

Experimenting with Progress Mappings for the Sweep-Line Analysis of the Internet Open Trading Protocol*

Guy Edward Gallasch¹, Chun Ouyang¹, Jonathan Billington¹, and
Lars Michael Kristensen^{2**}

¹ Computer Systems Engineering Centre
School of Electrical and Information Engineering
University of South Australia
Mawson Lakes Campus, SA 5095, AUSTRALIA

Email: guy.gallasch@postgrads.unisa.edu.au, chun.ouyang@unisa.edu.au,
jonathan.billington@unisa.edu.au

² Department of Computer Science, University of Aarhus
IT-parken, Aabogade 34, DK-8200 Aarhus N, DENMARK
Email: kris@daimi.au.dk

Abstract. The sweep-line occurrence graph method exploits a behavioural notion of progress found in many systems. This allows states to be deleted that will not be revisited during occurrence graph generation, allowing fewer states to be stored in main memory for the necessary comparisons, thus providing savings in both memory and time. Properties of the system (such as deadlocks) can then be verified on-the-fly. This method is relatively new and needs to be evaluated on a range of examples. One class of protocols that seems to be suited to sweep-line analysis is transaction protocols. This is because transaction protocols often have an occurrence graph that starts with a request and finishes with the request being satisfied (or not). Thus there is a natural progression of states as the transaction proceeds. This paper provides insight into how to design a progress mapping, central to the use of the sweep-line method, for a transaction protocol known as the Internet Open Trading Protocol (IOTP). IOTP is quite complex and is modelled using hierarchical Coloured Petri Nets (CPNs). The sweep-line method is particularised for CPNs and three progress mappings are developed for IOTP. The results show that naive choices for the progress mapping leads to unnecessary regeneration of states, due to the mapping not being monotonic. Refinement of the mapping leads to a monotonic progress measure, which allows results to be obtained for IOTP that were not previously possible.

Keywords: State space methods, Occurrence graph methods, Sweep-line, State explosion problem, Internet Open Trading Protocol, Coloured Petri Nets, Verification.

1 Introduction

State space (occurrence graph) methods encompass the paradigm of analysis techniques that involve generation of all or part of the reachable state space (occurrence graph) of a system in order to answer verification questions. This paradigm is one of the main analysis methods for Coloured Petri nets (CPNs) [19,20] and has been used successfully to analyse and verify many systems (for examples see [1,21]). These methods have an advantage over theorem proving techniques [35] in that the mathematics can be neatly contained in automated software tools such as Design/CPN [5,8].

One disadvantage that has been the subject of much research is that of the *state explosion problem*. Even for relatively simple systems, the number of reachable states can be very large. The unfortunate result of state explosion is that in many cases, the entire occurrence graph is too large to fit into computer memory. This has led to a number of so-called *state space reduction* techniques to alleviate the problem.

A good survey of reduction techniques is provided in [35]. These techniques may be classified into three main categories. The first are those that represent the occurrence graph

* Supported by an Australian Research Council (ARC) Discovery Grant (DP0210524).

** Supported by the Danish Natural Science Research Council

in a condensed or compact form, such as symmetry reduction [7, 9, 18]. The second class explores only a subset of the reachable states. Partial order methods [31, 34, 38] such as stubborn sets fall into this category. The third class involves deleting or throwing away states or state information during exploration and include *bit-state hashing* [15, 16, 39], *state space caching* [11, 13, 14] and the *pseudo-root* technique [30].

The sweep-line exploration method belongs to the third category. It guarantees full coverage of the occurrence graph but differs from the state space caching and pseudo-root techniques in the way that states are selected for deletion. By exploiting *progress* in the model being analysed, a *progress mapping* can be defined which identifies states that are guaranteed not to be reached again [6] or are unlikely to be reached again [24].

The Internet Open Trading Protocol (IOTP) [3, 4] is an electronic commerce protocol developed by the Internet Engineering Task Force (IETF). The core of IOTP is a set of electronic transactions that reflect common trading activities, such as purchasing goods or depositing funds, over the Internet. The specification of IOTP, published as Request For Comments (RFC) 2801 [3], was the largest RFC developed by IETF to that time, spanning 290 pages. The RFC however contains an informal narrative description of IOTP, and so far no complete implementation of IOTP yet exists [17, 32].

A hierarchical CPN model of IOTP was created [29] and further improved [27] to cover most protocol features in RFC 2801. A set of desired properties of IOTP (e.g., correct termination) were investigated in [28] and revealed errors in the design of IOTP. Changes to RFC 2801 were suggested and a revised IOTP CPN model [26] developed. This paper has arisen from attempts to analyse the revised IOTP CPN, which presents a practical challenge due to the large number of reachable states for increasing parameter values.

The purpose of this paper is to apply the sweep-line method to the revised IOTP CPN in order to obtain results for larger parameter values than is possible using conventional analysis [26] and to provide more experience in applying the sweep-line method to practical examples. Another objective is to provide a comparison between the effectiveness of sweep-line analysis when using a progress mapping based on general protocol properties and one based on IOTP-specific properties when analysing the revised IOTP CPN.

The rest of this paper is organised as follows. Section 2 provides a description of the sweep-line method. Sections 3 and 4 introduce the Internet Open Trading Protocol and its CPN model, respectively. The derivation of three different progress mappings and some insights into the process for doing this are presented in Section 5 and the experimental results obtained by using them are presented in Section 6. Finally, some concluding remarks and future work are presented in Section 7. We assume that the reader is familiar with the basic concepts of CPNs and reachability analysis.

2 The Sweep-line Method

We present the sweep-line method in the context of Coloured Petri nets [19, 22] as we are using the sweep-line method to analyse a CPN model. The method is, however, not specific to CPNs, but applicable to a wide range of modelling languages and formalisms.

The sweep-line method is based on the notion of *progress* within the system being modelled. Systems exhibit progress in different ways. One example is found in transaction protocols such as IOTP, where interacting protocol entities move through a series of interactions (called *exchanges* in IOTP) towards a final completed state. IOTP is described in more detail in the next section. Communication protocols in general exhibit progress through sequence numbers and retransmission counters. The key concept behind the sweep-line method is that if we can

quantify the progress of a system in each state, then we can identify the states with a lower *progress value* that cannot be reached from states with a higher progress value. When states are no longer reachable we do not need to keep them in memory for comparison with each newly generated state.

The notion of progress is captured formally in a *progress measure* [6, 24]. Importantly a progress measure specifies a *progress mapping* ψ from states to progress values that are ordered. In this paper we shall use the natural numbers \mathbb{N} as the set of progress values and their usual order relations (e.g. $\leq, <, >$). We firstly introduce the concept of a CPN instrumented with a progress mapping ψ .

Definition 1. *A CP-net with a progress mapping is a tuple $CPN_\psi = (CPN, \psi)$ where CPN is a Coloured Petri net (defined in [19]) and ψ is a progress mapping given by $\psi : \mathbb{M} \rightarrow \mathbb{N}$ where \mathbb{M} is the set of possible markings for CPN .*

From [19], let $[M_0\rangle$ be the set of *reachable* markings of CPN and $be \in BE$ be a binding element enabled in marking M . If, for a given progress mapping $\psi, \forall M, M' \in [M_0\rangle, M[be\rangle M' \Rightarrow \psi(M) \leq \psi(M')$ then the mapping ψ is *monotonic* with respect to the reachability relation and implies that if $\psi(M) > \psi(M')$ for $M, M' \in [M_0\rangle$ then $M' \notin [M\rangle$.

We may consider that the mapping ψ induces an ordered partition on the set of reachable markings. Once all successors of all markings with a particular (minimum) progress value have been generated, then, for a monotonic progress mapping, we can delete the markings (that are not of interest) with this progress value, freeing up memory, and reducing the time spent comparing new markings with those already generated. The overhead is calculating the progress value for each state, and ensuring that markings are processed in a least-progress-first order.

The monotonicity of the progress mapping can be checked during occurrence graph (OG) generation as all arcs in the OG are traversed by the sweep-line method. If, however, $\psi(M') < \psi(M)$ for some $M[be\rangle M'$ then we have a *regress edge*:

Definition 2. *Let OG_{CPN} be the occurrence graph of CPN in CPN_ψ as given in Definition 1. An occurrence of binding element $be \in BE$ of CPN in marking $M \in [M_0\rangle$, leading to marking M' in which $\psi(M') < \psi(M)$ is called a **regress edge** with respect to ψ of OG_{CPN} .*

Regress edges may lead to new markings or to markings that have already been explored but subsequently deleted from memory. The sweep-line algorithm has no way of distinguishing between these two types of markings and so must treat all destinations of regress edges as if they have not yet been explored. Exploration does not continue along regress edges, however the destinations of regress edges are marked as roots (initial states) for a subsequent sweep of the OG. To guarantee termination of the algorithm these states are also marked as *persistent*. Persistent states cannot be deleted and so each state can be marked at most once as a root state for a subsequent sweep.

An algorithm for the sweep-line method that takes into account non-monotonic progress mappings was presented in [24]. We present a modified version in Fig. 1 for CPN_ψ . Let there be a set ROOTS which contains the starting markings of a sweep; a set PERSISTENT which contains markings that cannot be deleted; a set UNEXPLORED which contains all the markings generated so far within a sweep that have not had their successors explored; a set PROCESSED which stores markings which have had their successors generated in a sweep; a set SUCCESSORS which holds the successors of a given marking; and a set DM which stores all dead markings. The algorithm selects and removes a marking from UNEXPLORED that has the minimum progress value among all states in UNEXPLORED (lines 12 and 13), adds it to

```

1: ROOTS  $\leftarrow \{M_0\}$ 
2: PERSISTENT  $\leftarrow \emptyset$ 
3: UNEXPLORED  $\leftarrow \emptyset$ 
4: PROCESSED  $\leftarrow \emptyset$ 
5: SUCCESSORS  $\leftarrow \emptyset$ 
6: DM  $\leftarrow \emptyset$ 
7: while ROOTS  $\neq \emptyset$  do
8:   UNEXPLORED  $\leftarrow$  ROOTS
9:   ROOTS  $\leftarrow \emptyset$ 
10:  while UNEXPLORED  $\neq \emptyset$  do
11:    (* Generate the successors of a node in UNEXPLORED that has the lowest progress value *)
12:    Select  $M \in$  UNEXPLORED such that  $\forall M' \in$  UNEXPLORED,  $\psi(M) \leq \psi(M')$ 
13:    UNEXPLORED  $\leftarrow$  UNEXPLORED  $\setminus \{M\}$ 
14:    PROCESSED  $\leftarrow$  PROCESSED  $\cup \{M\}$ 
15:    SUCCESSORS  $\leftarrow \{M' | M[be)M'\}$ 
16:    if SUCCESSORS =  $\emptyset$  then
17:      DM  $\leftarrow$  DM  $\cup \{M\}$ 
18:    else
19:      ROOTS  $\leftarrow$  ROOTS  $\cup \{M' \in$  SUCCESSORS  $| \psi(M') < \psi(M)$  and  $M' \notin$  PERSISTENT $\}$ 
20:      PERSISTENT  $\leftarrow$  PERSISTENT  $\cup \{M' \in$  SUCCESSORS  $| \psi(M') < \psi(M)\}$ 
21:      UNEXPLORED  $\leftarrow$  UNEXPLORED  $\cup \{M' \in$  SUCCESSORS  $| \psi(M') \geq \psi(M)$  and  $M' \in$  PROCESSED $\}$ 
22:    end if
23:    (* Delete states that have a progress value less than those in UNEXPLORED *)
24:    PROCESSED  $\leftarrow$  PROCESSED  $\setminus \{s \in$  PROCESSED  $| \forall M' \in$  UNEXPLORED,  $\psi(s) < \psi(M')\}$ 
25:  end while
26: end while

```

Fig. 1. The Generalised Sweep-line Algorithm, based on the algorithm from [24].

the set of processed markings (line 14) and generates all successors of this marking (line 15). If there is no successor, the marking is added to the set of dead markings (line 17). If any regress edges are detected, their destination markings (if not already marked as persistent) are added to ROOTS as initial states for the next sweep (line 19) and marked as persistent (line 20). Destinations of non-regress edges that have not already been processed are added to UNEXPLORED (line 21). Deletion of states occurs on line 24.

The example shown in Fig. 2 (from [25]) illustrates the behaviour of the sweep-line method. This figure shows three snapshots of the OG during OG exploration. Arc labels have been omitted to simplify the diagram. The states are arranged from left to right in ascending progress order. Nodes that have been explored and deleted are represented as empty circles. Nodes currently in memory (but that have not yet been explored) are solid black circles. Nodes yet to be discovered are grey circles. In Fig. 2 (a) the states M_0 and M_1 have been explored and

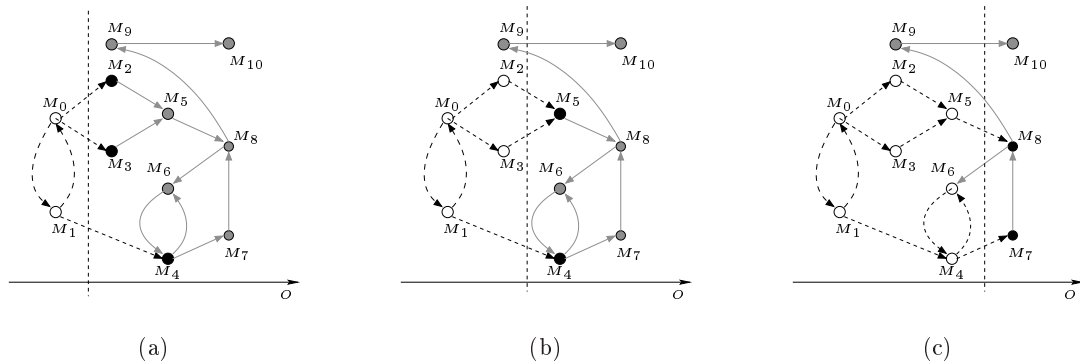


Fig. 2. Snapshots of sweep-line occurrence graph exploration.

subsequently deleted because they have a smaller progress value than the minimal progress value among the unprocessed states M_2 , M_3 and M_4 . The conceptual sweep-line is shown as a vertical dashed line, immediately to the left of the unprocessed states.

Exploring in least-progress-first order means that either M_2 or M_3 will be explored next. When both have been explored, the sweep-line moves to the right and M_2 and M_3 are deleted, giving the situation shown in Fig. 2 (b). M_4 , M_5 and M_6 will be explored and eventually the situation shown in Fig. 2 (c) will be obtained. When M_8 is explored two regress edges are identified, one going to the previously explored state M_6 and the other to the unexplored state M_9 . Note that the algorithm does not know that M_6 was previously explored, as it was deleted. The algorithm marks both M_6 and M_9 as persistent and flags them as roots for a subsequent sweep. In the subsequent sweep, M_{10} is discovered, along with the re-exploration of M_4 , M_7 and M_8 . Because M_6 and M_9 are persistent, the regress edges to M_6 and M_9 discovered in the re-exploration of M_8 do not induce a further sweep. The correctness of the sweep-line algorithm (both termination and full OG coverage) was proved in [24].

One drawback of the sweep-line method is that users need to define and supply their own progress mapping. Steps have been taken towards automatic generation of ψ for low-level Petri nets [33] and compositional systems [23].

3 The Internet Open Trading Protocol (IOTP)

IOTP [3] focuses on consumer-to-business e-commerce applications. It defines five *trading roles* to identify the different roles that organisations can assume while trading. These are *Consumer*, *Merchant*, *Payment Handler* (a bank), *Delivery Handler* (a courier firm) and *Merchant Customer Care Provider*. The core of IOTP is an *Authentication* transaction and five payment-related transactions named *Purchase*, *Deposit*, *Withdrawal*, *Refund* and *Value Exchange*. Each transaction comprises a sequence of IOTP message exchanges between trading roles, where each IOTP message comprises a set of pre-defined trading blocks. IOTP [3] currently uses HTTP [10] as its transport mechanism.

3.1 Document Exchanges and Transactions

IOTP defines a set of document exchanges as building blocks for creating transactions. These are: *Authentication*, *Brand Dependent Offer*, *Brand Independent Offer*, *Payment*, *Delivery*, and *Payment-and-Delivery*. An Authentication transaction consists of just an Authentication (document) exchange. A Purchase transaction comprises an optional Authentication, an Offer (either a Brand Dependent Offer or a Brand Independent Offer), and then, a Payment exchange, a Payment followed by a Delivery exchange, or a Payment-and-Delivery exchange. A Deposit, Withdrawal, or Refund transaction starts with an optional Authentication, an Offer, and a Payment exchange. Finally, a Value Exchange transaction begins with an optional Authentication followed by an Offer and two Payment exchanges in sequence.

Below, we consider an example of a Purchase transaction comprised of an Authentication, a Brand Dependent Offer, a Payment and a Delivery exchange. Figure 3 shows a possible sequence of messages exchanged between the four trading roles involved in the transaction.

In the beginning the Consumer decides to buy goods and so sends a Purchase Request (event 1) to the Merchant. This event initiates a Purchase transaction, however it is not part of Baseline IOTP [3] and is handled by HTTP [10].

Upon receiving the Purchase Request, the Merchant starts an Authentication document exchange (events 2-4) to verify the *bona fides* of the Consumer. In IOTP's terminology, the

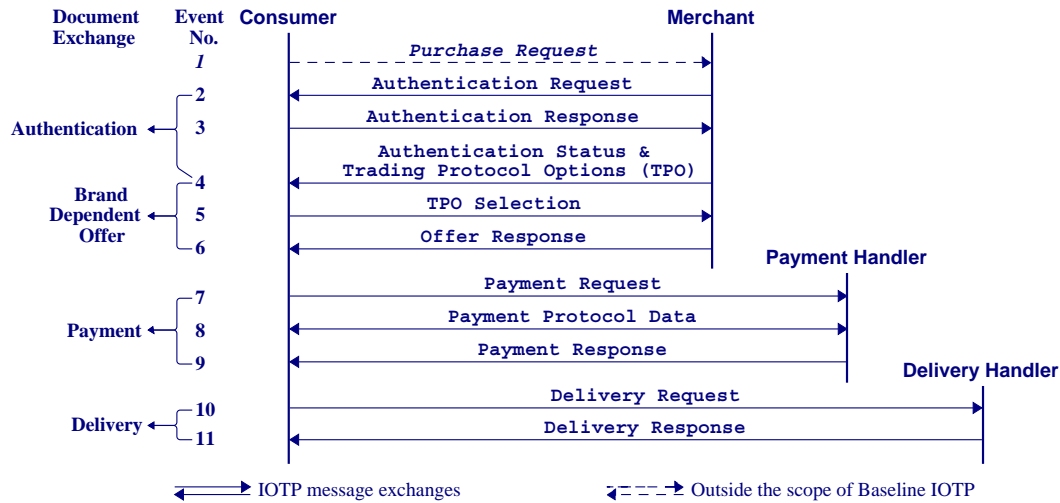


Fig. 3. A possible sequence of message exchanges in a Purchase transaction.

Merchant acts as the *Authenticator* and the Consumer the *Authenticatee*. At first, an Authentication Request is issued by the Merchant (event 2), specifying the authentication algorithm to be used. As a result, the Consumer replies with an Authentication Response containing the authentication data obtained using the above algorithm (event 3). After verifying the Consumer's response, the Merchant generates an Authentication Status indicating that the authentication is successful (part of event 4).

Once the authentication completes, the Merchant continues to a Brand Dependent Offer document exchange in our example (events 4-6) by providing the Consumer a list of Trading Protocol Options (TPO). This includes the available payment methods and associated payment protocols. The message combining the TPO and the above Authentication Status is then sent to the Consumer (event 4). The Consumer chooses one of the options, and sends it back as a TPO Selection (event 5). The Merchant uses the selection to create and send back an Offer Response (event 6), which contains details of the goods to be purchased together with payment and delivery instructions.

Next, a Payment document exchange starts between the Consumer and the Payment Handler (events 7-9). After checking the Offer Response for purchase details, the Consumer sends the Payment Handler a Payment Request (event 7). The Payment Handler checks the Payment Request, and if valid, the payment is conducted using Payment Protocol Data exchanges (event 8) as determined by the encapsulated payment protocol (e.g., Secure Electronic Transaction). After the payment protocol data exchange has finished, the Payment Handler sends a Payment Response (event 9) containing the payment result (e.g., receipt).

Finally, a Delivery document exchange is carried out between the Consumer and the Delivery Handler (events 10-11). After checking the Payment Response, the Consumer sends the Delivery Handler a Delivery Request (event 10). The Delivery Handler schedules the delivery and sends the Consumer a Delivery Response (event 11) containing details of the delivery, and possibly the actual delivery if the goods are electronic (e.g., an e-journal).

It should be mentioned that in a Brand Independent Offer the TPO Selection in event 5 does not occur. A Brand Dependent Offer occurs when the Merchant offers some additional benefit (e.g., price discount) in the Offer Response that depends on the specific *payment brand* (e.g., VISA or MasterCard) chosen in the Consumer's TPO Selection. In the Brand Independent Offer, the Offer Response is independent of the TPO and so the TPO Selection

(event 5) does not happen. Also, IOTP defines a combined TPO and Offer Response message (combining events 4 and 6) for a Brand Independent Offer.

3.2 Transaction Cancellation and Error Handling

A Cancel message is used for transaction cancellation and an Error message for reporting errors and instigating retransmissions. A transaction may be cancelled by any trading role engaged in that transaction. For example, in the Purchase transaction shown in Fig. 3, the Merchant would cancel the transaction if the Consumer's Authentication Response failed. Error handling is concerned with how trading roles handle technical errors and exceptions that occur during a transaction. For example, in Fig. 3, the Merchant may re-send the TPO upon reception of an Error message when expecting the Consumer's TPO Selection. Also, IOTP defines a *message identifier* to uniquely identify IOTP messages at each local trading role. Only duplicates have the same message identifier.

4 A Revised IOTP CPN Model

RFC 2801 [3] contains an informal narrative description of IOTP and suffers from ambiguities and incompleteness. We have created a CPN model of IOTP for six Authentication and Payment-related transactions [29] and further improved it to include procedures for error handling and arbitrary cancellation [27]. Analysis of the IOTP CPN model [28] revealed two main errors in the current design of IOTP, where the Payment-related transaction fails to terminate correctly. This motivated the development of a revised IOTP specification to eliminate the identified errors. To see if the revised protocol was correct, we revised the previous IOTP CPN [27]. In this section, we describe the revised IOTP CPN model only to the level of detail necessary to understand the derivation of the progress mappings in Sect. 5. A presentation of the complete CPN model can be found in [26].

4.1 Net Structure

Figure 4 shows the *hierarchy page* for the revised IOTP CPN. There are 31 pages organised into four hierarchical levels, giving a logical structure that can be validated against RFC 2801.

The first (top) level has one page named IOTP_TopLevel. The second level comprises four pages: Consumer, Merchant, PHandler and DHandler, corresponding to four trading roles¹. We refer to these pages as *trading role pages*. Each trading role page has a set of subpages specifying the possible Authentication and payment-related transactions for that trading role. All these subpages, which we call *transaction pages*, constitute the third level of the model. The initial letter of a trading role is used as a suffix of the name of transaction pages modelled for that trading role. For example, the page Consumer has four subpages modelling six transactions for the Consumer. The three transactions Deposit, Withdrawal and Refund use the same procedure and therefore are modelled on one page named Deposit_C/Withdrawal_C/Refund_C. Each transaction page is further decomposed into a set of subpages modelling the document exchanges that are used to construct the transaction as well as error handling and cancellation procedures. All these subpages, which we call *exchange level pages*, constitute the fourth level of the model. For example, the page Purchase_C has six subpages modelling the six document exchanges used to implement the Purchase transaction for the Consumer and two others for

¹ The Merchant Customer Care Provider, currently not used in any transaction, is not modelled.

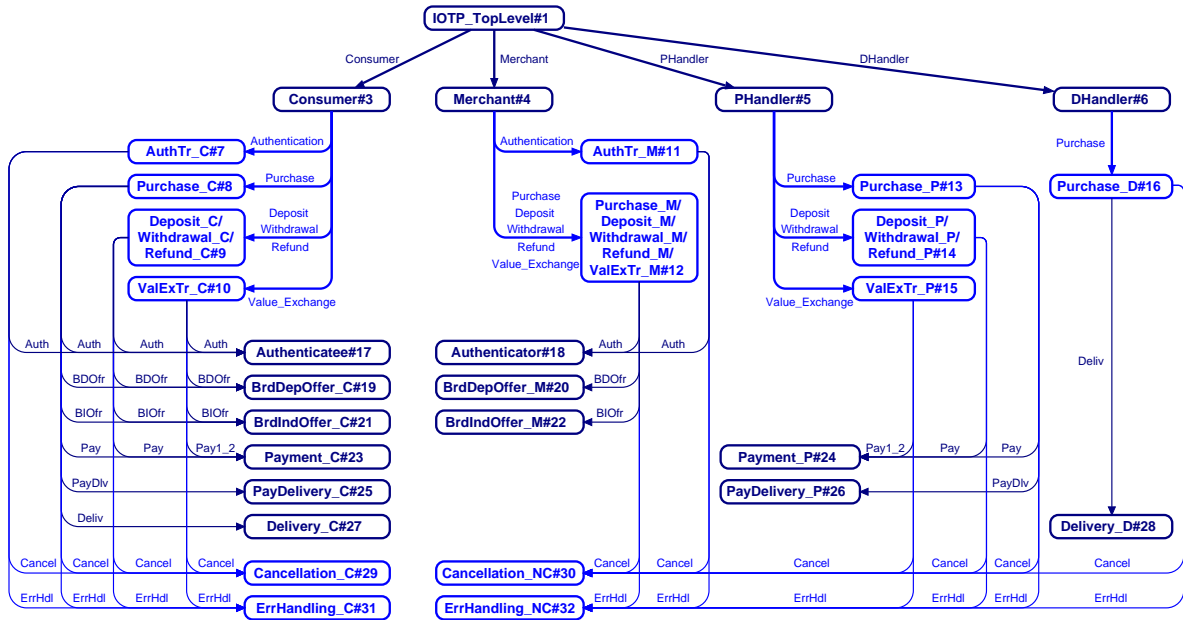


Fig. 4. The Hierarchy page.

error handling and cancellation. Since a document exchange involves two trading roles, we have modelled each exchange as a pair of pages - one for each of the trading roles involved in the document exchange. For example, the pages Authenticatee and Authenticator in Fig. 4 represent an Authentication exchange where the Consumer is authenticated by the Merchant.

4.2 Global Declarations

Figure 5 shows part of the global declarations that define the colour sets and variables for IOTP messages and trading role states. The declarations not shown in Fig. 5 define functions used in the model.

IOTP messages are modelled by the colour set `lotpMsg` (line 15) as a *list of trading blocks*, as derived from their XML definition in RFC 2801. The colour set `TradingBlk` (lines 8-14) is defined as the *union* of different kinds of trading blocks of IOTP messages. A trading block describes several attributes, some of which are specified by the first four colour sets (lines 1-5). The colour set `TwoErrAttrsOfErrComp` (line 6) is defined as a *product* of the two colour sets `Severity` (line 4) and `ErrorCode` (line 5), and models an *Error Block* containing these two attributes. The colour set `MsgId` (line 17), defined as *integers*, models the message identifier. The colour set `Message` (line 18), defined as the product of `lotpMsg` and `MsgId`, specifies that each IOTP message has a message identifier. Since IOTP currently operates over HTTP, we consider a reliable transport medium with no loss, duplication or re-ordering. Because of this, a list of Messages is defined by `MsgQueue` (line 20) for use when modelling a First-In-First-Out queue of messages between trading roles. The colour set `TradingRole` (line 21) enumerates the four trading roles. The colour set `TRxTRxMQ` (line 22) is defined as a product of a sender `TradingRole`, a receiver `TradingRole` and the `MsgQueue` between the two trading roles.

Trading role states are modelled by the colour set `State` (line 31), which is the product of the following five colour sets. `InternalState` (lines 25-26) specifies the eight internal states of a trading role in a document exchange. `Exchange` (line 27) represents the six document exchanges and a value `NoExch` indicating no document exchange occurs. `R_Counter` (line 29) models the message retransmission counter used in the error handling procedure, which increments its

```

(* Selected Attributes *)
1 color lotpTransType = with Authentication | Purchase | Deposit | Withdrawal | Refund |
2   ValueExchange;
3 color DelivExch = with True | False;
4 color Severity = with TransientError | HardError;
5 color ErrorCode = with MsgErr | MsgBeingProc;
6 color TwoErrAttrsOfErrComp = product Severity * ErrorCode;
7 var trtype: lotpTransType; var dlv: DelivExch;

(* Trading Blocks and IOTP Messages *)
8 color TradingBlk = union TransRefBlk:lotpTransType +
9   AuthReqBlk + AuthRespBlk + AuthStatusBlk +
10  TpoBlk + TpoSelectionBlk + OfferRespBlk:DelivExch +
11  PayReqBlk + PayExchBlk + PayRespBlk +
12  DeliveryReqBlk + DeliveryRespBlk +
13  ErrorBlk:TwoErrAttrsOfErrComp +
14  CancelBlk;
15 color lotpMsg = list TradingBlk;
16 var m, rm, sm: lotpMsg;

(* IOTP Message with Message Identifier *)
17 color MsgId = int;
18 color Message = product lotpMsg * MsgId;
19 var id, rid, sid: MsgId;

(* Message Queue between Trading Roles *)
20 color MsgQueue = list Message;
21 color TradingRole = with Consumer | Merchant | PHandler | DHandler;
22 color TRxTRxMQ = product TradingRole * TradingRole * MsgQueue;
23 var role: TradingRole;
24 var q, rq, sq: MsgQueue;

(* Trading Role State and Message Buffer *)
25 color InternalState = with READY | COMPLETED | CANCELLED |
26   LISTEN | WAIT | HOLD | HOLD_WAIT | FINISHED;
27 color Exchange = with Auth | BDOfr | BIOfr | Pay | PayDiv | Deliv | NoExch;
28 val RCmax = 1; (* Maximum number of message Re-transmissions *)
29 color R_Counter = int with 1..RCmax; (* Message Re-transmission Counter *)
30 color MldxRC = product MsgId * R_Counter;
31 color State = product InternalState * lotpTransType * Exchange * MldxRC * MsgId;
32 color StaxBfr = product State * lotpMsg;
33 var s: InternalState; var exch: Exchange; var rc: R_Counter;

```

Fig. 5. Definitions of colour sets and variables for the revised IOTP CPN model.

value by one upon each message retransmission until it reaches the maximum value RCmax (line 28). MldxRC (line 30) records the MsgId of the previously sent message along with its R_Counter. The last colour set in State represents the MsgId of the previously received message, used to check for duplicates in the error handling procedure. The colour set StaxBfr (line 32) combines the State of a trading role with the lotpMsg residing in the message retransmission buffer for that trading role.

4.3 The Top Level Page

Figure 6 shows the IOTP_TopLevel page that provides an abstract view of IOTP. The four substitution transitions, Consumer, Merchant, Payment Handler and Delivery Handler, represent IOTP's procedures for the corresponding trading roles. The place Transport, typed by colour set TRxTRxMQ, models the transport medium over which the trading roles communicate.

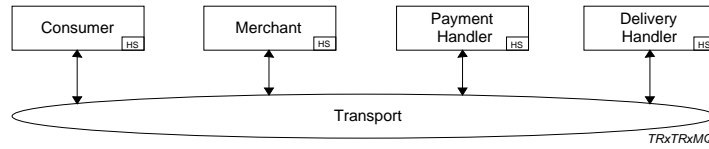


Fig. 6. The IOTP_TopLevel page.

4.4 Trading Role Pages

Both Consumer and Merchant are involved in all six transactions. Figure 7 depicts the two corresponding pages Consumer and Merchant, which we use as representative examples of the four trading role pages in the CPN model. Each transaction is abstractly represented by a substitution transition which has the same name as the transaction. Each place has a name starting with C or M, is typed by the product set *StaxBfr*, and models the state of the Consumer or Merchant with its message retransmission buffer in one of the six transactions. For brevity, we refer to these places as *C_places* or *M_places*. The exception is the Transport place on each page in Fig. 7 which has been described above.

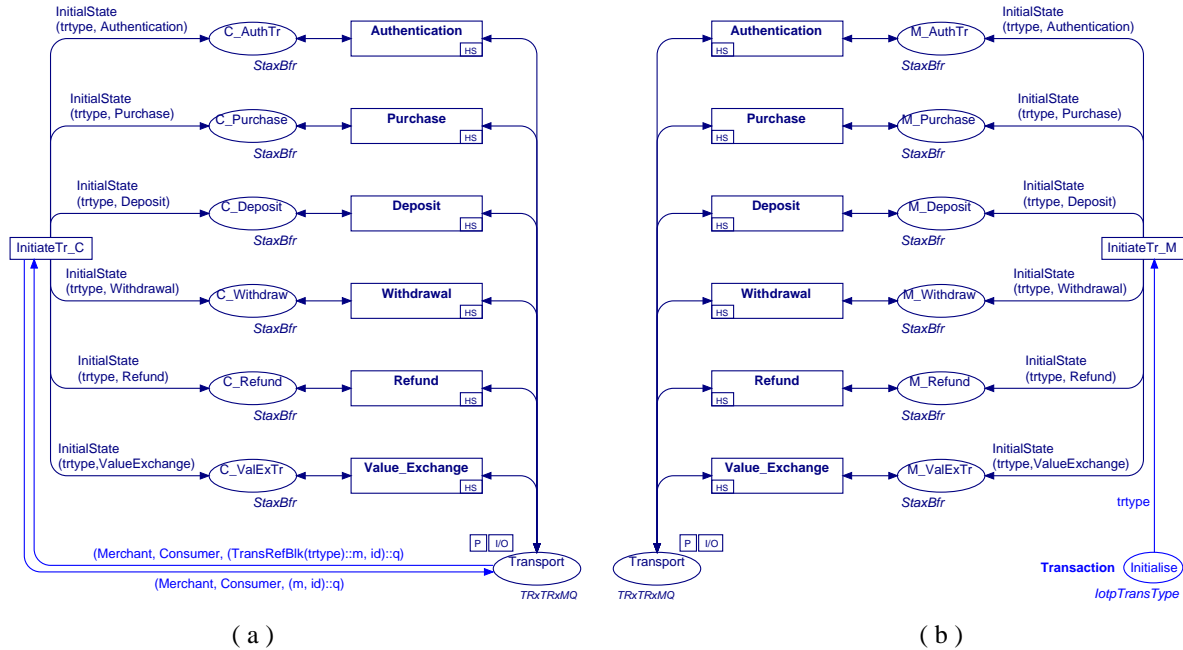


Fig. 7. Two trading role pages: (a) Consumer and (b) Merchant.

In Fig. 7 (b), the page Merchant has another place called Initialise (bottom right), which stores the transaction to be initiated by the Merchant. It is typed by the colour set *lotpTransType* and has an initial marking given by a constant Transaction that can be set to any of the six transactions. Place Initialise has an output arc to the only ordinary transition called *InitiateTr_M*. The arc is inscribed by variable *trtype* representing any token available in place Initialise. The arc from transition *InitiateTr_M* to each *M_place* is inscribed by a function named *InitialState* (see below), modelling how the initial state of the Merchant is determined upon a certain transaction type.

Figure 8 defines the function *Initialise*. It has two parameters: *trtype* and *trtypevalue*, both of type *lotpTransType*. If the two parameters have the same value, the function returns $1'((\text{READY}, \text{trtype}, \text{NoExch}, (0,0), 0), [])$ (of *StaxBfr*), representing the initial state of a trading role. If not, no token value is returned (defined as empty). For example, in Fig. 7 (b), if

Transaction has the value Purchase, upon occurrence of InitiateTr_M, a token that indicates the Merchant is ready to carry out a Purchase transaction will be added to place M_Purchase.

```

1 fun InitialState(trtype, trtypevalue:IotpTransType): StaxBfr =
2   = if trtype=trtypevalue
3     then 1'((READY, trtype, NoExch, (0, 0), 0), [])
4     else empty

```

Fig. 8. Function used to determine the initial state of a trading role.

The page Consumer in Fig. 7 (a) has only one ordinary transition, InitiateTr_C, used to initiate an appropriate transaction for the Consumer on receiving the first IOTP message from the Merchant. It has both input and output arcs associated with place Transport. The input arc has an inscription specifying the first IOTP message (represented by (TransRefBlk(trtype)::m, id)) from the Merchant. The output arc indicates that processing of a *Transaction Reference Block*, which conveys transaction type information (TransRefBlk(trtype)) in an IOTP message, is complete, and thus the block is removed from the input message buffer. Similarly, the token returned by function InitialState is added to one of the six C_places on occurrence of transition InitiateTr_C, modelling that the corresponding transaction has been initiated at the Consumer.

5 Sweep-line Exploration of IOTP

Analysis of the revised IOTP focuses on the six Authentication and Payment-related transactions. For each transaction, there are also different cases we can analyse by changing the maximum value of the message retransmission counter ($RCmax$, line 28 of Fig. 5) for each of the four trading roles. Once the number of retransmissions reaches $RCmax$, the transaction will be cancelled. However, RFC 2801 does not define a value for $RCmax$. We therefore have modelled $RCmax$ as an unbounded parameter, resulting in an infinite number of possible configurations. In [26] the revised IOTP CPN was analysed for $0 \leq RCmax \leq 3$. When $RCmax$ was increased to 4, the OG of both the Purchase and Value Exchange transactions became too large to manage with the available computer resources at that time. It is our intention to alleviate the problems of state explosion for the revised IOTP with $RCmax$ greater than 3 by applying the sweep-line method.

Two approaches were taken to define a progress mapping for the application of the sweep-line method to the revised IOTP CPN. The first was to look only at features common to many protocols, such as sequence numbers and retransmission counters. The second was to develop a progress mapping based on specific features of IOTP and hence take advantage of behavioural properties of the IOTP CPN. These progress mappings are presented in Sections 5.1 and 5.2 respectively. A third progress mapping, combining the generic and specific progress measures, is presented in Section 5.3.

5.1 Generic Progress Mapping

Sequence numbers and retransmission counters are features common to many protocols. It would be nice if a generic progress mapping, giving reasonable performance, could be derived from these common features.

Each trading role in IOTP maintains its own retransmission counter and sequence number (the message identifier) for the message it most recently transmitted. The message identifier is used to differentiate messages when interacting with other trading roles and retransmissions

are used to recover from (processing delay and transient) errors. As an example, let us consider the Consumer trading role. In a Purchase transaction, the Consumer trading role keeps this information in the token on place `C_Purchase` on the Consumer page. This token belongs to the `StaxBfr` colour set (see Fig. 5). To obtain the message identifier and retransmission counter of the Consumer we can define functions `GetMessIDConsumer` and `GetRCConsumer` to extract them from the token on `C_Purchase` in a given marking $M \in [M_0]$ of the IOTP CPN. Analogous functions can be defined for the Merchant, Payment Handler and Delivery Handler.

Let us define $TR = \{\text{Consumer, Merchant, Payment Handler, Delivery Handler}\}$ to be the set of four trading roles. One strategy to create a *generic* progress mapping would be a straight sum of the values of these variables over all trading roles, i.e. providing the mapping

$$\psi_{generic_1}(M) = \sum_{tr \in TR} (\text{GetMessID}_{tr}(M) + \text{GetRC}_{tr}(M))$$

Intuitively, this may not be the best solution, as many combinations of message identifiers and retransmission counters will result in the same progress value. It is desirable to have many progress values to give the potential for greater reduction, but there is a trade-off with the number of regress edges introduced (we want this to be low) and also the number of states discovered with higher progress values that are not immediately explored but are nonetheless stored in memory and cannot be deleted until later. We also note that, in general, a sequence number represents a *more significant* measure of progress than a retransmission counter, as generally a retransmission counter is kept for each message, then reset for the next message. We use this knowlegde to construct a more elaborate progress mapping by giving a weight to the message identifier component of the progress values. This weight is $(RCmax + 1)$ and is one larger than the maximum possible value of `GetRCtr`. This ensures that an increment in the message identifier is always more significant than any increment in the retransmission counter. A second generic progress mapping is then defined as:

$$\psi_{generic_2}(M) = \sum_{tr \in TR} ((RCmax + 1) * \text{GetMessID}_{tr}(M) + \text{GetRC}_{tr}(M))$$

This is the generic progress mapping we use in our experiments in Section 6.1. Note that $\psi_{generic_2}$ may not exhibit the optimum combination of message identifiers and retransmission counters with respect to the IOTP CPN. For example, each trading role could be given its own weight to further differentiate the progress of states, but assigning anything but an arbitrary weighting to each trading role may require IOTP-specific knowledge, hence violating our intentions for the general nature of this progress mapping.

An important feature of this progress mapping is its incorporation of the parameter $RCmax$. As $RCmax$ increases, the set of generated progress values also increases. It is hoped that this progress mapping will *scale* well with $RCmax$. Scalability is a desirable property of a progress mapping.

5.2 IOTP-Specific Progress Mapping

In IOTP, all six Authentication and Payment-related transactions are implemented via combinations of document exchanges. Within a transaction, progress is exhibited by the execution of successive document exchanges. Within a document exchange, progress is exhibited by the internal state changes of the trading roles. Accordingly, we can define a progress mapping which is based on the current document exchange in the transaction and also the internal states of trading roles within each document exchange.

Table 1. The four mappings ψ_m , ψ_c , ψ_{ph} and ψ_{dh} enumerating and ordering the internal states of the four trading roles (Merchant, Consumer, Payment Handler and Delivery Handler) in a document exchange.

Trading Role Internal States	$\psi_m(M)$	$\psi_c(M)$	$\psi_{ph}(M)$	$\psi_{dh}(M)$
READY	1	1	1	1
LISTEN	2	4	-	-
HOLD	-	5	2	-
WAIT	3	2	3	2
HOLD_WAIT	4	3	4	-
FINISHED	5	6	-	-
COMPLETED	6	7	5	3
CANCELLED	7	8	6	4
no state	0	0	0	0

For a given transaction, the sequence of internal states of each trading role is fixed. The internal state of each trading role can be obtained by considering the token on the place corresponding to the given transaction type on the corresponding Trading Role page, as described in Section 4.4 for the Consumer and Merchant. To capture the progress from the trading role internal states within a document exchange, we define four progress mappings ψ_m , ψ_c , ψ_{ph} and ψ_{dh} , which respectively enumerate the internal states of the Merchant, the Consumer, the Payment Handler and the Delivery Handler according to the progress represented by each. Table 1 lists the sequences of trading role internal states for a Purchase transaction. Not all trading roles can enter all states and so no mapping has been defined for these cases (indicated by ‘-’ in the table.) The bottom row of Table 1 indicates that trading roles that have not yet entered any state (no state) map to 0.

We also define a mapping to capture the progress of the execution of document exchanges in a transaction. An Authentication (Auth) exchange always takes place at the beginning of a transaction (that requires authentication of the Consumer). An Offer exchange, either Brand Dependent (BDOfr) or Brand Independent (BIOfr), never occurs before an Authentication exchange but must occur before any other document exchange. A Payment (Pay1) or a Payment-and-Delivery (PayDlv) exchange is carried out after an Offer exchange only, and a Delivery (Deliv) exchange happens after a Payment exchange only. A Value Exchange transaction involves two Payment exchanges, where the second Payment (Pay2) must follow the first Payment (Pay1) exchange. It can be seen that there is always a choice between a Brand Dependent Offer and a Brand Independent Offer in any Payment-related transaction.

This knowledge of the behaviour of IOTP is useful in two ways. The first is that it allows us to reason about the order in which the document exchanges occur. IOTP is sequential in its ordering of document exchanges within a transaction and thus we can give a higher progress value to those states in which the trading roles have progressed further. Secondly, we can use this knowledge to explore the part of the occurrence graph for each document exchange combination sequentially. Conceptually, when arranging the OG in a least-progress-first manner, this is tantamount to reshaping the OG to be *long and thin* rather than *short and wide* to minimise the number of states with a given progress value, which, in our experience, is beneficial to the performance of the sweep-line method. To illustrate this, we describe the effect of reshaping the OG for a Purchase transaction. The Purchase transaction OG can be initially divided into two parts based on the choice of a Brand Dependent or Brand Independent Offer. We choose to always explore the part relating to the Brand Dependent Offer first, and then the part involving the Brand Independent Offer. Similarly, there is always a choice between a Payment followed by a Delivery exchange and a (combined) Payment-and-Delivery exchange in a Purchase transaction, and so the corresponding transaction OG following each Offer

exchange can be further divided into two parts. In this case, we choose to explore first the part with Payment followed by Delivery, then the part with the Payment-and-Delivery exchange.

By defining a mapping that reflects these observations, a transaction OG can be generated in such a way that on one hand the correct progress of the transaction is preserved (i.e. the progress mapping results in monotonic progress) and on the other hand a narrower OG is obtained. Such a mapping is defined in Table 2 as ψ_{exch_comb} which enumerates all possible combinations of document exchanges executed at each of the four trading roles. Similarly to identifying the internal state of each trading role, the document exchange that a trading role is involved in can be obtained from the token on the place corresponding to the given transaction on the corresponding Trading Role page (see Section 4.4). A trading role with no document exchange information (represented by '-') is not involved, or has not yet started a transaction. In Table 2, the value $\psi_{exch_comb}(M)$ is incremented by 26 for each different combination of document exchanges. The offset of 26 was chosen as it is one greater than the maximum sum over all trading roles of the trading role internal state progress mappings ($7+8+6+4=25$, see Table 1) in a document exchange. This is important, as an offset of one greater than 25 guarantees that there will be no overlap between the progress mappings in ψ_{exch_comb} for any two combinations of document exchange states at each trading role.

Having identified sources of IOTP-specific progress, we can now define the full progress mapping $\psi_{specific}$ for IOTP as follows:

$$\psi_{specific}(M) = \psi_{exch_comb}(M) + \psi_m(M) + \psi_c(M) + \psi_{ph}(M) + \psi_{dh}(M)$$

The performance of the sweep-line with this progress mapping is discussed in Section 6.2.

5.3 Combination of Generic and Specific Progress Mapping

The mapping $\psi_{specific}$ takes advantage of knowledge of the sequence of states that trading roles progress through, and also of the sequential nature of IOTP operations. It does, however, lack potential for scalability. Ideally, we would like a progress mapping for the IOTP CPN to incorporate the parameter $RCmax$ in such a way as to scale with $RCmax$. The generic progress mapping $\psi_{generic_2}$ 'grows' with the parameter $RCmax$ so by combining this with the specific progress mapping we hope to obtain a progress measure with the advantages of both.

Rather than simply adding together the progress values $\psi_{generic_2}(M)$ and $\psi_{specific}(M)$ for each state M , we take note of the fact that message identifiers and retransmission counters increment *within* a particular document exchange. When combining the progress measures, we give a weighting to the IOTP-specific progress values to make an increment in the IOTP-specific progress values more significant than any increment in the generic progress values. This weight must be one more than the maximum value of $\psi_{generic_2}$. To determine this value we must first determine an upper bound on the message identifier values.

IOTP has 15 different (non-error and non-cancel) message types, each of which may be used at most once during a transaction. (We omit the details of these messages for brevity.) Each new message sent by a trading role is given a message identifier one greater than the previously sent message. Retransmissions of the same message are given the same message identifier as the original. The receiver of one of the 15 message types may generate an error message requesting a retransmission. Each error message is given a new (incremented) message identifier. Thus, in the worst case, sending a (non-error and non-cancel) message to a peer trading role will cause the receiving trading role's internal message identifier to increment by $RCmax$ (after $RCmax$ retransmissions) or by $RCmax + 1$ if the message being sent stimulates

Table 2. The mapping ψ_{exch_comb} enumerating all possible combinations of document exchanges at each of the four trading roles in a transaction.

<i>Merchant</i>	<i>Consumer</i>	<i>PHandler</i>	<i>DHandler</i>	$\psi_{exch_comb}(M)$
NoExch	-	-	-	0
Auth	-	-	-	26
Auth	NoExch	-	-	52
Auth	Auth	-	-	78
BDOfr	-	-	-	104
BDOfr	NoExch	-	-	130
BDOfr	Auth	-	-	156
BDOfr	BDOfr	-	-	182
BDOfr	Pay1	-	-	208
BDOfr	Pay1	NoExch	-	234
BDOfr	Pay1	Pay1	-	260
BDOfr	Deliv	Pay1	-	286
BDOfr	Deliv	Pay1	NoExch	312
BDOfr	Deliv	Pay1	Deliv	338
BDOfr	PayDlv	-	-	364
BDOfr	PayDlv	NoExch	-	390
BDOfr	PayDlv	PayDlv	-	416
BDOfr	Pay2	Pay1	-	442
BDOfr	Pay2	Pay2	-	468
BIOfr	-	-	-	494
BIOfr	NoExch	-	-	520
BIOfr	Auth	-	-	546
BIOfr	BDOfr	-	-	572
BIOfr	Pay1	-	-	598
BIOfr	Pay1	NoExch	-	624
BIOfr	Pay1	Pay1	-	650
BIOfr	Deliv	Pay1	-	676
BIOfr	Deliv	Pay1	NoExch	702
BIOfr	Deliv	Pay1	Deliv	728
BIOfr	PayDlv	-	-	754
BIOfr	PayDlv	NoExch	-	780
BIOfr	PayDlv	PayDlv	-	806
BIOfr	Pay2	Pay1	-	832
BIOfr	Pay2	Pay2	-	858

a response from the receiving trading role. Thus $15(RCmax + 1)$ is an upper bound on message identifiers, where 15 is the number of messages and $(RCmax + 1)$ is the maximum increase in message identifier induced by each message. Cancel messages terminate the transaction and are never retransmitted and thus are not taken into account.

Based on this, an upper bound on the value of $\psi_{generic_2}$ is $4(15(RCmax + 1)^2 + RCmax)$ where $RCmax$ is the maximum possible value of $GetRC_{tr}$ and the 4 comes from the summation over all 4 trading roles. This is quite a conservative upper bound, as each trading role transmits only a subset of the set of 15 messages. The weight needs to be larger than this value, and so we obtain the following combined progress mapping:

$$\psi_{comb}(M) = \psi_{generic_2}(M) + (4(15(RCmax + 1)^2 + RCmax) + 1) * \psi_{specific}(M)$$

The performance and scalability of this progress measure is discussed in Section 6.3.

In [12] the sweep-line method was used to verify the Wireless Transaction Protocol (WTP) [36] within the Wireless Application Protocol (WAP) [37]. Conventional occurrence graph generation could only be used for retransmission counters up to 5 due to state explosion. In contrast to our method of mapping directly to \mathbb{N} , a monotonic progress mapping was

defined as a *progress vector* in \mathbb{Z}^5 , embedded into the total ordering (\leq) on integers. The five elements of progress were the state of each of the two interacting protocol entities, the retransmission counters in each of the two interacting protocol entities and the inverse of the number of messages left in the channel once one or both of the protocol entities had reached their final states. This allowed WTP to be verified using sweep-line for retransmission counters up to 8, the value specified in [36] for WAP operating over IP networks. The progress mapping in [12] is similar to our definition of ψ_{comb} as we also use protocol entity states and retransmission counters to gauge the progress of IOTP. In addition, we use message identifiers but we do not use the number of messages left in the channel after a transaction has completed.

6 Experimental Results

The IOTP CPN was analysed with the computer tool Design/CPN, using conventional OG generation and a prototype implementation of the sweep-line method using the generic progress mapping $\psi_{generic_2}$, the IOTP-specific progress mapping $\psi_{specific}$ and the combined progress mapping ψ_{comb} . The results are shown in Tables 3, 4 and 5 respectively. We present results from the Purchase Transaction only, as this transaction exercises all elements of the IOTP CPN model.

When analysing IOTP, one of the desired properties of IOTP is valid transaction termination. If a transaction terminates properly, each of the trading roles that have started the transaction must enter a valid terminal state, i.e. **COMPLETED**, indicating that a transaction terminates successfully, or **CANCELLED**, indicating the transaction is cancelled. We are able to check the dead markings obtained and verify that they all have this property.

To eliminate differences in performance due to the efficiency of state storage, a custom hashing function for storing states was implemented and used in both the conventional generation and the sweep-line generation. All experiments were conducted on a 2.6GHz Pentium 4 with 1Gb of memory.

It is worth noting that the node deletion mechanism in the sweep-line implementation used in this paper differs from the experimental implementation used in [6] and [24], where node deletion was initiated every time a statically defined number of new nodes were explored. In addition, as reported in [24], the former method for node deletion performed poorly and became the dominant time factor when exploring large occurrence graphs. The implementation used in this paper matches the algorithm more closely, in that states with a progress value $n \in \mathbb{N}$ are deleted as soon as the minimum progress value over all unprocessed states is greater than n .

6.1 Generic Progress Mapping

Table 3 contains the OG statistics when using the sweep-line method with $\psi_{generic_2}$. Column 1 shows the value of the *RCmax* parameter for which the OG was generated. The second, third and fourth columns show the number of states and arcs generated by conventional exploration (the number of states and arcs in the full OG) and the total time taken. Column 5 shows the peak number of states stored in memory at any one time using the sweep-line method. Columns 6 and 7 show the total number of states swept (explored) and arcs traversed with the sweep-line method. The total time for sweep-line exploration is shown in column 8. The number of dead markings discovered is shown in column 9. Columns 10 and 11 show the factor of reduction in space and time when using the sweep-line method as compared to conventional generation. We were limited to $RCmax = 4$ for conventional generation and

$RCmax = 5$ with the sweep-line method before computer memory and time constraints forced us to abandon generation. We were unable to generate the full OG (using the conventional method) for $RCmax = 5$ so the number of states and arcs in the full state space and the theoretical reduction are shown in brackets.

The first thing to notice is that this progress mapping is non-monotonic, as indicated by having swept more states than are in the full state space. This was not unexpected, as the message identifiers and retransmission counters reset to 0 at various times during an IOTP transaction. The factor of reduction in states is relatively constant over this range of $RCmax$ values, staying around 1.7 times more efficient in space. This is not a particularly useful reduction. It is worth noting that the progress mapping does not scale as well as we had hoped, hence the relatively constant (but worsening) reduction in space. The reduction in time is due to a lower peak state storage and so each new state does not need to be compared with as many existing states.

Table 3. Sweep-line statistics for the analysis of the IOTP CPN using $\psi_{generic_2}$.

RCmax	Conventional			Sweep-line				Dead markings	Reduction in	
	States	Arcs	hh:mm:ss	Peak	Total	Arcs	hh:mm:ss		space	time
0	2144	5243	00:00:03	1462	2534	6338	00:00:04	85	1.47	0.75
1	12695	37947	00:00:30	7141	17521	52675	00:00:36	85	1.78	0.83
2	47931	161437	00:04:21	26369	62158	208473	00:02:43	85	1.82	1.60
3	142499	518910	00:29:59	84853	176126	635844	00:09:37	85	1.68	3.12
4	361451	1389461	02:55:09	227012	430713	1638636	00:31:04	85	1.59	5.64
5	(816957)	(3266339)	-	523953	946571	3743931	01:33:39	85	(1.56)	-

6.2 IOTP Specific Progress Mapping

Table 4 contains the state space statistics when using the sweep-line method with $\psi_{specific}$. The table format is the same as in Table 3. This progress mapping results in a monotonic measure of progress (this is checked automatically by the tool during exploration) so the total number of states and arcs explored with the sweep-line method is identical to the total number of states and arcs in the full state space, hence we can infer the size of the full state space. The reduction in space and time is better than when using $\psi_{generic_2}$. The space reduction worsens as $RCmax$ increases, indicating that this progress measure also scales poorly. The worsening in reduction is caused by the number of states with a given progress value growing out of proportion with the total number of states, as $RCmax$ increases. This worsening in memory reduction seems to go against the usual trend, as in e.g., [2, 6, 12, 24] where the reduction in space increases with the size of the state space.

Table 4. Sweep-line statistics for the analysis of the IOTP CPN using $\psi_{specific}$.

RCmax	Conventional			Sweep-line				Dead markings	Reduction in	
	States	Arcs	hh:mm:ss	Peak	Total	Arcs	hh:mm:ss		space	time
0	2144	5243	00:00:03	191	2144	5243	00:00:04	85	11.21	0.75
1	12695	37947	00:00:30	1266	12695	37947	00:00:27	85	10.03	1.11
2	47931	161437	00:04:21	5528	47931	161437	00:02:10	85	8.67	2.01
3	142499	518910	00:29:59	18876	142499	518910	00:08:12	85	7.55	3.66
4	361451	1389461	02:55:09	61025	361451	1389461	00:27:02	85	5.92	6.48
5	(816957)	(3266339)	-	163776	816957	3266339	01:22:36	85	(4.99)	-
6	(1690370)	(6961367)	-	385077	1690370	6961367	03:44:06	85	(4.39)	-
7	(3260531)	(13739782)	-	819848	3260531	13739782	10:24:42	85	(3.98)	-

6.3 Combined Progress Mapping

Table 5 contains the state space statistics when using the sweep-line method with ψ_{comb} . The table format is again the same as used previously. The reduction in space obtained by using the combined progress mapping is identical to using $\psi_{specific}$ for small values of $RCmax$ but as $RCmax$ increases, the reduction in space does not worsen as rapidly. The reduction in time is approximately the same as when using $\psi_{specific}$ even though the peak state storage is lower, a fact that we attribute to the more complicated progress mapping function (hence a larger overhead in calculating the progress value for each state).

An unexpected result we discovered was that ψ_{comb} is monotonic, at least for the cases we examined ($0 \leq RCmax \leq 7$). We conjecture that this is due to the events causing a decrease in progress with respect to $\psi_{generic_2}$ are also causing an increase in progress with respect to $\psi_{specific}$. Due to the weighting we have given $\psi_{specific}$ in ψ_{comb} the increase in progress due to a changing trading role internal state or document exchange is greater than the decrease caused by the reset of a message identifier or retransmission counter.

The number of dead markings is constant for all configurations, whereas we would expect the number of dead markings to explode with $RCmax$. The reason is that the IOTP CPN resets retransmission counters upon termination. We can deduce that there are 85 ways that the IOTP CPN can terminate independently of retransmission counter values.

Table 5. Sweep-line statistics for the analysis of the IOTP CPN using ψ_{comb} .

RCmax	Conventional			Sweep-line				Dead markings	Reduction in	
	States	Arcs	hh:mm:ss	Peak	Total	Arcs	hh:mm:ss		space	time
0	2144	5243	00:00:03	191	2144	5243	00:00:04	85	11.21	0.75
1	12695	37947	00:00:30	1266	12695	37947	00:00:27	85	10.03	1.11
2	47931	161437	00:04:21	5528	47931	161437	00:02:10	85	8.67	2.01
3	142499	518910	00:29:59	17163	142499	518910	00:08:10	85	8.30	3.67
4	361451	1389461	02:55:09	46369	361451	1389461	00:27:02	85	7.80	6.48
5	(816957)	(3266339)	-	124693	816957	3266339	01:22:08	85	(6.55)	-
6	(1690370)	(6961367)	-	294108	1690370	6961367	03:44:53	85	(5.75)	-
7	(3260531)	(13739782)	-	628845	3260531	13739782	10:14:15	85	(5.18)	-

7 Conclusions and Future Work

This paper has presented some experiments in applying the sweep-line occurrence graph method to a CPN model of the Internet Open Trading Protocol. We have particularised the sweep-line method to CPNs, where we use the Naturals as the progress values and its usual order relation (\leq). This allows us to just associate a progress mapping with the CPN. We then present a revised abstract generalised sweep-line algorithm for CPNs, which includes the storing of dead markings and uses set operations.

We derived three progress mappings for the analysis of the IOTP CPN model and presented our intuition and rationale behind each. The first is a progress mapping based on generic properties of protocols. The second is based on properties specific to IOTP. The third is based on a combination of the first two.

The IOTP CPN is parameterised with a maximum number of retransmissions, $RCmax$. Using the combined progress mapping, we were able to analyse the IOTP CPN for larger values of $RCmax$ than was possible using conventional occurrence graph generation. In doing so we have demonstrated that the sweep-line method can be successfully applied to a complex real-life example.

We have represented our progress values as a summation of component progress values. An alternative representation would be to use vectors (as in [12]) to represent the progress values. This would allow easier investigation of different orderings and weightings of the components making up the progress mapping. It also allows use of the compositional sweep-line method [23] for analysis of the IOTP CPN. This method takes advantage of monotonic and non-monotonic components of the measure of progress to further reduce peak state storage. We intend to investigate this approach to determine what gains in space and time can be made.

The design of optimal (or even good) progress mappings for CPN models of practical systems is an open question. When defining progress mappings for IOTP, we have used rules of thumb aimed at giving a good progress mapping with respect to the peak number of states stored and the time required for OG exploration. The rules of thumb are: 1) the progress sources should be weighted according to their significance in the system; 2) regress edges should be avoided or the number kept as low as possible; 3) the number of progress values should be as high as possible; and 4) a progress mapping that works well for small configurations of the system will also work well for larger configurations. Currently, these rules are based on intuition and a limited amount of practical experience. As part of future work it would be useful to develop some formal and/or further practical justification for these rules to obtain a better understanding of what constitutes a good progress mapping. The phenomenon observed in this paper where the reduction decreased for larger configurations suggests that 4) is not a valid assumption in general. It would also be of interest to determine the best possible progress mapping that can be defined for the system. This would allow the analyst to gauge the potential reduction that can be obtained with the sweep-line method.

References

1. J. Billington, G. E. Gallasch, and B. Han. A Coloured Petri Net Approach to Protocol Verification. In *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 210–290. Springer-Verlag, 2004.
2. J. Billington, G.E. Gallasch, L.M. Kristensen, and T. Mailund. Exploiting equivalence reduction and the sweep-line method for detecting terminal states. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 34(1):23–37, January 2004.
3. D. Burdett. Internet Open Trading Protocol - IOTP Version 1.0. RFC 2801, IETF, April 2000.
4. D. Burdett, D.E. Eastlake, and M. Goncalves. *Internet Open Trading Protocol*. McGraw-Hill, 2000.
5. S. Christensen, K. Jensen, and L.M. Kristensen. *Design/CPN Occurrence Graph Manual*. Department of Computer Science, University of Aarhus, Denmark. On-line version: <http://www.daimi.au.dk/designCPN/>.
6. S. Christensen, L.M. Kristensen, and T. Mailund. A Sweep-Line Method for State Space Exploration. In *Proceedings of TACAS 2001*, volume 2031 of *Lecture Notes in Computer Science*, pages 450–464. Springer-Verlag, 2001.
7. E.M. Clarke, R. Enders, T. Filkorn, and S. Jha. Exploiting Symmetries in Temporal Logic Model Checking. *Formal Methods in System Design*, 9(1/2):77–104, 1996.
8. Design/CPN Online. <http://www.daimi.au.dk/designCPN/>.
9. E.A. Emerson and A.P. Sistla. Symmetry and Model Checking. *Formal Methods in System Design*, 9(1/2):105–131, 1996.
10. R. Fielding et al. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, IETF, June 1999.
11. P. Godefroid, G.J. Holzmann, and D. Pirotin. State-Space Caching Revisited. *Formal Methods in System Design*, 7(3):227–241, 1995.
12. S. Gordon, L.M. Kristensen, and J. Billington. Verification of a Revised WAP Wireless Transaction Protocol. In *Proceedings of ICATPN'02*, volume 2360 of *Lecture Notes in Computer Science*, pages 182–202. Springer-Verlag, 2002.
13. G.J. Holzmann. Tracing protocols. *AT&T Technical Journal*, 64(10):2413–2433, December 1985.
14. G.J. Holzmann. Algorithms for Automated Protocol Validation. *AT&T Technical Journal*, 69(2):32–44, 1990.

15. G.J. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall, 1991.
16. G.J. Holzmann. An Analysis of Bitstate Hashing. *Formal Methods in System Design*, 13(3):287–305, 1998.
17. InterPay I-OTP.
URL: <http://www.ietf.org/proceedings/01aug/slides/trade-1/index.html>, August 2001.
18. K. Jensen. Condensed State Spaces for Symmetrical Coloured Petri Nets. *Formal Methods in System Design*, 9(1/2):7–40, 1996.
19. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Vol. 1, Basic Concepts*. Springer-Verlag, 2nd edition, 1997.
20. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Vol. 2, Analysis Methods*. Springer-Verlag, 2nd edition, 1997.
21. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Vol. 3, Practical Use*. Springer-Verlag, 1997.
22. L.M. Kristensen, S. Christensen, and K. Jensen. The Practitioner's Guide to Coloured Petri Nets. *International Journal on Software Tools for Technology Transfer*, 2(2):98–132, 1998.
23. L.M. Kristensen and T. Mailund. A Compositional Sweep-line State Space Exploration Method. In *Proceedings of FORTE'02*, volume 2529 of *Lecture Notes in Computer Science*, pages 327–343. Springer-Verlag, 2002.
24. L.M. Kristensen and T. Mailund. A Generalised Sweep-Line Method for Safety Properties. In *Proceedings of FME'02*, volume 2391 of *Lecture Notes in Computer Science*, pages 549–567. Springer-Verlag, 2002.
25. L.M. Kristensen and T. Mailund. Efficient Path Finding with the Sweep-Line Method using External Storage. In *Proceedings of the International Conference on Formal Engineering Methods (ICFEM'03)*, volume 2885 of *Lecture Notes in Computer Science*, pages 319–337. Springer-Verlag, 2003.
26. C. Ouyang. *Formal Specification and Verification of the Internet Open Trading Protocol using Coloured Petri Nets*. PhD thesis, Computer Systems Engineering Centre, School of Electrical and Information Engineering, University of South Australia, Adelaide, Australia, June 2004.
27. C. Ouyang and J. Billington. An improved formal specification of the Internet Open Trading Protocol. In *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC 2004)*, pages 779–783, Nicosia, Cyprus, 14-17 March 2004. ACM Press.
28. C. Ouyang and J. Billington. Formal Analysis of the Internet Open Trading Protocol. In *Proceedings of the 1st International Workshop on Theory Building and Formal Methods in Electronic/Mobile Commerce (TheFormEMC)*, Toledo, Spain, 1-2 October 2004, in press. Springer-Verlag.
29. C. Ouyang, L.M. Kristensen, and J. Billington. A formal and executable specification of the Internet Open Trading Protocol. In *Proceedings of 3rd International Conference on Electronic Commerce and Web Technologies*, volume 2455 of *Lecture Notes in Computer Science*, pages 377–387, Aix-en-Provence, France, 2-6 September 2002. Springer-Verlag.
30. A.N. Parashkevov and J. Yantchev. Space Efficient Reachability Analysis Through Use of Pseudo-Root States. In *Proceedings of TACAS'97*, volume 1217 of *Lecture Notes in Computer Science*, pages 50–64. Springer-Verlag, 1997.
31. D. Peled. All from One, One for All: On Model Checking Using Representatives. In *Proceedings of CAV'93*, volume 697 of *Lecture Notes in Computer Science*, pages 409–423. Springer-Verlag, 1993.
32. Standard SMart Card Integrated SettLEment System Project SMILE Project.
URL: <http://www.ietf.org/proceedings/99mar/slides/trade-smile-99mar>, April 1999.
33. K. Schmidt. Automated Generation of a Progress Measure for the Sweep-Line Method. In *Proceedings of TACAS'04*, volume 2988 of *Lecture Notes in Computer Science*, pages 192–204. Springer-Verlag, 2004.
34. A. Valmari. A Stubborn Attack on State Explosion. In *Proceedings of CAV'90*, volume 531 of *Lecture Notes in Computer Science*, pages 156–165. Springer-Verlag, 1990.
35. A. Valmari. The State Explosion Problem. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 429–528. Springer-Verlag, 1998.
36. WAP Wireless Transaction Protocol Specification. June 2000 Conformance Release. Available via: <http://www.wapforum.org/>.
37. Wireless Application Protocol. Specifications available via: <http://www.wapforum.org/>.
38. P. Wolper and P. Godefroid. Partial Order Methods for Temporal Verification. In *Proceedings of CONCUR'93*, volume 715 of *Lecture Notes in Computer Science*, pages 233–246. Springer-Verlag, 1993.
39. P. Wolper and D. Leroy. Reliable Hashing without Collision Detection. In *Proceedings of CAV'93*, volume 697 of *Lecture Notes in Computer Science*, pages 59–70. Springer-Verlag, 1993.