Creation of Estelle Specifications Using Predicate Logic

Germanas Budnikas, Tadeuš Lozovski and Miroslav Šeibak

Faculty of Economics and Informatics in Vilnius, University of Bialystok, Lithuania

Summary
This paper presents an approach showing how first order predicate logic sentences can be used creating Estelle specifications. Some knowledge based techniques for creation of specifications are briefly introduced. Knowledge about Estelle specifications and its application are given too. An approach for application of first order predicate logic formulas to construct Estelle specifications is discussed — predicates and relations used to describe elements of an extended finite state machine and the specification language are described in detail. A mapping between predicate logic formulas and specification language constructs is defined for the use during specification construction process. In order to illustrate the approach explained in the paper, an alternating bit protocol model is used.

Key words:
Estelle specification, first order predicates, alternating bit protocol.

1. Introduction

There exist methods and software tools based on knowledge engineering techniques intended for the creation of formal specifications or programs. Their examples could be a suggested method that uses graphical and textual notations for the creation of formal specifications in logic from problem domain description [1]; Knowledge Based Software Assistant tool [2] that produces formal executable specifications; Acquisition of Requirements and Incremental Evolution of Specifications tool [3]; a system that translates semi-formal specifications to VDM [4].

Formal specification languages such as Estelle, Promela and SDL are widely used designing distributed information processing systems [5]. In this paper it is shown how predicate logic formulas can be used to gather knowledge about a problem domain together with Estelle specification elements as well as how they could be applied for creation of the specification in Estelle language. Next, an introduction to Estelle specifications is given. Estelle specification language [6] is based on a model of an extended finite state machine [7]. For extending the last a programming language Pascal is used. The specification in Estelle language describes hierarchically structured system of non-deterministic components that interact using messages through bi-directional links between ports (so called interaction points). Each component is an entity of the module. The module is defined by a header and a body which is connected to that header. Interaction points of the module entities are connected with bi-directional channels. Each of such points has an associated to it an unlimited FIFO queue, which saves all messages sent to the module entity through this interaction point. A module body consists of the following sections: descriptions of initializations and descriptions of transitions. A description of constants, parameter types, variables, procedures and functions using a usual Pascal-style without any order make a description part. Description of a channel includes enumeration of all possible messages that can be transmitted through a given channel as well as types of parameters for each messages and name of roles that is assigned to each side of the channel, for example "sender" and "receiver" of messages. The initialization section defines initial values for some module variables. The transition section defines actions of a module. An internal behaviour of a module is characterized in terms of system states. The initial state is defined in the initialization section. Each transition contains a condition for a state change as well as a transition action. The transition condition consists of one or several statements of the form from (a transition from one state to another), when (when an input signal arrives), provided (a logical condition for state change) and a delay (when a timer expires). A transition action consists of operator to and a transition block, which in its turn, consists of Pascal operators with some restrictions and special Estelle operators.

Estelle specification languages likewise other popular languages — Lotos and SDL, which is considered as standard [8], is mainly used for specification and analysis of distributed system, for example — telecommunication protocols.

The creation of Estelle specification consists of the following phases. Concepts and relations that characterise specification elements are elicited from a conceptual description. These main elements are modules, their interaction points, links, signals, states, variables. The first order predicate logic is used to describe these concepts and relations. The specification is constructed in two steps — during the first one, its structural part is defined; during the second one — the specification structure is filled with a behavioural description of how modules are interacting using defined mapping between described predicate logic sentences and specification language elements. The approach described in the paper is illustrated by creating Estelle specification of an alternating bit protocol using

Manuscript received December 5, 2012
Manuscript revised December 20, 2012

first order predicate logic as a formalisation step towards the specification.

The paper is structured as follows. Section 2 presents a definition of predicate logic sentences that describe elements of an extended finite state machine and Estelle specification constructs. These predicates and logical formulas are used to construct a structural part of the specification and are given in section 3. An approach is explored using well known illustrative example of an alternating bit protocol in section 4. Section 5 describes relation of the proposed approach to the closest investigations in the field. Conclusion sums up the paper.

2. Definition of Predicate Logic Sentences for Estelle Specification

Since a predicate logic sentences will be used for a creation of Estelle specifications, it has to contain knowledge about a specification structure and a related model to that specification — extended finite state machine. This model will be used as a framework according which knowledge will be extracted from an analysed system conceptual description. First order predicate logic can be used as a knowledge representation mean.

Knowledge acquired in several stages. At the first one, targets of a developed predicate logic description are identified. They are the following: the first order predicate sentences will be used as an intermediate step to produce Estelle formal specification; knowledge about an analysed system as well as about the formal model related to the specification will be used. During the next stage, both the concepts and relations will be characterised. They are the following:

- Concepts:
  a) Modules; their interaction points; links and their end points;
  b) Input and output signals, and their parameters; states, timers, variables.
- Relations are:
  a) Interconnection between modules;
  b) Transition relations defining state changes;

During the last stage the first order predicate logic is used for the description of concepts and relations, which were defined in the previous stage. The following main parts are introduced.

1. Set of functions, which consists of arithmetic and logical operators: +, –, /, ×, *not*, =, <>. These operators are used for the description of state change.
2. Predicates, which describe relations between variables.

2.1. Predicates, which describe elements of some Estelle specification language constructs:

- An identification of a system module — Module (em_i), where em_i – the *i*-th module $i=\overline{1,n}$, n – number of modules in a system;
- A description of the interaction point iip_i^j of the module em_i - Ip $\left(em_i, iip_i^j\right)$, where $i=\overline{1,n}$, $j=\overline{1,l}$, l – the number of interaction points in the *i*-th module.
- A description of the elementary link ln_i and its end points ep_i^1 and ep_i^2 Link $\left(ln_i, ep_i^1, ep_i^2\right)$, where $i=\overline{1,k}$, k - the number of elementary links in a system, ep_i^1, ep_i^2 .
- A description of an interconnection of two modules em_{i_1}, em_{i_2} , which are connected through their interaction points $iip_{i_1}^{j_1}$ and $iip_{i_2}^{j_2}$ - $\texttt{Connect}\left(em_{i_1}, iip_{i_1}^{j_1}, em_{i_2}, iip_{i_2}^{j_2}\right)$, where $i_1, i_2=\overline{1,n}, i_3=\overline{1,k}, j_1=\overline{1,r}, j_2=\overline{1,l}$.
- A description of the *j*-th timer of the *i*-th module - $\texttt{Timer}\left(em_i, j\right)$.

2.2. Predicates that describe elements of an extended finite state machine:

- A predicate, which describes a module em_i input signal sn_i^j and its components $scv_j^1,\ldots,scv_j^{c_{ic}^j}$ that arrives at the interaction point iip_i^j , has the following view:
  $\texttt{InputSignal}\left(em_i, iip_i^j, sn_i^j, scv_j^1,\ldots,scv_j^{c_{ic}^j}\right)$,
  where $i=\overline{1,n}$, $j=\overline{1,l}, t=\overline{1,c_{ic}^j}$.
- Output signals and it components are described in the same way:
  $\texttt{OutputSignal}\left(em_i, iip_i^j, sn_i^j, scv_j^1,\ldots,scv_j^{c_{oc}^j}\right)$,
  where $i=\overline{1,n}$, $j=\overline{1,r}, t=\overline{1,c_{oc}^j}$.
- State related variables are described using the predicate
  $\texttt{Variables}\left(em_i, ccv_i^j, dcv_i^1,\ldots,dcv_i^{c_{dc}^i}\right)$,
  where $i=\overline{1,n}$, ccv_i^j corresponds to the *j*-th symbolic state label of the module em_i and $dcv_i^1,\ldots,dcv_i^{c_{dc}^i}$ are variables.
- A set of states of the module em_i is described by the predicates:

$\texttt{StateSet}\left(em_i, ccv_i^s, ccv_i^t\right)$,
$\texttt{CurrentState}\left(em_i, ccv_i^s\right)$

where ccv_i^s, ccv_i^t correspond to symbolic labels of source and target states. Using the predicates of the form `StateSet` a set of all states together with their transitions are described, while predicate `CurrentState` defines current module state and obviously can be only one for a certain module.

- An initial system module state is described using the logical formula:

$$\texttt{Init}\left(em_i\right) \rightarrow \texttt{CurrentState}\left(em_i, ccv_i^s\right) \wedge \texttt{Variables}\left(em_i, ccv_i^s, dcv_i^1, \ldots, dcv_i^{c_{dc}^i}\right).$$

Additional predicate has to be introduced for the description of dependencies of state change. Gating predicate that defines supplementary conditions for state changes can be written using predicate `Gate`, which truth depends on the truth of logical conditions $LC_1, \ldots, LC_{p_m}$ that check values of state components:

$$\texttt{Gate}\left(em_i, dcv_i^1, \ldots, dcv_i^{c_{dc}^i}\right) \equiv LC_1 \wedge \ldots \wedge LC_{p_m},$$

where $i = \overline{1,n}$, p_m – the number of logical conditions for the *m*-th formula that describes a state change. The number of `Gate` predicates used to describe a supplementary condition, when characterising a dependency, may be increased if the condition is written in the disjunctive normal form. Then predicate `Gate` will represent parts of the original condition written in the conjunctive normal form. This will cause an appearance of several formulas describing the same gating predicate but with different supplementary conditions.

- Transition relations:

  Logical formula that describes the transition relation that depends on an input signal arrival and when an output signal is sent out is as follows:

$$\texttt{CurrentState}\left(em_i, ccv_i^s\right) \wedge \texttt{StateSet}\left(em_i, ccv_i^s, ccv_i^t\right) \wedge \texttt{InputSignal}\left(em_i, iip_i^j, sn_i^j, scv_j^1, \ldots, scv_j^{c_{ic}^j}\right) \wedge \texttt{Variables}\left(em_i, ccv_i^s, dcv_i^1, \ldots, dcv_i^{c_{dc}^i}\right) \wedge \texttt{Gate}\left(em_i, dcv_i^1, \ldots, dcv_i^{c_{dc}^i}\right) \rightarrow \texttt{Variables}\left(em_i, ccv_i^t, dcv_i'^1, \ldots, dcv_i'^{c_{dc}^i}\right) \wedge$$

$$\texttt{OutputSigna}\ \left(em_i, iip_i^{z*}, sn_i^{z*}, scv_z^{1*}, \ldots, scv_z^{c_{oc}^z*}\right) \wedge \texttt{CurrentState}\left(em_i, ccv_i^t\right)$$

where variables marked with '*' sign represent new values, $i = \overline{1,n}$, $j = \overline{1,l}$, $z \in [1,r]$.

Other transition relations like *state changes and a signal is not sent out* is described in the same way although without corresponding predicates `InputSignal` and `OutputSignal`.

- A logical formula that describes a signal transmission between two modules:

$$\texttt{Connect}\left(em_{i_1}, oip_i^j, em_k, iip_k^t\right) \wedge \texttt{OutputSignal}\left(em_i, oip_i^j, sn_i^j, scv_j^1, \ldots, scv_j^{c_{oc}^j}\right) \rightarrow \texttt{InputSignal}\left(em_k, iip_k^t, sn_k^t, scv_j^1, \ldots, scv_j^{c_{oc}^j}\right),$$

where $i \in [1,n]$, $k \in [1,n]$, $j = [1,r]$, $t \in [1,l]$.

Next section describes how the defined first order logic predicates and logical formulas are used for the creation of Estelle specifications.

3 Creation of structural and behavioural descriptions of Estelle specification

A creation of Estelle specifications can be comprised into two steps. At the first one — a structural part of Estelle specification is defined (see Fig. 1). It contains definitions of

- channels "`channel`", its interaction points "`by ...`" and lists of transferred data;
- modules "`module`" and its interaction points "`ip`";
- blocks with definitions of
  - states and its variables "`state`", "`var`";
  - initial states "`initialize`";
  - transition relations "`trans`";
  - module interconnection "`Connect`".

```
specification abstract_ABP;

channel CONN(Role1,Role2);
  by Role1:
    Packet(data: integer;
        ack_bit: integer);
  by Role2:
    Ack(ack_bit: integer);

module Sender activity;
  ip iip: CONN(Role1)
            individual queue;
end;

body SenderBody for Sender;

  state S1, S2, S3;
  var ...;

  initialize to S_i
      begin
        ...
      end;
  trans
    from S2 to S3
       begin
         output iip.Packet();
       end;
  trans
    from S3 to S1 when iip.ACK
      provided (...)
        begin
          ...
        end;

end; { SenderBody }

modvar
  S: Sender;
  C: Canal;

    initialize
      begin
        init S with SenderBody;
        connect S.iip to C.iip1;
        ...
       end;
end.
```

Fig. 1 Example of structural part of Estelle specification. Source: made by author.

While defining specification structural part, knowledge, which is stored in the following predicates

```
Module(),
Ip(),
Link(),
Connect(),
StateSet(),
Init, CurrentState()
InputSignal(),
OutputSignal(),
Gate(),
```

is used.

During the second stage of the specification creation, a behavioural description of the specification is defined using the predicates and the formulas, which define state transitions. See the following table (table 1) for a detailed mapping.

Table 1: Mapping between first order predicates and Estelle specification language constructs.

Predicates	Estelle specification language construct
Module(), Ip()	module, ip
Link()	channel, by
Connect()	connect
StateSet()	state
Variables()	var
Init, CurrentState(S_i)	initialize to S_i
Gate()	provided
InputSignal()	when
OutputSignal()	output

Source: made by authors.

4 Alternating Bit Protocol Example

In this section an illustration of the proposed approach is given. In the first subsection first order predicate logic description to be used for creation of Estelle specification is constructed. In the second subsection of this chapter, a constructed Estelle specification using previously defined predicates and logical formulas is presented.

4.1 Creation of the First Order Predicate Logic Description of Alternating Bit Protocol

Conceptual description of alternating bit protocol

Protocol describes information transmission between two protocol entities – Sender and Receiver. Sender, after sending its package, starts a timer and waits for an acknowledgement. After timer expiration, Sender's packet is re-sent. It is considered that packet transmission cycle is over when an acknowledgement is got. Each transmitted packet additionally has a number, which can be 0 or 1 (it defines the protocol name). Receiver sends an acknowledgement when gets a packet. It is considered that packets may be lost during their transmission.

Predicate logic description

Identified concepts and relations that were described earlier are suitable without changes. Knowledge defined in these stages is common for any domain because its predicates and logical formulas, in our case, are used for the solution of the same task—the creation of Estelle specification.

Predicates and formulas that were defined during the last knowledge acquisition stage (see section 2) that describe the considered example are presented below. Module Sender will only be presented here (module Channel will be used too where it is necessary to show a connection to the module Sender).

Concepts and relations

Predicatates and logical formulas

System modules:

`Module` (Sender), `Module` (Channel)

Interaction points of modules:

`Ip` (Sender, iip), `Ip` (Channel, iip1), `Ip` (Channel, iip2)

Elementary links:

`Link` (CONN, Role1, Role2)

Interconnection of modules:

`Connect` (Sender, iip, Channel, iip1), `Connect` (Channel, iip1, Sender, iip).

Input signals:

`InputSignal` (Sender, iip, ACK, ack_bit).
`InputSignal` (Channel, iip1, PACKET, pck_bit).

Output signals:

`OutputSignal` (Sender, iip, PACKET, pck_bit),

State related variables:

`Variables`(Sender, S2, data, pck_bit)

Set of states:

`StateSet`(S1, S2), `StateSet`(S2, S3), `StateSet`(S3, S1), `StateSet`(S3, S2), `CurrentState`(S2).

Initial system module state:

`Init`(Sender) → pck_bit *= 0 ∧ `CurrentState`(S2) ∧ `Variables`(Sender, S2, data, pck_bit).

Transition relations:

`CurrentState`(s) ∧ `StateSet`(S1, S2) → `CurrentState`(S2).

S2 → *S3*

`CurrentState`(s) ∧ `StateSet`(S2, S3) ∧ `Variables`(Sender, s, data, pck_bit) → `OutputSignal` (Sender, iip1, PACKET, data,pck_bit) ∧ `Variables`(Sender, S3, data, pck_bit) ∧ `CurrentState`(S3).

S3 → *S1*

`CurrentState`(s) ∧ `StateSet`(S3, S1) ∧ `InputSignal` (Sender, iip, ACK, ack_bit) ∧ `Variables` (Sender, s, data, pck_bit) ∧ ack_bit = pck_bit → pck_bit*= 1–pck_bit ∧ `Variables` (Sender, S1, data, pck_bit) ∧ `CurrentState`(S1).

S3 → *S2*

`CurrentState`(s) ∧ `StateSet`(S3, S2) ∧ `InputSignal` (Sender, iip, ACK, ack_bit) ∧ `Variables` (Sender, s, data, pck_bit) ∧ ack_bit <> pck_bit → `Variables` (Sender, S2, data, pck_bit) ∧ `CurrentState`(S2).

S3 → *S2*

`CurrentState`(s) ∧ `StateSet`(S3, S2) ∧ `Timer` (Sender) ∧ `Variables` (Sender, s, data, pck_bit) → `Variables` (Sender, S2, data, pck_bit) ∧ `CurrentState`(S2).

Logical formula that describes signal passing between two modules:

`Connect` (Sender, iip, Channel, iip1) ∧ `OutputSignal` (Sender, iip1, PACKET, data, pck_bit) → `InputSignal` (Channel, iip1, PACKET, pck_bit).

4.2 Creation of the Behaviour Description for Alternating Bit Protocol Specification

Presented in the previous section predicates and logical formulas are used to define Estelle specification structure as well as supplementing the structure with the behavioural description. Resulted fragment of the specification is presented below.

```
specification ABP;

channel CONN(Role1,Role2);
  by Role1:
    Packet(data: integer;
        pck_bit: integer);
  by Role2:
    Ack(ack_bit: integer);

module Sender activity;
  ip iip: CONN(Role1) individual queue;
end;

body SenderBody for Sender;

  state S1, S2, S3;
  var data, pck_bit: integer;

  initialize to S2
      begin
        pck_bit := 0
      end;
  trans
    from S1 to S2
      begin
        { next message generating }
      end;
  trans
    from S2 to S3
       begin
          output iip.Packet(m,pck_bit);
       end;
  trans
    from S3 to S1 when iip.ACK
```

```
      provided (pck_ack = pck_bit)
        begin
          pck_bit:=1-pck_bit;
        end;
  trans
    from S3 to S2 when iip.ACK
      provided ( ack_bit <> pck_bit )
        begin end;
  trans
    from S3 to S2 delay(tmout)
        begin end;

end; { SenderBody }

modvar
  S: Sender;
  C: Canal;

    initialize
      begin
        init S with SenderBody;
        connect S.iip to C.iip1;
        ...
       end;
end.
```

Resulted specification of the protocol can be used for verification experiment execution. A possible system for that is Verics [9].

5 Related Works

The approach proposed in the paper is very close to the one, presented by Budnikas [10]. Budinikas uses production rule representation in order to create Estelle/Ag specifications that are based on piece-linear aggregate formalism [11]. Estelle/Ag specifications are mainly based on Estelle specification syntax with the distinction that description of states and states transitions differ. In Estelle/Ag specifications states are defined using continuous and discrete state components. Continuous state components correspond to operations of an aggregate (analogy of a module in Estelle); state changes in the aggregate model (hence in Estelle/Ag specifications) are possible due to external events that are related to arrival of input signals (the same transition relation presents in Estelle) and due to internal events that are related to end of operations or timers (a change of state due to timer expiration is possible in Estelle too). Estelle/Ag specifications additionally contains definition of controlling sequences that permit to perform both simulation and verification tasks based on a single Estelle/Ag specification. Budnikas' approach suggests performing validation experiments before creating Estelle/Ag specifications — the following general protocol properties are checked: completeness, boundedness, absence of static deadlocks and reachability.

An approach of transformation of a piece-linear aggregate mathematical model to the first order predicate logic is presented in [12]. There, a description of an analysed system was done in terms of the piece-linear aggregate model using predicates and logical formulas. These formulas were compiled in Prolog in order to perform correctness analysis. The following properties were checked — completeness, boundedness, absence of static and dynamic deadlocks, reachability and customisable invariant properties.

Conclusion

The paper showed a possibility to create Estelle specifications using first order predicate logic. Predicate logic joined with extended finite state machine model can be considered as a formalisation step used to create formal Estelle specification. Analysis of the related work in the field let us formulate the succeeding investigations — to make an analysis of the constructed first order predicate description as it was done in [10, 12].

References

[1] Fuchs, N, Robertson, D.: Declarative Specification. Knowledge Engineering Review (special issue on Logic Engineering), Vol. 11(4), (1996) 317–331

[2] White, D.A.: The Knowledge Based Software Assistant: a Program Summary. In: Knowledge Based Software Engineering Proceedings (1991) 2–6

[3] Johnson, W.L., Feather, M.S., Harris, D.R.: The KBSA Requirements/Specification Facet: ARIES. In: Knowledge Based Software Engineering Proceedings (1991) 48–56

[4] d'Alameida, J., Achuthan, R., Radhakrishnam, T., Alagar, V.S.: Transformation of Semi–Formal Specifications to VDM. In: Knowledge Based Software Engineering '92 Proceedings (1992) 40–49

[5] Paweł Rychwalski, Jacek Wytrębowicz. Unix streams generation from a formal specification, 1-14 // Proc. of the 23rd IFIP WG 6.1 International Conference "Formal Techniques for Networked and Distributed Systems - FORTE2003", H.Konig, M.Heiner, A.Wolisz (Eds.) Springer (2003)

[6] Budkowski S., Dembinski P. An Introduction to Estelle: a specification language for distributed systems // Computer networks and ISDN Systems. - 1988, - Vol. 14, N.1 p.3-23

[7] G.J. Holzmann. The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, 2003, ISBN 0321228626.

[8] Information Processing Systems - Open Systems Interaction - ESTELLE: A Formal Description Technique based on an Extended State Transition Model: International standard. ISO 9074, 1989.

[9] P. Dembinski, A. Janowska, P. Janowski, W. Penczek, A. Półrola, B. Woźna, M. Szreter, A. Zbrzezny: Verics: A Tool for Verifying Timed Automata and Estelle Specifications. Proc. of the 9th Int. Conf. on Tools and Algorithms for

Construction and Analysis of Systems. LNCS, vol. 2619, 278-283, Springer-Verlag, 2003.
[10] G. Budnikas. Application of knowledge-based techniques for creation of ESTELLE/AG specifications. In proc. of the International Conference Modelling and Simulation of Business Systems, Eds. H. Pranevicius, E. Zavadskas, B. Rapp. Kaunas, „Technologija" , 2003: pp. 172-176. (ISI Proceedings) ISBN 9955-09-420-6.
[11] Pranevicius, H., Pilkauskas, V., Chmieliauskas, A.: Aggregate Approach for Specification and Analysis of Computer Network Protocols. Technologija, Kaunas (1994).
[12] H. Pranevicius, R. Ceponyte. Application of logic programming based for validation of computers network protocols aggregate specifications. Automatic and computing technique. 1992, No.2, p.22-27.

Germanas Budnikas received the B.S., M.S. and Ph.D. degrees in Informatics from Kaunas University of Technology in 1994, 1996 and 2004 respectively. His research interests include artificial intelligence, formal specifications, and their static and dynamic analyses.

Tadeuš Lozovski received the M.E. degrees, from Kaunas Technical University in 1967. He received the Dr. Eng. degree from Kaunas Technical University in 1985. After working as a research assistant and assistant professor (from 1991) in the Dept of Solid State Electronics, Vilnius University, and an associate professor (from 1999) he received the Dr. Sc. degree from Wroclaw Technical University (Poland) in 2001, he has been a professor at Bialystok University since 2007. His research interest includes nondestructive method research surface potential thin semiconductor and dielectric layer. He is a member of STIPL and SNPL Lithuania.

Miroslav Šeibak received the M.Sc. degree from Vilnius University in 1986. He received the Ph.D. degree from Vilnius University in 1993. After working as a research assistant in the Department of Differential Equations and Numerical Analysis at Faculty of Mathematics and Informatics, Vilnius University, he has been an assistant professor at the Faculty of Economics and Informatics of the University of Bialystok in Vilnius since 2007. His research interest includes numerical analysis.