Structured Analysis Using Hierarchical Predicate Transition Nets

Xudong He and Cheng-Hong Yang

Department of Computer Science
North Dakota State University
Fargo, ND 58105

Abstract

High-level Petri nets are a powerful formal method for modeling and analyzing systems; however their applications are mainly limited to small-scale research projects due to a lack of modularity and hierarchy within the formalism itself and a systematic approach in using the formalism. In our previous work, a methodology for constructing hierarchical and structured high-level Petri net specifications has been developed. In this paper, we further explore and refine our methodology for using hierarchical high-level Petri nets in systems analysis. Our new approach has adapted the mature results from the Data Flow Diagram method and is demonstrated through a library system.

1. Introduction

Petri nets are an excellent model for studying concurrent and distributed systems since they have a simple visual notation for specifying the structural aspect and the causal relationships among various modules of concurrent and distributed systems, as well as a well-defined formal semantics that supports formal analysis and verification of system properties and behaviors, which is the major advantage of Petri nets over many other informal graphical specification and analysis techniques. Another significant advantage of Petri nets is their executability that enables the dynamic simulation of systems' behaviors. Despite their advantages, Petri nets have the distinct disadvantage of producing very large and unstructured system specifications. Although the introduction of high-level Petri nets has greatly alleviated the above problems, to specify a very large system in terms of a high-level Petri net is still a formidable task and can result in a huge net too complicated to be readily understood. In order to overcome these problems, a methodology for introducing hierarchies into high-level Petri nets and for constructing structured high-level Petri net specifications is needed.

In [1], a methodology for constructing hierarchical and structured predicate transition net (a type of high-level Petri nets) specifications has been developed. The methodology consists of powerful refinement, abstraction, synthesis, and decomposition techniques, and a systematic notation for supporting the above transformation techniques so that a predicate transition net specification can be developed hierarchically and understood stepwisely. The methodology adapts the leveling technique in Data Flow Diagrams [2], and the state decomposition idea in Statecharts [3]. The methodology can significantly reduce the complexity of writing large predicate transition net specifications, as well as enhance the comprehensibility of predicate transition net specifications of large parallel and distributed systems.

In this paper, we further explore and refine our methodology for using hierarchical predicate transition nets in systems analysis. Our new approach has adapted the mature results from the Data Flow Diagram method and its application in modern systems analysis. The major steps and the associated techniques of the new approach are presented and demonstrated through a library system.

2. Hierarchical Predicate Transition Nets

A predicate transition net (PrT net in the sequel) is a seven-tuple \((P,T;F,A,L,R,M_0)\) where \(P\) is the set of predicates represented by circles, which are parameterized places; \(T\) is the set of transitions represented by bars or boxes; and \(F \subseteq (P \times T \cup T \times P)\) is the set of flow relation represented by arcs; \(A\) is the supporting structure that defines the domain, the function and the predicate symbols to be used in labels and expressions; \(L\) is a mapping that associates each arc in \(F\) with a label. Each label is a set of tuples with the same arity and of the form \((t_1,...,t_n)\), and each \(t_i\) can be either a constant or an individual variable; \(R\) is another mapping that associates some transition in \(T\) with a relational expression to restrict the firing of the transition; \(M_0\) is the initial marking that maps the set \(P\) of predicates into a multi-set of structured tokens - constant tuples. All arcs incident on the same predicate must have the same arity, and the arity of the predicate is defined to be the arity of its incident arcs.
A hierarchical predicate transition net (HPrT net in the sequel) is also a seven-tuple \((P,T,F,A,L,R,Mo)\), which differs from a regular PrT net in the following ways:

1. predicates / transitions can be represented by both solid or dotted circles / boxes, where a solid component refers to the lowest abstraction level, and a dotted component stands for either the abstraction or the refinement of an existing PrT net;
2. non-terminating arcs (one end being connected to the boundary of a dotted component) are used to indicate the flow relation between children nets with their outside environment;
3. labels can be formed by using the operators \(+\) and \(\ast\) to express non-determinancy and concurrency among combined data flows respectively.

To construct HPrT net specifications and to ensure the well-definedness of HPrT nets. Four transformation techniques and the associated rules were developed in [1], which include refinement, abstraction, synthesis and decomposition. Refinement adds details to an existing HPrT net by replacing a component of the HPrT net with another HPrT net. Abstraction shrinks an existing HPrT net by substituting a subnet of the HPrT net with a single component. Synthesis introduces additional components into an existing HPrT net. Decomposition separates a HPrT net into several smaller HPrT nets. The details and formal definitions of these concepts can be found in [1].

3. A Practical Development Approach

In [1], the following general guidelines for writing a HPrT net specification were given:

1. The development of an initial HPrT net by
   * analyzing the system to distinguish the major passive and active components of the system, passive components are those capturing the distinct states, and active components are those making transitions from one state to another,
   * representing passive and active components by predicates and transitions respectively, and connecting the predicates and transitions with directed arcs,
   * inscribing arcs with appropriate labels to reflect data flows, constructing relational expressions to restrict the firing conditions of transitions if necessary, and marking the lowest level components;
2. The introduction of hierarchies into the initial HPrT net by applying the abstraction and refinement transformation techniques repeatedly; and
3. The creation of modularity in HPrT nets by using the decomposition and synthesis techniques repeatedly.

The above guidelines are very useful, but maybe too abstract for systems analysts to use. Systems analysts are generally familiar with traditional systems analysis techniques such as Data Flow Diagrams (DFDs in the sequel) and the process of using them. Although HPrT nets are quite different from and more complicated than DFDs, they have many similarities such as graphical notation, the leveling, and the balancing concepts. Therefore the mature results of using DFDs in systems analysis can be adapted in HPrT nets and make the HPrT net method a more practical and applicable systems analysis tool.

In developing and discussing our new development approach for HPrT specifications, we use the following small Library System given in [4]:

A small Library System has the following types of transactions:

1. Check out a copy of a book;
2. Return a copy of a book;
3. Add a copy of a book to the library;
4. Remove a copy of a book from the library;
5. List the book titles by a particular author;
6. List the books being borrowed by a borrower;
7. Search the last borrower of a particular copy of a book.

There are two types of users: staff users and ordinary borrowers. Transactions (1) to (7) are restricted to staff users, however that ordinary borrowers can perform transaction (6) to find out the list of books currently checked out by themselves. The system must also satisfy the following constraints: (a) all copies in the library must be either available for checkout or be checked out, (b) no copy of a book may be both available and checked out at the same time, (c) a borrower may not have more than a predefined number of books checked out at one time, and (d) a borrower may not have more than one copy of the same book checked out at one time.

3.1. The Environmental Model

As in the Data Flow Diagrams, to develop an initial HPrT net specification of a system we need to identify the boundary and interfaces between the system and its external environment. The result is the top level HPrT net called context diagram. A context diagram consists of the system under consideration represented by a single predicate (circle), external entities represented by transitions (boxes). The process for constructing a context diagram is:

1. Identify and name the system,
2. Determine and name external entities,
3. Define and label data flows.

Since we are only concerned about the logical library system, a book here refers to its information, which includes the title, author name, and a serial number. In the specification, the following convention is used: capital
letters are used to label structural data flows, and small letters are used to name elementary data flows.

The first two steps are simple and straightforward and the third step needs some careful analysis. Fig. 1 shows the context diagram of the Library System, where the whole system is named Library System, the external entities are ordinary users named User and staff users named Staff.

From the requirement description and especially transaction (6), the only possible data flow from an ordinary user to the Library System is an identification number (id) to find out the books borrowed; and the data flow from the Library System to a user is a list of book titles (LT). The possible data flows from a staff member to the Library System are (a) an identification number (id) and the book information (BK) to borrow a book, (b) a book (BK) to return, (c) a book (BK) to be added to the library, (d) a book (BK) to be removed, (e) an author’s name (an) to get the list of titles, (f) a user’s identification number (id) to find out all the borrowed book titles, and (g) a serial number (sn) of a book to find out the last borrower. The possible data flows from the Library System to a staff member are (a) a list of book titles (LT) by a particular author, (b) a list of book titles (LT) borrowed by a particular user, and (c) a user’s name (un) who last borrowed a particular copy of a book. To abstract away the details of each event, we group the input data (IP) and output data (OP) by their structures, for example input data flows (b), (c), and (d) have the same structure (cm, BK) where cm is the command and distinguishes the data flows for different events. The label formulation operator * means an AND relationship, which specifies the non-determinacy among different events.

### 3.2. The Initial Behavioral Model

The traditional approach to obtain a detailed system specification called the behavioral model here is to refine the environmental model stepwisely, which is generally useful for small systems, but may not be successful for large systems as pointed out in [2]. A more realistic approach is to identify all possible events from the requirement description and develop simple HPrT nets for the events. Most events are asynchronous, however their results may have impacts on other events at some later time, and thus are the components of the system states (history). System states are markings of a HPrT net captured by predicates. Therefore to obtain a HPrT specification for the whole system, we only need to apply the synthesis transformation technique to connect the HPrT nets from individual events through the common predicates. The process for constructing the initial behavioral model is:

1. Identify all events and their inputs and outputs,
2. Design a simple HPrT net for each event,
3. Connect the HPrT nets through common predicates.

We apply the above process to the Library System. Since the events are clearly stated in the requirement description, step 2 can be carried out immediately. The Check Out event is triggered by the incoming of a request and involves the updates of the borrower’s record (USER RECORD) and the book record (BOOK RECORD), which is specified in Fig. 2. The specifications of other events are similar to that of event Check Out. The label formulation operator * means an AND relationship, which specifies the concurrency among involved data flows.

The Step 3 is straightforward, however its result is normally both too complicated and lacking details, and thus subject to various transformations.

### 3.3. The Completed Behavioral Model

On the one hand, the initial behavioral model can be very complicated due to the large number of events and has to be abstracted and decomposed. On the other hand, the initial behavioral model lacks details and needs to be refined. Furthermore, we also have to define the constraints for the bottom-level transitions and assign an initial marking to the bottom-level predicates. The process for completing the behavioral model is:

1. Abstract and decompose the initial behavioral model,
2. Refine the resulting model and define constraints,
3. Assign an initial marking.

#### 3.3.1. Abstraction and Decomposition

Since the synthesized behavioral model is normally large, some transformations are needed. There are two kinds of general transformations: abstraction and decomposition. Abstraction is a vertical transformation technique, which hides details in a hierarchy. Sometimes label renaming is needed during the abstraction in order to keep the correct data and control flows, and to avoid name clashes. This kind of transformation is very much like the modularity concept in a typical modern high-level programming language. The labels serve as the channels for parameter passing and unique identification. Decomposition is a horizontal transformation technique, which reduces the complexity of specifications by dividing them into small related pieces. Although decomposition is the reverse of synthesis, a logical re-grouping of related transitions and predicates is usually necessary.

We apply the abstraction technique to the Library System, and obtain the results shown in Fig. 3. Fig. 3 (1) is an abstraction of the synthesized behavioral model, and is also a refinement of the predicate Library System in the
Events in the synthesized behavioral model can be very complicated and need further refinement. Refinement is of algorithmic nature, which routes input data flows in order to produce output data flows. During the refinement process, processing details are introduced, which must represent the functionality of the event, preserve the constraints stated in the requirements, and impose as fewer restrictions on the design and implementation as possible. Conditional expressions are constructed for reflecting the constraints in the requirements, for routing data flows, and for establishing the relationship between the incoming and outgoing data flows. The constraints are automatically inherited from parents to children.

The refinement of events Check Out, Return and Add are shown in Fig. 4 to 6 respectively. Some refinements are very simple, and others are quite complicated and need to be further refined.

In Fig. 4, the relational expression id = id1 ∧ BK = (bt1, an1, sn1) specifies that only the borrower's record and requested book record enable the transition, and are subject to change. The relational expression (1 bks1 i = 5 ∨ ∃(bt', an', sn') ∈ bks1. (bt' = bt ∧ an' = an ∧ sn' ≠ sn) ∨ cb1 ≠ λ) means that if (a) the borrowing limit is reached (5 is used as the limit in this paper), (b) the borrower tries to borrow a second copy of the same book, or (c) the request book has been checked out, then the request is denied and no change is made to the records BR2 = BR1 ∧ UR2 = UR1; otherwise (1 bks1 i < 5 ∧ ∃(bt', an', sn') ∈ bks1. (bt' ≠ bt ∨ an' ≠ an ∨ sn' = sn) ∧ cb1 = λ) the request is granted, and the changes to the book and borrower records are made according to the relation id = id1 ∧ id2 = id1 ∧ un2 = un1 ∧ bks2 = bks1 ∨ (BK) ∧ (bt2, an2, sn2) = BK ∧ fb2 = fb1 ∧ cb2 = id.

The assignment of an initial marking is straightforward. In the Library System, we can put some tokens in the predicates User Record and Book Record. The bks component of every token (id, un, bks) in the predicate User Record is set to empty set Φ; and the components fb and cb of every token (bt, an, sn, fb, cb) in the predicate Book Record are set to empty strings λ. The initial number of books is n.

Therefore we obtain a set of hierarchical specification of the Library System, which consists of the Data Dictionary, Figures 1 and 3 to 6, and the refinements of other events.

4. Analysis of Specifications

The major advantage of hierarchical predicate transition nets over other informal graphical specification methods is that they are rigorous and have a well-defined semantics; therefore formal analysis of system properties is possible. The analysis of HPR net is basically based on the algebraic definitions of labels, the relational expressions, and the initial marking.

4.1. Formalization of System Properties

The system properties can be easily expressed as first order logic formulas, or more precisely, as first order temporal logic formulas [5]. For simplicity purpose, we only consider the system properties as classical first order logic formula in this paper. The system properties of the Library System can be formalized as follows:

(a) all copies in the library must be either available for checkout or be checked out:

∀ (bt, an, sn, fb, cb) ∈ {BR1, ..., BRn}. (cb = λ ∨ ∃ (id, un, bks) ∈ {UR1, ..., URm}. (cb = id ∧ (bt, an, sn) ∈ bks));

(b) no copy of a book may be both available and checked out at the same time:

∀ (bt, an, sn, fb, cb) ∈ {BR1, ..., BRn}. (cb = λ = (id, un, bks) ∈ {UR1, ..., URm}. ((bt, an, sn) ∈ bks));

(c) a borrower may not have more than a predefined number of books checked out at one time:

∀ (id, un, bks) ∈ {UR1, ..., URm}. (bks ≤ 5);

(d) a borrower may not have more than one copy of the same book checked out at one time:

∀ (id, un, bks) ∈ {UR1, ..., URm}. (∀ (bt1, an1, sn1), (bt2, an2, sn2) ∈ bks. (bt1 ≠ bt2 ∧ v an1 ≠ an2)).

4.2. Proof of System Properties

To prove the system properties formulated in the above step, we have to use the control structures and the constraints defined for transitions as inference rules. Since classical first order logic instead of first order temporal logic is used, an induction based proof technique is to be used. We demonstrate the induction proof technique in proving the property (a) of the Library System as follows, and the proofs of other properties are basically the same.

Proof outline of property (a) of the Library System:

Step 1: under the initial marking, no book has been checked out,

∀ (bt, an, sn, fb, cb) ∈ {BR1, ..., BRn}. (cb = λ)
which implies (a);
Step 2: assume (a) holds after k events;
Step 3: prove (a) holds after k + 1 events:

We examine each event in turn:
• Event Check Out:

Case 1 (structure of the HPnT net in Fig. 4):
\[ \text{bks1} < 5 \land \text{cb1} = \lambda \lor (\text{bt}, \text{an}, \text{sn}) \subseteq \text{bks1}. \ (\text{bt} \neq \text{bt}\lor \text{an} \neq \text{an} \lor \text{sn} = \text{sn}) \lor \text{id} = \text{id1} \land \text{id2} = \text{id1} \land \text{un2} = \text{un1} \land \text{bks2} = \text{bks1} \lor \{ \text{BK} \} \land (\text{bt2}, \text{an2}, \text{sn2}) = \text{BK} \land \text{fb2} = \text{fb1} \land \text{cb2} = \text{id} \Rightarrow
\]
\[ \text{id} = \text{id1} \land \text{id2} = \text{id1} \land \text{un2} = \text{un1} \land \text{bks2} = \text{bks1} \lor \{ \text{BK} \} \land (\text{bt2}, \text{an2}, \text{sn2}) = \text{BK} \land \text{fb2} = \text{fb1} \land \text{cb2} = \text{id} \Rightarrow
\]
from the structure in Fig. 4

\[ \text{BR2} = (\text{bt2}, \text{an2}, \text{sn2}, \text{fb2}, \text{cb2}) \land \text{UR2} = (\text{id2}, \text{un2}, \text{bks2}) \land \text{cb2} = \text{id2} \land (\text{bt2}, \text{an2}, \text{sn2}) \subseteq \text{bks2}
\]
which together with the assumption implies (a);

Case 2:
\[ (\text{bks1l} = 5 \land \text{cb1} \neq \lambda \lor 3 (\text{bt'}, \text{an'}, \text{sn'}) \subseteq \text{bks1}. \ (\text{bt'} = \text{bt} \land \text{an}' = \text{an} \land \text{sn'} = \text{sn}) \lor \text{id} = \text{id1} \lor \text{id2} = \text{id1} \land \text{un2} = \text{un1} \land \text{bks2} = \text{bks1} \lor \{ \text{BK} \} \land (\text{bt2}, \text{an2}, \text{sn2}) = \text{BK} \land \text{fb2} = \text{fb1} \land \text{cb2} = \text{id} \Rightarrow
\]
from the structure in Fig. 4

\[ \text{BR2} = (\text{bt2}, \text{an2}, \text{sn2}, \text{fb2}, \text{cb2}) \land \text{UR2} = (\text{id2}, \text{un2}, \text{bks2}) \land \text{cb2} = \text{id2} \land (\text{bt2}, \text{an2}, \text{sn2}) \subseteq \text{bks2}
\]
which together with the assumption implies (a);

• Event Return:

\[ \text{BK} = (\text{bt1}, \text{an1}, \text{sn1}) \lor \text{BK} \subseteq \text{bks1} \land (\text{bt2}, \text{an2}, \text{sn2}) = \text{BK} \land \text{fb2} = \text{cb1} \land \text{cb2} = \lambda \land \text{id2} = \text{id1} \land \text{un2} = \text{un1} \land \text{bks2} = \text{bks1} \lor \{ \text{BK} \}
\]
\[ \Rightarrow \text{from the structure in Fig. 5}
\]
\[ (\text{bt2}, \text{an2}, \text{sn2}, \text{fb2}, \text{cb2}) = \text{BK} \land \text{cb2} = \lambda
\]
which together with the assumption implies (a);

• Event Add:

\[ (\text{bt2}, \text{an2}, \text{sn2}) = \text{BK} \land \text{fb2} = \lambda \land \text{cb2} = \lambda \land \text{n2} = \text{n1} + 1
\]
\[ \Rightarrow \text{from the structure in Fig. 6}
\]
\[ (\text{bt2}, \text{an2}, \text{sn2}, \text{fb2}, \text{cb2}) = \text{BK} \land \text{cb2} = \lambda
\]
which together with the assumption implies (a).

Other events do not affect the book status; therefore (a) holds from the assumption in Step (2).

5. Conclusions

In this paper, we explored our methodology for using hierarchical predicate transition nets in systems analysis, which has adapted the mature results from the Data Flow Diagram method and its application in modern systems analysis. The major steps for constructing HPnT specifications from given requirements were presented and demonstrated through a library system. An induction proof technique for analyzing system properties was given, which can be partially mechanized. The main advantages of HPnT nets are (1) a very good balance between formality and comprehensibility, (2) strong specification power, (3) scalability, (4) verifiability, and (5) executability. Our results show that hierarchical predicate transition nets are not only a very useful formal method for research purposes but also a very practical tool for general systems specification and analysis.

We are implementing a CASE tool to build, verify and execute HPnT nets. Our future research topics include (1) the study of formal semantics of HPnT in terms of modern algebra [6] so that various equivalent structural transformations can be automatically performed, and (2) the investigation of first order temporal logic proof techniques for the properties of HPnT nets.

References


Data Dictionary

an, an1, an2
bks, bks1, bks2
bt, bt1, bt2
cb1, cb2
cm = A | C | D | F | L | I | R | IS
fb1, fb2
id, id1, id2
n1, n2
sn, sn1, sn2
un, un1, un2

* Author Name *
* Books Borrowed by a User *
* Book Title *
* Current Borrower *
* Command Initials *
* Former Borrower *
* User ID *
* # of Books in the Library *
* Serial Number *
* User Name *
* A Book *
* Book Record *
* Book Records *
* Book Titles *
* User Record *
* A Book *
* A Book *
* A Book *
* A Book *
* A Book *
* A Book *
* A Book *
* A Book *
* A Book *
* A Book *

OP = (cm, id, BK) + (cm, BK) + (cm, st)
LBI = ni + BRi + URi + LBR
IP = (cm, id, BK) + (cm, BK) + (cm, st)
The Environmental Model

(1) Library System

(2) Library Status

(3) Transact

Check Out Event in the Initial Behavioral Model

The Behavioral Model after Abstraction

Refinement of Check Out

Refinement of Return

Refinement of Add