Verification of Signaling Protocols for Telecommunication Services in Intelligent Network

Bruno Blašković, Ignac Lovrek
Department of Telecommunications
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, HR-10000 Zagreb, Croatia
bruno.blaskovic@fer.hr

Abstract

The paper deals with verification of signaling protocols for telecommunication services. The model of signaling protocol synthesis based on elementary communicating processes and information flows they exchange is proposed. Elementary communicating processes represent basic service building blocks and exchange information by using process protocols. Service protocol is produced as a composition of process protocols for elementary processes involved. Verification problems related to elementary process protocols and service consistency are discussed. Some examples describing verification of signaling protocols for intelligent network services are included.

1 Introduction

Signaling protocols based on common channel signaling concepts are used to support calls and services in telecommunications. There are two distinct points that make difference between call and service signaling protocols and related design problems as well. The first one is the number of functional entities involved in call and service handling. Beside originating and terminating entities relevant for calls, additional switching, control and resource entities are involved in service handling, so multiple signaling relations are used. The second point are models of call and service processing themselves. Basically, calls are supported by single building blocks on the originating and terminating side. They exchange information by using complex monolithic signaling protocols for establishing, maintaining and releasing the call. On the other hand, services in an intelligent network are implemented by interconnecting different service building blocks. A segment of the signaling protocol is defined for each such block. It provides interchange of signaling information between different functional entities in the network. A connection pattern describing interconnection of service building blocks for a specific service is referred to as a service logic.

The basic idea regarding specification and verification of signaling protocols for telecommunication services is (i) to define elementary communicating processes representing basic service building blocks and their signaling protocols, and (ii) to define service description which allows composition of the partial process protocols into the overall service protocol. The approach exercised in this paper can be stated as follows: "generate signaling protocol from consistent service description". Consistent service description includes validated requirements for intelligent network protocol and verified service description is a base for service protocol synthesis. Verification problems arise at both levels, i.e. elementary communicating processes and their protocols as well as service description have to be checked.

Section 2 describes signaling protocols in intelligent network while Section 3 introduces an operational model of signaling protocol synthesis. Verification of signaling protocol, with some examples describing methods and tools used is given in Section 4. Conclusions include some future research directions.

2 Signaling in intelligent networks

The main difference between traditional and intelligent network regarding signaling comes from service creation, implementation and handling concepts [3].

Two types of functional entities are used for call switching: Call Control Agent Functions (CCAF) and Call Control Functions (CCF). Solutions like Telephone User Part (TUP) and Integrated Services User Part (ISUP) provide monolithic signaling protocols between originating and terminating CCAF/CCF entities used on a per call basis (Figure 1). Such entities are sufficient to describe traditional networks.

Intelligent network architectures introduce new functional entities like Service Switching Functions (SSF), Service Control Functions (SCF), Special Resource Functions (SRF), and Service Data Functions

(SDF). Signaling information is now exchanged between these entities by using Transaction Capabilities Application Part (TCAP) and Intelligent Network Application Protocol (INAP). Functional entities are implemented as application service elements (FE(INAP)), based on INAP recommendations. Lower layer protocols are the same in both cases, i.e. Message Transfer Part (MTP) and Signaling Connection Control Part (SCCP) (Figure 2).

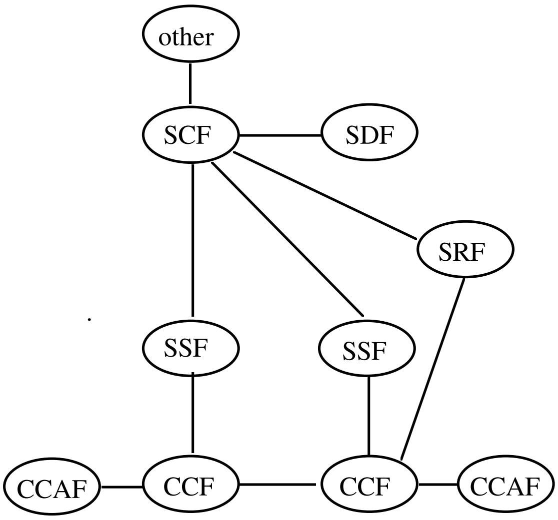

Figure 1: Functional entities and signaling relations in intelligent networks

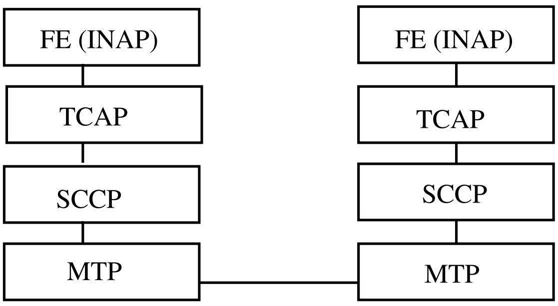

Figure 2: Common channel signaling in intelligent networks

One of the most important intelligent network concepts is service building by using various reusable service-independent blocks and connecting them according to a service logic. That reflects on signaling protocols, too. Instead of one complex protocol assuring all information needed for call processing, a number of transactions must be performed in order to collect and exchange information needed to support the service.

Service design must follow this general principle, therefore an operational model for synthesis of signaling protocols is proposed and an experimental prototype developed.

3 Synthesis of signaling protocols for telecommunication services

Operational model for signaling protocol synthesis consists of the following steps:

(i) definition of information elements and information flows,
(ii) definition of elementary communicating processes,
(iii) specification and verification of protocols for elementary communicating processes,
(iv) description of a telecommunication service,
(v) verification of telecommunication service consistency,
(vi) composition of elementary communicating processes protocols into the service protocol.

Operational model is designed mainly for services in intelligent network environment, although it is possible to extend the model to some other classes of distributed processes.

3.1 Information elements and information flows

An information element (InE) represents a unit of information to be exchanged between network entities in order to support call and/or service handling (e.g. digits 0-9, AddressComplete, DatabaseID, ...). Information elements are assembled into information flows (InF) if they can be exchanged as a unique information unit (e.g. InF Query consists of four information elements, DatabaseID, InformationKey, RequestedInfoType, RequestedInfo). All InEs and InFs for a specific service or a set of services under consideration are defined by using BNF notation as follows:

IE =: (InE_1, InE_2, ..., InE_i, ..., InE_n)
IF =: (InF_1, InF_2, ..., InF_i, ..., InF_m)
<InF>::= <InE>+

Information elements can be derived from three different starting points:

- elementary events from call and service processing,
- entities described through abstract data types (ASN.1, ACT ONE or similar),
- data structure.

As a small example, some events like fite (Forward Interworking Telephone Events), bite (Backward Interworking Telephone Events), and spite (Switching Processing Interface Telephone Events) are listed in Figure 3.

3.2 Elementary communicating processes

Elementary communicating process (ECP) is the basic block used for service description. Each service consists of several chains of ECPs.

```
fite(1,”Digit 1,2... or 0, code 11 or 12, end-of-pulsing (ST) signal”,1,1,1,1,1,1,1).
fite(37,”Call modification request (CMR)”,0,0,0,0,0,0,32).
fite(38,”Call modification completed (CMC)”, 0,0,0,0,0,0,33).
fite(A,”Call Set-up Information”,0,0,0,0,0,0,A).

bite(1,”spare”,0,0,0,0,0,0,0).
bite(2,”Address-complete,charge”,1,0,1,1,0,2,1).
bite(8,”Call unsuccessful”, 2,1,0,0,0,0,0).
bite(24,”Answer, re-answer”,0,0,18,18,0,0,0).
bite(25,”Clear-back”,4,3,19,19,2,12,0).

spite(1,”Activate reg. function (physical reg. or eq. funct.)”,”Task”).
spite(2,”Register function activated”,”Internal input”).
spite(3,”Deactivate register function”,”Task”).
spite(4,”Set up speech condition”,”Task”).
```

Figure 3: fite, bite and spite examples

ECP is introduced as an atomic, communication-oriented part of a service. ECP is defined as a quintuple:

ECP = (sInF, BaP, Cf, Df, p(ECP))

where:

- sInF (set InF) denotes a set of information flows interchanged between corresponding ECPs,
- BaP (Basic Process) describes "non-communication" part of service operations covered by ECP,
- Cf (Control flow) stands for a flow of control data which initiate start and termination of ECP,
- Df (Data flow) is flow of data to be exchanged between ECPs and
- p(ECP) is the protocol of ECP.

A set of elementary communicating processes allows all services to be constructed.

ECP generic model (Figure 4) comprises a set of information flows conveying data to be exchanged with other ECPs, as follows:

sInF = (<sd, SET>*, <sd, NOSET>*, <dd_in>*, <dd_out>*, <cf_in>+, <cf_out>+)

where:

<sd, SET>*	static data defined in some other ECP,
<sd, NOSET>*	static data not defined in some other ECP,
<dd_in>*	input dynamic data,
<dd_out>*	output dynamic data,
<cf_in>+	input control data,
<cf_out>+	output control data.

Static data are service dependent; they are common for all service instances. Dynamic data are service instance dependent; they are different for each ECP instance. They describe actual service usage. Some static and dynamic data goes implicitly with control data. Control data define logical start/termination of ECP.

ECP is defined through facts in production rules. Production rules supervise corespondent ECP as well as logical consistency during ECP definition phase.

Different service building blocks used in the intelligent network conceptual model may be mapped to elementary communicating processes, for instance service independent building block (SIB) from the intelligent network capability set 1 or High-level SIB, Process, Action, from the intelligent network capability set 2 [12].

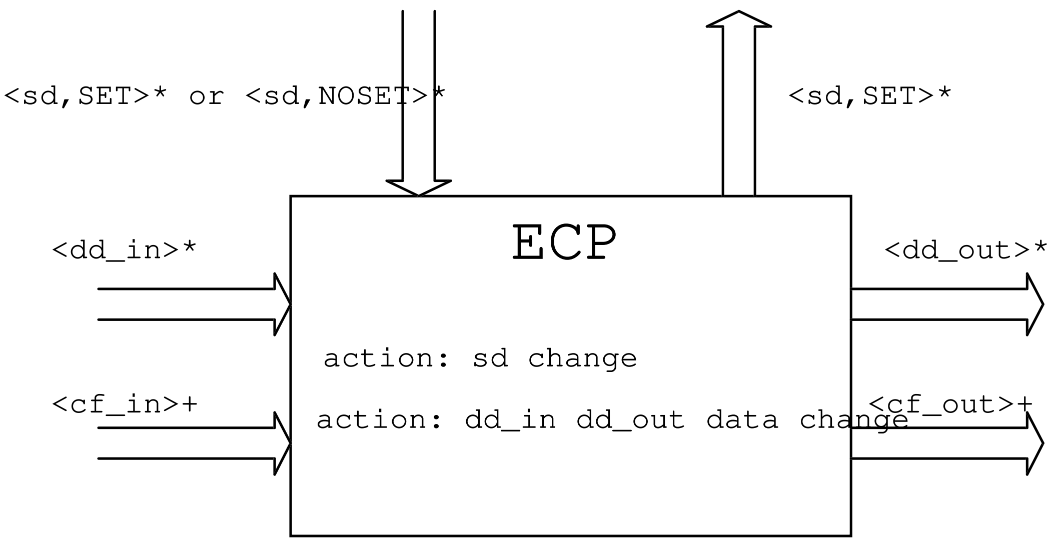

Figure 4: ECP generic model

3.3 Specification and verification of protocols for elementary communicating processes

Specification and verification of protocols for elementary communicating processes is the next step in signaling protocol synthesis.

ECP protocols (p(ECP)) are specified by using extended finite state machine model (FSM). Each p(ECP) has an initial state (InitSt) and final state (FinalSt), which are defined as external states and used for composing service protocol from p(ECP)s. All other states are internal. Typically, ECP has only one p(ECP), but in some occasions more than one p(ECP) is associated to an ECP.

The definition of p(ECP) is done by the following quadruple:

p(ECP) = (K, Σ, s_0, A)

where:

K = (InitSt, FinalSt, <InternalSt>+),

Σ = (<InF>+),

s_0 = InitSt,

$A \subseteq K \times \Sigma \times K$.

p(ECP) protocol is defined for each functional entity involved when distributed implementation of an ECP is considered. Finite state machine model of p(ECP) is verified by reachability tree based algorithms.

3.4 Description of telecommunication services

Each service description (SD) is defined by a set of partially ordered ECPs (ECP chains) and represented by a directed acyclic precedence graph or net (gSD) in the following way:

gSD = (SD, G, startECP, endECP)

where:

- SD represents a set of vertices denoting ECPs,
- G represents a set of edges where g_{ij} denotes a directed edge from ECP_i to ECP_j and describes the partial ordering or precedence relation between processes, i.e. ECP_i must be completed before ECP_j,
- startECP and endECP are the first and the last ECP in the service, respectively.

ECP_i precedes ECP_j if at least one InF describing input control data (<cf_in>+) of ECP_j equals to an InF describing output control data (<cf_out>+) of ECP_j.

Operators pre and post give the following relations:

pre (startECP) = (),

pre (ECP_j) = (ECP_i),

post (ECP_i) = (ECP_j),

post (endECP) = ().

Partial ordering of ECPs is defined by using AND - OR logic. For ECP_j and ECP_k such that:

post (ECP_i) = (ECP_j, ECP_k),

pre (ECP_j) = (ECP_i),

pre (ECP_k) = (ECP_i),

one of the two relations can hold:

(i) mutual exclusion

ECP_j OR ECP_k or

(ii) concurrency

ECP_j AND ECP_k.

The notion of service description is equivalent to the notion of service logic used in the intelligent network conceptual model [4,10].

Figure 5 gives a service description example with several ECPs chained by using OR/AND mechanism.

3.5 Verification of telecommunication service consistency

Service consistency proof is reduced to data flow consistency proof: "if all information flows reach regular terminating places, service is proved to be consistent, otherwise previous synthesis tasks have to be repeated".

The net from previous step is translated to a high level Petri Net. Predicate transition net (Pr/T) is chosen as a tool for service description [11]. A brief description of the Pr/T net follows.

Let U be a finite set of constants and let V be a finite set of variables. W is used to denote the union of U and V, $W = U \cup V$, and W^k to denote the set of k-tuples over W. Predicate transition net is the structure:

(P, T, F, L)

such that:

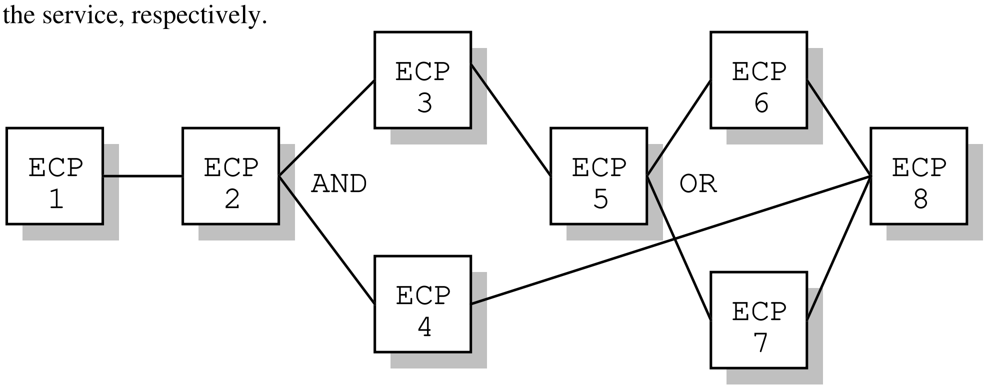

Figure 5: Service description

- (P, T, F) is a Petri net structure, where P denotes a finite set of places, T denotes a finite set of transitions and F denotes a finite set of arcs. The following conditions hold:
  $P \cap T = \emptyset$,
  $P \cup T \neq \emptyset$,
  $F \subseteq (P \times T) \cup (T \times P)$,
  domain (F) $\cup$ codomain (F) = $P \cup T$.
- L is a function from the set of arcs F to the power set over collection of tuples $W \cup ... \cup W^n$ for some n. The tuples for the arcs associated with a place p $\in$ P are of the same length.

Elements of P, T, F and L of the Pr/T net are called predicates (places), transitions, arcs and arc labels, respectively.

A marking denoted by M assigns tokens of different kinds to each place. A transition t $\in$ T is said to be enabled in M if there are enough "right" tokens in each input place of the transition. The transition firing causes removal of tokens from its input places and insertion of tokens to its output places. The arc label l $\in$ L defines how many and which kinds of tokens will be removed from input places or added to output places.

Given a service description gSD, it is transformed into a Pr/T net in the following way:

- each vertex (ECP_i) is modeled as a transition t_i,
- an initial place is connected as an input place to a transition representing startECP,
- each edge at ECP_i output is represented by a place, when AND logic is considered; these places are separate input places for the members of the set post (ECP_i),
- all edges at ECP_i output are represented by a common place, when OR logic is considered; this place is an input place shared between all members of the set post (ECP_i),
- a final place is connected as an output place to a transition representing endECP,
- arcs labels are derived from a set of information flows, sInF, exchanged between ECPs; they define data to be consumed by ECP (input arcs) or produced by ECP (output arcs),
- information flows, InFs, are modelled by markings which represent data vectors. Initial marking shows data necessary to start a service, final marking shows data to terminate it correctly.

Token flow in Pr/T net models information flow. Computation Tree Logic formulas (CTL) are used to check markings. Verification is based on a reachability tree analysis. Special attention has to be paid to irregular/inconsistent states in service description. Service logic description is consistent if all final markings are regular.

Consistent service logic serves as a basis for protocol definition, which is the next step in protocol synthesis.

3.6 Composition of protocols for elementary communicating processes into the service protocol

In this step the composition of p(ECP)s for all elementary communication processes involved into a service protocol (p(SD)) is performed. The service protocol is defined as follows:

p(SD) = (InitSt, <FinalSt>+, <InitFinSt>+,
<p(ECPi)j>+, ϕ(<FinalSt>+))

where:

InitSt	initial state,
<FinalSt>+	set of final states,
<InitFinSt>+	set of mid-states (joint states: initial state of p(ECP_{i+1}) and final state of p(ECP_i); post (ECP_i) = ECP_{i+1}),
<p(ECP_i)$_j$>	j-th p(ECP_i),
ϕ(<FinalSt>+)	CTL formula describing final state markings.

If more than one p(ECP) exists for some ECP, only one protocol, relevant for a service under consideration, is selected.

The composition of p(ECP)s is performed in two ways: as sequential or concurrent (parallel) composition. Two protocols, p(ECP_j) and p(ECP_k), are composed sequentially if ECPs are ordered such that:

post (ECP_j) = ECP_k.

For concurrent composition of p(ECP_j) and p(ECP_k), the following relations hold:

post (ECP_i) = ECP_j,
post (ECP_i) = ECP_k,
ECP_j AND ECP_k.

An example of sequential composition is shown in Figure 6.

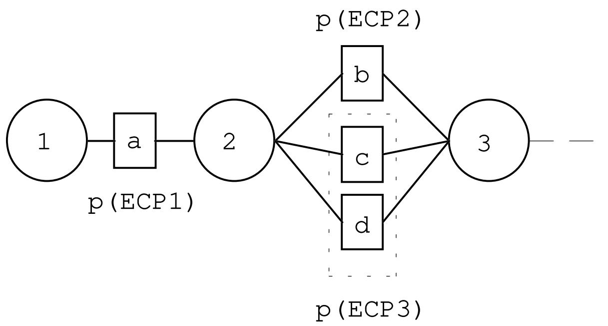

Figure 6: Composition of elementary process protocols

Three elementary process protocols, p(ECP1), p(ECP2) and p(ECP3), and three states, 1, 2 and 3, are defined. p(ECP1) is a single FSM (FSM_a). State 1 is its initial and state 2 final state. State 1 is also the initial state of p(SD), so p(ECP1) runs as the first protocol. After that, due to OR semantics, p(ECP2) with a single FSM (FSM_b) or p(ECP3) with two FSMs

(FSM_c, FSM_d) is executed. State 2 is the common initial state for p(ECP2) and p(ECP3), state 3 is the common final state for p(ECP2) and p(ECP3). Protocols p(ECP1) and p(ECP2) are executed sequentially, p(ECP2) after p(ECP1). The same stands for p(ECP1) and p(ECP3).

4 Verification of signaling protocols

Verification of signaling protocols includes verification of elementary process protocols and verification of service description, i.e. of service consistency. Namely, it can be shown that composition preserves protocol properties or that addition of new transitions while previous are verified preserves protocol properties. This approach can be compared with [13] where each new global state is interactively verified. Instead of interactive synthesis, ECPs hold information for p(ECP)s that are sequentially composed. If each p(ECP) is verified then composition is also verified and p(SD) is verified. It is up to user to decide what kind of formulae will represent users view of protocol correctness.

After analyzing protocol verification techniques, reachability tree analysis as well as state explosion reduction techniques (sleep-sets, symmetries, partial order reduction), SPIN/PROMELA approach is chosen for elementary process protocols [2, 6, 7, 8, 9].

Service consistency is verified by using PROD analyzer for predicate transition nets [5].

Experimental tool for verification of signalling protocols is based on CLIPS, PROMELA/SPIN and PROD tools where CLIPS is a "glue" which integrates them all into a single system. The tool prototype includes CLIPS solutions for all production rule based definitions [1].

AAB service (Automatic Alternative Billing) from the intelligent network capability set 1 is used as a case study [3]. Short prose description of the service follows: "The AAB service enables a user to make a call from any telephone and for the call charge to be billed to the users account which is specific to this service, and which does not refer either to the calling line or to the called line. This service enables a user to call another user and ask him to receive the call at his expenses."

This protocol synthesis example does not reflect commercial requirements and constraints and serves only for methodology illustration.

4.1 Preparatory steps - definition of information elements, information flows and elementary communicating processes

All information elements, information flows and elementary communicating processes must be defined. Some of information elements and information flows for AAB service are shown in Figure 7, more detailed description is given in [3].

For the service under consideration, the following ECPs, derived from corresponding SIBs, are introduced:

- Translate (determines output information from input information: change of dialled digits as required by the service logic),
- Verify (provide confirmation that information received is syntactically consistent with the expected form of such information),
- Screen (perform a comparison of an identifier against a list to determine whether the identifier has been found in the list: user authentication),
- Charge (determine special charging treatment of the call, where special refers to any charging in addition to that normally performed by the basic call),
- Userinteraction (allows information to be exchanged between the network and a call party, where a call party can be either a calling or a called party).

All ECPs are defined through CLIPS deffacts constructs; one of them, ECP "Screen" (SCR), is listed in Figure 8 where static data (ScreenListIdentifier, CIDFP_Screen, CIDFP_Error), dynamic data (Identifier) and control data (match, NOmatch) are declared.

4.2 Specification and verification of elementary process protocols

Elementary process protocols are specified by using FSM. Typical FSM definition of p(ECP) as well as message sequence between functional entities is shown in Figure 9. Send primitives are denoted by "!", the receiving ones by "?". Three information units are exchanged between functional entities A and B: message, confirm and answer.

ECP "Screen" is considered again. For this ECP, functional entities SSF, SCF, SRF and SDF are relevant. Therefore, four elementary protocols are introduced: SCR.ssf, SCR.scf, SCR.srf and SCR.sdf. FSM specifications for two of them, SCR.scf (fsm.aab.SCR.scf) and SCR.sdf (fsm.aab.SCR.sdf), are shown in Figure 10. Both FSMs have three states (0, 1, 2). They communicate by using two messages (Query, QueryResult).

p(ECP) protocol descriptions are translated from FSM into PROMELA language for verification purposes. An example describing "Screen" elementary protocols from Figure 10 is given in Figure 11.

Simulation of p(ECP) generates message sequence chart for particular functional entity within p(ECP). A simple example is shown in Figure 12, where message sequence consists of Query followed by QueryResult.

InF	InE
FurnishChargingInfo	CallID BillingChargingCharacteristics
ApplyCharging	CallID ChargingBillingCharacteristics PartyToCharge SendCalculationToScPIndicator
ApplyChargingReport	CallResult
RequestNotificationChargingEvent	SequenceOfChargingEvent
EventNotificationCharging	CallID EventTypeCharging EventSpecificInformation Charging LegID
SendChargingInformation	CallID BillingChargingCharacteristics LegID
Query	DatabaseID InformationKey RequestedInfoType RequestedInfo
InF:QueryResult (msg:ReturnResultFromQuery)	DatabaseID InformationKey RequestedInfoType RequestedInfo
ConnectToResource	CallID IPRoutingAddress PartyID
PromptAndCollectUserInformation	SRFConnectID InformationToSend DisconnectionfromIPForbidden CollectedInfoFormatReceivedInfo
CancellAnnouncement	OperationIdentifier
CollectedUserInformation	SRFConnectID InformationToSend DisconnectionFromIPForbidden CollectedInfoFormatReceivedInfo
DisconnectForwardConnection	SRFConnectID

Figure 7: Information elements and information flows for AAB service

```
(deffacts trScr “Screen (SIB)”
 (static_in sd_Scr_ScreenListIndicator)
 (static_out sd_Scr_CIDFP_Screen)
 (static_out sd_Scr_CIDFP_Error)
 (dynamic_in dd_Scr_Identifier)
 (cntrl_out SCR_match)
 (cntrl_out SCR_NOmatch))
```

Figure 8: ECP "Screen" definition (CLIPS notation)

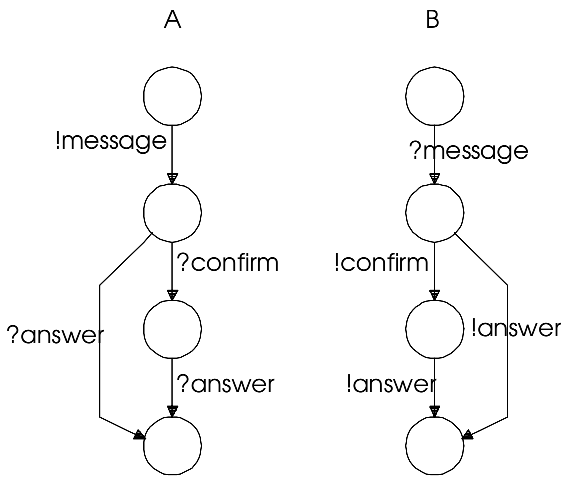

Figure 9: Typical FSM definition of p(ECP)

```
BEGIN.fsm.aab.SCR.scf
STATE 0 stSCR00 “initial state” aab.scr.scf
OUTPUT Query scf2sdf
NEXTSTATE stSCR1
ENDSTATE 0
STATE 1 stSCR01 ““ aab.scr.scf
INPUT ReturnResultFromQuery scf2sdf
NEXTSTATE 1 stSCR2
ENDSTATE 1
STATE 2 stSCR02 ““ aab.scr.scf
ENDSTATE 2
END.fsm.aab.SCR.sdf
```

```
BEGIN.fsm.aab.SCR.sdf
STATE 0 stSCR00 “initial state” aab.scr.scf
INPUT Query scf2sdf
NEXTSTATE 0 stSCR1
ENDSTATE 0
STATE 1 stSCR01 ““ aab.scr.scf
OUTPUT ReturnResultFromQuery scf2sdf
NEXTSTATE 2 stSCR2
ENDSTATE 1
STATE 2 stSCR02 ““ aab.scr.scf
ENDSTATE 2
END.fsm.aab.SCR.sdf
```

Figure 10: FSM specification of SCR.scf and SCR.sdf

```
/*Scr simulation*/
---A--- ---B---
!Query
 ?Query
 !QueryResult
?QueryResult
3 processes created
```

Figure 12: Message sequence for p(ECP) "Screen"

```
/*Scr PROMELA*
/
mtype = { Query, QueryResult}

chan scf2sdf = [1] of { byte };

proctype fsm_aab_SCR_scf()
{
S0:
 atomic {
  scf2sdf!Query -> printf(“!Query\n”); goto S1;
 };
S1:
  atomic {
    scf2sdf?[QueryResult] -> scf2sdf?QueryResult;
printf(“?QueryResult\n”);
    };
}

proctype fsm_aab_SCR_sdf()
{
S0:
 atomic {
  scf2sdf?[Query] -> scf2sdf?Query; printf(“
?Query\n”); goto S1;
 };
S1:
  atomic {
    scf2sdf!QueryResult; printf(“
!QueryResult\n”);
    };
}

init { atomic
  {
    printf(“fsm.aab.SCR\n\n”);
    printf(“---A--- ---B---\n”);
    run fsm_aab_SCR_scf();
    run fsm_aab_SCR_sdf();
  }
```

Figure 11: PROMELA definition of p(ECP) "Screen"

4.3 Service description and verification

The net describing AAB service is shown in Figure 13. The following ECPs are chained: Userinteraction (UI-a, UI-b), Verify (Ver), Screen (Scr), Translate (Tra), Charge (Cha). Formal description of AAB service as a semantic network is done by using CLIPS rules.

Figure 14 presents the predicate transition net for service description verification. Transitions correspond to ECPs as follows: trUIa (UI-a), trVS (combined Ver and Scr), trTC (combined Tra and Cha), trUIb (UI-b). Beside places which define initial and final states and relations between ECPs, two additional places describe static data and error conditions.

Verifying this net, service description consistency can be proved. It is required that each final state in predicate transition net can be reached as final state.

The small PROD script used for that purpose is presented in Figure 15. The numbers in Figure 15 represent information flows names, while "stSTATE" represent states. "dspan" is command for finding paths (depth first search) from initial to final states. The whole command can be interpreted as: "find all paths between initial state and final state with defined set of tokens in it". In this example, information flow number "36" is expected to be found in the final state stCha_FINAL (regular termination of service).

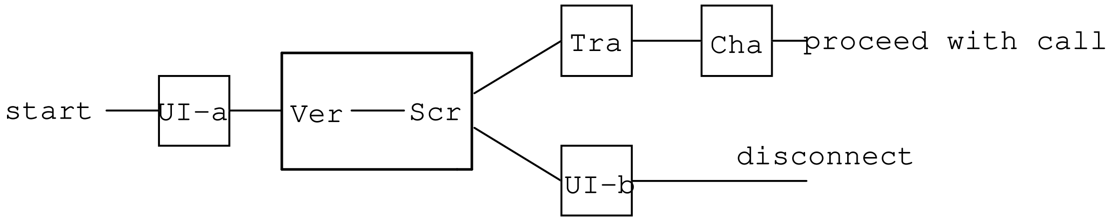

Figure 13: Net describing AAB service

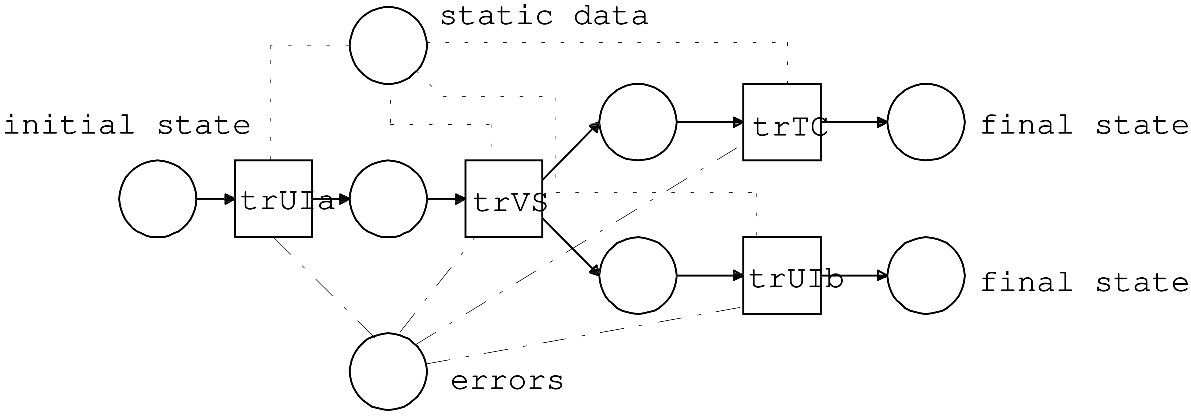

Figure 14: Predicate transition net for AAB service

```
query volatile verbose dspan(true,true)
    stCha_FINAL == <.36.>
query volatile verbose dspan(true,true)
    stUI_FINAL == <.38.>
query volatile verbose dspan(true,true) \
    stERROR ==
    <.29.>+<.30.>+<.33.>+<.35.>+<.37.>
query volatile verbose dspan(true,true)
    stERROR ==
    <.29.>+<.30.>+<.33.>+<.35.>+<.37.> && \
    stUI_FINAL == <.38.> && \
    stCha_FINAL == <.36.>
```

Figure 15: PROD script for service analysis

The number of states analyzed in this example is about over 110. First three "query" commands examine state by state while the last one does it at once.

Finally, service protocol, p(SD), is defined (the complete listing of PROMELA definition of AAB service is ~8K long).

5 Conclusions

The method for specification and verification of signaling protocols for telecommunication services based on definition of elementary communicating processes representing basic service building blocks and their signaling protocols, and service description which allows composition of the partial process protocols into the overall service protocol has been discussed. The method is suitable for all upper layer protocols based on the service concept. The method is applied for synthesizing intelligent network signaling protocols. The multiparadigmatic approach of declarative (rule-based) and procedural (analysis-based) paradigm is achieved by CLIPS programming environment, which cooperates with PROMELA/SPIN and PROD verification tools. The applicability of the method to the early design phases, especially for development concept formalization within a service creation environment has been shown. Future work will also include the extension of verification tool prototype.

References

[1] "CLIPS Version 6.0 - Reference manual I, II, III", ed. JSC-25013, Lyndon B. Johnson Space Center, 1993.

[2] "Proceedings of the First SPIN Workshop", ed. J.-Ch. Gregoire, INRS-Telecommunications, Montreal, QC, October 1995.

[3] "Q12xx: Intelligent Network", *Q-Series Intelligent Network Recommendations*, ITU-T, Helsinki, March 1993.

[4] B. Blašković, "Interpreted Petri Nets as a Validation Tool for Telecommunication Service Protocol Synthesis", *Proceedings of ConTEL93 Conference on Telecommunications, in: ITA - Informacija, Telekomunikacije, Automati*, Vol. 12, No. 1-3, pp. 379-386, 1993.

[5] P. Gronberg, M. Tiusanen, K. Varpaaniemi, "PROD - A Pr/T-Net Reachability Analysis Tool", *Series B: Technical Reports*, No. 11, Helsinki University of Technology, Espoo, Finland, June 1993.

[6] G. J. Holzmann, "Algorithms for Automated Protocol Verification", *AT&T Technical Journal*, No. 1, pp. 32-44, January/February 1990.

[7] G. J. Holzmann, "*Basic Spin Manual*", ATT&T Bell Laboratories, Murray Hill, New Jersey.

[8] G. J. Holzmann, "Design and validation of protocols: a tutorial", *Computer Networks and ISDN Systems*, Vol. 25, pp. 981-1017, 1993.

[9] G. J. Holzmann, "Protocol Design: Redefining the State of the Art", *IEEE Software Magazine*, No. 1, pp. 17-22, January 1992.

[10] I. Lovrek, "Petri Net Based Knowledge Representation for Intelligent Networks", *Proceedings of 10th International Conference on Intelligent Control*, Monterey, CA, 1995.

[11] T. Murata, D. Zhang, "A Predicate-Transition Net Model for Parallel Interpretation of Logic Programs", *IEEE Transaction on Software Engineering*, Vol. 14, No. 4, pp. 481-497, April 1988.

[12] Yun-Chao Hu, "IN CS-2 Enhancements to the Global Function Plane", *Workshop Record of the Intelligent Network Workshop 95*, Ottawa, Canada, 1995.

[13] Y. Zhang, K. Takahashi, N. Shiratori, S. Noguchi, "An Interactive Protocol Synthesis Algorithm Using a Global State Transition Graph", *IEEE Transactions on Software Engineering*, No. 3, pp. 394-404, March 1988.