

# Qualitative Simulation of Temporal Concurrent Processes Using Time Interval Petri Nets

Vadim Bulitko

*Department of Computing Science  
University of Alberta  
Edmonton, Alberta T6G 2H1, CANADA*

David C. Wilkins

*Beckman Institute  
University of Illinois at Urbana-Champaign  
Urbana, IL 61801, USA*

---

## Abstract

This paper presents a formalism called Time Interval Petri Nets (TIPNs), which are designed to support a qualitative simulation of temporal concurrent processes. One of the key features of TIPNs is a uniform use of time intervals throughout the model. This enables a natural and efficient representation of temporal uncertainty in inputs, outputs, and intermediate states of the qualitative simulation. This is required because the exact time of key events, such as the start time of a fire crisis, is typically not known with certainty. Likewise, output conclusions of the qualitative simulation include earliest time and guaranteed time of key events that can be used by a decision maker to select the most appropriate action.

Results are described of a TIPN-based qualitative simulator constructed in the domain of ship damage control. The simulator was created to replace an existing quantitative simulator which was too slow to support envisionment-based real-time decision making in this domain. The experimental results showed a speedup of four to five orders of magnitude which enables hyper-real time qualitative prediction of consequences of multiple competing actions. An automated shipboard damage control decision-making system incorporating a TIPN-based qualitative simulator achieved a 318% improvement over human subject matter experts in a large-scale simulated exercise of over 500 scenarios.

*Key words:* Qualitative simulation, Real-time decision making, Bounded rationality, Petri nets, Intelligent systems, Temporal reasoning, Envisionment-based control policies, Damage control.

---

## 1 Introduction

Decision-making systems in artificial intelligence can often make better decisions when it is possible to envision the consequences of alternative actions prior to making a decision (Russell and Wefald, 1991a,b; Korf, 1990; Baum and Smith, 1997). This paper focuses on qualitative simulation of temporal concurrent processes.

The organization of this paper is as follows. In section 2, the application domain is introduced and the modelling constraints on the inputs and outputs of a qualitative simulation are described. In section 3, a review is given of relevant prior research. In section 4, a formal description is given of a Petri net formalism for modelling this domain. It is illustrated with a detailed example in section 5. Subsequently, section 6 describes the empirical evaluation, and covers the degree of speedup, scalability, and accuracy. Conclusions and directions for future research are presented in section 7.

## 2 Ship Damage Control and Its Modelling Constraints

This section introduces the domain of decision making for ship damage control, and then describes the constraints on a qualitative simulation that are relevant to the decision making in this domain. The constraints relate to the form of temporal information to and from the qualitative simulator, and the speed of the simulator.

In the domain of ship damage control, a major decision making task relates to controlling crises that involve fire, smoke, flooding, pipe ruptures, hull ruptures, and electrical and mechanical system deactivation (Wilkins et al., 2001). Toward this end, the main decision-maker, called a Damage Control Assistant, determines the nature and extent of ship crises from ship sensors and human investigator reports, and addresses the crisis using a limited number of automated systems and human repair parties. A qualitative simulator allows a human and/or automated Damage Control Assistant (Bulitko and Wilkins, 2002b, 1999c,b,a) to make better decisions by envisioning the consequences of alternative actions that can be made.

A modelling constraint relevant to the design of a qualitative simulator is temporal uncertainty of the occurrence of critical events. Temporal uncertainty is present in the input information to the simulator, the processes that are simulated, and the output of the simulator. Examples of each of these three situations follow.

One important input event to a qualitative simulator in the domain of ship

---

*Email addresses:* bulitko@ualberta.ca (Vadim Bulitko), dcw@uiuc.edu (David C. Wilkins).

damage control for which there is temporal uncertainty is the time of the start of a crisis, such as a fire crisis. The existence of a fire crisis is often first known because of a fire alarm. A fire alarm, even when it is accurate, only places an upper bound on the start time of an associated fire crisis. Especially when a fire originates in a location distant from the fire alarm, the start time can be considerably earlier than the time of the fire alarm. Therefore, when there is a fire alarm it is only possible to specify the *start* time of the associated fire crisis event as a time interval:

```
IF      fire-alarm(x,ta) & no-damage(x,tn)
THEN   on-fire(x,[tn,ta])
```

Here  $x$  is the compartment identifier,  $ta$  is the time stamp of the fire alarm, and  $tn$  is the last known reading of intact status for the compartment. Therefore, our temporal uncertainty on the actual start time of the fire is given to the simulator as time interval  $[tn,ta]$ . Note that the interval has no indication of when the fire is out but merely of its start time.

One important process event used in a qualitative simulation of a fire crisis is the spread of a fire from one room or compartment to another. This is dependent on many factors that are only known approximately, such as the thickness of the walls, the door openings between the rooms and whether they are open or closed, the amount of material in the rooms, and its type. Different conditions will yield different fire spread rules, each of which will be approximated by a different temporal interval. For example, if a ship compartment is hot and there are no fire boundaries in place around this compartment, then the fire will spread to its horizontal neighbors in three to five minutes. Thus, we have the rule:

```
IF      on-fire(x,[t,t']) & neighbor(x,y)
THEN   on-fire(y,[t+3,t'+5])
```

Again,  $x$  and  $y$  are the compartment identifiers and  $[t,t']$  is the interval time stamp of the original fire.

There are many important output events from a qualitative simulator that can only be time stamped with intervals. Following our fire crisis example, one can imagine that extinguishing the fire in compartment  $y$  will take between 10 and 17 minutes. Consequently, the time stamp of event “fire is out” is best expressed as an interval:

```
IF      on-fire(x,[t1,t2]) & fighting-fire(x,[t3,t4])
THEN   fire-out(x,[t3+10,t4+17])
```

Here we assume that the fire fighting started after the fire is known to be in the compartment with certainty:  $t3 > t2$ . Then  $t3+10$  defines the earli-

est possible extinguishing time while  $\tau_{4+17}$  indicates the guaranteed fire-out time. Such data are often instrumental when it is desired to know when a fire is extinguished because the fire fighters are needed elsewhere. The time interval stamped output from the simulator provides the earliest time that this event can occur, and also the guaranteed time it will occur. Or suppose it is important to extinguish a fire before it reaches a room that contains gasoline tanks. The qualitative simulation would allow a determination of the fire suppressant technique needed to bring the pessimistic upper end of the time interval before the fatal fire-spread point (a more detailed example can be found in section 6.1). Hence, one requirement for a proposed qualitative simulator for this domain is to handle explicit temporal uncertainty of events and delays.

The decision-making task of ship damage control is inherently time critical since the more time is spent, for instance, in determining the exact location and size of a fire, the larger the fire will usually become. How much speedup is desired? In the domain of ship damage control, at a given point in time there are usually about 50 alternative decisions for which it would be helpful to envision 20 minutes into the future. If the goal is to do the total simulation in 5 seconds, this would require a simulation system that can operate  $50 * 20 * 60 / 5 = 12,000$  times faster than real-time. Different crisis management domains will have different speedup requirements; in the particular proof of principle domain used for experimentation, a speedup of four to five orders of magnitude was taken as the target.

Physical system and behavioral system simulators are one approach to envisionment (Heath, 1996; Shou et al., 2001; NIST, 2002; Grois and Wilkins, 2001). A limitation of this approach in complex domains arises in real-time decision making, when the time available for simulation is very limited and there is a high-cost of not making a quick decision (Bulitko and Wilkins, 1999b). Hence, the need for a qualitative approach.

### 3 Related Research

It is possible to imagine a qualitative simulation for the described domain of ship damage control based on a number of different techniques. Major examples include Qualitative Process Theory (Forbus, 1984), rule-based production systems (Forgy, 1981, 1982) and Petri nets (Merlin, 1974; van der Aalst, 1993; Jensen, 1997). None of these formalisms is designed and used to handle the type of temporal intervals that were described in the previous section.

This paper shows that Petri nets (Peterson, 1981; Murata, 1989; Donatelli and Kleijn, 1999) enhanced with explicit temporal interval logic are suitable for envisionment for decision-making domains where the modelling of real-time concurrent processes plays a central role in the reasoning. Previous work

that develops Petri Net extensions for AI intelligent reasoning includes (Sil, 1995) for probabilistic reasoning, (Zhang and Murata, 1996) for rule-based logic systems, (Medeiros et al., 1999) for fuzzy work-flow management, and (Costa Miranda, 1999) for multi-agent systems.

Numerous extensions of classical P/T nets (Peterson, 1981) have been proposed in the last several decades (Merlin, 1974; van der Aalst, 1993; Jensen, 1997). Most of them target the tasks of verification. Thus, the models are required to capture all essential domain elements precisely. Then certain properties of the Petri Net models are analyzed, proved, and linked to analytical properties of the system modelled (Berthomieu and Menasche, 1983; Berthomieu and Diaz, 1991). The verification and formal provability of these approaches limit the representation convenience of the modelling means as well as increase the model running time. For instance, TPNs by Merlin are essentially propositional and require structure replications to model the same phenomenon in different contexts (e.g., the fire crisis modelling subnet would have to be replicated several hundred times for compartments of a moderate size ship). Additionally, no explicit token time stamps are supported. Several other extensions, most notably TCPNs (Jensen, 1997) and ITCPNs (van der Aalst, 1993), do provide context support via token colors but assume point-value time stamps for their tokens. In order to represent temporal uncertainty in process duration, *domain simulation* is often carried out with Monte Carlo like stochastic methods. Therefore, numerous randomized runs of the same Petri net are needed to estimate the resulting state distribution.

We found using intervals as token time stamps to represent the earliest possible time and the guaranteed time of an event natural and effective in the real-time decision making domains. Such interval time stamps can be computed efficiently with a single-pass algorithm as detailed below.

#### 4 TIPN: Formal Representation

This section formally introduces Time Interval Petri Nets. An effort is made to follow similar terminology of (Jensen, 1997; van der Aalst, 1993). The formal description of TIPNs begins with the basic concepts and then defines the concept of enablement.

The definitions introduced in this section are illustrated with simplified ship damage control examples. An informal and more incremental introduction is presented in detail in the next section. The reader may wish to skip directly to it on the first reading.

**Definition 1** Formally a Time Interval Petri Net (TIPN) is a tuple  $\langle \Sigma, P, T, A, AT, PC, TM, TD, AE, PI \rangle$  where:

- $\Sigma$  is the set of color sets (i.e., token identifier types);

- $P$  is the set of places;
- $T$  is the set of transitions;
- $A$  is the set of arcs. It is required that

$$P \cap T = A \cap T = A \cap P = \emptyset;$$

- $AT$  is the arc type function mapping  $A$  to

$$(P \times T \times A_{types} \times ArF) \cup (T \times P \times A_{types} \times ArF).$$

Here  $A_{types}$  is the set of possible arc types:

$$A_{types} = \{regular, inhibitory, double-ended\}.$$

$ArF$  is the set of arities (i.e.,  $\{0, 1, 2, \dots\}$ ). Thus,  $AT$  takes an arc as an input and returns the arc's properties as a 4-tuple:  $\langle place, transition, arc\_type, arity \rangle$  (for input arcs) or  $\langle transition, place, arc\_type, arity \rangle$  (for output arcs). Output arcs are limited to the regular type;

- $PC$  is the place color function which restricts the type of colors for any tokens the place can hold. Formally:

$$PC : P \rightarrow \Sigma;$$

- $TM$  is the matching function defined on the transitions. It specifies the parts of token identifiers that have to match to enable the transition. Such matching is usually implemented using variable names in the token colors (a la in Prolog clauses);
- $TD$  is the transition time delay function which affects the token time intervals as they pass through the transition. Formally:

$$TD : T \rightarrow Time_i,$$

where  $Time_i$  is the set of time intervals:

$$Time_i = \{[t_b, t_e] \mid t_b, t_e \in \mathbb{R}^+ \ \& \ t_b \leq t_e\}.$$

Having transition time delays independent of the scenario time speeds up TIPN operation within qualitative simulation as well as inductive machine learning methods for TIPNs (Bulitko, 2000; Bulitko and Wilkins, 2002a);

- $AE$  is the arc expression function that assigns operators to the output arcs (i.e., the arcs that originate at a transition and end at a place). Formally:

$$AE : A_o \rightarrow RF$$

where  $RF$  is the set of computable functions. Here  $A_o$  is the set of output arcs:

$$A_o = AT^{-1}(P \times T \times A_{types} \times ArF) \cap A;$$

- $PI$  is the initialization expression that computes the initial multiset of tokens for a given place:

$$PI : P \rightarrow C(P)_{MS}.$$

Here subscript  $MS$  indicates a multiset.

We illustrate this definition with a concrete TIPN in Example 2 below. It should be noted that unlike CP-nets, TIPNs do not need a special color to represent temporal information since each token is *always* time stamped. A time stamp is an interval from set  $Time_i$  as defined above. Intuitively, a time stamp interval indicates a belief as to when a particular state was *acquired* or *entered*. Notice that a time stamp does *not* indicate the ending time of the *state* represented by the token. For any token  $x$  function  $bt(x)$  returns the beginning time of  $x$ 's time stamp while  $et(x)$  gives us the ending time. In other words, the interval  $[bt(x), et(x)]$  accounts for uncertainty in the state acquisition time.

**Example 1** Suppose place “Engulfed” holds a token labelled  $\langle \{ [Compartment, 3-370-0-E] \}, [5:44, 6:32] \rangle$ . This is interpreted as “It is believed that compartment 3-370-0-E became engulfed sometime between 5:44 and 6:32 scenario time”.

The series of examples used throughout this paper to illustrate Petri nets relate to the temporal process of fire spreading among compartments of a ship. Petri nets model the various factors that effect the rate of spread and extinguishment, such as the ship topology, availability of fire fighters, availability of firefighting resources such as water, or firefighting actions such as setting fire boundaries or hosing the fire with water. The Petri net models the state of each compartment, such as its being normal, engulfed, extinguished or destroyed.

#### 4.1 TIPN State Specification

A TIPN changes its state through time. The dynamic state of a TIPN is, thus, specified via its “marking” as follows.

**Definition 2** A TIPN is said to have a marking  $M$  if there is a function  $M : P \rightarrow (C(P) \times Time_i)_{MS}$ . A place  $p$  is said to hold a multi-set of tokens  $X$  iff  $M(p) = X$ .

**Example 2** Consider the simple TIPN whose graphical representation is shown in Figure 1. The net expresses the following domain dependency: “If a compartment is engulfed and no fire boundaries are set then the fire will spread to neighbor compartments with a delay of 5 to 7 minutes”. In the figure, the net is shown in a particular state with compartment 3-370-0-E engulfed and no fire boundaries set for it. The net shown in Figure 1 is formally described by the tuple  $\langle \Sigma, P, T, A, AT, PC, TM, TD, AE, PI \rangle$ :

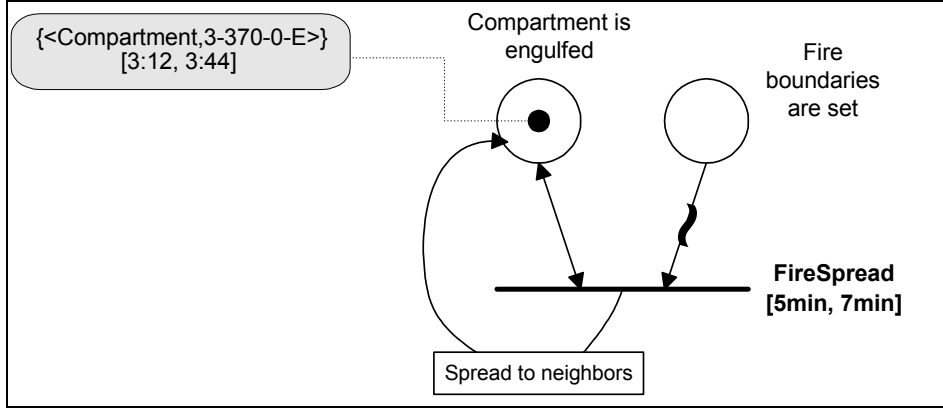


Fig. 1. A fire-spread TIPN presented in the graphical format

- $\Sigma = \{Compartment, Compartment \times Station\}$  as there are two color sets: *Compartment* and *Compartment*  $\times$  *Station*;
- The TIPN has two places:  $P = \{ \text{“Compartment is engulfed”}, \text{“Fire boundaries are set”} \}$ ;
- There is only one transition:  $T = \{ \text{“FireSpread”} \}$ ;
- The net has three arcs:  $A = \{a_1, a_2, a_3\}$ ;
- Input arc  $a_1$  connects place “Compartment is engulfed” to transition “FireSpread” and is a double-ended arc:  $AT(a_1) = \langle \text{“Compartment is engulfed”}, \text{“FireSpread”}, \text{double-ended}, 1 \rangle$ ;
- Input arc  $a_2$  connects place “Fire boundaries are set” to transition “FireSpread” and is a inhibitory arc:  $AT(a_2) = \langle \text{“Fire boundaries are set”}, \text{“FireSpread”}, \text{inhibitory}, 1 \rangle$ ;
- Output arc  $a_3$  connects transition “FireSpread” to place “Compartment is engulfed” and is a regular arc:  $AT(a_3) = \langle \text{“FireSpread”}, \text{“Compartment is engulfed”}, \text{regular}, 1 \rangle$ ;
- Tokens in place “Compartment is engulfed” are labelled (“colored”) with the values from color set *Compartment*:  $PC(\text{“Compartment is engulfed”}) = Compartment$ ;
- Tokens in place “Fire boundaries are set” are colored with tuples from *Compartment*  $\times$  *Station* color set:  $PC(\text{“Fire boundaries are set”}) = Compartment \times Station$ ;
- Transition “FireSpread” matches tokens based on the *Compartment* component of their colors:  $TM(\text{“FireSpread”}) = \{Comp\}$ . Here the color of tokens in place “Compartment is engulfed” is expressed with a single variable [*Comp*] while the color of tokens in place “Fire boundaries are set” has two variables: [*Comp*, *Stat*]. Variable *Comp* is the one to match;
- When fired the transition off-sets the time stamps by the delay interval of [5, 7] minutes:  $TD(\text{“FireSpread”}) = [5, 7]$ ;
- Arc expression for  $a_3$  is defined as  $AE(a_3) = SpreadToNeighbors(Comp)$ . Function  $SpreadToNeighbors(Comp)$  takes variable *Comp* as its input and returns multiple tokens for all adjacent compartments;
- Both places are empty at the beginning:  $PI(\text{“Compartment is engulfed”}) =$

$\emptyset$ ,  $PI(\text{“Fire boundaries are set”}) = \emptyset$ .

The net is shown with the following marking:

- Place “Compartment is engulfed” has one token in it:  $M(\text{“Compartment is engulfed”}) = 1' \{ \text{Compartment, 3-370-0-E}, [3:12, 3:44] \}$ ;
- The other place is presently empty:  $M(\text{“Fire boundaries are set”}) = \emptyset$ .

#### 4.2 TIPN Firing Sets

So far, this section has covered the representational aspect of TIPNs. The next part of this section introduces the inferential concept of a “firing set” that plays an important role in running the nets. While TIPN markings introduced in the previous section represent states of the modelled system, TIPN firing sets allow the modeler to describe the system’s dynamics.

**Definition 3** Enabling places for a transition are the set of input places connected to the transition through regular or double-ended arcs.

**Definition 4** Disabling places for a transition are the set of input places connected to the transition through inhibitory arcs.

**Definition 5** A firing set  $S_{t,i}$  for transition  $t$  is a minimal set of tokens relevant to a transition firing, that is, either making a transition ready to fire (i.e., enabling it) or prohibiting it from firing (i.e., disabling it). A firing set is comprised of all relevant tokens in the enabling places, called enabling tokens and denoted by

$$ET(S_{t,i})$$

and all relevant tokens in the disabling places, called disabling tokens and denoted by

$$DT(S_{t,i}).$$

Note that one transition can have several firing sets simultaneously. The set of all firing sets for a TIPN  $E$  with an initial marking  $M_0$  is denoted by

$$FS(E, M_0).$$

**Example 3** In this example we will illustrate the concepts of enabling and disabling places and tokens and firing sets. A marked TIPN is shown in Figure 2. The net expresses the following domain dependency: “If a compartment is engulfed and no fire boundaries are set then the fire will spread to neighbor compartments with a delay of 5 to 10 minutes”. The net is presented in a particular state showing compartments 3-370-0-E and 3-78-0-M engulfed. No fire boundaries are set for the latter compartment while the former space has very delayed fire boundaries ( $[5:33, 6:12]$  versus  $[3:12, 3:14]$ ). Thus, the net has two firing sets as illustrated. Each firing set represents a potential domain fire spread:



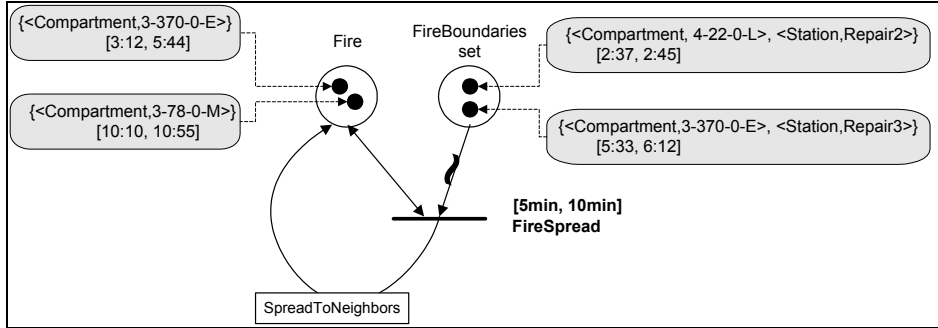


Fig. 3. Illustration of firing times - both transitions are enabled.

$$[ftb, fte] = \begin{cases} [eft, \min(gft, edt)] & \text{if } eft \leq \min(gft, edt), \\ \text{not defined} & \text{(transition disabled)}. \end{cases} \quad (4.1)$$

Here  $eft$  (earliest firing time),  $gft$  (guaranteed firing time), and  $edt$  (earliest disabling time) are defined as follows:

$$eft(S_{t,i}) = \max_{x \in ET(S_{t,i})} (bt(x)) \quad (4.2)$$

$$gft(S_{t,i}) = \max_{x \in ET(S_{t,i})} (et(x)) \quad (4.3)$$

$$edt(S_{t,i}) = \begin{cases} \min_{x \in DT(S_{t,i})} (bt(x)) & \text{if } DT(S_{t,i}) \neq \emptyset, \\ \infty & \text{otherwise.} \end{cases} \quad (4.4)$$

Recall that  $bt(x)$  is the earliest time event  $x$  may have started and  $et(x)$  expresses our belief as to when the event  $x$  started for sure. Intuitively, transition  $t$  with firing set  $S_{t,i}$  can fire if both of the following conditions are satisfied: (i) all enabling tokens are present and (ii) none of the disabling tokens is present.

Thus, the earliest time a transition can fire (i.e.,  $eft$ ) is when all enabling tokens ( $ET(S_{t,i})$ ) of a particular firing set ( $S_{t,i}$ ) are in the enabling places. Likewise, the firing conditions are guaranteed to be satisfied at time  $gft$  except for the disabling token “interference” which is guaranteed to be absent until  $edt$ . We found this approximation to work fairly well for qualitative simulation in the ship board damage control domain (see section 6 for details). Furthermore, section 4.5 discusses the underlying assumptions and possible modifications to the firing mechanism presented above.

**Example 4** Figure 3 illustrates the concepts introduced with the TIPN used in the previous example. The net has two firing sets detailed below:

Transition:	FireSpread
Enabling places:	{Fire}
Disabling places:	{FireBoundaries set}

Firing Set #1:

Enabling tokens: { <<Compartment,3-370-0-E>, [3:12,5:44]> }

Disabling tokens: { <<Compartment,3-370-0-E,<br><Station,Repair3>, [5:33,6:12]> }

eft = 3:12

gft = 5:44

edt = 5:33

firing time = [3:12, 5:33]

Firing Set #2:

Enabling tokens: { <<Compartment,3-78-0-M>, [10:10,10:55]> }

Disabling tokens: { none }

eft = 10:10

gft = 10:15

edt = infinity

firing time = [10:10, 10:15]

#### 4.4 TIPN Enablement

Finally, the concept of “enablement” completes the set of definitions needed to introduce TIPN inference. It allows reasoning about transitions that are ready to run (fire).

**Definition 7** Two transitions  $a$  and  $b$  with firing sets  $S_{a,i}$  and  $S_{b,j}$  are said to be independently and concurrently enabled iff firing one of them will not disable the other. Note that  $a$  and  $b$  may refer to the same transition but with different firing sets ( $i \neq j$ ). In that case we say that the transition is multiply enabled within itself.

Recall that firing times are defined as time intervals to reflect the temporal uncertainty as to when a change in the states can occur. An additional source of uncertainty is the duration of the change. For instance, fire spread between adjacent compartments can take 3 to 7 minutes depending on the wall thickness, the amount of insulation, heat distribution, and the air flow. Thus, similar to Merlin in (Merlin, 1974; Merlin and Faber, 1976) we assign delay intervals to our transitions. In the example, transition “Fire Spread” will be labelled with delay interval  $[3, 7]$ . Thus, if the transition was to fire at some point between 10 and 12 minutes scenario time (due to the uncertainty in interval time stamps of the enabling and disabling tokens), the interval time stamps of the resulting tokens will be  $[10 + 3, 12 + 7] = [13, 19]$  minutes scenario time\*. This is formalized in the following definition.

---

\* Note that if the firing time  $t$  were known precisely then the time stamps of the resulting tokens would be  $[t + 3, t + 7]$ . Not knowing the exact moment  $t$ , we approximate the set of intervals  $\{[t + 3, t + 7] \mid 10 \leq t \leq 12\}$  with a single interval  $[13, 19]$ .

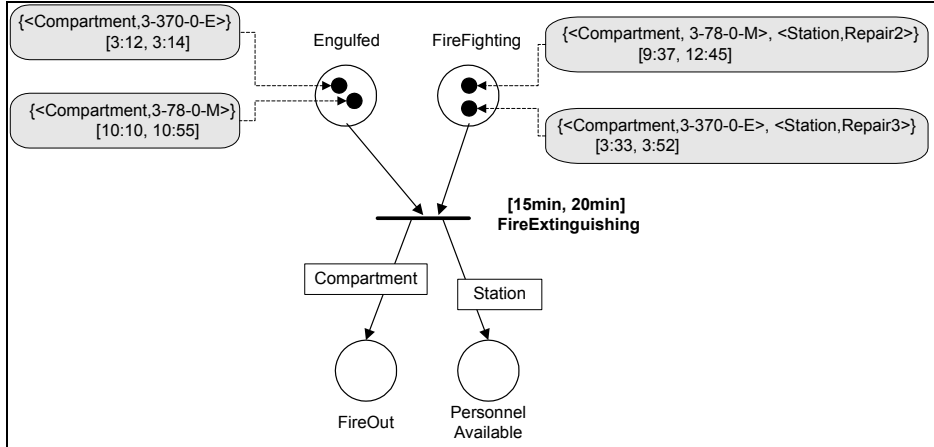


Fig. 4. Transition firing: the initial configuration.

**Definition 8** A transition  $t$  with time delay interval  $[\Delta_{min}, \Delta_{max}]$  and firing set  $S_{t,j}$  with firing time  $[ftb, fte]$  is said to fire iff it consumes all enabling tokens (except the ones connected to the transition via double-ended arcs) and adds appropriate tokens to its output places. All such added tokens will have identical time stamps of  $[ftb + \Delta_{min}, fte + \Delta_{max}]$ . Each token's color will be fed into the arc expression function  $AE(a_o)$  for each output arc  $a_o$ . The output of the function will determine the new tokens' color.

Analogously to CP-nets (Jensen, 1997), the functions (or arc expressions) are defined in terms of the variables on the input token colors. Intuitively, a transition firing represents a change of state, the time delay interval indicates the likely range of state change duration, and the output arc functions define attributes of the new state.

**Example 5** We will now consider a simple TIPN that models putting out a fire at a very coarse level. For each of the enabled firing sets we will specify the firing times. It turns out that in this example, the transition is multiply enabled within itself (i.e., has multiple independently enabled firing sets associated with it). Figure 4 shows the initial configuration. The net expresses the following domain dependency: "If a compartment is engulfed and fire fighting is in progress then the fire will be put out in 15 to 20 minutes. The firefighting personnel will then be available for other tasks". The net is presented in a particular state showing compartments 3-370-0-E and 3-78-0-M engulfed. Two firing sets are formed indicating that both fires will be put out:

Transition:	FireExtinguishing
Enabling places:	{Engulfed, FireFighting}
Disabling places:	{none}

Firing Set #1:	
Enabling tokens:	{ <<Compartment,3-370-0-E>, [3:12,3:14] >, <<Compartment,3-78-0-M>, [10:10,10:55] > }

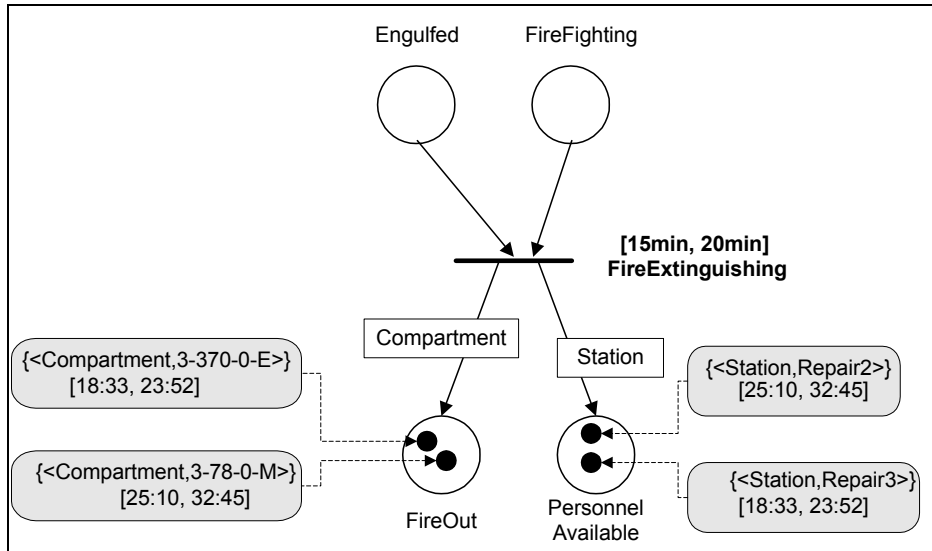


Fig. 5. Transition firing: the resulting configuration.

```
<{<Compartment,3-370-0-E>,
  <Station,Repair3>}, [3:33,3:52]> }
```

```
Disabling tokens: { none }
eft = 3:33
gft = 3:52
edt = infinity
firing time = [3:33, 3:52]
```

Firing Set #2:

```
Enabling tokens: { <{<Compartment,3-78-0-M>}, [10:10,10:55]>
  <{<Compartment,3-78-0-M>,
    <Station,Repair2>}, [9:37,12:45]> }
Disabling tokens: { none }
eft = 10:10
gft = 12:45
edt = infinity
firing time = [10:10, 12:45]
```

The time stamps of the new tokens are calculated as the firing times plus the delay intervals. Figure 5 presents the marking after both firing sets fire showing the new tokens generated (notice that the transition has multiple output arcs with different arc expressions). After both firing sets are fired, all consumable enabling tokens are removed from the enabling place and the following new tokens are placed into the output places:

Firing Set #1:

```
firing time           [3:33, 3:52]
transition's time delay [15:00, 20:00]
new tokens' time stamp [18:33, 23:52]
```

variable bindings	{ Comp = 3-370-0-E, Stat = Repair3 }
left output arc function	Comp
right output arc function	Stat
new token in place FireOut	<{<Compartment, 3-370-0-E>}, [18:33, 23:52]>
new token in place PersonnelAvailable	<{<Station, Repair3>}, [18:33, 23:52]>
Firing Set #2:	
firing time	[10:10, 12:45]
transition's time delay	[15:00, 20:00]
new tokens' time stamp	[25:10, 32:45]
variable bindings	{ Comp = 3-78-0-M, Stat =Repair2 }
left output arc function	Comp
right output arc function	Stat
new token in place FireOut	<{<Compartment, 3-78-0-M>}, [25:10, 32:45]>
new token in place PersonnelAvailable	<{<Station, Repair2>}, [25:10, 32:45]>

#### 4.5 TIPN Inference

Like many other computational formalisms, TIPN representation is complemented with an inference engine. Similarly to the first order predicate logic that allows for various inference mechanisms (e.g., forward chaining, backward chaining, etc.), TIPNs can handle inference in various ways. Correspondingly, the inference attributes such as soundness, completeness, practical feasibility, convergence, and so forth vary. To date several inference engines have been developed for TIPNs. They differ in terms of the specific scheme used for firing sets. The following definition presents a basic mechanism developed.

**Definition 9 (Inference Mechanism).** Given a marked TIPN, we compute the firing times for all firing sets and execute the firing set with the earliest (i.e., lowest) firing time (*eft*). If several firing sets are enabled with exactly the same earliest firing time (*eft*) then one of them is fired (the choice can be resolved arbitrarily but consistently for the TIPN). After the firing set is executed, the marking is updated as outlined in sections 4.3 and 4.4.

Note that several assumptions are made:

- (1) The assumption of “non-conservatism” (or “optimism”): if a firing set is enabled (i.e., the firing time interval is not empty) then it will fire as long as there does not exist another firing set with an earlier firing time (*eft*).

The assumption is referred to as “non-conservative” or “optimistic” due to the following fact. Even though token time stamps  $[bt(x), et(x)]$  represent temporal uncertainty intervals inasmuch as state  $x$  may have been accrued at *any* time between  $bt(x)$  and  $et(x)$ , we optimistically use  $bt(x)$  *only* for calculating  $eft$ . The ending time ( $et(x)$ ) is used together with the maximum transition time delay ( $\Delta_{max}$ ) to propagate the temporal uncertainty as the transitions fire. While the ending times ( $et(x)$ ) are not taken into account when making firing decisions in the basic inference engine presented here, they play an important role in inductive machine learning methods for TIPNs (Bulitko, 2000; Bulitko and Wilkins, 2002a).

- (2) We always fire the firing set with the lowest  $eft$  time first. The ties are resolved arbitrarily but consistently for a given TIPN.

Relaxing the two assumptions will result, for instance, in a more conservative firing logic. Such a mechanism would be useful for modelling domains with a great deal of event and temporal uncertainty in which these assumptions can be frequently violated. Accordingly, one of the future research avenues would be an exploration of this variant of the firing logic.

## 5 TIPN Illustration with an Example

We will now illustrate the concepts introduced in the previous section within the context of a simple fire spread example. For exposition purposes, the domain phenomenon will be first modelled using the knowledge representation and inference of the classical place transition Petri nets (P/T-nets) (Peterson, 1981) and then will be extended to achieve the necessary fidelity. Such presentation reveals the motivation behind the design choices made in developing TIPNs.

In order to model the process of fire spread with traditional modelling tools one needs to set up a numeric simulator to compute combustible material distribution, gas zones, plumes, heat transfer, wall temperatures, ignition properties, fire suppressant effects, and so forth (Shou et al., 2001). Since fire fighting personnel are involved in the process, another behavioral simulator is needed to model personnel travel time, human fatigue, communication mistakes, injuries, oxygen-breathing apparatus operation, and many other non-trivial aspects (Grois and Wilkins, 2001). The inherent complexity of these types of simulation prevents multiple alternatives extending, say 20 minutes into the future, from being simulated in a matter of seconds, hence the motivation for the TIPN approach. As with all abstractions of first-principles models, the TIPN approach is not as accurate, but it provides sufficient accuracy in a limited amount of time to support the decision-making.

The process of constructing a TIPN begins with a model of the qualitative-type patterns augmented with temporal information that underlie the phenomena

in question. Using this approach, the complex processes outlined in the previous paragraph are abstracted into the following domain dependency (greatly simplified for illustration purposes):

“for any compartment  $X$  if  $X$  is hot and the fire boundaries around  $X$  are not set then the fire will spread to neighbor compartments  $Y$  in 3 to 4 minutes”

In the first order predicate calculus (FOPC) it can be expressed as follows (here and below we use Prolog notation):

```
ignited(Y,Tnew) :-
    hot(X,Told),
    not fire_boundaries(X,Told),
    neighbor(X,Y),
    delay(Tnew,Told,3,4).
```

Here predicate `ignited(Y,Tnew)` holds when compartment  $Y$  is ignited at time  $Tnew$ . Likewise, predicate `hot(X,Told)` indicates the fact that compartment  $X$  is of a high temperature at time  $Told$ . Predicate `fire_boundaries(X,Told)` establishes that fire boundaries are set around compartment  $X$  at time  $Told$  (i.e., that the compartment walls are being cooled down to prevent fire spread). Predicate `neighbor(X,Y)` holds if and only if compartment  $X$  is adjacent to compartment  $Y$  thereby specifying the ship topology. Finally, predicate `delay(Tnew,Told,3,4)` tells us that  $Tnew$  and  $Told$  are 3 to 4 minutes apart.

We will introduce TIPNs by starting out with a simplified dependency expressible in propositional logic and moving to FOPC. This gradual upgrade will be paralleled by the transition P/T nets to Time Interval Petri Nets.

### 5.1 Place-Transition Petri Nets

Figure 6 presents the first step in this process. We start out with no variables and no time delays. At this stage we are expressing merely that “if *it* is hot and no fire boundaries are set then *it* becomes ignited”.

The propositional logic sentence `ignited :- hot, no_fbs.` relates to the Petri Net as follows. Each large circle (called “place”) corresponds to an atomic sentence. If a small black dot (called “token”) is present inside the circle then the corresponding atomic sentence is considered to hold. The horizontal bar (called a “transition”) represents the implication. The arrows (called “arcs”) indicate the preconditions and post effects. Once all preconditions are met (i.e., there are tokens in all relevant places) the transition “fires”. The process of firing retracts tokens from all enabling places (i.e., preconditions) and deposits tokens to the output places (i.e., the effect places). Thus, the tokens can be said to flow from the enabling places to the output places.

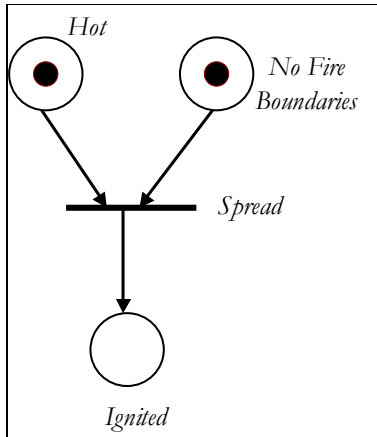


Fig. 6. Step 1: Place-Transition Petri Net corresponding to propositional logic sentence `ignited :- hot, no_fbs`. Note that firing the transition will render the preconditions false as the enabling tokens will be withdrawn. This is not the case with the logical sentence and the physical reality and can be fixed with double-ended arcs as shown below.

### 5.2 Colored Petri Nets

There are several aspects of the fire spread example that the representation described in the previous section doesn't model. One such aspect is the inability to model explicit context. This section shows how this limitation is overcome via introducing compartment variables and time stamps. The addition of these variables to our propositional statement makes it into the first-order predicate calculus. In our evolving example, the variable `T` will express time while the variable `X` will take its values over the set of compartments. Our sentence becomes `ignited(X,T) :- hot(X,T), no_fbs(X,T)`. Correspondingly, the Petri Net model is upgraded with so-called "colors" (Jensen, 1997). Colors are arbitrary data tags attached to the tokens. The data includes compartment identifiers (e.g., 3-370-0-E) and time stamps (e.g., [2:32, 3:14]). Time stamps are represented with intervals accounting for the temporal uncertainty of the predicate's becoming true. The extended representation is illustrated in Figure 7. By extending the Petri Net in such a way, we gain the ability to use a *single* net for modelling *many* different compartments, similar to using a single algorithm to compute square root for many different numbers.

### 5.3 Time Interval Petri Nets

The model described in the previous section is able to model explicit context but doesn't model temporal delays, which are vital in domains with time-critical decision making. In FOPC this can be addressed by adding an extra predicate `delay(TimeOld, TimeNew, MinimumDelay, MaximumDelay)`. Within the TIPN formalism this is handled by the addition of a delay interval to the transition as shown in Figure 8. As the tokens move through the transition their time stamps get appropriately adjusted. In the example above, we can

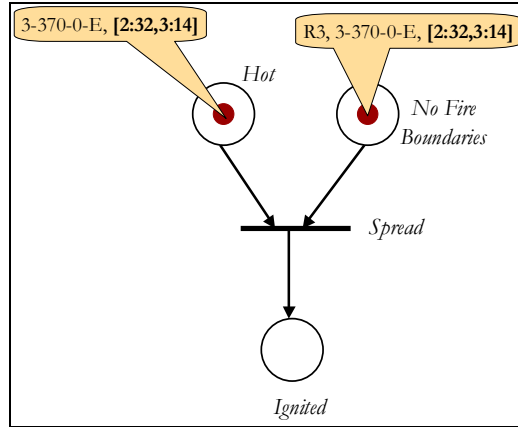


Fig. 7. Step 2: Time Interval Petri Net corresponding to first-order predicate logic sentence:  $\text{ignited}(X,T) :- \text{hot}(X,T), \text{no\_fbs}(X,T)$ . Again, double-ended arcs are needed to keep preconditions true after firing the transition.

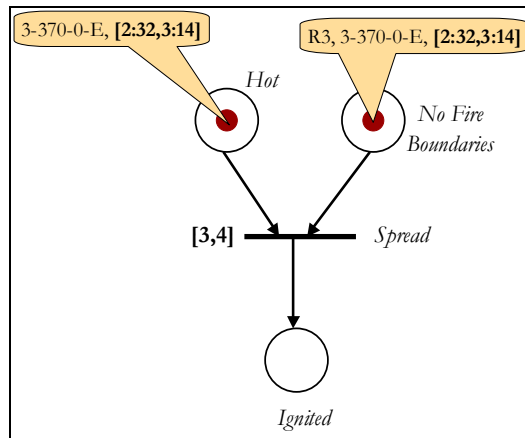


Fig. 8. Step 3: our model is extended with temporal delays. In this case the transition “Spread” gets delay interval [3,4] associated with it. The corresponding FOPC sentence is this:  $\text{ignited}(X,T') :- \text{hot}(X,T), \text{no\_fbs}(X,T), \text{delay}(T,T',3,4)$ . Double-ended arcs are in order to prevent “undoing” preconditions.

easily simulate that fire spread takes between 3 and 4 minutes.

In the evolving Petri Net representation, it is still not easily possible to specify the spatial aspect of fire spread. In other words, we are in the need to adjust not only the time stamps but also the colors of the tokens passing through the transition. This can be achieved by attaching an “output operator” to the transition’s arcs leading to its output places. In Figure 9 we will associate a box labelled “Neighbors” with the arc leading to place “Ignited”. The operator will create multiple output tokens corresponding to all neighbor compartments. In FOPC the parallel effect is achieved through introducing an extra predicate neighbor:

$\text{ignited}(Y,T') :-$

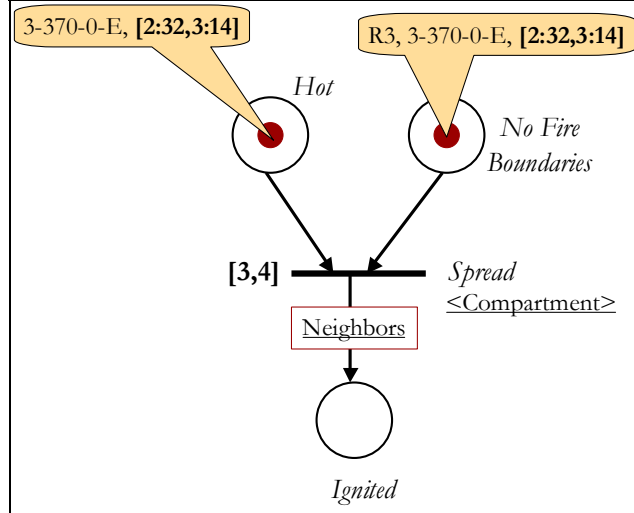


Fig. 9. Step 4: delay interval  $[3,4]$  increases passing tokens' time stamps, arc operator "Neighbor" affects the color of tokens. It will create output tokens for all neighbor compartments. In the FOPC the parallel effect is achieved through introducing an extra predicate `neighbor: ignited(Y,T') :- hot(X,T), no_fbs(X,R,T), delay(T,T',3,4), neighbor(X,Y)`. Double-ended arcs are needed to prevent "undoing" preconditions.

```

hot(X,T),
no_fbs(X,R,T),
delay(T,T',3,4),
neighbor(X,Y).

```

#### 5.4 TIPNs with inhibitory arcs

At this point both our models (TIPN and FOPC) are very close to what we originally desired. Currently, however, tokens always get retracted as the transition fires. This does not correspond to the reality as the fact that fire spreads from compartment  $X$  to compartment  $Y$  does not render compartment  $X$  cold (i.e., *not hot*). This drawback can be addressed by introducing so-called double-ended arcs (Figure 10). They operate just like regular arcs except the corresponding enabling tokens remain in their places. Note, that unlike the case of classical P/T nets, implementing double-ended arcs with regular return arcs can be problematic as all time-stamps get adjusted by the delay interval of the transition (see the next section for details).

The reader may have also noticed that we currently have no way to express negation but through introducing negated concepts (atoms) such as "No Fire Boundaries Are Set". It would be more convenient to represent negation explicitly. The justification for doing so is similar to that of using NOT-gates in electronic circuit design. Explicit negation facilitates connections between

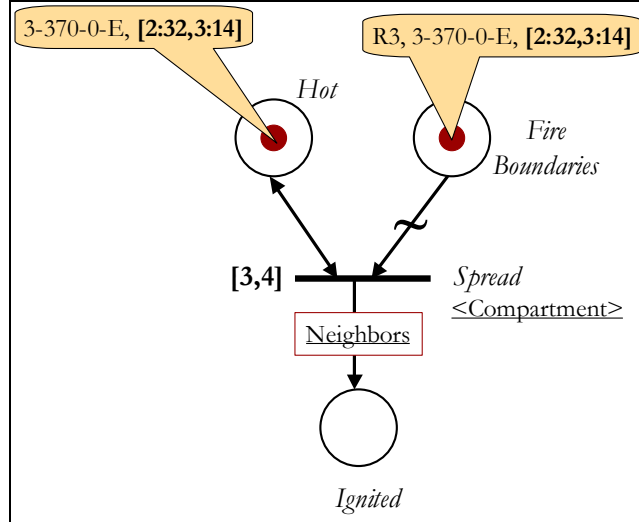


Fig. 10. Step 5: We finish the extension process by adding two new arc types: inhibitory arcs marked with tilde and double-ended arcs. The corresponding FOPC clause is: `ignited(Y,T') :- hot(X,T), not fbs(X,R,T), delay(T,T',3,4), neighbor(X,Y).`

states and their negated counter-parts and thereby increases modularity of TIPN design.

In TIPNs explicit negation is accomplished with a special arc type. Such arcs are called “inhibitory arcs” and are marked with a tilde in the diagrams. In FOPC we simply add negation connective (**not**):

```
ignited(Y,T') :-
    hot(X,T),
    not fbs(X,R,T),
    delay(T,T',3,4),
    neighbor(X,Y).
```

The illustration is now complete. The examples used were considerably simplified for presentational clarity. More realistic examples are demonstrated in the next section and in particular in Figure 12.

## 6 TIPN Evaluation

In this section we discuss TIPN fitness for hyper-real-time qualitative simulation.

### 6.1 TIPNs Attributes Relevant to Qualitative Simulation

TIPNs were found effective for qualitative simulation of concurrent temporal processes. In particular:

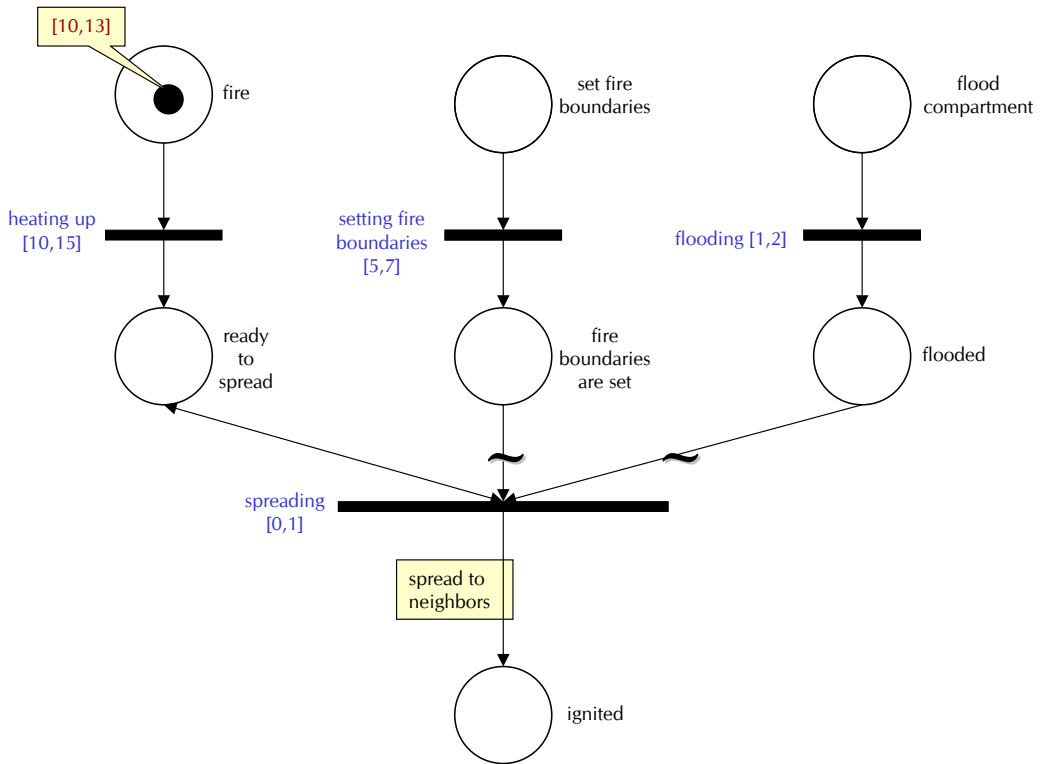


Fig. 11. The fire is about to spread to a missile compartment. Should the damage control coordinator order setting fire boundaries or flood the engulfed compartment?

- The following example demonstrates an advantage of using interval time stamps. Suppose, a fire is about to spread to a missile compartment. Its spread that normally takes 10 to 15 minutes can be prevented by cooling the walls (i.e., setting the fire boundaries around the compartment) or by flooding the compartment. The former method may take 5 to 7 minutes as a fire-fighting team needs to travel to the location, hook up water hoses, and start cooling down the walls with the hose water. On the other hand, flooding a compartment is done via installed sprinklers activated remotely. Consequently, it takes between 1 and 2 minutes. Unfortunately, flooding a compartment with sea water destroys the equipment located in the compartment and worsens the overall ship stability. Finally, the fire in the original compartment is known to have started between scenario time of 10 (a missile hit) and 13 (the fire alarm went off) minutes. The current time is 13 minutes. Should the damage control coordinator flood the compartment or order setting fire boundaries? What if the current time is 17 minutes?

The answer to these questions can be efficiently computed with a TIPN model shown in Figure 11. The fire will be ready to spread at  $[20,28]$  (i.e., no earlier than 20 and no later than 28 minutes scenario time). Unless fire boundaries are set or the compartment is flooded it will take 0 to 1 minute to ignite the neighboring missile compartment. If the order to set fire boundaries is dispatched at time=13 then the fire boundaries will be

set at [18,20] which is *guaranteed* to prevent fire spread at [20,28]. However, if the order is issued at time=17 then the fire boundaries can be set up as early as 22 minutes but are not guaranteed to be in place until time=24. Therefore, there is possible unprotected window of [20,22] (between the earliest time the fire can spread and the earliest time the fire boundaries can be set up). Therefore, fire boundaries are the preferable way of stopping the fire spread until scenario time=15. After that flooding the compartment is guaranteed to be reliable up to time=18.

As discussed in section 4.5, TIPNs can be made more or less “pessimistic” by adjusting the firing logic. With the rules presented earlier in the paper, the simulated ignition will take place unless fire boundaries or flooding is administered as early as 20 minutes into the scenario.

- TIPNs’ color mechanism enables explicit representation of process context (Bulitko and Wilkins, 1999c). Consequently, similar processes (e.g., fire spread) can be simulated by the same TIPN subnet regardless of the specific fire location.
- Additionally, the fidelity of TIPN simulation can be refined at the cost of enlarging the net. In the fire spread example, the designer can opt for simulating Bravo and Alpha class fires differently and replace the single “fire spread” TIPN subnet with two different TIPNs. Each subnet will then capture the details of fire spread of its own class. For instance, a Bravo-class fire TIPN may have a shorter delay interval in its fire spread transition as oil-based Bravo-class fires tend to be hotter and spread faster than their Alpha-class counterparts.
- Large systems can be simulated with a collection of TIPNs each responsible for a part of the original system. The TIPN components are connected simply by overlaying their input and output places. The benefits of modular TIPN design parallel these of procedure/class modularity in traditional programming languages and include faster development, labor division among developer teams, more effective debugging, and upgrades.
- As most Petri nets, TIPNs have a native graphical representation which facilitates drag-and-drop style visual development, running, and debugging of TIPNs. In some domains this enables interactive knowledge acquisition sessions with subject matter experts who may lack programming background.
- Unlike Monte-Carlo runs used with certain types of colored Petri nets, TIPN-based simulation doesn’t require multiple runs to generate a predicted state and therefore can be executed in a small fraction of the scenario time. We report empirical findings in the subsequent sections.
- Similar to optimized matching algorithms for productions (Forgy, 1982), TIPN data tokens are natively linked to *relevant* transitions. This streamlines TIPN running and speeds up the overall qualitative simulation.
- TIPNs support interacting processes and shared resources in an intuitive and efficient fashion. In particular, a shared resource (e.g., water pressure in vessel’s pipe network) that can be used for several different processes (e.g., fire fighting or equipment cooling) is represented as a place linked

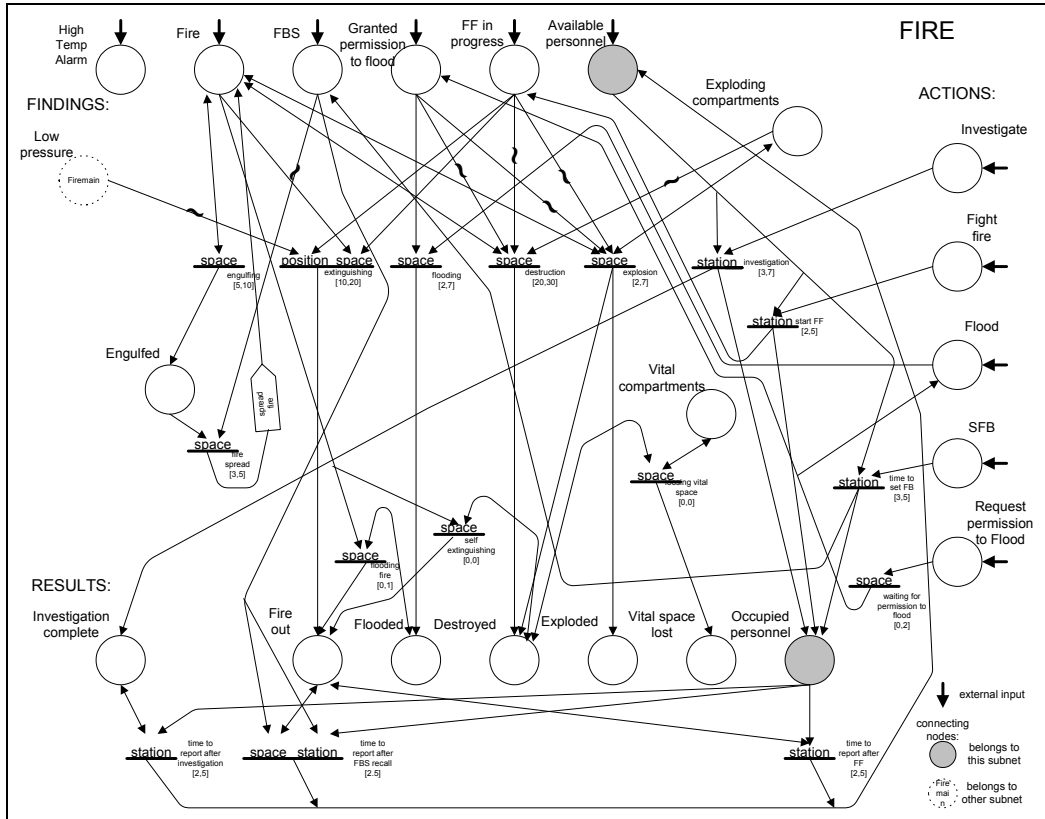


Fig. 12. Time Interval Petri Net used to model fire spread in Minerva 5.2 and the experiments described in this paper.

to several transitions. Therefore, firing one transition (i.e., engaging the resource) automatically disables the others. Together with the specific inference mechanism presented above this offers a natural conflict resolution scheme.

## 6.2 Empirical Studies

The issues of practical applicability of TIPNs as an environment module include: (a) the degree of speed-up over traditional numeric simulators, (b) TIPN scalability, and (c) the accuracy of environment. To provide one data point regarding these issues, a series of experiments were conducted in the domain of ship board damage control.

Figure 12 shows a fragment of the TIPN model used to simulate ship damage crises management. Other TIPN models of a similar complexity can be found in (Bulitko, 1998, 2000). This particular fragment models fire spread including the actions of the fire-fighting crews. Other TIPN fragments model aspects of ship damage control such as flooding and firemain management.

The ship is broken into compartments or spaces. The fire spread places ship

compartments in one of the following states: normal, engulfed, destroyed, extinguished, and flooded. Damage control personnel are based off designated compartments called stations. The TIPN shown in Figure 12 takes as input the following variables: “low pressure” indicating an insufficient water pressure in the firemain sea-water pipe network, “high temperature alarm” indicating a specific alarm going off, “fire” indicating a fire report for a particular set of compartments, “FBS” indicating the fact that fire boundaries are set on a compartment, “Granted permission to flood” indicating that the captain allowed to flood a compartment, “FF in progress” representing fire fighting efforts in progress in a compartment, and “Available personnel” listing all un-engaged damage control personnel. These variables describe the state of the system. The second category of the variables refer to the actions whose effects we seek to envision. These include: “investigate” representing the action of checking a compartment for fire, “fight fire” order, “flood” compartment order, “SFB” standing for “set fire boundaries on a compartment” order, and “request permission to flood” representing a request to the captain.

The results of envisionment are represented via the following output variables: “investigation complete” indicates completion of a compartment investigation, “fire out” marks the extinguished state of a compartment, “flooded” indicates a flooded compartment, “destroyed” compartment state refers to the complete burn-out, “exploded” compartment state indicates an explosive combustion in a compartment, “vital space lost” means that a critical compartment is destroyed by an explosion, and “occupied personnel” stores the engaged personnel assignments.

In addition to its inputs and outputs, the net also has a few internal variables. These are “engulfed” representing a compartment on fire, “explosive compartments” listing all compartments with explosive materials in them, and “vital compartments” referring to critical spaces on the ship.

### *6.3 Comparison between a Numeric Simulator and a TIPN*

The experimentation to measure the accuracy of the TIPN involved four fire-spread scenarios of different scale. The scenarios had 1, 5, 10, and 20 initial ignitions (primary damage events) at the beginning of each scenario correspondingly. No fire suppressants were introduced. The DC-SIM damage control numeric simulator (Shou et al., 2001) was compared to the TIPN shown above on these scenarios. The prediction interval was varied in the following steps: 1, 5, 10, and 20 minutes. Thus, 16 envisionment runs were conducted for the numeric simulator paralleled by 16 runs with the TIPN envisioner.

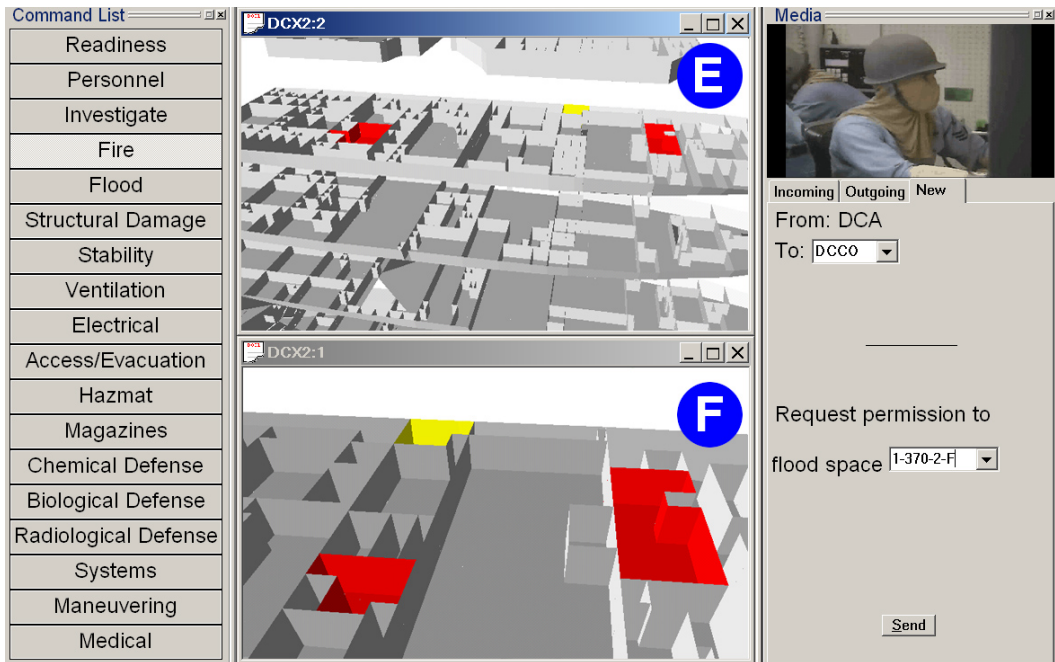


Fig. 13. Visualization of the physical shipboard damage progression. Compartments are colored-coded according to the type and extent of damage in them. Interactions among the stations are shown as well.

### 6.3.1 DC-SIM Numeric Simulation

In order to appreciate the complexity of the task, a brief overview of the DC-SIM numeric simulator (Shou et al., 2001) is presented in this section.

DC-SIM uses first-principles and compiled numeric methods to model major systems aboard a DDG-51 naval vessel including: fire and smoke (combustion, radiation, internal free convection, vent convection, wall conduction, boundary layer effect, direct fire fighting, fire boundaries, ventilation (de-smoking)) and flooding (firemain network, valves, pumps, ruptures). Finally, a comprehensive visualization module allows a visualization of the damage states in real-time or retroactively (Figure 13).

The degree of fidelity allows DC-SIM to be used for training of Damage Control Assistants (DCAs) at the Surface Warfare Officer School (SWOS) in Newport, RI. Toward this end, DC-SIM serves as a component of an immersive damage control trainer called DC-Train (Wilkins et al., 2001). In particular, DC-SIM allows placing a ship in a crisis state and then simulating the progression of the crisis through time based on the laws of physics and the damage control response actions taken by ship personnel. Observing such phenomena in real-time (Figure 13) and solving multiple damage-control scenarios provides Navy officers with training that would not otherwise be available, since it is cost-prohibitive to repeatedly inflict major damage to an actual ship in order to provide DCAs with whole-task training in real-time crisis management.

Number of primary fires	Envisionment interval (seconds)	TIPN computation time (seconds)	Degree of speed-up
1	1200	0.010	120,000x
5	600	0.014	42,857x
5	1200	0.015	80,000x
10	600	0.015	40,000x
10	1200	0.016	75,000x
20	600	0.016	37,500x
20	1200	0.031	38,710x

Table 1

Degree of speed-up of the TIPN envisionment module over a numeric simulator.

### 6.3.2 Degree of Speed-Up and Scalability

The numeric simulator was able to keep up its real-time operation even when simulating 20 simultaneous primary damage events that would in turn result in many more uncontrolled fire spreads. Table 1 shows the timings for the experiments and the degree of speed up. As mentioned above, primary fires were introduced at the very beginning of each scenario and were let burn uncontrollably. The envisionment/simulation was terminated at the end of the envisionment interval. For the purpose of this paper the numbers for the numeric simulator computation time are approximated using the simulated scenario time. In practice, the actual computation time was within 10% of the estimate.

As a summary, speed-ups of three to five orders of magnitude were feasible with the TIPN envisionment module. It should be also noticed that increasing number of primary fires did slow down the TIPN module though it was still above the desired speedup of 12,000x described in Section 1 of this paper.

### 6.3.3 Accuracy of Envisionment

As table 2 illustrates, the agreement between the numeric simulator and the TIPN envisioner was perfect (0 false positives and 0 false negatives) for the prediction intervals under 10 minutes. Longer envisionment intervals caused some differences between fire spread envisioned by the TIPN module and the fire spread modelled by the numeric simulator. Some of the discrepancies can be explained through the very basic initial design of the TIPN fire spread model that, for example, assumed that fire *always* spreads to *all neighbor compartments*. At the same time, the numeric simulator had a much finer spread model involving wall temperatures, probability, and presence of combustible

Number of primary fires	Envisionment interval (minutes)	False positives (fires envisioned by TIPN but not the simulator)	False negatives (fires envisioned by the simulator but not TIPN)	TOTAL discrepancy
1	1	0	0	0
1	5	0	0	0
1	10	12	0	12
1	20	20	4	24
5	1	0	0	0
5	5	0	0	0
5	10	11	0	11
5	20	19	4	23
10	1	0	0	0
10	5	0	0	0
10	10	13	6	19
10	20	20	13	33
20	1	0	0	0
20	5	0	0	0
20	10	5	11	16
20	20	7	23	30

Table 2

Envisionment accuracy of the TIPN module versus the numeric simulator measured in the number of fires. The total discrepancy is the sum of false positives and false negatives.

materials.

#### 6.4 Use of TIPN in a Decision Making System

Are these discrepancies crucial for the task of damage control? We tested the TIPN envisionment module in more realistic and practically useful settings of the ship damage control domain. A much greater set of physical and personnel activities phenomena were handled (Bulitko, 1998). TIPNs were used as an envisioner within a rule-based real-time decision-making system Minerva-DCA (Bulitko and Wilkins, 1999a; Bulitko, 1998, 2000). The system's tasks were (i) to maintain situation awareness by taking in damage reports and sensor readings aboard a naval vessel as well as actively verify crisis reports by send-

ing investigator teams and (ii) to provide a casualty response by dispatching damage control teams and activating damage control devices (Wilkins et al., 2001).

On every decision-making cycle Minerva-DCA’s rule-based deliberation module suggests a set of feasible damage control actions to take. For instance, a fire in a missile compartment can be addressed by hosing it down with a fire-fighting agent (action 1) or, alternatively, by flooding the entire compartment (action 2).

The TIPN envisionment module is then used to predict effects of proposed actions in hyper-real time. In the example above, qualitative simulation of action 1 may lead to a terminal explosion while action 2 will lead to a loss of the weapons in the compartment while saving the ship.

A static state evaluator is then invoked on the predicted states. Minerva-DCA used a machine learned one-hidden layer feed-forward artificial neural net to generate numeric scores of domain states. Finally, the action leading to the highest ranked predicted state is selected for domain execution.

To evaluate the system’s performance, 160 scenarios were run within the DC-Train ship damage control simulator at the Navy’s Surface Warfare Officer School (SWOS) in Newport, Rhode Island. These scenarios were judged by experts to be realistic and approximately 60 different Navy officers solved them. We compared the performance of Minerva-DCA (with the TIPN envisioner) to that of the Navy officers at SWOS. Any scenario had one of the three outcomes: “ship lost”, meaning that a major disaster such as a missile compartment explosion was reached; “ship possibly saved”, meaning that at 25 minutes scenario time the ship was still alive yet there were active crises; and “ship saved”, meaning that there were no active crises at the 25-minute mark. The results are shown in Figure 14.

The use of Minerva with the TIPN envisionment module resulted in 117 out of 160 ships being saved. This is a 318% improvement over human DCAs wherein 28 ships were saved. Likewise, TIPN-equipped Minerva lost 21 ships, or 46% fewer than the Navy officers did.

## 7 Summary and Conclusions

The following main contributions were presented in the paper:

- (1) Properties of qualitative simulation as a tool for selecting better actions within real-time decision-making are considered. Previous research is reviewed and its fitness for the task is discussed.
- (2) A Petri nets based approach to support decision-making through qualitative simulation is presented. Petri nets formalism is known for its solid

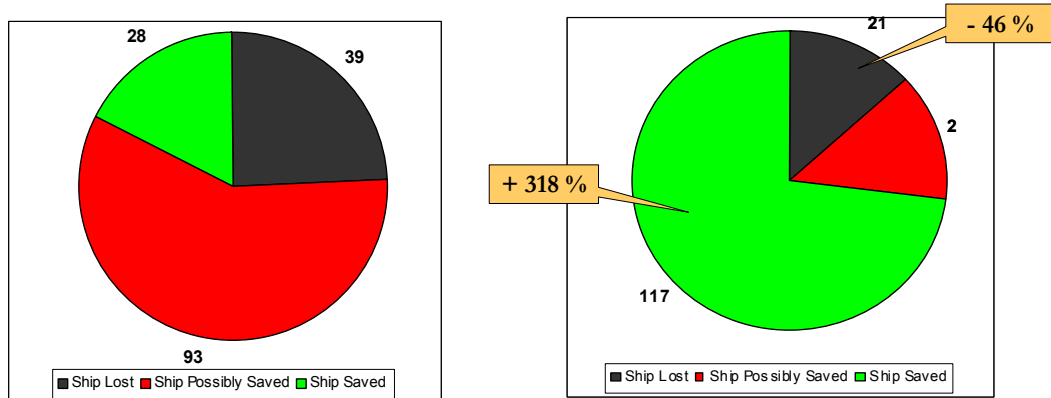


Fig. 14. Performance of Minerva with a TIPN environment module in the domain of ship damage control vs. human experts. The left pie chart shows the distribution of outcomes for scenarios presented to Navy experts. The chart on the right presents the distribution for Minerva-DCA.

- theoretical base, clear syntax and semantics, intuitive graphic representation, and native concurrency support (Peterson, 1981; Murata, 1989).
- (3) Shortcomings of classical P/T nets with respect to the task are discussed and the formalism is appropriately extended. The main extension is temporal intervals to model temporal uncertainty. The new formalism is hence called Time Interval Petri Nets (TIPNs).
  - (4) TIPNs are effective for qualitative simulation of concurrent temporal processes as they explicitly support temporal uncertainty in states and state changes, the context of processes simulated, modularity of the design, visual debugging and development, and adjustable degree of simulation fidelity. Furthermore, methods for automated and semi-automated acquisition of TIPNs have been designed and implemented (Bulitko, 2000; Bulitko and Wilkins, 2002a).
  - (5) The framework has been applied to the real-time decision-making domain of ship damage control for the tasks of automated problem-solving and intelligent tutoring (advising, critiquing, and scoring). In a large exercise involving 160 simulated ship crisis scenarios, a TIPN-equipped decision-making system showed a 318% improvement over Navy officers by saving 89 more ships.

Subsequently, the directions for future research include:

- (1) A TIPN model encodes what is commonly known as the next state function  $\delta(s_t, a) = s_{t+1}$  where  $s_t$  is the world's state at time  $t$ ,  $a$  is the action the agent takes at time  $t$ , and  $s_{t+1}$  is the state of the world at time  $t + 1$ . In the framework presented in this paper we used function  $\delta$  to predict the effects (i.e., to calculate  $s_{t+1}$ ) given the current situation ( $s_t$ ) and the action ( $a$ ) considered. Another interesting way of using  $\delta$  is to solve the equation above for action  $a$  given the current state  $s_t$  and the desired

future state  $s_{t+1}$ . Using the model in such a way will allow us to produce the action to take given where we are and where we want to be. If the equation allows for multiple solutions then a more refined model can be used to arbitrate among them.

- (2) A large number of formal analysis techniques have been developed for Petri nets. Examples include computing deadlocks, cycles, equivalence, coverability, and reachability (Peterson, 1981). A line of future research is to show how these techniques are valuable when using Petri nets for physical and intelligent agent simulation for a decision-making system. Some initial efforts in this direction are reported in (Bulitko, 2000).

## Acknowledgements

Participation of Sebastian Magda, Anthony Czupryna, Jr., David Fried, Valeriy Bulitko, and Dan Roth deserve a special recognition. We would also like to express our gratitude to the anonymous reviewers of Artificial Intelligence Journal. Their detailed and extremely useful feedback lead the paper to its current state. The research has been supported in part by ONR Grant N00014-95-1-0749, ARL Grant DAAL01-96-2-0003, NRL Contract N00014-97-C-2061, an NSERC discovery grant, and University of Alberta internal funding.

## References

- Baum, E., Smith, D., 1997. A Bayesian approach to relevance in game-playing. *Artificial Intelligence* 97, 195–242.
- Berthomieu, B., Diaz, M., March 1991. Modelling and verification of time dependent systems using Time Petri Nets. *IEEE Transactions on Software Engineering* 17 (3), 259–273.
- Berthomieu, B., Menasche, M., 1983. An enumerative approach for analyzing time petri nets. *IFIP Congress Series* 9, 41–46.
- Bulitko, V., 1998. *Minerva-5: A multifunctional dynamic expert system*. Master’s thesis, Department of Computer Science, University of Illinois at Urbana-Champaign.
- Bulitko, V., 2000. *Envisionment-based scheduling using time interval petri networks: Representation, inference, and learning*. Ph.D. thesis, University of Illinois at Urbana-Champaign.
- Bulitko, V., Wilkins, D., June 1999a. Automated instructor assistant for ship damage control. In: *Proceedings of the 11th Innovative Applications of Artificial Intelligence '99 conference*. Orlando, USA, pp. 778–785.
- Bulitko, V., Wilkins, D., 1999b. Damage control domain: Using petri nets for intelligent scheduling. In: Portinale, L., Valette, R., Zhang, D. (Eds.), *Proceedings of the Workshop on Application of Petri Nets to Intelligent System Development*. Williamsburg, USA, pp. 14–25.
- Bulitko, V., Wilkins, D., 1999c. Using petri nets to represent context in blackboard scheduling. In: *Proceedings of the American Association for Artificial Intelligence (AAAI) Workshop on Reasoning in Context for AI Applications*. Orlando,

- Florida, USA.
- Bulitko, V., Wilkins, D., 2002a. Inductive learning for time interval petri nets. *Journal of Machine Learning Research* (submitted).
- Bulitko, V., Wilkins, D., 2002b. Real-time decision making for shipboard damage control. In: *Proceedings of the AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*. AAAI Press, Edmonton, Alberta, pp. 37 – 46.
- Costa Miranda, M., 1999. Modeling and analysis of a multi-agent system using colored petri nets. In: Portinale, L., Valette, R., Zhang, D. (Eds.), *Proceedings of the Workshop on Application of Petri Nets to Intelligent System Development*. Williamsburg, USA, pp. 59–70.
- Donatelli, S., Kleijn, J., 1999. *Applications And Theory of Petri Nets 1999*. Vol. 1639. Springer.
- Forbus, K. D., 1984. Qualitative process theory. *Artificial Intelligence* 24, 85–168.
- Forgy, C., 1981. OPS5 user’s manual. Tech. Rep. CMU-CS-81-135, Department of Computer Science, Carnegie-Mellon University.
- Forgy, C., 1982. Rete: A fast algorithm for the many samples/many objects match problem. *Artificial Intelligence* 19 (1), 17–37.
- Grois, E., Wilkins, D. C., 2001. Comprehensive intelligent agent descriptions for DC-Train 4.0. Tech. Rep. UIUC-BI-KBS-2001-0038, Beckman Institute, University of Illinois, Urbana-Champaign.
- Heath, M., 1996. *Scientific Computing: An Introductory Survey*. McGraw Hill Text.
- Jensen, K., 1997. *Colored Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. Monographs in Theoretical Computer Science. Springer-Verlag.
- Korf, R. E., 1990. Real-time heuristic search. *Artificial Intelligence* 42 (2-3), 189–211.
- Medeiros, S., Xexeo, G., de Souza, J., 1999. Fuzzy petri nets for dynamic workflow in gis environment. In: Portinale, L., Valette, R., Zhang, D. (Eds.), *Proceedings of the Workshop on Application of Petri Nets to Intelligent System Development*. Williamsburg, USA, pp. 38–46.
- Merlin, P., Faber, D., Sept 1976. Recoverability of communication protocols. *IEEE Transactions on Communication* 24 (9), 1036–1043.
- Merlin, P. M., 1974. A study of the recoverability of computing systems. Ph.D. thesis, Irvine: University of California, available from Ann Arbor: Univ Microfilms, No. 75–11026.
- Murata, T., 1989. Petri nets: Properties, analysis, and applications. In: *Proceedings of IEEE 77*. pp. 541–580.
- NIST, 2002. Cfast reference documentation. <http://fast.nist.gov>, national Institute of Standards and Technology.
- Peterson, J., 1981. *Petri Nets Theory and Modeling of Systems*. Prentice-Hall, Inc.
- Russell, S. J., Wefald, E. H., 1991a. *Do the right thing: Studies in limited rationality*. MIT Press.
- Russell, S. J., Wefald, E. H., 1991b. Principles of metareasoning. *Artificial Intelligence* 49.
- Shou, G., Wilkins, D., Hoemann, M., Mueller, C., Tatem, P., Williams, F., 2001. Supervisory control system for ship damage control: Volume 2 – scenario generation and physical ship simulation of fire, smoke, flooding, and rupture. Tech.

- Rep. NRL/MR/6180-01-8572, Naval Research Laboratory, Washington, D.C.
- Sil, J., 1995. Intelligent expert and learning systems using petri nets. Ph.D. thesis, Jadavpur University.
- van der Aalst, W., 1993. Interval timed colored petri nets and their analysis. In: Marsan, M. (Ed.), Applications and Theory of Petri Nets. Vol. 691 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, pp. 453–472.
- Wilkins, D., Sniezek, J., Tatem, P., Williams, F., 2001. The DC-SCS supervisory control systems for ship damage control: Volume 1 – design overview. Tech. Rep. NRL/MR/6180-01-8559, Naval Research Laboratory, Washington, D.C.
- Zhang, D., Murata, T., 1996. Fixpoint semantics for a petri net model of definite clause logic programs. Advances in the Theory of Computation and Computational Mathematics 1, 155–194.