CERTIFIED REASONING ABOUT REAL-TIME SYSTEMS USING SCHEDULING ALGEBRA

A PREPRINT

David Pereira[*]
CISTER / ISEP - P.Porto
xxx
Porto, Portugal
drp@isep.ipp.pt

Who else?
xxx xxx
xxx
xx@yy.pt

February 4, 2020

ABSTRACT

Software application are becoming more complex, and their particular usage to control Cyber-Physical Systems that support several tasks of our daily activities, demand that these applications have to satisfy strict scheduling rules to ensure their correct operation, under strict timing constraints. Paradigmatic examples are ADAS applications that are necessary for the development of future autonomous vehicles, advanced robotics and its application in the context of Industry 4.0, or even applications for the financial sector. However, developing applications with strict timing constraints is know to be a difficult task. Therefore, frameworks that allow to specify and reasoning about these constraints can be of great value to the designers and developers of such applications. In this paper, we revisit *Scheduling algebra* and mechanise it in the type theory of Coq. The outcome of this mechanisation effort is a certified framework that allows to encode concurrent applications with timing constraints and reason about their timed, interleaved computations.

Keywords First keyword · Second keyword · More

1 Introduction

Applications are growing in complexity due to the increasing on the number of internal components, increased interaction with the physical environment (including with humans), and are highly connected. Many operations need to ensure strict timing requirements, e.g., autonomous driving, drone control, factory automation with robotics, among many others. It therefore becomes imperative for software developers to have access at rigorous methods that allow to precisely specify timed schedules and verify if such schedules are ensured at the run-time of the application.

Current industrial practice in

2 Scheduling Algebra

Scheduling algebra (SA) was introduced by van Glabbeek and Rittgen [] as an algebraic theory of scheduling. Its syntax allows to build terms that specify tasks built from actions that have an associated duration, while the semantic interpretation if that of sets of possible *schedules* satisfied by such specifications.

In this section we introduce and syntax and semantics of scheduling algebra, and show how we have encoding them in Coq.

***

Add an example of what type of property we are intending to formalize, and how one intends to reason about those.

[*]all who helped.

Explain at the high-level what is the semantic model we are going to consider (what type of traces are we considering), i.e., how we intend to formalize schedules so that we can algebraically reason about them.
**

2.1 Syntax

The language of SA, denoted $\mathcal{L}_{SA}$, is that of terms that allow to specify the concurrent execution of actions that have an associated execution time. It assumes a finite set of actions $\mathcal{A} = \{a_1, \ldots, a_n\}$ with $n \geq 1$ and a representation of time as the set of real numbers. A special empty action ϵ is assumed. Pairs $(a, t) \in \mathcal{A} \times \mathcal{T}$, denoted $[a]^t$, specifies actions a that take t units of time to execute (also referred to a has having duration t). In this work, we consider discrete, tick-based time and as so, time is interpreted as the set of natural numbers.

Terms $\tau \in \mathcal{L}_{SA}$ are inductively defined as follows:

$$\tau ::= \varepsilon \mid a^t \mid a^t \odot \tau \mid t \odot \tau \mid \Delta(\tau) \mid [\tau] \mid \tau \cup \tau \mid \tau \Cap \tau,$$

such that ϵ denotes the empty action, $[a]^t \odot \tau_1$ denotes the sequential execution of a during t time units followed by the term τ_1, $\tau_1 \cup \tau_2$ denotes the non-deterministic choice between executing τ_1 or τ_2, and $\tau_1 \Cap \tau_2$ denotes the concurrent execution of τ_1 and τ_2.

The encoding in Coq of the above concepts is straight forward, by defining two ADTs, `Action` and `term`, as presented below.

```
Inductive Action : Type :=
(** Empty action *)
| noAct : Action
(** Some action *)
| aAct : ℕ → Action.

Inductive term : Type :=
(** Empty process (lifting of empty action to processes) *)
| empProc  : term
(** Execution of a single action during some time *)
| singleProc : Action → nat → term
(** Execution of an action during some time, followed by other process(es) *)
| seqProc  : Action → nat → term → term
(** Offsetting the execution of a process for a given number of time units *)
| offsetProc : nat → term → term
(** Unbounded delay of a process *)
| delayProc : term → term
(** Elimination of delays *)
| elimProc : term → term
(** Non-deterministic choice between two processes *)
| choiceProc : term → term → term
(** Concurrent execution of processes *)
| forkProc : term → term → term.
```

We consider syntactic shortcuts for representing: 1) the execution of a single action $\langle a \rangle^t$ which denotes the term $[a]^t \odot \epsilon$, and starting the execution of any action with an offset of t time units $t \odot \tau$ which denotes the term $[\epsilon]^t \odot \tau$.

2.2 Semantics

The core concepts supporting the semantics of SA is that of a schedule, which is represented by a triple $\langle \sigma_1, \sigma_2, \sigma_3 \rangle$, such that σ_1 is a multiset of triples (a, i, f) with $a \in \mathcal{A}$ being an action, $i \in \mathbb{N}$ being the initial point in time when a starts to execute, and $f \in \mathbb{N} \setminus \{0\}$ being the point in time when a stops its execution, $\sigma_2 \subseteq N$ is a set of points referring to starting points of the actions in σ_1, and σ_3 is a set of so-called *anchor points*, that is, points where actions have stopped execution. Note that since σ_1 is a multi-set, the tiple (a, i, f) can occur several times, denoting the parallel execution of a starting at i and finishing at f. We refer to elements (a, i, f) as *bounded-timed actions* (BTA) since they denote the execution of the action a during f units of time, starting at point in time i.

Formally, SA is interpreted over set whose elements have the type $\mathcal{S} = M(\mathcal{A}, \mathbb{N}, \mathbb{N} - 0) \times 2^{\mathbb{N}} \times 2^{\mathbb{N}-\{0\}}$ such that each element $s \in \mathcal{S}$ is a bounded-timed action, and the two remaining components are subsets of the natural numbers containing all the initial execution times and final-execution times, so called-anchors, of those actions.

Before introducing the complete interpretation of schedule terms, we provide an intuitive example of what is an expected schedule for the simple program $a^1 \odot b^2 \Cap a^1$ which executes two concurrent tasks, one composed of a sequence of BTA and another with just a single one. The execution of the program goes as follows. On program start, the action a runs for a single unit of time, representing the BTA $(a, 0, 1)$. The corresponding schedule is $\langle\{(a, 1, 0), (a, 1, 0)\}, \{0, 1\}, \{1\}\rangle$. Now, b must execute for 2 units of time, starting in time 1 and hence we have $(b, 1, 2)$. Consequently, we obtain the schedule $\langle\{(a, 1, 0), (a, 1, 0), (b, 2, 1)\}, \{0, 1, 3\}, \{1, 3\}\rangle$.

Definition 1 *Let $S_1 = \langle\sigma_1^1; \sigma_2^1; \sigma_3^1\rangle$ and $S_2 = \langle\sigma_1^2; \sigma_2^2; \sigma_3^2\rangle$ be two schedules. The union of S_1 with S_2, denoted $S_1 \cup S_2$ is the schedule $\langle\sigma_1^1 \cup_m \sigma_1^2; \sigma_2^1 \cup \sigma_2^2; \sigma_3^1 \cup \sigma_3^2\rangle$, such that union of schedules $\cup_m$ is the classic union over multisets.*

3 Examples of citations, figures, tables, references

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

[1, 2] and see [3].

The documentation for `natbib` may be found at

`http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf`

Of note is the command `\citet`, which produces citations appropriate for use in inline text. For example,

```
\citet{hasselmo} investigated\dots
```

produces

> Hasselmo, et al. (1995) investigated…

`https://www.ctan.org/pkg/booktabs`

3.1 Figures

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

See Figure 1. Here is how you add footnotes. [2] Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

3.2 Tables

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

See awesome Table 1.

[2] Sample of the first footnote.

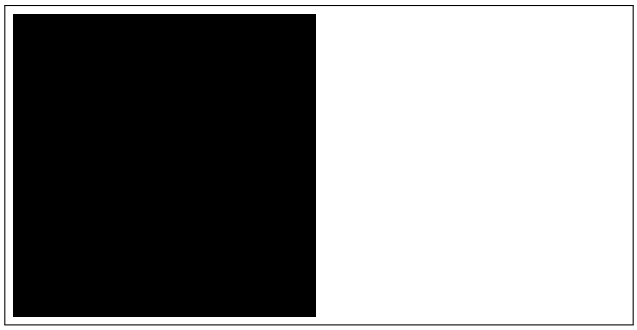

Figure 1: Sample figure caption.

Table 1: Sample table title

	Part	
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

3.3 Lists

- Lorem ipsum dolor sit amet
- consectetur adipiscing elit.
- Aliquam dignissim blandit est, in dictum tortor gravida eget. In ac rutrum magna.

References

[1] George Kour and Raid Saabne. Real-time segmentation of on-line handwritten arabic script. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 417–422. IEEE, 2014.

[2] George Kour and Raid Saabne. Fast classification of handwritten on-line arabic characters. In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, pages 312–318. IEEE, 2014.

[3] Guy Hadash, Einat Kermany, Boaz Carmeli, Ofer Lavi, George Kour, and Alon Jacovi. Estimate and replace: A novel approach to integrating deep neural networks with existing applications. *arXiv preprint arXiv:1804.09028*, 2018.