# Capacity Timed Automata

Jonas Groth, Anders Hesselager, Søren Larsen, and Torkil Olsen

Aalborg University, Department of Computer Science
Selma Lagerlöfs Vej 300
9220 Aalborg East, Denmark
{jgroth07,ahesse07,slars07,tolse09}@student.aau.dk
Supervisor: René Rydhof Hansen

**Abstract.** We define and study the class of *Capacity Timed Automata* (CTA) which are *Priced Timed Automata* (PTA) extended with negative cost rates on edges and locations. The motivational example is the satellite AAUSAT-II designed and constructed by Aalborg University. During its orbit it will periodically be in the shadow of the Earth and not be able to charge the battery via its solar panels. In this situation it must operate with *energy optimal task scheduling* to ensure it does not risk running out of power. We use CTA to model the system and use the introduced capacity variables to represent the charge of the satellite's battery. A proof sketch showing that the reachability problem for CTA is undecidable is also provided. We define an algorithm translating CTA to *Linear Hybrid Automata* (LHA), which allows the use of model checking tools like HyTech and Kronos for verification. The expressiveness of CTA is discussed, leading to a proof that the class CTA is a strict subset of the class LHA. Furthermore we translate the LHA to a *StopWatch Automata* (SWA) with the algorithm presented in [CL00]. With the system represented as an SWA we verify certain properties of the system operating the satellite by using the model checking tool UPPAAL.

## 1 Introduction

In the shadow of the Earth a satellite in orbit is running on its current power reserve. It currently has no way of recharging its battery. We pose the question of how to model such an energy critical real-time system and be able to verify that the satellite will operate as expected and not run out of power.

Characteristics of the class of problems to be modelled are:

- There exists one or more resources that can increase or decrease as the system runs and time elapses
- The resources have a lower and upper bound e.g. a battery with zero or full charge
- The resources can increase or decrease in system states or as a result of executing operations
- The system is expected to run indefinitely and with the resources never exceeding the bounds

*Related work.* Several different formalisms for defining real-time systems exist today. Many of which are widely used in academic settings.

The model of *Timed Automata* (TA) was introduced in [AD94] and has become a standard formalism for modelling real-time systems. With the model checking tool, UPPAAL [UPP10], it is possible to verify real-time systems modelled as networks of timed automata.

The *Priced Timed Automata* (PTA) formalism, described in [BLR04], is an extension of timed automata that adds the ability to specify the rate at which cost accumulates during execution. The cost can represent e.g. power consumption in a task scheduling system. The PTA modelling formalism can be used to describe e.g. scheduling problems and with the tool, UPPAAL CORA [COR10], it is possible to perform cost-optimal reachability analysis. UPPAAL CORA is a branch of UPPAAL optimized for Cost Optimal Reachability Analysis. Practical examples of application of the formalism are shown in [BLR04].

*Weighted Timed Automata* (WTA) is another name for PTA. Some research describing negative costs in WTA has been made. In [PB08] several important properties of these automata are discussed. The main contribution of [PB08] is decidability results for several properties such as the existential problem is shown decidable for finite and one-clock weighted automata.

*StopWatch Automata* (SWA) is the class of timed automata extended with stopwatches; that is the derivative of a variable in a location can be either 0 or 1 [CL00]. The classes SWA and *Linear Hybrid Automata* (LHA) are timed language equivalent as proven in [CL00] and a method for translating SWA into equally expressive LHA. SWA can be modelled and a number of approximate verifications can be performed using the SWA-extension of UPPAAL.

*Hybrid Automata* are a strong extension of TA used to model systems with both discrete and continuous components, i.e. a hybrid system [CL00], [Hen96]. The restricted class of hybrid automata, LHA [ACH$^+$95], are hybrid automata where invariants, guards and activities are linear expressions over a set of variables $V$. A *linear expression* $\phi(\bar{v})$ over $V$ is of the form $\Sigma a_i v_i$ with $a_i \in \mathbb{Z}$, $v_i \in V$ [CL00].

None of the existing formalisms provide an intuitive way of modelling the proposed class of problems. We therefore propose a new formalism to bridge the gap between the simplicity of TA and the expressiveness of more powerful formalisms, such as the above mentioned. TA, PTA and WTA are not capable of expressing the properties needed to define the defined class of problems. LHA and SWA can model the problem, but are too general in their expressiveness.

*Our contribution.* We define the formalism *Capacity Timed Automata* (CTA) as an extension of PTA. The extension is a PTA with negative cost rates on edges and locations as well as the introduction of multiple capacity variables with a lower and upper bound for each variable's value. The capacity variables can be used for representing power level, water level etc. We will provide a proof sketch showing that the reachability problem for CTA is undecidable.

We will translate CTA to LHA algorithmically. The algorithm is straightforward and serves well to show both the similarities and differences between CTA and LHA. We will give a proof showing that CTA is a strict subset of LHA. The translation is useful since it is possible to translate the resulting LHA into an SWA without loss of expressive power and furthermore use UPPAAL for approximate verification.

*Motivating example.* The AAUSAT-II satellite is orbiting Earth. The satellite can perform a set of operations which require some amount of power to execute. As the satellite does not always have the opportunity to recharge its batteries, e.g. due to being in the shadow of Earth, it is interesting to verify if certain properties hold w.r.t. power consumption over time. Violating power contrains, especially in space, can have devastating consequences.

*Outline of the paper.* In Section 2 we define CTA and prove the undecidability of the reachability problem for CTA. Section 3 states that CTA can be translated into a corresponding LHA. In Section 4 we show the expressiveness of CTA. Section 5 is focused on the practical interest of the translation and we give an example of a CTA that can be verified using our translation to LHA, the translation in [CL00] from LHA to SWA and the UPPAAL version extended with SWA. Finally we conclude in Section 6.

## 2 Capacity Timed Automata

The CTA formalism is an extension of PTA, described in [BLR04]. The extension allows negative costs on edges and locations. We define $\beta(X)$ as the set of simple expressions over the set of clocks $X$, used to define guards and invariants.
A simple expression is of the form $x \bowtie k$ where $x \in X$, $\bowtie \in \{<, \leq, =, \geq, >\}$ and $k$ is a non-negative integer. This definition is the same as the definition of clock constraints for TA given in [AILS08]. We define $\mathbf{\Delta}(C)$ as the set of simple expressions over the set of capacity variables $C$, used to define guards and invariants. A simple expression is of the form $c \bowtie r$ where $c \in C$, $\bowtie \in \{<, \leq, =, \geq, >\}$ and $r$ is an integer.

**Definition 1 (Capacity Timed Automaton).** *A Capacity Timed Automaton over a finite set of clocks $X$ and a finite set of actions Act is a 8-tuple:*

$$A = (L, l_0, C, C^0, C^\sigma, E, I, P)$$

*where*

- *$L$ is the set of locations*
- *$l_0 \in L$ is the initial location*
- *$C$ is the set of real valued capacity variables*
- *$C^0$ is the set of initial values for the capacity variables*

- $C^\sigma : C \longrightarrow \mathbb{Z} \times \mathbb{Z}$ *assigns bounded intervals for the capacity variables s.t.* $c_j^\sigma = [k_1, k_2]$*, where* $k_1, k_2 \in \mathbb{Z}$*, is the interval in which the capacity variable* $c_j \in C$ *is bounded*
- $E \subseteq L \times \beta(X) \times \mathbf{\Delta}(C) \times Act \times 2^X \times L$ *is the set of edges where* $2^X$ *is the set of clocks that will be reset to* 0*.*
- $I : L \longrightarrow \beta(X) \times \mathbf{\Delta}(C)$ *assigns clock and capacity invariants to locations*
- $P : L \cup E \longrightarrow \mathbb{Z}^C$ *assigns cost rates to edges and locations s.t.* $P(e)(c_j)$ *is the cost of an edge e for capacity variable* $c_j$ *and* $P(l)(c_j)$ *is the cost rate of* $c_j$ *in location l*

In CTA it is worth to notice that the bounds given by $C^\sigma$ are strict bounds in the sense that a capacity variable may never exceed the bounds. An edge in CTA between two locations contain guards for both clocks and capacity variables, $\beta(X)$ and $\mathbf{\Delta}(C)$ respectively. We may also specify a set of clocks, $2^X$, that will be reset to 0. An example of CTA can be seen in Example 1.

To define the semantics of the CTA formalism we use a Capacity Transition System (CTS) shown in Definition 2. With CTS we add to the notion of the Labelled Transition System [AILS08] (LTS) formalism as done in [BLR04] for *Priced Transition System*. The transition relation is a partial function from transitions to real values, allowing transitions with both positive and negative real-valued costs. When the transition relation is applied on two locations $l$ and $l'$ with action $a$ and a cost $p$ we write $l \xrightarrow{a}_p l'$. In light of recent, we write $\xrightarrow{a}\!\!\!\!\!\rightarrow_p$ for the impossible transition relation. That is, either a non-existing label is used, or the cost $p$ results in one or more capacity variables violating the invariants in the target location $l'$ or exceeding their bound $c^\sigma$.

**Definition 2 (Capacity Transition System).**
*A Capacity Transition System is a tuple* $\mathcal{T} = (S, s_0, \Sigma, \longrightarrow)$*, where S is a (possibly infinite) set of states,* $s_0 \in S$ *is the initial state,* $\Sigma$ *is a set of labels, and* $\longrightarrow : (S \times \Sigma \times S) \hookrightarrow \mathbb{R}$ *is a partial function from transitions to real valued numbers.*

With the definition of CTS we describe the semantics of a CTA in Definition 3. We denote a state in CTS by $(l, v, q)$ where $l$ is a location in CTA, $v$ is a valuation of the clocks $X$ and $q$ is a valuation of the capacity variables $C$.

**Definition 3 (Semantics of a Capacity Timed Automaton).**
*The semantics of a Capacity Timed Automaton* $A = (L, l_0, C, C^0, C^\sigma, E, I, P)$ *over clocks X and actions Act is given by a Capacity Transition System* $\mathcal{T} = (S, s_0, \Sigma, \longrightarrow)$*, where* $S = \{(l, v, q) \in L \times \mathbb{R}^X_{\geq 0} \times \mathbb{R}^C | v, q \models I(l)$ *and* $q(c_j) \models C^\sigma(c_j)$ *for all* $c_j \in C\}$*, is the set of states that satisfy the invariants,* $s_0 = (l_0, v_0, q_0)$ *is the initial state for* $v_0$ *evaluating to zero for all clocks in X and* $q_0(c_j) = c_j^0$ *for all* $c_j \in C$ *and* $c_j^0 \in C^0$*,* $\Sigma = Act \cup \mathbb{R}_{\geq 0}$ *is the set of labels, and* $\longrightarrow$ *consists of discrete and delay transitions as defined below.*

We now describe the two types of transition in the CTS w.r.t CTA as mentioned in Definition 3. In Definition 4, discrete transitions are described. When following an enabled discrete transition, i.e. following an edge where the invariant of the source location holds, the guards on the edge evaluate to true and the invariant in the target location evaluates to true, the target location is enabled, the clocks in the reset set are set to zero and the capacity variables in the set of prices are updated accordingly. Let $gx$ and $gc$ denote the guards for clocks and capacity variables respectively.

**Definition 4 (Discrete transitions).**
*A transition $(l, v, q) \xrightarrow{a}_p (l', v', q')$ is a discrete transition iff there is an edge $(l, gx, gc, a, r, l')$ from $l$ to $l'$, s.t. the guards, $gx$ and $gc$, evaluate to true in the source state $(l, v, q)$, $v'$ is derived from $v$ by resetting all clocks in the reset set $r$, and the capacity variables are updated by $q' = q + P(e)$.*

A delay transition is defined as time passing while remaining in the same location. We denote this delay by $d \in \mathbb{R}_{\geq 0}$. The definition of a delay transition is given in Definition 5. A delay transition is only enabled when the invariant of the location holds both before and after the delay. The capacity variables are updated according to the product of cost rates in the location and the duration of the delay.

**Definition 5 (Delay transitions).** *A transition $(l, v, q) \xrightarrow{d}_p (l', v', q')$ is a delay transition iff $q' = q + d \cdot P(l)$, $v' = v + d$, and the invariant for clocks and capacity variables of $l$ is satisfied by the source, target and all intermediary states i.e. for all non-negative delays $d'$ less than or equal to $d$ we have $v + d' \models I(l)$ and $q + d' * P(l) \models I(l)$.*

*Example 1.* To exemplify the CTA formalism, consider the CTA $A$ in Figure 1. The example models a simple charge/discharge scenario with one clock $x$, one capacity variable $c_0$ with initial value 20 and three states, where $l_0$ is a discharge state and $l_1$ and $l_2$ are charge states. The capacity variable is universally bound by the interval $c_0^\sigma$=[3,23].

A trace of a CTA is a sequence of alternating discrete and delay transitions in the corresponding CTS.

Two different traces $\alpha_0$ and $\alpha_1$ of $A$ are shown below. Note how it is not possible to perform a delay transition in $l_0$ with delay $d = 9$ since the cost rate is $-2$, resulting in $c_0 = 2$. This violates the bound interval $c_0^\sigma$ specified.

$$\alpha_0 = (l_0, x = 0, c_0 = 20) \xrightarrow{9}_{-2}$$

$$\begin{aligned} \alpha_1 = (l_0, x = 0, c_0 = 20) &\xrightarrow{4}_{-2} (l_0, x = 4, c_0 = 12) \rightarrow_0 \quad (l_1, x = 0, c_0 = 12) \\ &\xrightarrow{5}_{1} \quad (l_1, x = 5, c_0 = 17) \rightarrow_{-1} (l_2, x = 0, c_0 = 16) \\ &\xrightarrow{2}_{3} \quad (l_1, x = 2, c_0 = 22) \rightarrow_0 \quad \cdots \end{aligned}$$

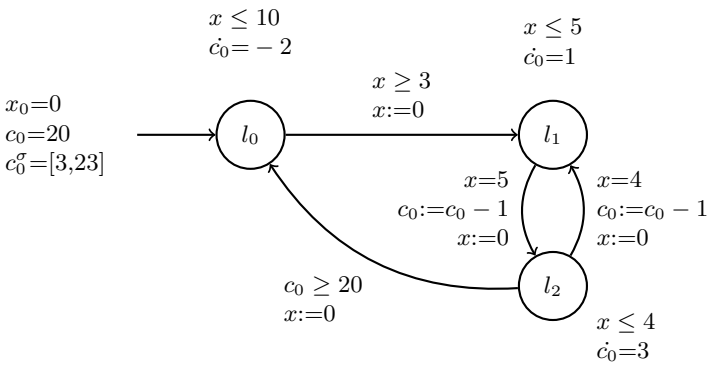



**Fig. 1.** A capacity timed automaton $A$ showing a simple charge/discharge example.

### Undecidability of the reachability problem for CTA

By the following proof sketch we conclude that the reachability problem for CTA is undecidable.

**Theorem 1.** *The reachability problem for CTA is undecidable*

We will prove this theorem by using *Minsky Machines* for which the halting problem is undecidable, as proven in [Hir94]. A Minsky Machine is a two-counter machine which has two different labelled commands; $L : c := c + 1$; *goto* $L'$ and $L : if\ c = 0\ then\ goto\ L'\ else\ c := c - 1$; *goto* $L''$ and a $HALT$ command. For the formal definition see [Hir94]. We can reduce the halting problem for Minsky Machines to the reachability problem for CTA and thus prove that it is undecidable.

*Proof sketch.* Let $A$ be a Minsky Machine. We start by creating a CTA, $A_{CTA}$, with the capacity variables $c_1$ and $c_2$ which correspond to the counter variables of $A$. For each labelled instruction $L$ in $A$ we create a location in $A_{CTA}$ with the label $L$. We then create the edges based on the goto of the instructions in $A$. If an instruction is $L : c := c+1$; *goto* $L'$ we create an edge from the location $L$ in $A_{CTA}$ to the location $L'$ with a cost $c' := c + 1$. For the instruction $L : if\ c = 0\ then\ goto\ L'\ else\ c := c - 1$; *goto* $L''$ we create two edges from the location $L$ in $A_{CTA}$ to the location $L'$ with the guard $c \leq 0$ and to the location $L''$ with the guard $c > 0$ and a cost $c' := c - 1$ respectively. If the instruction is $HALT$ we create no links as this location is an end location. We then create a start location, $S$, with an edge to the location of the initial instruction of $A$.

By construction, the Minsky Machine halts iff there is a reachable location from $S$ in $A_{CTA}$ that corresponds to a $HALT$ instruction in $A$. As the halting problem for Minsky Machines is undecidable we can conclude that the reachability problem for CTA is also undecidable.

## 3 Translating Capacity Timed Automata to Linear Hybrid Automata

### 3.1 Preliminaries

In the previous section we proved that the reachability problem for CTA is undecidable.

By translating a CTA to an LHA we can use tools such as HyTech [HHWT97] or Kronos [BDM+98] to make an approximate model verification. Furthermore, using the translation from LHA to SWA given in [CL00], we can verify certain properties using UPPAAL which has support for SWA in the development version.

Before giving the translation we describe the class LHA. We use the definition for syntax and semantics given in [CL00] as these are the basis for the translation from LHA to SWA, described in the same paper.

#### Linear Hybrid Automata

Linear hybrid automata is a restricted class of hybrid automata where invariants, guards and activities are linear expressions as described in the introduction. Linear guards and invariants can be described as *linear constraints* which are propositional formula using the three boolean connectives $\vee, \wedge, \neg$ over linear boolean expressions of the form $\phi(\bar{x}) \bowtie c$, where $\bowtie \in \{<, =, >\}$ and $c \in \mathbb{N}$. $\mathcal{LC}(V)$ is the set of linear constraints. *Linear assignments* over $V$ are of the form $\bar{v} := A.\bar{v} + \bar{b}$ where A is an $n \times n$ matrix with coefficients in $\mathbb{Z}$ and $\bar{b}$ is a vector of $\mathbb{Z}^n$. The linear assignments form the set $\mathcal{LA}(V)$. For more in-depth explanations of linear assignments and constraints we recommend reading [ACH+95] Section 3 and [CL00] Section 2.1.

**Definition 6 ( Syntax of a Linear Hybrid Automaton [CL00]).** *A* linear hybrid automaton $\mathcal{H}$ *is a 7-tuple* $(N, l_0, V, A, E, Act, Inv)$ *where:*

- $N$ *is a finite set of locations,*
- $l_0 \in N$ *is the initial location,* $v_0$ *is the initial valuation,*
- $V$ *is a finite set of real-valued variables,*
- $A$ *is a finite set of actions,*
- $E \subseteq N \times \mathcal{LC}(V) \times A \times \mathcal{LA}(V) \times N$ *is a finite set of edges;* $e = \langle l, \gamma, a, \alpha, l' \rangle \in E$ *represents an edge from the location* $l$ *to the location* $l'$ *with the linear guard* $\gamma$*, the label* $a$ *and the linear assignment* $\alpha$*.*
- $Act \in ((\mathbb{Z} \times \mathbb{Z})^V)^N$ *where* $Act(l)(x) = [u_1, u_2]$ *means that the first derivative of* $x$ *in location* $l$ *lies in the compact bounded interval* $[u_1, u_2]$ *of* $\mathbb{Z}$*.*
- $Inv \in \mathcal{LC}(V)^N$ *assigns a linear invariant to any location.*

**Definition 7 (Semantics of a Hybrid Automaton [CL00]).** *The semantics of a hybrid automaton* $\mathcal{H}(N, l_0, V, A, E, Act, Inv)$ *is a timed transition system* $S_{\mathcal{H}} = (Q, q_0, \Sigma, \longrightarrow)$ *where* $Q = N \times \mathbb{R}^V$, $q_0 = (l_0, v_0)$ *is the initial state* ($v_0(x) = 0$, $\forall x \in V$) *and* $\longrightarrow$ *is defined by:*

$$\langle l, v\rangle \xrightarrow{a} \langle l', v'\rangle \text{ iff } \exists (l, \gamma, a, \alpha, l') \in E \text{ s.t. } \begin{cases} \gamma(v) = tt, v' = \alpha(v) \text{ and} \\ Inv(l')(v') = tt \end{cases}$$

$$\langle l, v\rangle \xrightarrow{d} \langle l', v'\rangle \text{ iff } \begin{cases} l = l' \text{ and } \exists d \in Act(l) \text{ s.t. } v' = v + d.t \text{ and} \\ \forall 0 \leq d' \leq d, v + d'.t \in Inv(l) \end{cases}$$

### 3.2 Translation

We now present the algorithm for translating a CTA into an equivalent LHA.

*Translating locations.* Create the locations in LHA corresponding to the locations in CTA.

- For each location $l_n \in L_{CTA}$ create the location $l_n \in N_{LHA}$.

*Translating variables.* In CTA there are capacity variables and clock variables. The capacity variables only increase or decrease as an effect of cost rates on edges and in locations. The clock variables increase with time and can be reset. On the contrary there is only one type of variables in LHA; the variables $v \in V$. They are used for both clock and capacity variables. Clocks are not implicit in LHA so there must be an activity in every state specifying that the clock variables should increase over time. This increase is implicit in CTA. The first of the two steps of creating the variables in LHA are creating the variables corresponding to the capacity variables in CTA.

- For each capacity variable $c_n \in C_{CTA}$ create the variable $c_n \in V_{LHA}$.

The second step creates variables corresponding to the clocks and adds activities in each location that increase the clock value in delays, i.e. $\dot{x}_n = 1$.

- For each clock $x_n \in X_{CTA}$ create the variable $x_n \in V_{LHA}$ and for each location $l_k \in N_{LHA}$ create the activity $Act(l_k)(x_n) = 1$.

*Translating actions.* Create actions in LHA according to the actions in CTA.

- For each action $a \in Act_{CTA}$ create the action $a \in A_{LHA}$.

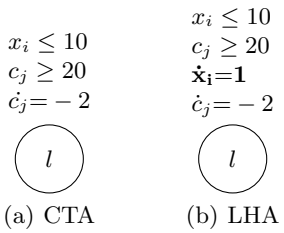
*Translating invariants.* Translating the invariants is a straightforward task. They are translated directly to the corresponding invariants in LHA.

- For each invariant $I_{CTA} : L_{CTA} \longrightarrow \beta(X_{CTA}) \times \Delta(C_{CTA})$ create the invariant $Inv_{LHA} \in \mathcal{LC}(V_{LHA})^{N_{LHA}}$.

*Translating activities.* The activities in CTA defining what the variables evaluate to in the different states are translated into corresponding activities in LHA.

– For each cost rate in a location, $P_{CTA}(l_n)(c_m) = u_i$, where $u_i \in \mathbb{Z}$, create the activity $Act_{LHA}(l_n)(c_m) = u_i$.

Figure 2 shows the translation of a location, with invariants and cost rates.



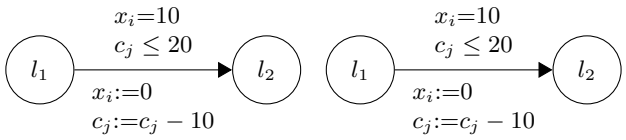

(a) CTA (b) LHA

**Fig. 2.** Translation of a location.

*Translating edges.* The string representation of edges in CTA and LHA are similar to some extent. The source and target locations can be translated directly as well as the action. The guards on edges in LHA are a union of clock and capacity variable guards in CTA. The assignments in LHA are a union of the reset set and the set of costs on the edge.

– For every edge $e = \langle l, gx, gc, a, r, l' \rangle \in E_{CTA}$, create the edge $e = \langle l, \gamma, a, \alpha, l' \rangle \in E_{LHA}$ s.t. $l = l, \gamma = gx \cup gc, a = a, \alpha = r \cup q' = q + P(e)$, where *P(e)* is the set of cost rates on the edge and $l' = l'$.

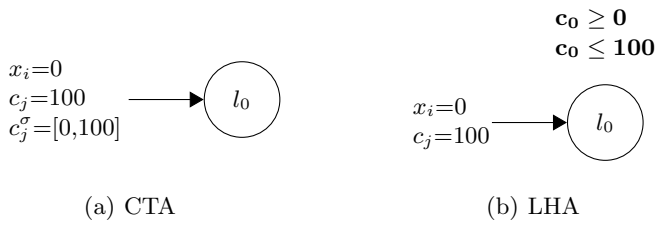
Figure 3 shows the translation of a discrete transition, between two locations, with guards, assignments and costs on the transition.




(a) CTA (b) LHA

**Fig. 3.** Translation of locations with discrete transition.

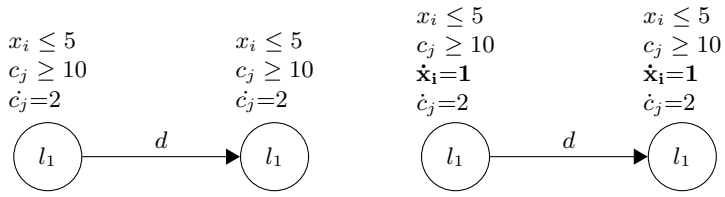
*Translating initial valuations.* Figure 4 shows the translation of the initial location with the initial valuations.




(a) CTA (b) LHA

**Fig. 4.** Translation of the initial location and bound for capacity variables.

Figure 5 shows the translation of a delay transition.




(a) CTA (b) LHA

**Fig. 5.** Translation of a location with delay transition.

*Translating the bounded intervals that capacity variables are bounded by.* For representing the bounded interval for the capacity variables, we create an invariant for each capacity variable $c_j \in C$ in each location $l_n \in L_{CTA}$ shown in 4.

– For each $c_j^\sigma = [k_1, k_2] \in C^\sigma$ create the invariant $k_1 \leq c_j$ and $c_j \leq k_2$ in each location, $l_n \in L_{LHA}$.

## 4 Capacity Timed Automata Expressiveness

We show that CTA are not as expressive as LHA by giving an example of an LHA which cannot be translated into a CTA.

**Theorem 2.** *The class CTA is not as expressive as the class LHA in terms of problems that can be modelled.*

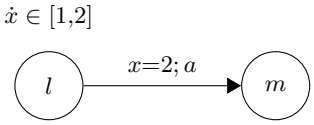



**Fig. 6.** Example of an LHA that cannot be expressed as a CTA.

Our proof is based on Figure 6 which is given in [CL00], Figure 2. The example describes a simple LHA $H$ with only two states. The first state, $l$, has a derivative in the interval $[1, 2]$. The proof is done by a counter example.

*Proof.* The main problem with translating this LHA to CTA is the derivative in location $l$. In [Hen96], Definition 1.3 bullet 4, and in [CL00], Definition 1 bullet 6, it is defined that the derivative of a variable in a location is a real number within a bounded integer interval. The CTA formalism only supports integer derivatives and thus the derivative cannot be translated in cases where the bounded interval is not guarded by $u_1 = u_2$ where $Act(l)(x) = [u_1, u_2]$ for the variable $x$ at location $l$.

This makes it impossible to model all LHA as CTA.

## 5 AAUSAT-II

AAUSAT-II is a small cubic satellite developed by students at Aalborg University, with the technical goal of testing a ACDS[1] system and a gamma ray detector [AAU02]

An *Electronic Power System* (EPS) module governs power supply to the other subsystems. A 13,3wh battery pack serves as the continuous power source [THH+05]. Whenever the satellites orbital path takes it out of the shadow of the Earth, the EPS can charge the battery from an array of solar panels.

Other subsystems include an *On-Board Computer* (OBC) for processing and distributing commands to other subsystems and a *Communication System* (COM) that maintains a link to a ground station for data and operator command communication.

### The COM subsystem

AAUSAT-II and all its subsystems implies a complex and comprehensive model, which is not within the scope of this paper. For the purpose of a practical example the COM subsystem is used. COM modelled as a CTA is depicted in Figure 7. Please refer to Appendix A for a complete CTA of COM with all guards and invariants.

[1] The Attitude Determination and Control System (ACDS) provides initial stabilisation of AAUSAT-II and attitude tracking for scientific experiments

AAUSAT-II has four different power modes: Initialization, Safe, Recovery and Nominal mode. Each of these modes have different implications for COM. In Initialization mode (**Init.**) and Safe mode (**Safe**) the COM module is powered off. The battery may be charged (**charge**) when in Safe mode.

When the battery is sufficiently charged, the power mode is changed to Recovery mode (**Rec.**). In this mode, COM is powered and may listen for commands (**listening**) or transmit data (**TX**) back to Earth. The OBC is also booted in this mode.

When OBC is successfully booted, the mode is changed to Nominal (**bootOk**). COM can now listen for commands (**listening**), transmit data (**TX**) and receive commands (**RX**). At any time in both Recovery and Nominal mode, EPS can change the mode to Safe (**batteryLow**) if the battery is too low.

One capacity variable $c_0$ is used to represent the battery, and two clocks $x_0$ and $x_1$ are used to control transitions between states and the actions COM can perform. The values for $c_0$, $c_0^\sigma$, invariants and guards are derived from [AGK+05] and [Gro].

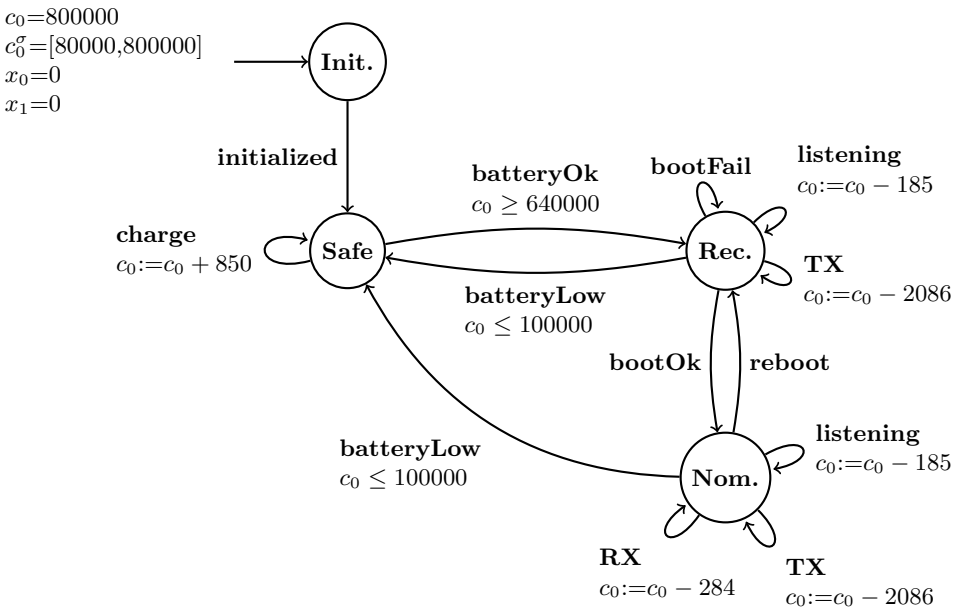



**Fig. 7.** The AAUSAT-II COM subsystem as a CTA.

Using the algorithm given in Section 3 we can translate the CTA in Figure 7 to an LHA and then to a SWA. The COM subsystem modelled as a LHA and

SWA, can be seen in Appendix B and Appendix C, respectively. Note that in the translation from LHA to SWA, the capacity variable is represented as two clocks $v0p$ and $v0m$. It follows that $v0p$ maintains the accumulated positive cost and $v0m$ the negative. Intuitively, we have that $c_0 = v0p - v0m$.

### Verification results

Given the COM subsystem encoded as an SWA, we can utilize UPPAAL to verify whether the model satisfies certain properties or requirements.

In UPPAAL, a property must be specified as a query. UPPAAL uses a simplified version of *Timed Computation Tree Logic* (TCTL) for the query language [BDL05]. Queries consist of *path formulae* and *state formulae*. With state formulae we may specify individual states, whereas path formulae allow us to quantify over traces in the model. Combining state and path formulae, we can verify queries such as: Does a trace exist where a certain state holds at some point, or, for all traces in the model is a given state always satisfied?

Below, we offer three different verification results for the COM subsystem.

**Requirement:** It is critical that the COM subsystem, and the satellite in general, respects the bounds defined on the power supply. That is, COM must satisfy the bound $c_0^\sigma$ with respect to $c_0$.
**Query:** For all traces in COM, is the bound $c_0^\sigma$ respected when the locations **Safe**, **Rec.** and **Nom.** are satisfied.
**Result:** This property holds.

**Requirement:** The satellite relies on the COM system to recieve and transmit operational commands and data, and as such, it must be satisfied that COM can listen for communication from the Earth station.
**Query:** Does a trace exist s.t. the location **listening** is satisfied.
**Result:** This property maybe holds.

**Requirement:** While the battery is being charged, all other subsystems are either shut down or in standby. Spending unnecessary time charging a fully charged battery, reduces execution time of the other subsystems. Thus, the battery should only be charged whenever $c_0$ is below the upper bound of $c_0^\sigma$.
**Query:** For all traces with location **charge** satisfied, is $c_0$ below the upper bound of $c_0^\sigma$.
**Result:** This property holds.

## 6 Conclusion

In this paper we defined capacity timed automata for modelling problems which require both positive and negative cost rates in locations and on edges. We use the notion of capacity variables to model varying values, such as the charge of a battery, general capacity scenarios and the AAUSAT-II example.

We presented a proof sketch that shows the reachability problem for CTA to be undecidable. The proof sketch is based on a reduction of the halting problem for Minsky Machines to the reachability problem for CTA.

An algorithm was presented that translates any CTA into an equivalent LHA. Following the translation it is proven that CTA is a strict subset of LHA by means of a counterexample.

We applied the translation algorithm to the COM subsystem of the AAUSAT-II satellite represented as a CTA, and showed that it could be translated to an equivalent LHA. By translating the LHA into an SWA, the model was encoded in UPPAAL and three sample verifications of the COM subsystem were provided. One of which was to verify that the COM subsystem never violated the bounds of the capacity variable, which proved to hold. It can be concluded that it is possible to do a two-step translation of CTA into SWA and to some extent verify properties of the model.

We have developed a formalism that is expressive enough to model the capacity related problems mentioned in the introduction while eliminating some of the complexity compared to HA and SWA.

### Future Work

For future work in the area of CTA a proof of the correctness of the translation is required. Furthermore it may be interesting to investigate parallel composition of CTA models. Concurrent execution and parallel composition is an integral part of not only timed automata but also SWA and LHA, and is needed to model more complex capacity problems.

**Acknowledgements.** We would like to thank Kim G. Larsen for valuable input and feedback during the process of writing this paper.

## References


[AAU02] AAU. *AAUSAT II*, 2002. As Seen on 26. May, 2010.

[ACH+95] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:pp. 3–34, 1995.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, Vol. 126(Issue 2):pp. 183–235, 1994.

[AGK+05] Bo Ørsted Andersen, Claus Grøn, Rasmus Hviid Knudsen, Claus Nielsen, Kresten Kjær Sørensen, and Dan Taagaard. The aausat-ii communication system. Technical report, 2005.

[AILS08] Luca Aceto, Anna Inglfsdttir, Kim Guldstrand Larsen, and Jiří Srba. *Reactive Systems - Modelling, Specification and Verification*. Number ISBN: 978-0-521-87546-2 in 1st edition. Cambridge University Press, 2008.

[BDL05] Gerd Behrmann, Alexandre David, and Kim G. Larsen. *A Tutorial on UPPAAL*, 2005. As Seen on 27. May, 2010.

[BDM+98] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A model-checking tool for real-time systems. *Lecture Notes in Computer Science*, 1486:pp. 298–302, 1998.

[BLR04] Gerd Behrmann, Kim Guldstrand Larsen, and Jacob Illum Rasmussen. Priced timed automata: Algorithms and applications. *LNCS*, Vol. 3657:pp. 162–182, 2004.

[CL00] Franck Cassez and Kim Larsen. The impressive power of stopwatches. *LNCS*, Vol. 1877:pp. 138–152, 2000.

[COR10] UPPAAL CORA. Uppaal cora. http://www.cs.aau.dk/ behrmann/cora/, 2010. As seen 18. May, 2010.

[Gro] AAUSAT-II Group. *Power Budget*. http://aausatii.space.aau.dk/homepage /en/dok/Power_budget.pdf. As seen 24. May, 2010.

[Hen96] Thomas A. Henzinger. The theory of hybrid automata. *IEEE Computer Society Press*, pages pp. 278–292, 1996.

[HHWT97] Thomas A. Henzinger, Pei-Hsin Ho, , and Howard Wong-Toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:pp. 110–122, 1997.

[Hir94] Yoram Hirschfeld. Petri nets and the equivalence problem. *Lecture Notes in Computer Science*, 832:pp. 165–174, 1994.

[PB08] Kim G. Larsen Nicolas Markey Jiří Srba Patricia Bouyer, Uli Fahrenberg. Infinite runs in weighted timed automata with energy constraints. *Lecture Notes in Computer Science*, 5215:pp. 33–47, 2008.

[THH+05] Michael Torp, Magnus Møberg Hansen, Jon Hagedorn, Elmar Reinart Fjallheim, and Malte Rasmus Østergaard Löfstedt. Preliminary design document: Power subsystem design. Technical report, 2005.

[UPP10] UPPAAL. Uppaal. http://www.uppaal.com/, 2010. As seen 18. May, 2010.

## A

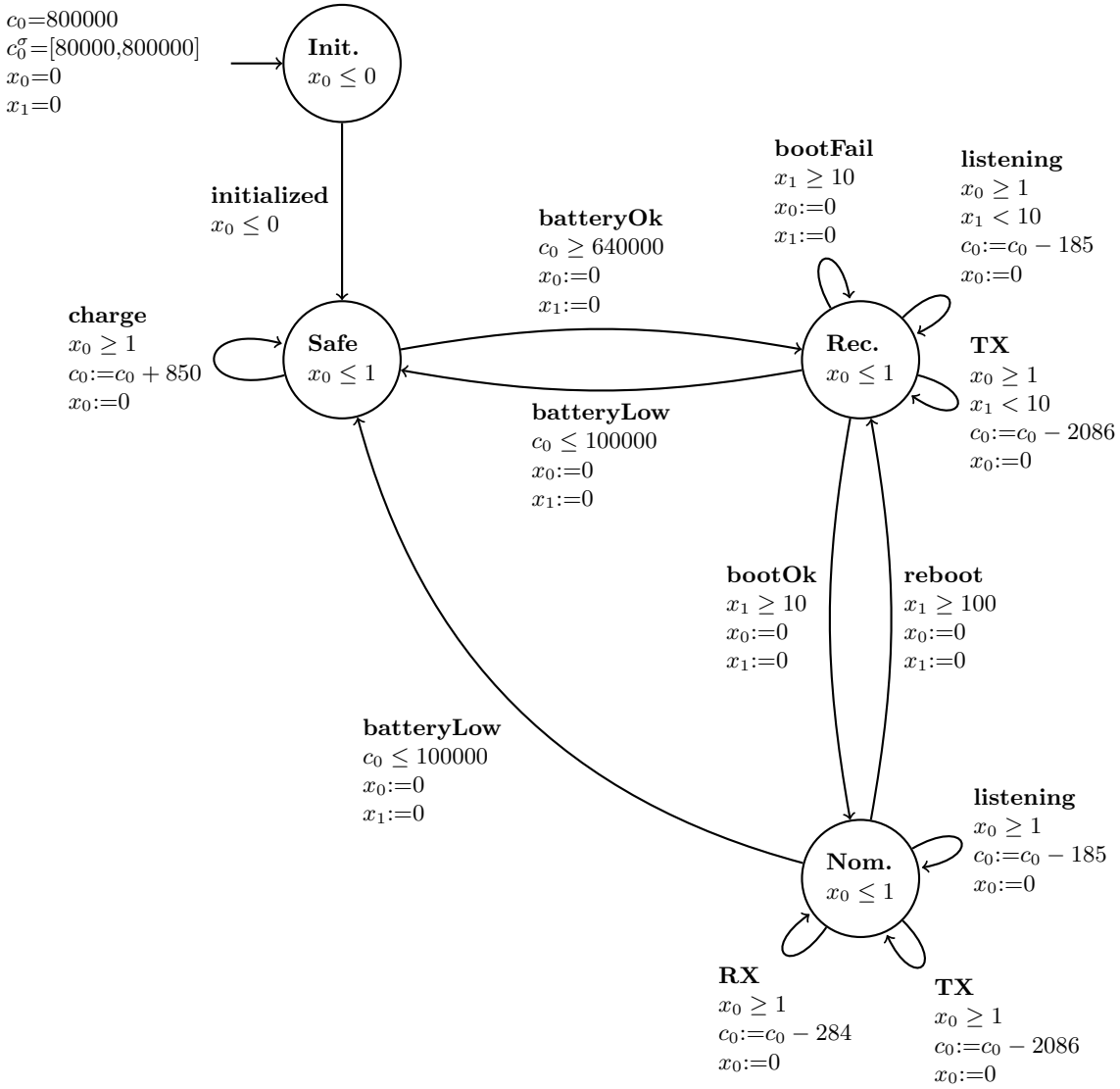



**Fig. 8.** The AAUSAT-II COM subsystem as a CTA

## B

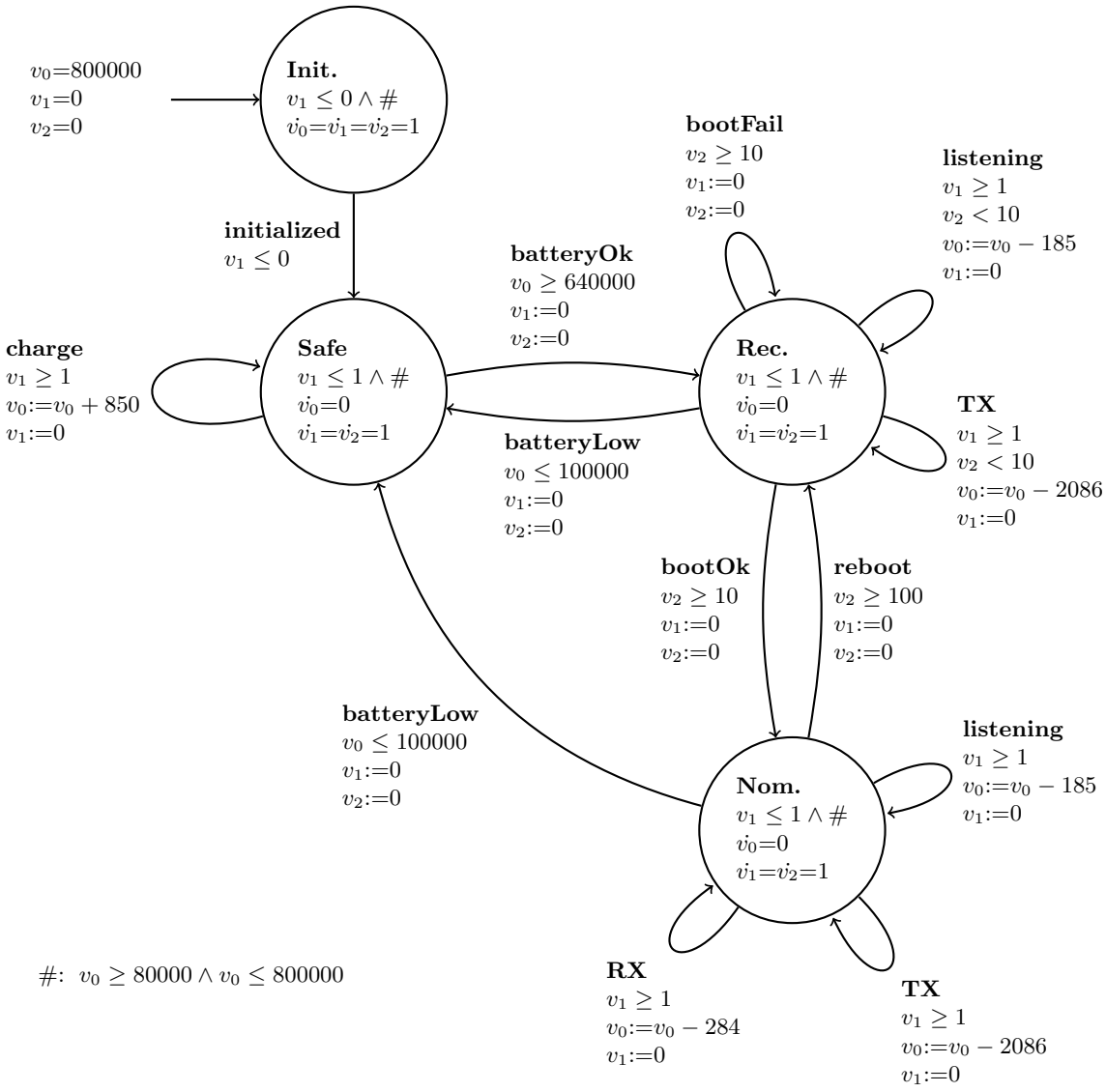



**Fig. 9.** The AAUSAT-II COM subsystem as an LHA. Note that the invariants denoted by # applies to all locations.

**C**

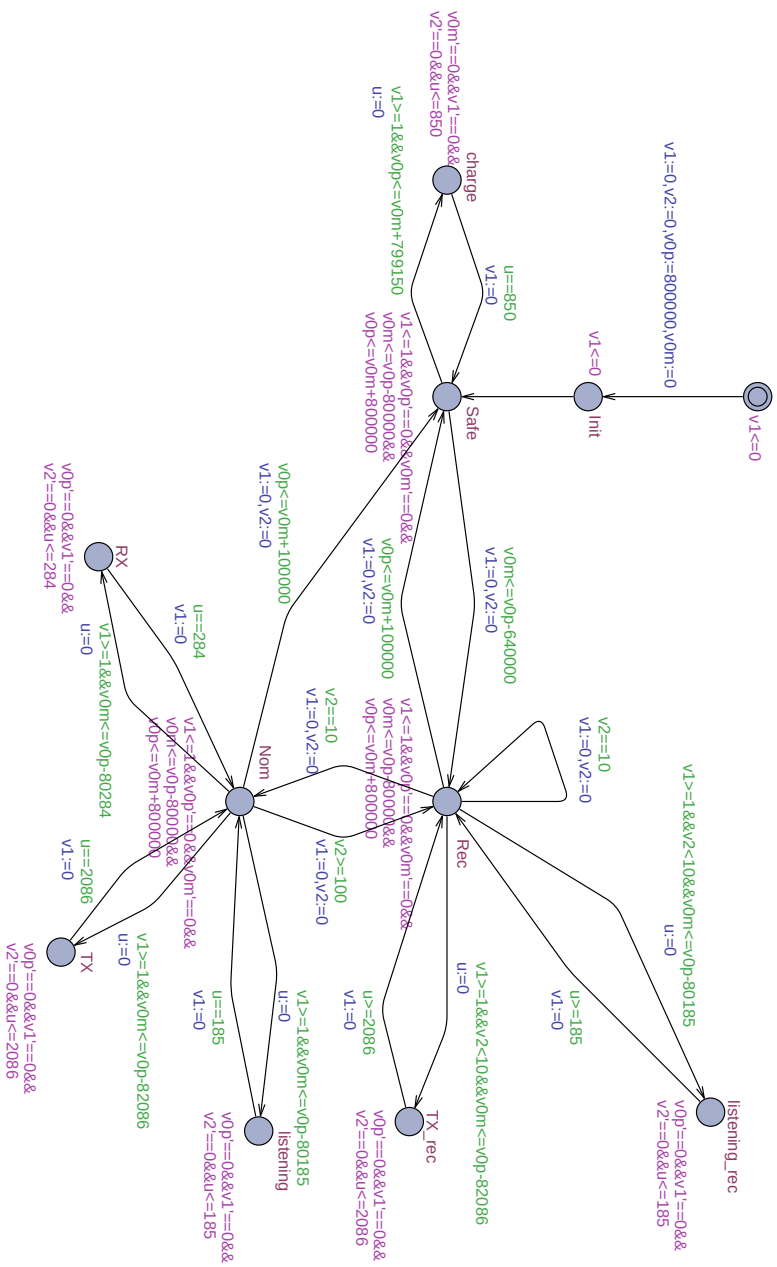



**Fig. 10.** The AAUSAT-II COM subsystem as a SWA in UPPAAL

## Résumé

We define and study the class of Capacity Timed Automata (CTA) which are Priced Timed Automata (PTA) extended with negative cost rates on edges and locations. The motivational example is the satellite AAUSAT-II designed and constructed by Aalborg University. During its orbit it will periodically be in the shadow of the Earth and not be able to charge the battery via its solar panels. In this situation it must operate with energy optimal task scheduling to ensure it does not risk running out of power.

We use CTA to model the system and use the introduced capacity variables to represent the charge of the satellite's battery. A proof sketch showing that the reachability problem for CTA is undecidable is also provided. We define an algorithm translating CTA to Linear Hybrid Automata (LHA), which allows the use of model checking tools like HyTech and Kronos for verication. The expressiveness of CTA is discussed, leading to a proof that the class CTA is a strict subset of the class LHA. Furthermore we translate the LHA to a StopWatch Automata (SWA) with the algorithm by Kim G. Larsen and Frank Cassez presented in The Impressive Power of Stopwatches.

With the system represented as an SWA we verify certain properties of the system operating the satellite by using the model checking tool UPPAAL. We perform three sample verications of the COM subsystem. One of which is to verify that the COM subsystem never violates the bounds of the capacity variable, which is proven to hold. From this we conclude that it is possible to do a two-step translation of CTA into SWA and to some extent verify properties of the model.



Jonas Groth

Anders Hesselager

Søren Larsen

Torkil Olsen