Comparing the Expressiveness of Timed Automata and Timed Extensions of Petri Nets

Jiří Srba*

Department of Computer Science, Aalborg University,
Selma Lagerlöfs Vej 300, 9220 Aalborg East, Denmark
srba@cs.aau.dk

Abstract. Time dependant models have been intensively studied for many reasons, among others because of their applications in software verification and due to the development of embedded platforms where reliability and safety depend to a large extent on the time features. Many of the time dependant models were suggested as real-time extensions of several well-known untimed models. The most studied formalisms include Networks of Timed Automata which extend the model of communicating finite-state machines with a finite number of real-valued clocks, and timed extensions of Petri nets where the added time constructs include e.g. time intervals that are assigned to the transitions (Time Petri Nets) or to the arcs (Timed-Arc Petri Nets). In this paper, we shall semi-formally introduce these models, discuss their strengths and weaknesses, and provide an overview of the known results about the relationships among the models.

1 Introduction

In formal modelling and verification of software and hardware systems there is an obvious need for considering time features and hence the study of so-called *time dependant models* has become increasingly important. The overall research in this area is motivated, among others, by the development of embedded platforms which use time features and should be reliable and correct [53].

As mentioned in [83], majority of these formalisms rely on the assumption of *orthogonality* between discrete and continuous (time delay) changes which significantly simplifies the underlying semantics of time dependant models. A run of the system can be then seen as a sequence of steps where continuous time progress and discrete events alternate. We are going to adopt such an approach also in this paper.

In what follows, we shall take a closer look at three prominent examples of time dependant systems, namely Networks of Timed Automata (NTA), Time Petri Nets (TPN) and Timed-Arc Petri Nets (TAPN). These models have existed for a relatively long period of time but they have been developed to a large extent

* The author is supported in part by a grant of the Ministry of Education of the Czech Republic, project No. 1M0545.

independently of each other, even though they share many common features. Citing [21]:"In spite of many technical resemblances and their overlapping application domains, few material was available until recently comparing expressiveness of these ... models." Fortunately, there has recently been a growing interest in mutual comparisons of different models that include real-time constructs. One reason for this development could be the recent availability of efficient verification tools for timed automata, which stimulates the translation approaches from TPN and TAPN to TA, rather than the development TPN/TAPN verification techniques that resemble those already available for timed automata.

In this paper we will, by means of examples, introduce the models of timed automata, time Petri nets and timed-arc Petri nets. Then we provide an overview of decidable and undecidable problems related to these models and present a summary of their strengths and weaknesses. Finally, we give an up-to-date overview of work that aims at comparing the relative expressive power of these models. In the last section we finish with a few concluding remarks and the possible future development in this area.

2 Time Dependant Models

Time dependant models are often obtained by extending the untimed ones with time constructs that enable to manipulate in different ways the passing of time. Two well-studied approaches include the extension of finite automata resp. networks of communicating finite automata with a number of real-valued clocks and different timed extensions of Petri nets.

2.1 Timed Automata and Networks of Timed Automata

Timed Automata. Timed automata were introduced by Alur and Dill [6, 7] and have by now been recognized as one of the classical formalisms for modelling real-time systems with dense time.

A *timed automaton* (TA) is a finite-state machine extended with a finite number of synchronous clocks. Transitions in the automaton are conditioned on the clock values and taking a transition can affect (reset) the values of selected clocks. A typical transition in a timed automaton looks like

$$\ell \xrightarrow{g,a,r} \ell'$$

where ℓ and ℓ' are *locations* (or *states*) of the automaton, g is a clock *guard*, a is a *label* (or *action*) of the transition, and r is a subset of clocks that are reset when the transition is taken. Guards are defined by the abstract syntax

$$g ::= x \bowtie k \mid x - y \bowtie k \mid g \wedge g$$

where x, y are elements from a given finite set of *clocks*, k is an integer, and $\bowtie \in \{\leq, <, \geq, >, =\}$. A timed automaton which does not contain any guard of the form $x - y \bowtie k$ is called *diagonal-free*.

Guards can be also associated with locations and then they are called *invariants*. Invariants restrict the amount of time that can be spent in a given location and we usually consider only invariants given by the abstract syntax: $g ::= x \leq k \mid x < k \mid g \wedge g$.

A *configuration* of a timed automaton is a pair of a location and a *clock valuation*, which is a function assigning to each clock a nonnegative real number (the time that has elapsed since the last clock reset). Consider an example of a timed automaton in Figure 1 where ℓ_0 is a given initial location.

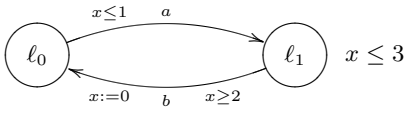

Fig. 1. Example of TA

Starting from the configuration $(\ell_0, [x = 0])$ where the value of the clock x is zero, we can delay for any nonnegative real number d and reach the valuation $(\ell_0, [x = d])$. This is called a *time elapsing step*. As long as we are in the configuration where $d \leq 1$, we can also perform a *discrete step* by taking the transition labelled by a. This is because the guard $x \leq 1$ is satisfied. We then reach the configuration $(\ell_1, [x = d])$. In the location ℓ_1 we can now delay for at most $3 - d$ time units because of the invariant $x \leq 3$ associated with the location ℓ_1. As soon as the value of the clock x is at least 2 we can also take the discrete transition labelled with b and return to the initial configuration $(\ell_0, [x = 0])$ because the value of the clock x is reset to 0 by taking this transition.

Networks of Timed Automata. Single timed automata can be also run in parallel. A *network of timed automata* (NTA) is a parallel composition of a finite number of timed automata where the actions are partitioned into the set of output (suffixed with !) and input (suffixed with ?) actions. A component in the parallel composition can make a discrete step under an action $a!$ only if there is another component ready to make a discrete step under the complementary action $a?$. In this case the two parallel components perform a handshake synchronization and move simultaneously to their new locations. Other types of synchronization, e.g. in the style of Arnold-Nivat [10] via synchronization functions, are also possible and studied. The parallel components may also perform a time elapsing step and in this case all the clocks age in a synchronous manner. Networks of timed automata are not more expressive than a single timed automaton (which can be shown by a standard product construction) but they are exponentially more concise.

2.2 Petri Nets

Untimed Petri Nets. Petri nets (PN) were first suggested in early sixties by Carl Adam Petri in his PhD thesis [73] and have since then become a popular and wide-spread model of distributed systems with many applications and a large number of academic as well as industrial tools (see [51] for an updated list). One of the main advantages of this model is its intuitive graphical representation. Consider an example of a Petri net in Figure 2.

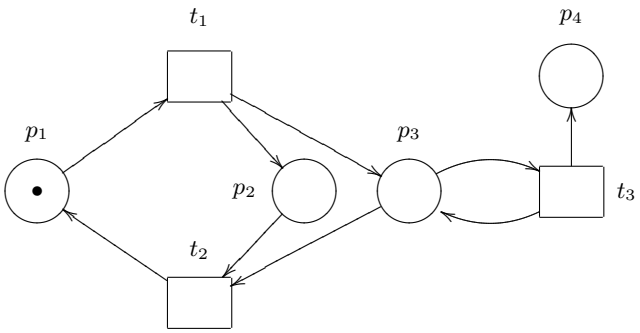

Fig. 2. Example of PN

The circles are called *places*, boxes are called *transitions* and arrows (also called *arcs*) can connect only a place to a transition, or a transition to a place. The dot in the place p_1 is called a *token* and it is connected with the behaviour of the net. One place can hold several tokens. A token assignment to the places is called a *marking*. A transition t, in a given marking, that has a token in every of its input places (those connected to t by an arc) is called *enabled* and can *fire* by consuming one token from every of its input places and producing a new token to every of its output places (having an incoming arc from the transition t).

In our example the transition t_1 is enabled and when it fires it consumes the token in the place p_1 and produces two new tokens, one into the place p_2 and the other into the place p_3. Now either the transition t_2 can fire and return the net to its initial marking, or the transition t_3 can fire, consume the token from p_3, and produce a new token into the places p_3 and p_4. Now the transition t_3 can fire again, leaving the token in p_3 and producing a second token into the place p_4. It is easy to see that by repeatedly firing the transition t_3, an arbitrary large number of tokens will be placed into p_4. This net is hence an example of so-called *unbounded net*.

Formally, a net is called *bounded* or *safe* if the number of tokens in all reachable markings is bounded by some a priori given constant. A special case when every place in any reachable marking contains at most one token is called *1-safe*.

Remark 1. Despite the infinite state-spaces of unbounded Petri nets, several properties like marking reachability, coverability, boundedness and others are still decidable (for an overview see e.g. [47, 46]), while strong bisimilarity and some other related problems are undecidable [54]. In order to compare the expressive power of Petri nets (and their extensions with time) with finite automata-based models, we usually consider only bounded nets. They are still useful for modelling and analyzing many real-life problems (for an overview of different case studies see e.g. [52]).

Extending Petri Nets with Time. Unlike timed automata, Petri nets offer several options where the time constructs can be associated to. For example *timed transitions Petri nets* were proposed in [76] where transitions are annotated with their durations. A model in which time parameters are associated with places is called *timed places Petri nets* and it was introduced in [82]. For an overview of the different extensions see e.g. [32, 88, 72].

In this paper we shall focus on two other, perhaps the most studied, extensions called Time Petri Nets of Merlin and Faber [66, 67] introduced in 1976 and the model of Timed-Arc Petri Nets first studied around 1990 by Bolognesi, Lucidi, Trigila and Hanisch [26, 50].

2.3 Time Petri Nets

In Time Petri Nets (TPN) [66, 67] each transition has an associated time interval which gives the earliest and latest firing time of the transition since it became enabled. One can think of this as every transition having an associated real-valued clock, which gets initialized at the moment the transition becomes enabled. The transition can fire as soon as the clock value reaches the earliest firing time and it *must* fire no later than the latest firing time, unless the transition got disabled by the firing of some other transition. This means that TPN can express *urgent* behaviour (also called the *strong* semantics). The precise semantics of the behaviour is, however, not completely obvious and several different variants can be considered [17]. It seems that the *intermediate semantics* is the most often used one. Here all transitions disabled after consuming the tokens of the transition being fired, as well as the firing transition itself, are reinitialized.

Consider the following example of TPN presented in Figure 3. It uses the underlying untimed net from Figure 2 enriched with time intervals on transitions. Assume that the clocks x_1, x_2 and x_3 are associated to the transitions t_1, t_2 and t_3, respectively. In the initial marking the clock x_1 gets initialized to the value 0 and as the firing interval of t_1 is $[2, 4]$, the only possible behaviour of the net is to delay any time between 2 and 4 time units and then fire the transition t_1. This produces two new tokens into p_2 and p_3 and the clocks x_2 and x_3 get initialized at the same time. Now the net has to wait for another 3 time units and latest by 4 time units the transition t_2 must fire and we reach the initial marking. This implies that t_3 is never firable and unlike the underlying untimed net, the TPN in our example is bounded and even 1-safe. Assume now that the time interval $[5, 7]$ associated with t_3 is replaced by $[2, 8]$. Now after firing t_1 and starting the

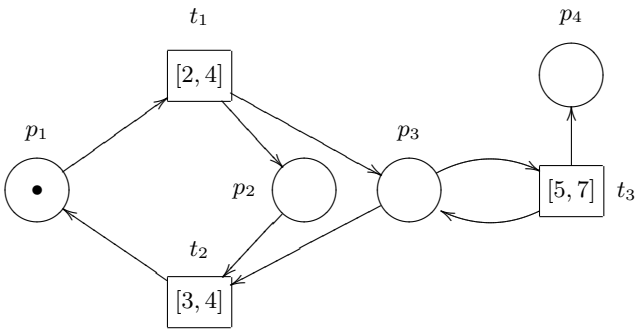

Fig. 3. Example of TPN

clocks x_2 and x_3, the transition t_3 is ready to fire already after two time units, and after another 1 time unit both t_2 and t_3 are enabled. Note that after firing t_3 both x_2 and x_3 get reinitialized to 0 because we adopted the intermediate semantics. In other semantics the clock x_2 might not get restarted.

2.4 Timed-Arc Petri Nets

The last extension of Petri nets that we consider in this paper is called *Timed-Arc Petri Nets* (TAPN) [26, 50]. Here the time entity (also called *age*) is associated with tokens. We can think of this as if every token in the net had its own private clock. The arcs from places to transitions are labelled by time intervals which restrict the age of tokens that can be used to fire a given transition. When new tokens are produced, their age is set by default to 0. The usually considered semantics is *non-urgent* (or *weak*), which means that tokens can grow older even if this disables the firing of certain transitions (sometimes for ever). Consider the following TAPN in Figure 4 with the same underlying untimed net as in Figure 2.

In the initial marking there is one token of age 0 in the place p_1. As the age of the token does not belong to the interval $[2, 4]$, the transition t_1 is not enabled yet, but only after two time units. Then anytime within another 2 units the transition can fire and produce two new tokens of age 0 into the places p_2 and p_3. Note, however, that due to the non-urgent semantics it is possible that the age of the token in p_1 grows beyond 4 and hence the transition t_1 gets disabled for ever. Should this happen, the token in the place p_1 is called *dead*. Assume now that we are in a marking with two tokens of age 0 in the places p_2 and p_3. After waiting for two time units, the transition t_3 becomes enabled and if it fires it resets the age of the token in the place p_3 to 0 and produces a new fresh token into the place p_4. By waiting for another two time units the age of the token in the place p_2 reaches the value 4 and the tokens in the places p_3 and p_4 will be of age 2. Now, for example, the transition t_2 can fire, consuming the tokens in places p_2 and p_3 and producing a fresh one into the place p_1. Another possible

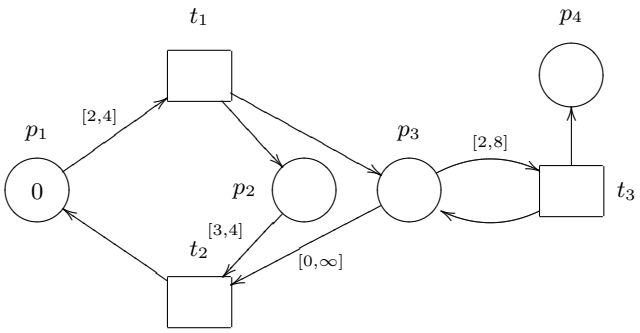

Fig. 4. Example of TAPN

behaviour (due to the non-urgent semantics) is that the transition t_3 keeps firing for ever after arbitrary delays between 2 and 8 time units. This shows that our TAPN is unbounded and moreover every token in the place p_4 will have its own unique age. This demonstrates that unlike for TA and TPN, we cannot rely only on a finite number of clocks associated with a given automaton/net.

Our example also shows that every time the transition t_3 fires, the age of the token in p_3 is reset to 0. In some applications this might be undesirable and hence in [84] the model was extended with so-called *read-arcs*, which allow to test for a presence of a token in a certain place but do not change its age. It was later shown in [30] that this extension is not only convenient from the modelling point of view but it also extends the expressiveness of the model.

3 Overview of Known Decidability Results

In this section we shall mention a selection of positive and negative decidability results for our three time dependant models.

3.1 Networks of Timed Automata

The *region graph* construction provides a universal tool for arguing about the decidability of several problems on timed automata [6, 7]. Using this technique it was shown, e.g., that reachability is decidable in PSPACE for TA [7] as well as for NTA [5], untimed language equivalence for TA is decidable in PSPACE [7], and untimed bisimilarity for TA is decidable in EXPTIME [59]. Practically more efficient algorithms are usually achieved by considering *zones* instead of regions (see e.g. [16]). Somewhat surprisingly even timed bisimilarity for timed automata is decidable. Using region graphs on a product construction, timed bisimilarity was shown to be decidable in EXPTIME [41].

Unfortunately, timed language equivalence for timed automata is undecidable [7]. In fact, even the universality problem (whether a given timed automaton generates *all* timed traces) is undecidable.

3.2 Time Petri Nets

It is known that even very simple classes of timed nets extended with the notion of urgency have the full Turing power [55] (can simulate e.g. Minsky two-counter machines) and hence most of the verification problems for TPN like reachability and boundedness are undecidable. For additional remarks see also [23, 74].

On the other hand, for bounded nets the *state class method* [24] can be often used to derive decidability results. Regarding the complexity, it is known that reachability (and also TCTL model-checking) of bounded TPN is PSPACE-complete [27].

3.3 Timed-Arc Petri Nets

In spite of the fact that reachability is decidable for ordinary Petri nets [65], it is undecidable for timed-arc Petri nets [78], even in the case when tokens in different places are not required to age synchronously [69]. On the other hand, coverability, boundedness and other problems remain decidable for TAPN [77, 3, 1], which are also known to offer 'weak' expressiveness, in the sense that TAPN cannot simulate Turing machines [25]. Coverability is decidable even for TAPN extended with read-arcs [30]. These results hold due to the monotonicity property (adding more tokens to the net does not restrict the possible executions) and the application of well-quasi-ordering (for a general introduction see [48]) resp. better-quasi-ordering [2] techniques.

When we consider the subclass of 1-safe TAPN, it is known that the reachability problem is no more difficult than in untimed 1-safe Petri nets and hence it is decidable in PSPACE [84]. This is the case also for 1-safe TAPN with read-arcs.

4 Strengths and Weaknesses of the Models

This section aims at providing a comparison of the three models w.r.t. to their modelling capabilities and their applicability for verification purposes.

4.1 Networks of Timed Automata

Pros. Timed automata are nowadays a widely used modelling formalism with rich theoretical foundations and a number of developed verification tools like UPPAAL [58], KRONOS [37], IF [38] and CMC [57]. A number of case studies (see e.g. [9, 14] for an overview) demonstrate that timed automata are a suitable formalism for modelling of systems of industrial sizes and the tools have already reached a reasonable degree of maturity and efficiency. A combination of an easily understandable syntax and semantics together with the support for C-like constructs and data structures (e.g. in the tool UPPAAL) makes this a widely applicable and successful approach to modelling and verification of time dependant systems.

Cons. On the other hand, timed automata are less convenient for modelling of certain types of applications like work-flow management systems, production lines with shared resources and other systems which require e.g. a dynamic creation of new processes. Here models based on Petri nets are most commonly used. It is also known that e.g. TPN are exponentially more concise than NTA [31, 29], so even though timed automata are as expressive as bounded Petri net based models, the size of TA models for certain time dependant systems might be unnecessarily large. As claimed in [45] timed automata also lack a support for high level composable graphical patterns to support systematic design of complex systems. The explicit use of invariants in timed automata also causes problems: during the design of a specification it is easy to introduce time deadlocks (time cannot progress and no transition is firable). This is usually interpreted as an inconsistency of the specification and should be avoided. However, such specification errors cannot occur neither in TPN (urgency is applicable only as long as some discrete transitions are still enabled) nor in TAPN (time is always allowed to progress). For further discussion on this issue see [83].

4.2 Time Petri Nets

Pros. The model of TPN has been around for several decades and it has proven to be useful for modelling of a wide range of real-time systems including work-flow processes, scheduling problems and others [15, 11, 75, 68]. It has an implicit notion of urgency suitable for modelling many real-life problems and implicitly avoids the construction of ill-defined timed systems. There are available a few public verification tools like TINA [22] and ROMEO [81].

Cons. Unbounded TPN are too expressive (have full Turing power) and are hence unsuitable for automatic verification. This means that most of the verification approaches are limited to bounded nets. Also the modelling power is restricted as TPN cannot model some useful features like e.g. a dynamic creation of new processes which carry their own time information. In fact the number of time clocks (associated to the transitions) is limited in advance by the structure of the net and lacks more flexibility. Last, the exact semantics is not only more difficult to understand than for the other two models but it also offers several variants [17]. Different papers use different variants of the semantics, though the intermediate semantics described also in this paper seems most common. This is in contrast with the conclusion drawn in [17] where the authors give arguments why other policies like e.g. the *persistent atomic semantics* should be preferred over the intermediate semantics. The current TPN tools are relatively new, have a limited number of constructs like data structures which include e.g. arrays, and they lack hierarchical modelling features.

4.3 Timed-Arc Petri Nets

Pros. TAPN are particularly suitable for modelling of manufacturing systems, work-flow management and similar applications. It is the author's opinion that

TAPN offer a more intuitive semantics than TPN; at least there are no competing variants of the semantics. TAPN provide an easy to understand interplay between the token game and the associated time features. For modelling of certain applications, especially systems where a number of identical time processes share the same pattern of behaviour [3, 4] but also others [70, 79, 80, 71], TAPN provide a convenient modelling formalism. By simply adding more and more tokens to the net, more and more processes (each carrying its own clock) can be modelled without any change to the net itself. Neither TPN nor NTA provide this feature. As several problems like e.g. coverability are decidable even for unbounded TAPN, a parametric reasoning is possible [3].

Cons. One of the major weaknesses of TAPN is the lack of the possibility to model urgent behaviour. This might be one of the reasons, together with the fact that the model was introduced much later than TPN, why TAPN deserved less attention among the scientists. Currently there are no publicly available tools, though some prototype implementations already exist.

5 Relationships Among the Models

In this section we will provide an overview of the expressiveness results for the studied models. As already mentioned, unbounded TPN and TAPN generate infinite state-spaces and e.g. the reachability problem is undecidable for both of them. Hence to draw a fair comparison between the nets and timed automata, most of the work focuses on the relationship between *bounded* or *1-safe* nets and timed automata. There are some exceptions like e.g. the work in [40] which translates unbounded TPN into UPPAAL-like NTA extended with unbounded integer arrays.

From TA/NTA to TPN. A translation from diagonal-free TA without invariants and strict constraints to 1-safe TPN preserving weak timed bisimulation was suggested by Haar et al. in [49]. In the paper they, however, consider only weak (non-urgent) semantics for TPN. Bérard et al. give in [18] a linear translation from diagonal-free TA with invariants to 1-safe TPN up to timed language equivalence. They also show that TA are strictly more expressive than TPN w.r.t. weak timed bisimilarity and in [19] (see also a forthcoming journal article [20]) they identify a strict subclass of TA which is equivalent with bounded TPN w.r.t. weak timed bisimilarity. In [21] Berthomieu et al. suggest to extend the TPN model with priorities and show that this is enough to establish an equivalence with NTA w.r.t. weak timed bisimilarity. Another reduction from TA to TPN, which includes also diagonal constraints and updates to integral values, was presented by Bouyer et al. in [31]. The reduction preserves timed language equivalence and works in linear resp. quadratic time, depending on what features of TA are included. This work, however, does not include invariants in TA. In [31] the authors also provide a translation from NTA to TPN, which preserves timed language equivalence, but introduces new deadlocks into the system behaviour.

From TPN to TA/NTA. Haar et al. provided in [49] a translation from 1-safe TPN to TA preserving weak timed bisimilarity. It improved the complexity of the previously known work based on enumerative methods [23, 24] and their translation is polynomial but only in the size of the TPN reachability graph. On the other hand, they allow only non-strict intervals and consider the weak (non-urgent) semantics for TPN, while the other papers focus on the standard strong (urgent) semantics. Another approach by Lime and Roux [62] extends these results to bounded TPN but also requires first a construction of the state class graph of the given bounded TPN. An extended version of their paper [63] provides an efficient reduction technique to decrease the number of clocks in the resulting timed automaton. A structural translation from TPN to NTA preserving weak timed bisimilarity, which does not require the construction of the state class graph, was proposed by Cassez and Roux in [40]. Their reduction uses NTA extended with arrays of (unbounded) integers and enables to translate even unbounded TPN into the extended NTA. If the input net is bounded, the values in the integer arrays are bounded too and automatic verification is hence possible. An implementation of the translation is available as a part of the TPN tool Romeo [81] and the results seem promising as documented on several case studies [40]. A possible problem with this approach is a potentially high number of clocks in the produced NTA. Recently D'Aprile et al. suggested in [44] an alternative technique for the translation from TPN to TA. Their method bypasses the construction of the state class graph (as used e.g. in [49, 62]) by considering only the underlying untimed reachability graph. It preserves timed bisimulation and TCTL properties. According to the experiments carried out by the authors, it is competitive with the other approaches on a number of case studies. On the other hand, it requires the underlying untimed net to be bounded, while the other approaches require only TPN boundedness. Empirical methods to deal with this limitation are outlined in the paper. Yet another approach is presented in [43] by Cortés et al. where the authors translate a more general model of TPN called PRES+ into UPPAAL-like timed automata, suggest several optimizations of the reduction, and provide two case studies. Their reduction works only for 1-safe nets and unfortunately there is no argument about the correctness of the translation.

From TA/NTA to TAPN and Backwards. The first result (we are aware of) which compares the expressive power of TA and TAPN is by Sifakis and Yovine [83] from 1996. They provide a translation of 1-safe timed-arc Petri nets (with urgent behaviour) into timed automata (with invariants) which preserves strong timed bisimilarity but their translation causes an exponential blow up in the size. Srba established in [84] a strong relationship (up to isomorphism of timed transition systems) between NTA without invariants and a superclass of 1-safe TAPN extended with read-arcs. When we are interested only in the reachability questions, the reductions work in polynomial time. Recently Bouyer et al. in [30] presented a reduction from bounded TAPN (with read-arcs) to 1-safe TAPN (with read-arcs) which preserves timed language equivalence (over finite words, infinite words and non-Zeno infinite words). This demonstrates that

NTA without invariants and bounded TAPN with read-arcs are timed language equivalent. The authors in [30] also provide a number of expressiveness results for several subclasses of TAPN with read-arcs.

From TPN to TAPN and Backwards. We are aware of only few detailed studies comparing TPN and TAPN. In [42] Cerone and Maggiolio-Schettini study several timed extensions of bounded Petri nets w.r.t. language equivalence. Regarding the two classes of our main focus they show that TPN and TAPN are language equivalent w.r.t. weak (non-urgent) semantics and that TPN form a subclass of TAPN when considering the strong (urgent) semantics. In [35] Boyer and Vernadat show that the inclusion of TPN in TAPN is strict (in the strong semantics). A further comparison of the different classes w.r.t. weak timed bisimilarity is provided in [34]. We should note that all the work mentioned so far in this paragraph uses the so-called single server semantics [36] for TAPN where the timing information to be remembered in every marking is constant. However, TAPN are mostly studied with the multi-server semantics where each token carries its own timing information. In this case, as concluded in [33], TPN express timed behaviour and TAPN express time behaviour *and* time constraints.

6 Conclusion

We have introduced three popular models of time dependant systems, compared their relative strengths and weaknesses and presented an overview of the known relationships across these models. Even though these formalisms share many common features and there exist mutual translations between the models, it is hard to say what should be *the* model for time dependant systems. Different applications might be modelled in different modelling formalisms with a very varying effort for a human modeller to create such models. We should also note that due to their complexity, the modelling tricks used in the translations between the different models are often unsuitable for a direct use by a human modeller. Nevertheless, the possibility of automatically translating all the time dependant models studied in this paper to e.g. a network of timed automata offers the option of creating hybrid tools that will enable to enter real-time models (or even their subparts) in all kinds of different formalisms, depending on the preference of the human modeller, and still have a clearly defined semantics and a support for automatic verification.

The tools for timed automata verification seem to be most developed at the moment. So while adding additional modelling features to Petri net based models, researchers should check whether these features can be translated to their TA counter-parts and the verification techniques/tools can be reused for them, rather than rediscovered. For example in the theory of TA there has been recently lots of research on extending timed automata with price/cost [60, 8, 28] and studying timed games [64, 39, 13, 12] as well as on-line testing [56, 61]. There is also a tool support implementing many of these theoretical results. Let us mention e.g. the tool UPPAAL-CORA [85] for cost-optimal reachability,

UPPAAL-TIGA [86] for timed games and UPPAAL-TRON [87] for on-line testing. It is likely that the existing translations from TPN and TAPN to NTA can be extended with such features and a future research might focus on studying efficient translation approaches that include several of these new aspects.

Acknowledgements. I would like to thank to Patricia Bouyer, Franck Cassez, Alexandre David and Olivier H. Roux for their comments on a draft of this paper.

References

[1] P.A. Abdulla, P. Mahata, and R. Mayr. Dense-timed Petri nets: Checking zenoness, token liveness and boundedness. *Logical Methods in Computer Science*, 3(1):1–61, 2007.

[2] P.A. Abdulla and A. Nylén. Better is better than well: On efficient verification of infinite-state systems. *Proceedings of 15th Annual IEEE Symposium on Logic in Computer Science (LICS'00)*, pages 132–140, 2000.

[3] P.A. Abdulla and A. Nylén. Timed Petri nets and BQOs. In *Proceedings of the 22nd International Conference on Application and Theory of Petri Nets (ICATPN'01)*, volume 2075 of *LNCS*, pages 53–70. Springer-Verlag, 2001.

[4] Parosh Aziz Abdulla, Johann Deneux, Pritha Mahata, and Aletta Nylén. Forward reachability analysis of timed Petri nets. In *Joint International Conferences on Formal Modelling and Analysis of Timed Systems (FORMATS'04) and Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'04)*, volume 3253 of *LNCS*, pages 343–362. Springer-Verlag, 2004.

[5] L. Aceto and F. Laroussinie. Is your model checker on time? On the complexity of model checking for timed modal logics. *Journal of Logic and Algebraic Programming*, 52–53:7–51, 2002.

[6] R. Alur and D. Dill. Automata for modelling real-time systems. In *Proceedings of the 17th International Colloquium on Algorithms, Languages and Programming (ICALP'90)*, volume 443 of *LNCS*, pages 322–335. Springer-Verlag, 1990.

[7] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[8] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. *Theoretical Computer Science*, 318(3):297–322, 2004.

[9] Tobias Amnell, Gerd Behrmann, Johan Bengtsson, Pedro R. D'Argenio, Alexandre David, Ansgar Fehnker, Thomas Hune, Bertrand Jeannet, Kim G. Larsen, M. Oliver Möller, Paul Pettersson, Carsten Weise, and Wang Yi. UPPAAL: Now, next, and future. In *4th Summer School on Modeling and verification of parallel processes (MOVEP'01)*, volume 2067 of *LNCS*, pages 99–124. Springer-Verlag, 2001.

[10] A. Arnold. *Finite Transition Systems.* Prentice-Hall, 1994.

[11] Raimundo Barreto, Sérgio Cavalcante, and Paulo Maciel. A time Petri net approach for finding pre-runtime schedules in embedded hard real-time systems. In *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops - W7: EC (ICDCSW'04)*, pages 846–851. IEEE Computer Society, 2004.

[12] Gerd Behrmann, Agnès Cougnard, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. UPPAAL-Tiga: Timed games for everyone. In Luca Aceto and Anna Ingolfdottir, editors, *Proceedings of the 18th Nordic Workshop on Programming Theory (NWPT'06), Reykjavik, Iceland*. Reykjavik University, 2006.

[13] Gerd Behrmann, Agnes Cougnard, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. UPPAAL-TIGA: Time for playing games! In *Proceedings of the 19th International Conference on Computer Aided Verification (CAV'07)*, number 4590 in LNCS, pages 121–125. Springer-Verlag, 2007.

[14] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM-RT'04)*, number 3185 in LNCS, pages 200–236. Springer-Verlag, 2004.

[15] Darlam Fabio Bender, Benoît Combemale, Xavier Crégut, Jean-Marie Farines, Bernard Berthomieu, and François Vernadat. Ladder metamodeling and PLC program validation through time Petri nets. In *Proceedings of the 4th European Conference on Model Driven Architecture — Foundations and Applications (ECMDA-FA'08)*, volume 5095 of *LNCS*, pages 121–136. Springer-Verlag, 2008.

[16] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 87–124. Springer-Verlag, 2003.

[17] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux. Comparison of different semantics for time Petri nets. In *Proceedings of the 3rd International Symposium on Automated Technology for Verification and Analysis (ATVA'05)*, volume 3707 of *LNCS*, pages 293–307. Springer-Verlag, 2005.

[18] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux. Comparison of the expressiveness of timed automata and time Petri nets. In *3rd International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *LNCS*, pages 211–225. Springer-Verlag, 2005.

[19] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux. When are timed automata weakly timed bisimilar to time Petri nets? In *Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'05)*, volume 3821 of *LNCS*, pages 273–284. Springer-Verlag, 2005.

[20] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux. When are timed automata weakly timed bisimilar to time Petri nets? *Theoretical Computer Science*, 2008. Forthcoming.

[21] B. Berthomieu, F. Peres, and F. Vernadat. Bridging the gap between timed automata and bounded time Petri nets. In *Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *LNCS*, pages 82–97. Springer-Verlag, 2006.

[22] B. Berthomieu, P-O. Ribet, and F. Vernadat. The tool TINA — construction of abstract state spaces for Petri nets and time Petri nets. *International Journal of Production Research*, 42(14):2741–2756, 2004.

[23] Bernard Berthomieu and Michel Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. Software Eng.*, 17(3):259–273, 1991.

[24] Bernard Berthomieu and Miguel Menasche. An enumerative approach for analyzing time Petri nets. In *Proceedings of IFIP Congress 1983 on Information Processing*, volume 9, pages 41–46. Elsevier Science Publishers, Amsterdam, 1983.

[25] T. Bolognesi and P. Cremonese. The weakness of some timed models for concurrent systems. Technical Report CNUCE C89-29, CNUCE–C.N.R., 1989.

[26] T. Bolognesi, F. Lucidi, and S. Trigila. From timed Petri nets to timed LOTOS. In *Proceedings of the IFIP WG 6.1 Tenth International Symposium on Protocol Specification, Testing and Verification (Ottawa 1990)*, pages 1–14. North-Holland, Amsterdam, 1990.

[27] Hanifa Boucheneb, Guillaume Gardey, and Olivier (H.) Roux. TCTL model checking of time Petri nets. Technical Report IRCCyN number RI2006-14, Nantes Cedex, 2006, updated in 2008.

[28] Patricia Bouyer, Ed Brinksma, and Kim Guldstrand Larsen. Staying alive as cheaply as possible. In *Proceedings of the 7th International Workshop on Hybrid Systems: Computation and Control (HSCC'04)*, volume 2993 of *LNCS*, pages 203–218. Springer-Verlag, 2004.

[29] Patricia Bouyer and Fabrice Chevalier. On conciseness of extensions of timed automata. *Journal of Automata, Languages and Combinatorics*, 10(4):393–405, 2005.

[30] Patricia Bouyer, Serge Haddad, and Pierre-Alain Reynier. Timed Petri nets and timed automata: On the discriminating power of zeno sequences. *Information and Computation*, 206(1):73–107, 2008.

[31] Patricia Bouyer, Pierre-Alain Reynier, and Serge Haddad. Extended timed automata and time Petri nets. In *Proceedings of the 6th International Conference on Application of Concurrency to System Design (ACSD'06)*, pages 91–100. IEEE Computer Society, 2006.

[32] Fred D.J. Bowden. Modelling time in Petri nets. In *Proceedings of the Second Australia-Japan Workshop on Stochastic Models*, 1996.

[33] M. Boyer and M. Diaz. Non equivalence between time Petri nets and time stream Petri nets. In *Proceedings of the 8th International Workshop on Petri Nets and Performance Models (PNPM'99)*, pages 198–207. IEEE Computer Society, 1999.

[34] M. Boyer and Olivier H. Roux. Comparison of the expressiveness of arc, place and transition time Petri nets. In *Proceedings of the 28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency (ICATPN'07)*, volume 4546 of *LNCS*, pages 63–82. Springer-Verlag, 2007.

[35] M. Boyer and F. Vernadat. Language and bisimulation relations between subclasses of timed Petri nets with strong timing semantic. Technical Report No. 146, LAAS, 2000.

[36] Marc Boyer and Michel Diaz. Multiple enabledness of transitions in Petri nets with time. In *Proceedings of the 9th international Workshop on Petri Nets and Performance Models (PNPM'01)*, pages 219–228. IEEE Computer Society, 2001.

[37] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: A model-checking tool for real-time systems. In *Proceedings of the 10th International Conference on Computer-Aided Verification (CAV'98)*, volume 1427 of *LNCS*, pages 546–550. Springer-Verlag, 1998.

[38] Marius Bozga, Susanne Graf, Ileana Ober, Iulian Ober, and Joseph Sifakis. The IF toolset. In *Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM-RT'04)*, volume 3185 of *LNCS*, pages 237–267. Springer-Verlag, 2004.

[39] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR'05)*, volume 3653 of *LNCS*, pages 66–80. Springer-Verlag, 2005.

[40] Franck Cassez and Olivier H. Roux. Structural translation from time Petri nets to timed automata. *Journal of Systems and Software*, 79(10):1456–1468, 2006.

[41] Kārlis Čerāns. Decidability of bisimulation equivalences for parallel timer processes. In *Proceedings of the 4th International Conference on Computer Aided Verification, (CAV'92)*, volume 663 of *LNCS*, pages 302–315. Springer-Verlag, 1993.

[42] Antonio Cerone and Andrea Maggiolio-Schettini. Time-based expressivity of timed Petri nets for system specification. *Theoretical Computer Science*, 216(1-2):1–53, 1999.

[43] L. A. Cortés, P. Eles, and Z. Peng. Verification of real-time embedded systems using Petri net models and timed automata. In *Proceedings of the 8th International Conference on Real-Time Computing Systems and Applications (RTCSA'02)*, pages 191–199, 2002.

[44] Davide D'Aprile, Susanna Donatelli, Arnaud Sangnier, and Jeremy Sproston. From time Petri nets to timed automata: An untimed approach. In *Proceedings of 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07)*, volume 4424 of *LNCS*, pages 216–230. Springer-Verlag, 2007.

[45] Jin Song Dong, Ping Hao, Shengchao Qin, Jun Sun 0001, and Wang Yi. Timed patterns: TCOZ to timed automata. In *Proceedings of the 6th International Conference on Formal Engineering Methods (ICFEM'04)*, volume 3308 of *LNCS*, pages 483–498. Springer-Verlag, 2004.

[46] J. Esparza. Decidability and complexity of Petri net problems — an introduction. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, pages 374–428. Springer-Verlag, 1998.

[47] J. Esparza and M. Nielsen. Decidability issues for Petri nets — a survey. *Bulletin of the European Association for Theoretical Computer Science*, 52:245–262, 1994.

[48] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, 2001.

[49] S. Haar, L. Kaiser, F. Simonot-Lion, and J. Toussaint. Equivalence of timed state machines and safe TPN. In *Proceedings of the 6th International Workshop on Discrete Event Systems (WODES'02)*, pages 119–126. IEEE Computer Society, 2002.

[50] H.M. Hanisch. Analysis of place/transition nets with timed-arcs and its application to batch process control. In *Proceedings of the 14th International Conference on Application and Theory of Petri Nets (ICATPN'93)*, volume 691 of *LNCS*, pages 282–299, 1993.

[51] F. Heitmann, D. Moldt, K.H. Mortensen, and H. Rölke. Petri nets tools database quick overview. `http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html`, Accessed: 11.6.2008.

[52] F. Heitmann, D. Moldt, K.H. Mortensen, and H. Rölke. Applications of Petri nets. `http://www.informatik.uni-hamburg.de/TGI/PetriNets/applications`, Accessed: 15.6.2008.

[53] Thomas A. Henzinger and Joseph Sifakis. The discipline of embedded systems design. *Computer*, 40(10):32–40, 2007.

[54] P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995.

[55] N.D. Jones, L.H. Landweber, and Y.E. Lien. Complexity of some problems in Petri nets. *Theoretical Computer Science*, 4(3):277–299, 1977.

[56] Moez Krichen and Stavros Tripakis. Black-box conformance testing for real-time systems. In *Proceedings of the 11th International SPIN Workshop on Model Checking of Software (SPIN'04)*, volume 2989 of *LNCS*, pages 109–126. Springer-Verlag, 2004.

[57] F. Laroussinie and K.G. Larsen. CMC: A tool for compositional model-checking of real-time systems. In *Proceedings of the FIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XI) and Protocol Specification, Testing and Verification (PSTV XVIII)*, pages 439–456. Kluwer, B.V., 1998.

[58] K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a Nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.

[59] Kim G. Larsen and Wang Yi. Time-abstracted bisimulation: Implicit specifications and decidability. *Information and Computation*, 134(2):75–101, 1997.

[60] Kim Guldstrand Larsen, Gerd Behrmann, Ed Brinksma, Ansgar Fehnker, Thomas Hune, Paul Pettersson, and Judi Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *LNCS*, pages 493–505. Springer-Verlag, 2001.

[61] Kim Guldstrand Larsen, Marius Mikucionis, and Brian Nielsen. Online testing of real-time systems using Uppaal. In *Proceedings of the 4th International Workshop on Formal Approaches to Software Testing (FATES'04)*, volume 3395 of *LNCS*, pages 79–94. Springer-Verlag, 2005.

[62] D. Lime and O.H. Roux. State class timed automaton of a time Petri net. In *Proceedings of the 10th International Workshop on Petri Net and Performance Models (PNPM'03)*, pages 124–133, 2003.

[63] Didier Lime and Olivier (H.) Roux. Model checking of time Petri nets using the state class timed automaton. *Journal of Discrete Events Dynamic Systems — Theory and Applications (DEDS)*, 16(2):179–205, 2006.

[64] Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *Proceedings of 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'05)*, volume 900 of *LNCS*, pages 229–242. Springer-Verlag, 1995.

[65] E.W. Mayr. An algorithm for the general Petri net reachability problem (preliminary version). In *Proceedings of the 13th Ann. ACM Symposium on Theory of Computing*, pages 238–246. Assoc. for Computing Machinery, 1981.

[66] Philip Meir Merlin. *A Study of the Recoverability of Computing Systems*. PhD thesis, University of California, Irvine, CA, USA, 1974.

[67] P.M. Merlin and D.J. Faber. Recoverability of communication protocols: Implications of a theoretical study. *IEEE Transactions on Communications*, 24(9):1036–1043, 1976.

[68] Luis Montano, Francisco José García Peñalvo, and José Luis Villarroel. Using the time Petri net formalism for specification, validation, and code generation in robot-control applications. *International Journal of Robotic Research*, 19(1):59–76, 2000.

[69] M. Nielsen, V. Sassone, and J. Srba. Properties of distributed timed-arc Petri nets. In *Proceedings of the 21st International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'01)*, volume 2245 of *LNCS*, pages 280–291. Springer-Verlag, 2001.

[70] Fernando L. Pelayo, Fernando Cuartero, Valentín Valero, Hermenegilda Macia, and Maria L. Pelayo. Applying timed-arc Petri nets to improve the performance

of the MPEG-2 encoding algorithm. In *Proceedings of the 10th International Multimedia Modelling Conference (MMM'04)*, pages 49–56. IEEE Computer Society, 2004.

[71] Fernando L. Pelayo, Fernando Cuartero, Valentin Valero, Maria L. Pelayo, and Mercedes G. Merayo. How does the memory work? by timed-arc Petri nets. In *Proceedings of the 4th IEEE International Conference on Cognitive Informatics (ICCI'05)*, pages 128–135, 2005.

[72] Wojciech Penczek and Agata Pólrola. *Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach*. Springer-Verlag, 2006.

[73] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Darmstadt, 1962.

[74] Louchka Popova-Zeugmann. On time Petri nets. *Elektronische Informationsverarbeitung und Kybernetik*, 27(4):227–244, 1991.

[75] Louchka Popova-Zeugmann, Monika Heiner, and Ina Koch. Time Petri nets for modelling and analysis of biochemical networks. *Fundamenta Informatica*, 67(1-3):149–162, 2005.

[76] C. Ramchandani. *Performance Evaluation of Asynchronous Concurrent Systems by Timed Petri Nets*. PhD thesis, Massachusetts Institute of Technology, Cambridge, 1973.

[77] V. Valero Ruiz, D. de Frutos Escrig, and O. Marroquin Alonso. Decidability of properties of timed-arc Petri nets. In *Proceedings of the 21st International Conference on Application and Theory of Petri Nets (ICATPN'00)*, volume 1825 of *LNCS*, pages 187–206. Springer-Verlag, 2000.

[78] V. Valero Ruiz, F. Cuartero Gomez, and D. de Frutos Escrig. On non-decidability of reachability for timed-arc Petri nets. In *Proceedings of the 8th International Workshop on Petri Net and Performance Models (PNPM'99)*, pages 188–196, 1999.

[79] Valentín Valero Ruiz, Juan José Pardo, and Fernando Cuartero. Translating TPAL specifications into timed-arc Petri nets. In *Proceedings of the 23rd International Conference on Applications and Theory of Petri Nets (ICATPN'02)*, volume 2360 of *LNCS*, pages 414–433. Springer-Verlag, 2002.

[80] Valentín Valero Ruiz, Fernando L. Pelayo, Fernando Cuartero, and Diego Cazorla. Specification and analysis of the MPEG-2 video encoder with timed-arc Petri nets. *Electronic Notes Theoretial Computer Science*, 66(2), 2002.

[81] Ch. Seidner, G. Gardey, D. Lime, M. Magnin, and O. Roux. Romeo: A tool for time Petri net analysis. `http://romeo.rts-software.org/`, Accessed: 11.6.2008.

[82] J. Sifakis. Use of Petri nets for performance evaluation. In *Proceedings of the Third International Symposium IFIP W.G. 7.3., Measuring, Modelling and Evaluating Computer Systems (Bonn-Bad Godesberg, 1977)*, pages 75–93. Elsevier Science Publishers, Amsterdam, 1977.

[83] J. Sifakis and S. Yovine. Compositional specification of timed systems. In *Proceedings of the 13th Annual Symposim on Theoretical Aspects of Computer Science (STACS'96)*, volume 1046 of *LNCS*, pages 347–359. Springer-Verlag, 1996.

[84] J. Srba. Timed-arc Petri nets vs. networks of timed automata. In *Proceedings of the 26th International Conference on Application and Theory of Petri Nets (ICATPN 2005)*, volume 3536 of *LNCS*, pages 385–402. Springer-Verlag, 2005.

[85] UPPAAL-Cora. `http://www.cs.aau.dk/~behrmann/cora/`, Accessed: 18.6.2008.

[86] UPPAAL-Tiga. `http://www.cs.aau.dk/~adavid/tiga/`, Accessed: 18.6.2008.

[87] UPPAAL-Tron. `http://www.cs.aau.dk/~marius/tron/`, Accessed: 23.6.2008.

[88] J. Wang. *Timed Petri Nets, Theory and Application*. Kluwer Academic Publishers, 1998.