Relaxed Timed Coloured Petri Nets - A Motivational Case Study *

Guy Edward Gallasch and Jonathan Billington

Computer Systems Engineering Centre
University of South Australia
Mawson Lakes Campus, SA, 5095, AUSTRALIA
Email: guy.gallasch@unisa.edu.au

Abstract. Transitions in a Timed Coloured Petri Net exhibit an *eagerness-to-execute* property due to the time semantics employed. Every transition will occur at the earliest model time that it becomes enabled, unless conflict prevents it from doing so. It is known that this property may preclude discovery of optimal schedules, i.e. those that achieve the goal in the shortest time. In this paper we propose a relaxation of the time semantics that eliminates the eagerness-to-execute property. We provide motivation for our proposal in the context of task scheduling and describe an attempt to model the effect of our proposal.
Keywords: Timed Coloured Petri Nets, Eagerness-to-Execute, Task Scheduling.

1 Introduction

The Computer Systems Engineering Centre has long been involved in collaborative projects that apply Coloured Petri Nets (CPNs) [11] to industrial-scale systems. Two significant projects have been with Australia's Defence Science and Technology Organisation (DSTO) [3] (and later with National Information and Communication Technology Australia [14]) on the topic of operational planning [4, 5, 10, 12, 13, 15, 16], and modelling logistics processes [6–9]. We have used both untimed and Timed CPNs within these projects.

Timed CPNs incorporate a time semantics that causes an *eagerness-to-execute* behaviour to be exhibited by transitions: transitions will occur at the earliest model time that they become enabled (colour enabled and ready [11]) unless conflict results in them becoming disabled. We have observed that when using CPNs to model a process for the scheduling of tasks with fixed durations (as part of the operational planning work) that this eagerness-to-execute property manifests itself as task schedules in which all tasks are executed as early as possible. When producing schedules of tasks that are intended to achieve a particular goal, this behaviour may preclude discovery of the optimal schedule, as it is possible that delaying particular tasks may result in a schedule with a smaller makespan (time taken to achieve the goal). An example of this is given in [5]. We have made similar observations in our modelling work on logistics processes.

Early work (e.g. [2]) on scheduling used Timed Petri Nets, where durations are associated with transitions, and considered cyclic systems. Closest to our work is that of van der Aalst [1] who considered non-cyclic scheduling problems when using a timed Petri net formalism that associates timestamps with tokens. The timestamps are incremented when transitions fire, by a value associated with the transition. This is very close to the semantics of Timed CPNs, which are a generalisation of this formalism. Aalst's formalism thus possesses the same eager-to-execute semantics and he recognised the same problem with failing to discover optimum

* This work was stimulated by the NICTA-UniSA Research Agreement "Modelling and Analysis of Operations Planning using Untimed CPNs" of April 2006.

schedules. He discusses removing the restriction that transitions must fire as soon as possible to allow discovery of optimal schedules from the reachability graph (RG), however, the results in [1] are restricted to eager-to-execute semantics.

As suggested by Aalst [1], in this position paper, we propose a modified semantics for Timed Coloured Petri nets that eliminates the eagerness-to-execute property by removing the restriction that transitions must fire as soon as possible. Section 2 relates our experience with using both untimed and Timed CPN models for the purpose of task scheduling. In Section 3 we present our proposal and illustrate it with a simple example. We discover that in some situations, such as task scheduling, relaxation of the time semantics in this way is problematic, hence we present a refined proposal in Section 4. An attempt to model the effect of our proposal is given in Section 5 in the context of task scheduling. The paper concludes in Section 6.

2 Timed or Untimed?

A simple task-scheduling CPN model, presented in [4], was developed as part of the operational planning work that did not use conventional timestamps but encoded time within tokens. The rationale behind the development of that model was to avoid the eagerness-to-execute property of Timed CPNs, so that the execution of a task could be delayed and not necessarily occur as soon as all of its starting requirements were satisfied.

As we discovered, using untimed CPNs and modelling time within tokens was not restrictive enough. This methodology resulted in sequences of actions in the RG that were, in fact, inconsistent with the encoded timing information in the corresponding nodes and arcs [4], so-called *infeasible* schedules. This stems from the lack of a *synchronisation mechanism* in the untimed model, such as that provided by the global clock in Timed CPNs. Because of this lack, the occurrence of transitions in the untimed model was not forced to obey temporal constraints in the sequencing of task starting and terminating events, as the encoded time information was calculated on a 'per transition' basis and written into tokens, somewhat independently of the sequence of transition occurrences.

One way of addressing this shortcoming of the untimed model is to generate the infeasible schedules but remove them or simply ignore them when analysing the RG. This is undesirable, as it addresses the effect of the problem but doesn't tackle its source. Another, more satisfactory solution is to not generate the infeasible schedules in the first place, but not preclude any feasible sequences. This is the motivation for our proposal in this position paper in Sections 3 and 4.

2.1 Using Untimed CPNs that Encode Time

In order to force transition occurrences in the model to obey temporal constraints, a synchronisation mechanism needs to be implemented, to prevent events occurring before they are 'allowed to', based on the timing information contained in the tokens. For example, if two tasks start at time 0, one with a duration of 5 and the other with a duration of 10, then the task with a duration of 10 cannot terminate before the task with a duration of 5 (when considering fixed durations).

The RG of an untimed model will capture all interleavings of (concurrently) enabled transitions. It is this property of untimed CPNs that makes them desirable for the discovery of all schedules of tasks, as this property manifests itself as the ability to delay the start of

tasks. However, this strength is also a weakness when modelling time within tokens. Because a nondeterministic choice is made whenever more than one transition is (concurrently) enabled, there is nothing to prevent the occurrence of a particular transition from being delayed indefinitely (provided that at least one other transition is always enabled). There is no simple net structure mechanism to force a transition to occur 'at a particular time'.

2.2 An Approach Based on Timed CPNs

Thankfully, however, Timed CPNs *do* provide such a synchronisation mechanism, that *does* force transitions to occur at a particular model time. However, as we know, it forces all transitions to occur as soon as they are able. What we would like to do is to allow *some* transitions to be delayed.

The problem can be summarised as follows: Untimed CPNs provide the flexibility to delay transitions, but not easily enforce time constraints, whereas Timed CPNs enforce timing constraints but do not allow transitions to be delayed. In a sense, these two approaches sit either side of the solution we are looking for. Our investigations in [5] indicate that it is simpler to relax the behaviour of Timed CPNs than introduce rigidity into Untimed CPNs.

3 Removing the Eagerness to Execute Property

In a Timed CPN, in any given marking, the next transition to fire is nondeterministically selected from only those colour-enabled transitions that are ready at the earliest model time. The gist of our proposal is to allow this selection to be made from all colour-enabled transitions. This is essentially how untimed CPNs behave, but with the global clock advancing to the appropriate value (the amount required to enable the selected colour-enabled transition). This follows the proposal given in [1].

This requires but a minor change to the definition of the enabling rule of Timed CPNs as given in [11]. In terms of *steps* (multisets of binding elements), Definition 11.6 in [11] specifies that a step, Y, is enabled at time t' in a timed marking, (M, t^*) (where t^* is the value of the global clock in marking M), if and only if the following properties are satisfied:

1. all binding elements in the step satisfy the guard of the corresponding transition;
2. sufficient tokens exist in all untimed input places to satisfy the untimed input arcs for all binding elements in the step;
3. appropriate timed tokens exist in all timed input places to satisfy the timed input arcs at time t' for all binding elements in the step;
4. the current global clock, t^*, is less than or equal to t'; and
5. t' is the smallest time value for which there exists a step satisfying conditions 1 to 4.

Our proposal modifies the fifth condition so that any step can be considered, not just the steps enabled at the smallest time value. We propose condition 5 to become:

5. t' is the smallest time value that satisfies conditions 1 to 4 for step Y.

Hence, we do not require t' to be the smallest time that satisfies conditions 1 to 4 over all possible steps, thereby allowing any step that is colour enabled to be executed, and the model time to advance only to the necessary time to enable that step. Note that condition 4 is still valid with our proposed time semantics, as t' does not need to be the minimum time at which

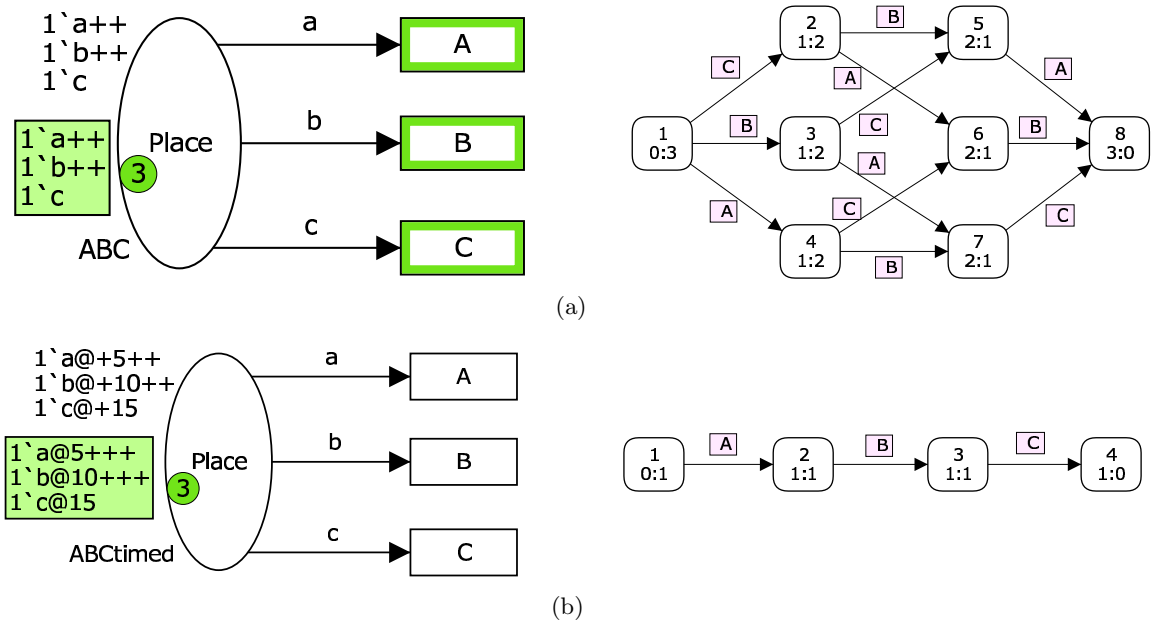

Fig. 1. (a) an Untimed CPN model and its Reachability Graph; and (b) a corresponding Timed CPN model and its Reachability Graph.

step Y is enabled, but rather greater than or equal to that minimum time. Our intention is for the modified condition 5 to allow steps to be enabled at times larger than the minimum required to enable a step, i.e. if step Y_1 is enabled at the earliest at time 5, and step Y_2 is enabled at the earliest at time 10, step Y_2 can occur at time 10, causing step 5 to be delayed until at least time 10. This modification still allows transitions to occur at the time they first become enabled, but it also allows transitions to be delayed. The delay is not arbitrary, but rather it corresponds to the increase in the global clock resulting from the occurrence of one or more other transitions.

As an example, consider the Untimed CPN and its RG in Fig. 1 (a). ABC is an enumerated colour set containing the values a, b and c. The three transitions, A, B and C, are concurrently enabled and can occur in any order, which is reflected in the RG. RGs are generated using interleaving and hence arcs of the RG represent a single binding element. In general a step enabled in the initial marking (marking 1) could comprise, for example, all three transitions. This corresponds to moving from marking 1 to marking 8 in one step.

Figure 1 (b) shows a Timed CPN version of the net in Fig. 1 (a) and its RG, in which the global clock is 0 in the initial marking. From the five conditions given above for the existing time semantics, the only step enabled by the initial marking is the step comprising transition A at time 5: A's guard is satisfied (condition 1); there are no untimed input places (condition 2); an 'a' token with timestamp 5 marks Place (condition 3); the current global clock, 0, is less than or equal to 5 (condition 4); and no other step exists that is enabled at a time less than 5 (condition 5). Transition A will occur at time 5, and for the same reasons B will occur at time 10, and C at time 15. From the firing rule of Timed CPNs, the values for the global clock for markings 2, 3 and 4 are 5, 10 and 15 respectively.

In the case of our relaxed timed semantics, the example in Fig. 1 (b) has 7 steps enabled in the initial marking (we omit the empty binding, $\langle\rangle$, from each binding element): 1‘A, 1‘B, 1‘C, 1‘A++1‘B, 1‘A++1‘C, 1‘B++1‘C, and 1‘A++1‘B++1‘C. Condition 1 is satisfied because

all guards are true; there are no untimed input places so condition 2 is satisfied; there are sufficient timed tokens on all timed input places and t' can be set to 5, 10 or 15 as necessary (condition 3); initially the global clock is zero, $t^* = 0$, so $t^* \leq t'$ (condition 4); and $t' = 5$ for 1`A, $t' = 10$ for 1`B and 1`A++1`B; and $t' = 15$ for the remaining four steps, to satisfy condition 5. When generating the RG we only consider interleaving and hence only the first 3 steps (1`A, 1`B and 1`C). On their occurrence we obtain markings 2, 3 and 4 shown in Fig. 1 (a), except that timing information has been added: timestamps as in Fig. 1 (b) and the value of the global clock being 0 for the initial marking, 15 for marking 2, 10 for marking 3 and 5 for marking 4 according to the firing rule for Timed CPNs. In marking 2, we can see that transitions A and B are concurrently enabled at $t' = 15$. t' must be set to 15 to satisfy both conditions 4 and 5. This leads to the successor markings 5 and 6, and their successor, marking 8, with the global clock remaining at 15 ($t^* = 15$). In marking 3, $t^* = 10$. C is enabled with $t' = 15$, to satisfy conditions 3 and 5 (thus condition 4 also holds), leading to marking 5 with $t^* = 15$. A is enabled with $t' = 10$ (to satisfy conditions 4 and 5) and results in marking 7, with $t^* = 10$. In marking 4, $t^* = 5$. B is enabled with $t' = 10$ (to satisfy conditions 3 and 5), leading to marking 7 and similarly C is enabled with $t' = 15$, resulting in marking 6. In marking 7, C is enabled with $t' = 15$ to satisfy conditions 3 and 5 (condition 4 is satisfied as $t^* = 10$), and when it occurs results in marking 8 ($t^* = 15$). Hence we can see that in this case the RG with relaxed time semantics is isomorphic to the untimed RG. This illustrates how our relaxed time semantics allows transitions to be delayed until after the occurrence of one or more other transitions enabled by a larger global clock.

4 Refining our Relaxed Time Semantics

The proposal in Section 3 follows our own initial belief and coincides with the proposal in [1]. But we have found situations in which this particular relaxation of the time semantics introduces behaviour that we do not desire. For example, our task scheduling models (see e.g. [13]) execute tasks of fixed duration using one transition to represent the start of the task and another transition to represent the termination of the task. If such a task has a fixed duration, d, then we desire the task termination transition to occur exactly d time units after the task start transition, and not be delayed. The time semantics resulting from our proposed change are too relaxed in this sense, as we can no longer enforce the execution of transitions at specific model times.

A refinement to our proposal is thus to allow only a subset of transitions to be delayed within a Timed CPN model. It is possible to modify the definition of Timed CPNs from [11] (Definition 11.4, for non-hierarchical Timed CPNs) so that the definition of transitions recognises two disjoint classes of transition: *fixed* transitions (not delayable) and *delayable* transitions; such that point 2 of Definition 11.4 from [11] becomes:

2. $T = T_f \cup T_d$ is a finite set of **transitions** comprising **fixed transitions**, T_f, and **delayable transitions**, T_d, such that $T_f \cap T_d = \emptyset$ and $P \cap T = \emptyset$.

The enabling rule must then be modified to disallow the enabling of any step that would require the global clock to advance past the model time at which any fixed transition is enabled. To do this, condition 5 of Definition 11.6 from [11] can be further modified from that in Section 3 to become:

5. t' is the smallest time value that satisfies conditions 1 to 4 for step Y, and there does not exist any other step comprising at least one fixed transition that is enabled at a time value less than t'.

By doing so, we are able to prevent certain transitions from being delayed, and hence in the context of task scheduling enforce fixed task durations. This proposal is more general than our first proposal in Section 3 in the sense that we can capture all the behaviour of the first proposal but also enforce the eagerness-to-execute behaviour exhibited by the current time semantics of Timed CPNs, whereas our first proposal cannot enforce such behaviour.

One of the motivations given in [11] for adopting the particular time semantics for Timed CPNs is to preserve occurrence sequences between the timed model and its corresponding untimed equivalent: the occurrence sequences exhibited by a Timed CPN model will be a subset of those exhibited by its corresponding Untimed CPN model [11]. This means that turning an untimed model into a timed model cannot introduce new behaviour in terms of occurrence sequences. A second motivation, for considering that the occurrence of transitions is an instantaneous event, is that the set of reachable markings of the Timed CPN is a subset (when excluding the global clock and timestamps) of those of the corresponding Untimed CPN, i.e. no new markings are introduced when time is introduced into an untimed model.

We conjecture that our proposal for relaxing the time semantics of Timed CPNs also preserves these properties. Further, we contend that the reachability graphs of relaxed timed CPNs (with no fixed transitions) are isomorphic to the corresponding untimed model.

5 Modelling our Proposal

In [5] we have attempted to produce a highly simplified version of the task scheduling model from [13] that exhibits behaviour similar to that which would result from implementation of the proposed time semantics from Section 4. We do so by using additional net structure to *induce* a delay.

Figure 2 shows the simplified task execution engine from [5]. The two transitions, Start and Terminate, model the start and termination of tasks. Tasks to be executed initially reside in the Idle Tasks place, moving to the Executing place while executing, and ending up in the Terminated place once executed. Resources and conditions provide constraints on the specific tasks that can be executed, and when. We wish for Start to be able to be delayed, but not Terminate, so that fixed durations for tasks can be enforced by the model. More details of the basic operation of the complete model, from which this highly simplified version was derived, can be found in [13].

In bold are two places, Semaphore 1 and Semaphore 2, both members of the Semaphore fusion set (but drawn separately for convenience), and arcs that connect these two places to the Start and Terminate transitions. These two places and their associated arcs implement the mechanism that mimics a relaxed time semantics, along with the net shown in Fig. 3, which we now explain.

In Fig. 3 is a place, Semaphore, and a transition, InduceDelay. The sole purpose of this page is to allow the model time to advance instead of a timed transition being forced to occur. The Semaphore place, part of the same fusion set as the two semaphore places in Fig. 2, contains a single token comprising a boolean and a time value. InduceDelay is enabled whenever the boolean part is true, and its occurrence changes the boolean part to false without changing the time value stored in the token.

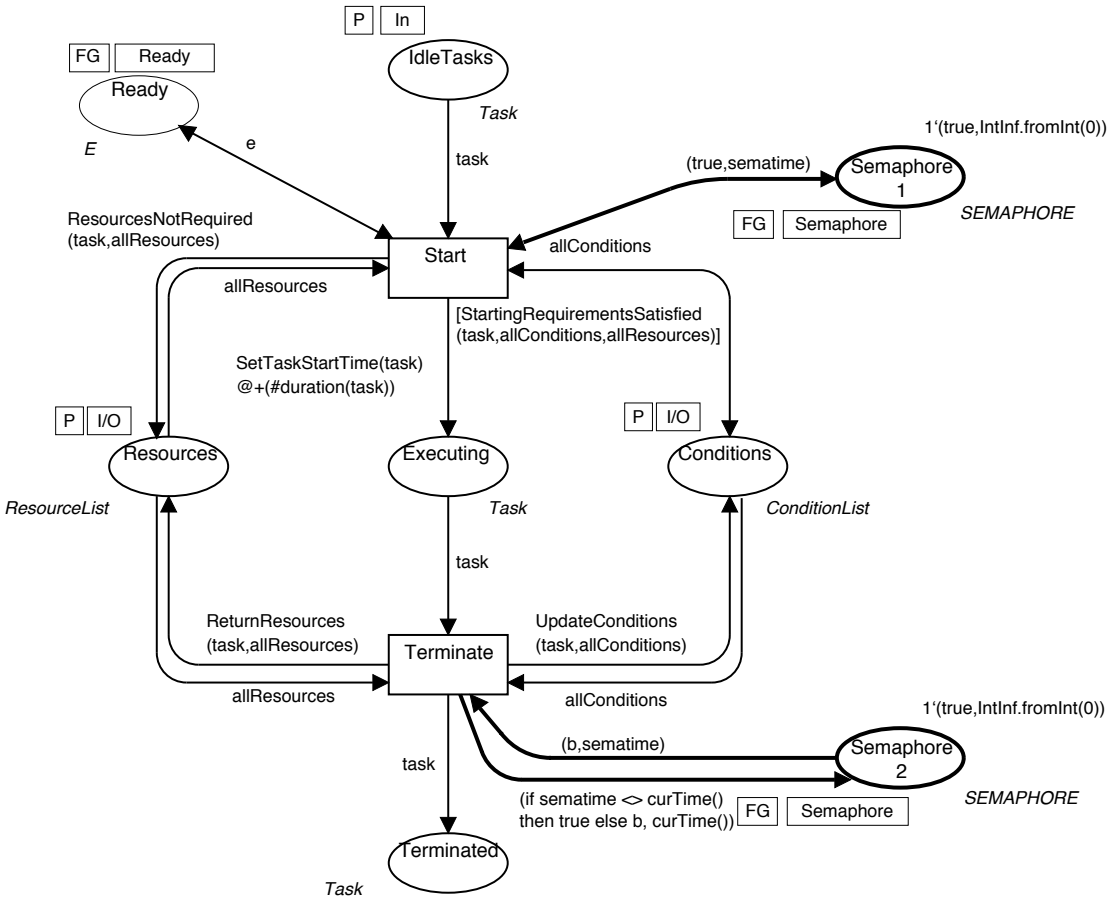

Fig. 2. Part of a Simplifed Task Execution Engine Model from [5].

Figure 2 shows how the token in Semaphore can be used to delay transitions. Transitions that we want to be able to delay are connected to the Semaphore place with an arc inscription requiring the semaphore to be true, as is the case with the Start transition in Fig. 2. InduceDelay is enabled at model time 0, and hence any number of transitions can occur at time 0 (provided they are colour-enabled and ready) before InduceDelay nondeterministically sets the semaphore to false. So, suppose Start was enabled at time 0. If InduceDelay occurs first, Start becomes disabled, and will not be enabled again until the semaphore is again set to true. Alternatively, Start could have occurred at time 0. Transitions that we don't wish to delay are connected to the Semaphore place by arcs that reset the boolean to true and update the time value to the current model time, as is the case with the Terminate transition. Once such a transition occurs, the timed transitions that were previously disabled (delayed) can now occur, if all other enabling conditions are still satisfied.

One emergent property of this implementation was that transitions can be delayed indefinitely, and indeed never occur. This is either a limitation or a bonus, depending on your point of view. For our operational planning work this was a bonus, as we also wanted to model the ability of tasks to be delayed indefinitely. Unfortunately, implementing a solution in this way results in state explosion caused by the additional InduceDelay transition and Semaphore place, and is not straightforwardly extended to larger models with more complex transition interactions.

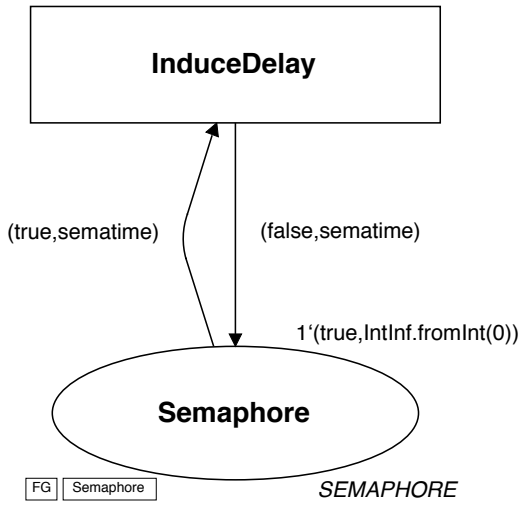

Fig. 3. The InduceDelay Transition to mimic our proposed relaxed time semantics.

6 Conclusions and Future Work

In this paper we propose a relaxation of the time semantics of Timed CPNs to allow some or all transitions to be excluded from the eagerness-to-execute behaviour imposed by the existing time semantics of Timed CPNs. This new time semantics has application in the area of task scheduling, including the modelling of logistics systems and operational planning.

The relaxed semantics is a true relaxation in the sense that it permits all behaviour of the current time semantics of Timed CPNs while also permitting all behaviour allowed by our first proposal in Section 3. Our intial proposal did also capture the eagerness-to-execute behaviour given by the current time semantics but not without also allowing the possibility that all transitions could be delayed, whereas our second proposal has the capability to enforce eagerness-to-execute behaviour.

If our conjecture that occurrence sequences and markings are preserved from Untimed CPNs under our new time semantics, it may follow that many, if not all, of the theory surrounding the properties and analysis capabilities of Timed CPNs will also hold for our new time semantics. Hence, our new time semantics has the potential to be used for the analysis of any system currently analysed by Timed CPNs. Establishing a firm theoretical foundation is thus of high significance and priority. If this can be established, and our proposal given tool support, we would like to investigate the use of this modified time semantics in a case study of significance, such as the operational planning work in [5, 13].

In the previous section we illustrated an attempt to model our proposed time semantics, however this becomes cumbersome and error-prone for large-scale models, as we discovered when we attempted to do something similar to the logistics distribution network model in [6,8]. Implementing the modified time semantics in a tool will allow the operational planning models to be used without any modification.

References

1. W.M.P. van der Aalst. Petri net based scheduling. *OR Spectrum*, 18:219–229, 1996.
2. J. Carlier and P. Chretienne. *Timed Petri Net Schedules*, volume 340 of *Lecture Notes in Computer Science*, pages 62–84. Springer, 1988.

3. Defence Science and Technology Organisation (DSTO). http://www.dsto.defence.gov.au.
4. B. Han G. E. Gallasch and J. Billington. COAST User Interface Design, Integration and Support, and the Development of an Untimed COAST Server. Technical Report CSEC-22, Computer Systems Engineering Centre Report Series, University of South Australia, June 2005, revised July 2005. (93 pages).
5. G. E. Gallasch and J. Billington. Modelling and Analysis of Operations Planning Using Untimed and Timed CPNs. Technical Report CSEC-27, Computer Systems Engineering Centre Report Series, University of South Australia, September 2006. (73 pages).
6. G. E. Gallasch, N. Lilith, and J. Billington. A Coloured Petri Net Model of a Defence Logistics Physical Network. Technical Report CSEC-25, Computer Systems Engineering Centre Report Series, University of South Australia, August 2006. (140 pages).
7. G. E. Gallasch, N. Lilith, and J. Billington. Coloured Petri Net Modelling of Defence Logistics. Technical Report CSEC-33, Computer Systems Engineering Centre Report Series, University of South Australia, July 2008. (198 pages).
8. G. E. Gallasch, N. Lilith, J. Billington, L. Zhang, A. Bender, and B. Francis. Modelling Defence Logistics Networks. *International Journal on Software Tools for Technology Transfer, special section on CPN'06*, 10(1):75–93, 2008.
9. G. E. Gallasch, C. Moon, B. Francis, and J. Billington. Modelling personnel within a defence logistics maintenance process. In *Proceedings of 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008)*, March 2008. (10 pages).
10. G. E. Gallasch J. Freiheit and J. Billington. About the Use of Untimed CPN Models for COAST and Further COAST Client Development Support. Technical Report CSEC-20, Computer Systems Engineering Centre Report Series, University of South Australia, December 2004. (33 pages).
11. K. Jensen and L.M. Kristensen. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems.* Springer, 2009.
12. L. M. Kristensen. The COAST Server, Design and Implementation. Technical Report CSEC-5, Computer Systems Engineering Centre Report Series, University of South Australia, July 2002.
13. L. M. Kristensen, P. Mechlenborg, L. Zhang, B. Mitchell, and G. E. Gallasch. Model-based Development of a Course of Action Scheduling Tool. *International Journal on Software Tools for Technology Transfer, special section on CPN'06*, 10(1):5–14, 2008.
14. NICTA - Australia's ICT Research Centre of Excellence. http://www.nicta.com.au.
15. L. Zhang, L. M. Kristensen, C. Janczura, G. E. Gallasch, and J. Billington. A Coloured Petri Net based Tool for Course of Action Development and Analysis. In *Formal Methods in Software Engineering and Defence Systems 2002, Proceedings of the* Satellite Workshops on Software Engineering and Formal Methods *and* Formal Methods Applied to Defence Systems, volume 12 of *Conferences in Research and Practice in Information Technology Series*, pages 125–134. Australian Computer Society Inc., 2002.
16. L. Zhang, L. M. Kristensen, B. Mitchell, G. E. Gallasch, P. Mechlenborg, and C. Janczura. COAST - An Operational Planning Tool for Course of Action Development and Analysis. In *Proceedings of the 9th International Command and Control Research and Technology Symposium (ICCRTS), Copenhagen, Denmark.*, 2004.