) aiming to increase their descriptive power. ECATNets are given interpretation in terms of a rewrite logic fitting into the same semantic framework as CATNets.

The practical usefulness of ECATNets is shown thru their application to a series of real-size communication protocols of the OSI environment. These studies let appear how ECATNets may be used to achieve modular specifications, which may be easily put together. Our specification approach is based on starting from implementation concepts rather than from theoretical ones, thus bridging the gap between the theory and the practice.

Keywords and phrases: High-level Petri nets, communication protocols, true concurrency, rewrite logic, abstract data types.

1. INTRODUCTION

ECATNets are a formalism motivated by the need of practical applications. They are based on CATNets, a general-purpose semi-graphical formalism for specifying concurrent objects. The formal definition of CATNets together with their semantics are given in \cite{3}. Before giving the definition of ECATNets, let us recall briefly some basic information about CATNets. CATNets are a form of high-level net/abstract data model which combines the strength of Petri nets with that of abstract data types. Petri nets are used for their foundation in concurrency and dynamics, while abstract data types are used for their data abstraction power and solid mathematical foundation. Moreover both formalisms are able to be processed by machine. Similar models are the OBJSA Nets \cite{1}, the Many-Sorted High-Level Nets \cite{6}, the Algebraic High-Level Nets with Capacities \cite{9}, the Petri Nets and Algebraic Specifications \cite{13}, and many others. Billington's work is perhaps closest to ours, since his work was motivated by a background in protocols and Numerical Petri Nets \cite{5}. However our work may be considered as generalizing the work in \cite{6}, since we are using abstract data types with axioms, while this is not the case in Billington's work. Moreover the inhibitor are concept defined in Billington's work and missing in our work on CATNets, has actually its equivalent in ECATNets.

CATNets are given semantics thru their interpretation in terms of a rewrite logic inspired from \cite{11}. The proposed logic acts as an axiomatisation allowing us to study the concurrent behavior of CATNets by deduction in such a logic. The rewrite logic consists of a set of axioms and a set of deduction rules. The axioms are rewrite rules describing transitions effects as elementary types of changes. The deduction rules allow us to draw valid conclusions about the evolution of the CATNet from these changes.

The objective of this paper is twofold.

First, we present the new syntactic notations and generalize the rewrite-theory defined for CATNets to deal with the finite capacity and Transition Condition cases.

Second we show thru a representative protocol action example the ability of ECATNets to specify, in a modular way, real-size OSI protocols. Our decomposition of the protocol into actions (specified by ECATNet modules) is motivated by a separation of concern strategy applied to each protocol entity, thus emphasizing the divide and conquer strategy used by the OSI world.

The paper is organized in the following way. In section 2 we give the definition of ECATNets. In section 3 we present the semantics of ECATNets thru their interpretation in terms of rewrite logic. In section 4, we show the applicability of ECATNets to OSI protocols. In section 5 we end by giving some concluding remarks.

2. ECATNets

Let us start this section by recalling some basic notions and notations from the field of algebraic specifications \cite{8}. A signature \text{SIG} = (\text{S, OP}) consists of a set \text{S}, the set of sorts and a set \text{OP}, the set of operation symbols together with their declaration which specifies the names of the domain and co-domain of each of the operation symbols. A SIG-algebra \text{A} is a pair \text{A} = (\text{SA}, \text{OP}_A) where \text{SA} = (\text{A}_i)_{i \in S} is the set of carriers and \text{OP}_A = (\text{op}_A)_{\text{op} \in \text{OP}} is the set of corresponding functions. When there is no ambiguity the name of an algebra will be used to denote the algebra itself as well as its set of carriers. We denote by \text{T}_{\text{SIG}}(\text{X}) the SIG-algebra of SIG-terms with variables in an \text{S}-sorted set \text{X}. Similarly, given a set \text{E} of SIG(\text{X})-equations, \text{T}_{\text{SIG}}(\text{X}) denotes the SIG-algebra of equivalence classes of SIG-terms with vari-
ables in $X$ modulo the equations $E$. We let $[x]_E$ or just $[x]$ denote the $E$-equivalence class of $x$. $MT_{S_0,E}(X)$ denotes the many-sorted free commutative monoid over $T_{S_0,E}(X)$ with $0$ the usual internal operation, and $\Phi_M$ the identity element. $MT_{S_0,E}(X)$ denotes the structure of equivalence classes of expressions of $MT_{S_0,E}(X)$ modulo the associative, commutative and identity (ACI) axioms of $E$. We let $[x]_E$ denote the equivalence class of $x$, w.r.t. to the ACI axioms for $E$. CATNets were defined on an algebraic structure, denoted $\text{CATNas}(X)$ and called syntactic structure [3]. In this section we just present the newly introduced extensions.

Definition 2.1 (CATNets syntactic structure)

Let $\text{CATNas}(X)$ be a CATNet syntactic structure. The CATNets syntactic structure denoted by $\text{CATNas}(X)^*$, is defined inductively as follows.

- $\text{CATNas}(X) \subseteq \text{CATNas}(X)^*$
- $\emptyset \in \text{CATNas}(X)^*$
- If $[m]_E \in \text{CATNas}(X)$ then $\neg [m]_E \in \text{CATNas}(X)^*$
- $\text{CATNas}(X)^*$ will be called the CATNet syntactic substructure.

Definition 2.2 (CATNet)

A CATNet is a structure $(P,T,IC,DT,CT,C,TC)$ where

- $P$ is a set of places and $T$ is a set of transitions;
- $\forall P \rightarrow S$ is a function that associates a sort with each place;
- $IC: (P \times T) \rightarrow \text{CATNas}(X)^*$ is a partial function such that for every $(p,t) \in \text{domain}(IC)$, if $IC(p,t) \in \text{CATNas}(X)$ then $IC(p,t) = \text{CATNas}(X)_p(t)$, else if $IC(p,t) \neq \emptyset$ then $\exists [m]_E \in \text{CATNas}(X)^*$ such that $IC(p,t) = \neg [m]_E$
- $DT: (P \times T) \rightarrow \text{CATNas}(X)$ is a function such that for every $(p,t) \in (P \times T)$, $DT(p,t) \in \text{CATNas}(X)_p(t)$;
- $CT: (P \times T) \rightarrow \text{CATNas}(X)$ is a function such that for every $(p,t) \in (P \times T)$, $CT(p,t) \in \text{CATNas}(X)_p(t)$;
- $C: P \rightarrow \text{CATNas}(X)\emptyset$ is a partial function such that for every $p \in \text{domain}(C)$, $C(p) \in \text{CATNas}(X)\emptyset$;
- $TC: T \rightarrow \text{CATNas}(X)\emptyset$ is a function such that for every $t \in T TC(t) \in \text{CATNas}(X)\emptyset$ where $X(t)$ is the set of variables occurring in $IC(p,t)$ (when defined), $DT(p,t)$ and $CT(p,t)$ for every $p \in P$. $X(t)$ will be called the transition context.

Definition 2.3 (marked CATNet)

A marked CATNet is an CATNet with a function $M: P \rightarrow \text{CATNas}(X)\emptyset$ such that for every $p \in P$, $M(p) \in \text{CATNas}(X)\emptyset$ and $M(p) \in C(p)$ if $p \in \text{domain}(C)$.

Comments

- $IC$, $DT$, and $CT$ are respectively the Input Condition, the Destroyed Tokens, and the Created Tokens. The extensions are related only to $IC$, and may be considered as an equivalent of the arc inhibitor concept as defined in [6].
- $C$ is a capacity (partial) function and $TC$ is a Transition Condition.
- The marking of an ECATNet is defined w.r.t. the capacity. When a place does not belong to $\text{domain}(C)$ it will be considered as a place with an infinite capacity.

3. CATNets SEMANTICS

In this section we will recall briefly the CATNets semantics, while putting emphasis on the semantics associated with the extensions introduced in the previous section. 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 enabled. The second condition is that $TC(t)$ is true. Finally the addition of $CT(p,t)$ to each output place $p$ must not result in $p$ exceeding its capacity when this capacity is finite. 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 $p$. Let us now precise what we mean by $IC(p,t)$ being enabled. Actually this enabling depends strongly on the syntactic notation used for representing $IC(p,t)$:

- Notation 1 $IC(p,t) = [m]_E$ with $[m]_E \in \text{CATNas}(X)_p(t)$; the enabling in this case means $[m]_E \in M(p)$. This is in fact the only enabling situation defined for CATNets (without extensions).
- Notation 2 $IC(p,t) = \neg [m]_E$ with $[m]_E \in \text{CATNas}(X)_p(t)$; the enabling in this case means $[m]_E \subset M(p)$.
- Notation 3 $IC(p,t) = \emptyset$; the enabling in this case means $M(p) = \Phi_M$. The last two notations are proper to ECATNets.

Definition 3.1 (CATNet rewrite theory)

Let $T$ be the set of transitions of a given CATNet, $\text{CATNas}(X)$ its syntactic substructure, and $B\Phi_{TS_0,E}(X)$ its semantic structure. An ECATNet rewrite theory is a set of triples $R \subseteq T \times (B\Phi_{TS_0,E}(X))^2 \times \text{CATNas}(X)_\text{boot}$. The elements of $R$ are called rewrite rules. A rewrite rule will be denoted by $t: U \rightarrow V$ if $\text{boolexp}; U, V \in B\Phi_{TS_0,E}(X)$; $\text{boolexp}$ is a logical conjunction of pairs of the form $ui \rightarrow vi$; $ui, vi \in \text{CATNas}(X)$.

Given a rewrite theory $R$, we say that $R$ entails a sequence $[|s_0|] \rightarrow [|s_1|] \rightarrow \cdots \rightarrow [|s_n|]$ where $[|s_0|], [|s_1|] \in (B\Phi_{TS_0,E}(X))^2$ is a pair of states, if $[|s_0|] \rightarrow [|s_1|]$ can be obtained by finite (and concurrent) applications of the following rules of deduction: Reflexivity, Congruence, Replacement [11], Splitting, Recombination, and Identity [3]. Let us in the following briefly comment the meaning of each one of these rules. The Reflexivity rule says that everything may be transformed into itself. The Congruence rule says that basic changes have to be correctly propagated. The Replacement rule is used when the context of a transition is not empty. The Splitting and Recombination rules allow us, by judiciously splitting and recombining different multisets of $E$-equivalence classes of terms to detect ECATNet computations exhibiting a maximum of parallelism. The Identity rule allows us to relate the identity element of the syntactic structure with the identity element of the semantic structure of an ECATNet.
which, given a state \( [s] \), seeks for all possible basic changes (the axioms) and tries to compose them (the deduction rules) in such a way to reach the next state \( [[t]g] \) corresponding to the intended concurrent computation of the net. The rules of deduction indicate that concurrent rewriting is performed modulo the intended concurrent computation of the net. The rules of deduction also allow free rewriting from syntactic constraints of a term representation (rewriting modulo \( E \)) and to deal with the concurrent behavior of the ECATNets (rewriting modulo \( ACI \)). In the following we will show and motivate how to associate a rewrite theory (i.e. the axioms) with a given ECATNet. A transition effect is represented by a rewrite rule, the general form of which is given by definition 3.1. The enabling aspect of the transition may be represented in two ways: in the left-hand side of the rewrite rule or in its conditional part. The first way is best suited when the syntax of IC\( (p,t) \) is given according to notation1. This choice was motivated and treated in [3]. The second way is necessary when the syntax of IC\( (p,t) \) is given according to notation2 and notation3. Let us mention that this second way may also be used to deal with finite capacities as well as with transition conditions. The firing aspect of the transition will be represented by a proof allowing us to deduce an ECATNet state from a previous one, by using our rewrite rule as an axiom and anyone of the deduction rules of our rewrite logic. A concurrent firing is thus represented by concurrent proofs. These proofs have to be composed in such a way to capture correctly the intended ECATNet behavior. In order to enhance the readability, we will start by treating the simplest case, i.e. an ECATNet consisting of one transition, one input place, one output place, and with arcs inscribed by closed multisets. In the following we will proceed by treating each notation separately. The symbols \( (\cap, \cup, \setminus, =) \) will be used with a multiset-like interpretation.

a) Form of the axioms for notation2

\[ ([p_0], DT(p_0) \cap M(p_0))g_0 \rightarrow [[p_0], CT(p_0))g_0] \]
\[ \Phi(t_0) \rightarrow \Phi(t_0) \]

When the place capacity \( C(p_0) \) is finite the conditional part of our rewrite rule will include the following component:

\[ [[p_0], CT(p_0))g_0 \rightarrow [[p_0], CT(p_0))g_0] \]

In the case where there is a transition condition \( TC(t) \) the conditional part of our rewrite rule will include the following component: \( [TC(t)] \rightarrow [true] \).

Comment: The form of an axiom corresponding to a transition with one input place and more than one output place is given in [3]; if one or more output place(s) has a finite capacity, the conditional part of the rewrite rule has to include a component of the form denoted by \( (Cap) \) for each one of these places. Then the situation where a transition has more than one input place may be considered as a composition-like of different basic ones.

4. USING ECATNets FOR SPECIFYING COMMUNICATION PROTOCOLS

Real computer communication protocols are complex objects. To overcome this complexity, ISO suggests, through its standard for Open System Interconnection, a layered model based on the divide and conquer strategy. In terms of protocol specification, this strategy allows us to replace a protocol specification by the specifications of each of its layers [10]. The different layers enclose however many similar functions such as data fragmentation and encapsulation, addressing, multiplexing, flow and error control. It may then be useful to provide the designer with a library of the basic specifications of such functions, so that he may concentrate more on the specific functions of the designed layer. Using of ECATNets allow us to specify a given protocol by decomposing it into a set of different standard actions; each of them being specified by an ECATNet module regrouping one or more protocol functions. Once given the specifications of the different actions, the specification of the protocol itself, may be obtained by a composition on places of the specifications of these actions. From the semantical point of view this means that the rules to associate with a given protocol are obtained merely by putting together the rules associated with its different actions. The different aspects of this approach are detailed in a series of case studies devoted to different real size (and complexity) OSI protocols. Let us in the following give a simple example to illustrate some aspects of our approach. The example is treating one representative action of a data-link layer protocol which may appear, for lack of context (space), far from complete. Before presenting the specification of this action, let us declare the used algebraic specifications.

The algebraic specifications involved in the ECATNet syntactic notations are in reality simple, and most of them are often available as basic library standards. In our case study we will suppose predefined the following type: \( mti \) (the integers mod \( 2^w + 1 \), with \( addw \) and \( subw \), as addition and substraction mod \( 2^w + 1 \) respectively), naturals and booleans. The others types may be specified in the following way, where an OBJ-like notation is used.

**OBJ I-FRAME / INTW TYPE-I LSDU**

- **sorts** I-frame
- **ops** \( r, \ldots \rightarrow \) : type-I intw lsdv \( \rightarrow \) I-frame
- **JB0**
- **OBJ WINDOW-EDGE / INTW**
  - **sorts** window-edge
  - **ops** \( \wedge \rightarrow \) : intw intw \( \rightarrow \) window-edge
  - **lwe_** : window-edge \( \rightarrow \) intw
  - **uwe_** : window-edge \( \rightarrow \) intw
  - **vars** \( I, J : \) intw
  - **eqns** \( \wedge \leq \) \( I, J \) = \( I \)
  - **uwe_ < \wedge \rightarrow \rightarrow \) \( (I, J) \rightarrow \) \( w \rightarrow \) else \( w \)
I-frame sending) by first presenting its ECATNet model, then giving a brief informal interpretation, finally rewrite rules.

Informal comment

The user requests an LSDU transmission by passing it to the sender through the access point represented by FROM_USER. P constructs an I-frame from the user supplied LSDU by appending it to the frame type DT, and a sender sequence number (available in VS), then the V(S) is incremented and the I-frame is sent, provided that the send-window and the transmission medium capacity will not be exceeded. At the same time a timer is started to it the frame.

Rewrite rules

DT-FRAMING: (FROM_USER,(DATA)) ⊗ (VS,[NS]) → (VS,[addw(NS,1)]) ⊗ (DT_TO_SEND,(<DT,NS,DATA>))

DT-SENDING: [(DT_TO_SEND, (<DT,NS,DATA>)) ⊗ (P_WINDOW-EDGE, φ₂) ⊗ (P_WINDOW-EDGE, [<LWEp, UWEpité)]) ⊗ (P-T_0, [TO]) ⊗ (P-T_0,φ₂) --> (P_WINDOW-EDGE, [<LWEp, UWEpité)]) ⊗ (R_LIST, (<DT,NS,DATA>)) ⊗ (TIMER, ([<TO,NS>]) ⊗ (MSG_Q, [<DT,NS,DATA>])) ⊗ (P-T_0, [TO]) # [[[NS > = LWEp] and (NS < UWEp)]) → [true]] and ([[<DT,NS,DATA>]) ⊗ M(MSG_Q)) ∩ Q(MSG_Q)]⊳ → [[[<DT,NS,DATA>]) ⊗ M(MSG_Q)]⊳]

Figure. I-frame sending action

5. CONCLUSION

In this paper we proposed an extension to CATNets (ECATNets) in order to increase their descriptive power, while preserving the semantic framework already set for CATNets. The practical use of our formalism was shown thru their application to a series of case studies treating real-size communication protocols in the OSI environment. Let us remark that the used semantic framework allows us to consider the studied protocol as concurrent complex objects rather than sequential ones as it seems to appear from the OSI standard documents. It is worthwhile to mention that this paper is primarily devoted to the descriptive aspects of the proposed formalism rather than to its proof ones (i.e. theorems and automatic analysis methods). Let us however remark that at the present time these problems are partially being investigated [7].

Acknowledgements: This work is supported by a grant from the Algerian Ministry of Higher Education under research contract number B2501/01/02/91.

6. REFERENCES


