Analysis of timed processes with data using algebraic transformations

Michel A. Reniers Yaroslav S. Usenko

Department of Mathematics and Computer Science,
Technical University of Eindhoven,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Abstract

The language of timed μCRL is an extension of an ACP-style process algebra-based language μCRL with time-related features. In this paper we describe this language and its equational axiomatization, and give an example specification. We outline the method of simplifying transformations based on this equational axiomatization, and illustrate it on this example. This transformation method allows for a time-free abstraction of the specification, which in turn enables the use of tools and techniques for verification of untimed systems. We prove some properties of the example using a known invariants technique.

Key words: Process Algebra, Real Time, Axioms, Verification.

1 Introduction

The language μCRL, see [13], offers a uniform framework for the specification of data and processes. Data are specified by equational specifications (cf. [5, 19]): one can declare sorts and functions working upon these sorts, and describe the meaning of these functions by equational axioms. Processes are described in process algebraic style, where the particular process syntax stems from ACP [6, 4, 9], extended with data-parametric ingredients: there are constructs for conditional composition, and for data-parametric choice and communication. As is common in process algebra, infinite processes are specified by means of (finite systems of) recursive equations. In μCRL such equations can also be data-parametric. As an example, for action a and adopting standard semantics for μCRL, each solution for the equation $\mathsf{X} = \mathsf{a} \cdot \mathsf{X}$ specifies (or "identifies") the process that can only repeatedly execute a, and so does $\mathsf{Y}(17)$ where $\mathsf{Y}(n)$ is defined by the data-parametric equation $\mathsf{Y}(n) = \mathsf{a} \cdot \mathsf{Y}(n+1)$ with $n \in \mathit{Nat}$.

Several timed extensions have been proposed for different kinds of process algebras. For an overview of ACP extensions with time we refer to [3]. According to [3], timed process algebras can be categorized by three criteria. *Discrete* vs. *continuous* time; *relative* vs. *absolute* time; *two-phase* vs. *timed-stamped* model. In [11] the language μCRL is extended with time, and in [21] a sound and complete axiomatization of timed μCRL is presented. In [15] some examples of specification and reasoning in timed μCRL are given. Timed μCRL makes use of absolute time, timed stamped model, and the time domain can be defined by the user (both discrete and continuous domains are possible). For a way to interpret timed automata [1] in timed μCRL we refer to [23, Chapter 6].

In this paper we present a method to describe and analyze real-time systems using μCRL and timed μCRL. It is assumed that some real-time system is described by means of a timed μCRL specification (see Section 2). Mostly such descriptions will contain operators such as parallel composition that complicate analysis. As a first step towards the analysis of such systems, we *linearize* the given description using the algorithm from [22] (see Section 3). The result is a *Timed*

Linear Process Equation (TLPE) which is equivalent to the original description and has a very simple structure. On this TLPE it is already possible to perform some analysis.

Next, in Section 4, we describe how a TLPE can be transformed into an LPE, i.e., a linear process equation without time. This transformation, called *time-free abstraction*, has been used for a more restricted class of timed μCRL specifications in [21]. Crucial steps in this transformation are that the TLPE is first transformed into a *well-timed* TLPE and second that it is transformed into a *deadlock-saturated* well-timed TLPE. Finally, all time-stamping is captured in the parameters of atomic actions. The result is an LPE for which the machinery of untimed μCRL can be put to use for further analysis. These are based on symbolic analysis of the specifications, such as invariants, term rewriting and theorem proving, or on explicit state space generation and model-checking.

We illustrate the respective steps of the proposed method on the Bottle Filling System from [15, Section 3] and [3, Section 4.2.5]. In comparison to those expositions, we apply a proved transformation method [22] and a general analysis technique based on invariants [7].

2 μCRL and Timed μCRL

Timed μCRL specifications contain algebraic specifications of several abstract data types. The only data types that are required are booleans and time. The algebraic specifications of booleans are standard and can be found for instance in [8, Chapter IV]. We assume a constant $\mathbf{t}$ and functions $\neg : \mathit{Bool} \to \mathit{Bool}$ and $\vee : \mathit{Bool} \times \mathit{Bool} \to \mathit{Bool}$.

Time Time can be represented in many different ways. In timed μCRL the time domain has to satisfy a set of properties. We present these properties as an algebraic specification of sort *Time* by defining its signature and the axioms. The signature of sort *Time* consists of: a constant $\mathbf{0}$; functions $\mathit{leq}, \mathit{eq} : \mathit{Time} \times \mathit{Time} \to \mathit{Bool}$, which are often abbreviated as $\leq$ and $=$, respectively; function $\mathit{if} : \mathit{Bool} \times \mathit{Time} \times \mathit{Time} \to \mathit{Time}$; and functions $\mathit{min}, \mathit{max} : \mathit{Time} \times \mathit{Time} \to \mathit{Time}$.

The axioms of sort *Time* are presented below. Many of the axioms are taken from, or inspired by [11, 16]. The axioms say that $\leq$ is a total order on the *Time* domain, and $\mathbf{0}$ is the least element.

$$t \leq u \wedge u \leq w \approx t \leq u \wedge u \leq w \wedge t \leq w \qquad \text{(Time1}')$$
$$t \leq u \wedge \neg\, w \leq u \approx t \leq u \wedge \neg\, w \leq u \wedge \neg\, w \leq t \qquad \text{(Time1}'')$$
$$\neg\, u \leq t \wedge u \leq w \approx \neg\, u \leq t \wedge u \leq w \wedge \neg\, w \leq t \qquad \text{(Time1}''')$$
$$\mathbf{0} \leq t \approx \mathbf{t} \qquad \text{(Time2)}$$
$$t \leq u \vee u \leq t \approx \mathbf{t} \qquad \text{(Time3)}$$
$$\mathit{eq}(t,u) \approx t \leq u \wedge u \leq t \qquad \text{(Time5)}$$
$$\mathit{min}(t,u) \approx \mathit{if}(t \leq u, t, u) \qquad \text{(Time6)}$$
$$\mathit{max}(t,u) \approx \mathit{if}(u \leq t, t, u) \qquad \text{(Time6}')$$
$$\mathit{if}(\mathbf{t}, t, u) \approx t \qquad \text{(Time7)}$$
$$\mathit{if}(\neg b, t, u) \approx \mathit{if}(b, u, t) \qquad \text{(Time8}')$$
$$\mathit{if}(b_1 \vee b_2, t, u) \approx \mathit{if}(b_1, t, \mathit{if}(b_2, t, u)) \qquad \text{(Time9)}$$
$$\mathit{if}(b_1, \mathit{if}(b_2, t, u), w) \approx \mathit{if}(b_1 \wedge b_2, t, \mathit{if}(b_1, u, w)) \qquad \text{(Time10)}$$
$$\mathit{if}(t \leq u \wedge u \leq t, t, u) \approx u \qquad \text{(Time11)}$$
$$t \leq \mathit{if}(b, u, w) \approx (b \wedge t \leq u) \vee (\neg b \wedge t \leq w) \qquad \text{(Time12)}$$
$$\mathit{if}(b, u, w) \leq t \approx (b \wedge u \leq t) \vee (\neg b \wedge w \leq t) \qquad \text{(Time13)}$$

The last seven axioms allow to eliminate *if* from any boolean expression containing subterms of sort *Time*. Every term of sort *Time* can be represented as $\mathbf{0}$, a variable, or as $\mathit{if}(b,t,u)$ where t and u are terms of sort *Time*. The above mentioned form has two extremes: one where all boolean terms b are variables, and another, where every variable of sort time (and $\mathbf{0}$) occurs at most once. The latter form is useful for proving time identities in the following way: if we order the time variables occurring in a term as $\mathbf{0} < t_1 < \ldots < t_n$, then with the help of the axioms we can transform every term of sort *Time* to the form $\mathit{if}(b_1, t_{i_0}, \mathit{if}(b_2, t_{i_1}, \ldots \mathit{if}(b_m, t_{i_{m-1}}, t_{i_m}) \ldots))$ with indices such that $t_{i_k} < t_{i_{k+1}}$. Moreover, the conditions $b_1, \ldots, b_m$ can be made pairwise distinct,

i.e. having the property that $i \neq j \rightarrow b_i \wedge b_j \approx \mathbf{f}$. In addition, the conditions $b_1, \ldots, b_m$ can be made such that if $eq(t_{i_k}, t_{i_{k+1}}) \approx \mathbf{t}$, then $b_k \approx \mathbf{t}$. This gives us a method for proving identities of sort *Time*.

Other Data Types. Any other data type in μCRL is specified in a similar way by providing a signature and axioms from which all other identities are derived. Other data sorts have generally different axioms, and sometimes induction principles (cf. [14]) are required to describe them.

Processes Next we define the binding-equational theory of timed μCRL by defining its signature and the axioms. The signature of timed μCRL consists of data sorts (or 'data types') including *Bool* and *Time* as defined above, and a distinct sort *Proc* of processes. Each data sort D is assumed to be equipped with a binary function $eq : D \times D \rightarrow Bool$. (This requirement can be weakened by demanding such functions only for data sorts that are parameters of communicating actions). The process operations are the ones listed below:

- actions $\mathsf{a} : \overrightarrow{D_\mathsf{a}} \rightarrow Proc$ where $\mathsf{a} \in ActLab$ is an action label and $\overrightarrow{D_\mathsf{a}}$ is a list of parameter types of a. It is assumed that the signature of timed μCRL is parameterized by the finite set of action labels *ActLab*.
- deadlock $\delta :\rightarrow Proc$. The constant δ models inaction, or the inability to perform actions.
- alternative composition $+ : Proc \times Proc \rightarrow Proc$. The process $p + q$ behaves like p or like q, depending on which of the two performs the first action.
- sequential composition $\cdot : Proc \times Proc \rightarrow Proc$. The process $p \cdot q$ first performs the actions of p, until p terminates, and then continues with the actions from q.
- conditional operator $\_ \lhd \_ \rhd \_ : Proc \times Bool \times Proc \rightarrow Proc$. The process term $p \lhd b \rhd q$ behaves like p if b is equal to $\mathbf{t}$, and if b is equal to $\mathbf{f}$ it behaves like q.
- alternative quantification $\sum_{d:D} : Proc \rightarrow Proc$, for each data variable d of sort D. The process $\sum_d p$ behaves like $p[d_1/d] + p[d_2/d] + \cdots$, i.e., as the possibly infinite alternative composition of processes $p[d_i/d]$ for any data term d_i of sort D.
- at-operator $\leftcurvearrowright : Proc \times Time \rightarrow Proc$. A key feature of timed μCRL is that it can be expressed at which time a certain action must take place. This is done using the at-operator. The process $p \leftcurvearrowright t$ behaves like the process p, with the restriction that the first action of p must start at time t. The process $p \leftcurvearrowright t$ can delay till at most time t. If p consists of several alternatives, then only those with the first actions starting at time t will remain in $p \leftcurvearrowright t$. The alternatives that start earlier than t will express that $p \leftcurvearrowright t$ can delay till that earlier time. The alternatives that start later than t will express that $p \leftcurvearrowright t$ can wait till time t (but not till that later time).
- initialization operator $\gg : Time \times Proc \rightarrow Proc$ and weak initialization operator $\ggg : Time \times Proc \rightarrow Proc$. The *initialization* operator $t \gg p$ expresses the process in which all alternatives of p that start earlier than t are left out, but an alternative to delay till time t is added. The *weak initialization* operator $t \ggg p$ expresses the process in which all alternatives of p that start earlier than t are replaced by the ability to delay till those earlier times. Thus the process $t \ggg p$ can delay till the same time as p, while $t \gg p$ can delay till at least time t, which can be longer than p could delay.
- parallel composition $\parallel : Proc \times Proc \rightarrow Proc$, left-merge $\lfloor\!\lfloor : Proc \times Proc \rightarrow Proc$, and communication merge $| : Proc \times Proc \rightarrow Proc$. The process $p \parallel q$ can first perform an action of p, first perform an action of q, or start with a communication, or synchronization, between p and q. The process $p \parallel q$ exists at time t only if both p and q exist at time t. The process $p \lfloor\!\lfloor q$ is as $p \parallel q$, but the first action that is performed comes from p. The action can only be performed if the other party still exists at that time. The process $p \mid q$ also behaves as

the process $p \parallel q$, except that the first action must be a communication between p and q. Furthermore, these actions must occur at the same time and have the same data parameters. The action resulting from such a communication is defined by the partial commutative and associative function $\gamma : ActLab \times ActLab \rightarrow ActLab$ such that $\gamma(\mathsf{a}_1, \mathsf{a}_2) \in ActLab$ implies that $\mathsf{a}_1, \mathsf{a}_2$ and $\gamma(\mathsf{a}_1, \mathsf{a}_2)$ have parameters of the same sorts. It is assumed that the signature of timed μCRL is parameterized by this function γ.

- encapsulation $\partial_H : Proc \rightarrow Proc$, for $H \subseteq ActLab$. The process $\partial_H(p)$ behaves as the process p where the execution of actions from the set H is prohibited.
- ultimate delay $\partial\mathcal{U} : Proc \rightarrow Proc$. The *ultimate delay* operator $\partial\mathcal{U}(p)$ expresses the process, which can delay as long as p can, but cannot perform any action.
- before operator $\ll : Proc \rightarrow Proc$. The *before* operator $p \ll q$ expresses the process in which all alternatives of p that start later than $\partial\mathcal{U}(q)$ are replaced by the abilities to delay till $\partial\mathcal{U}(q)$. Thus $p \ll q$ cannot delay longer than both p and q. The ultimate delay $\partial\mathcal{U}(p)$ of process p can be expressed in terms of $\ll$ as $\delta \ll p$. This process cannot perform actions and can delay as long as p could (because δ can delay till any time).

Another key feature of timed μCRL is that it can be expressed that a process can delay till a certain time. The process $p + \delta \cdot t$ can certainly delay till time t, but can possibly delay longer, depending on p. Consequently, the process $\delta \cdot \mathbf{0}$ can neither delay nor perform actions, and the process δ can delay for an arbitrary long time, but cannot perform any action. We follow the intuition that a process that can delay till time t can also delay till an earlier moment, and a process that can perform a *first* action at time t can also delay till time t.

The descending order of binding strength of operators is: $\cdot$, $\cdot$, $\{\gg, \ggg, \ll\}$, $\{\|, \between, |\}$, $\lhd\rhd$, $\sum$, $+$. In Appendix A the axioms of timed μCRL are given. Many of these axioms are taken from, or inspired by [21, 12].

To prove identities in timed μCRL we use a combined many-sorted calculus, which for the sort of processes has the rules of binding-equational calculus, for the sorts of booleans and time has the rules of equational calculus, while other data sorts may include induction principles which could be used to derive process identities as well. We note that the derivation rules of binding-equational calculus do not allow to substitute terms containing free variables if they become bound.

The operational semantics (SOS) of timed μCRL and soundness and completeness proofs of the axiomatization are presented in [21]. The axiomatization used here is an extension of the axiomatization in [21] with a number of axioms that are derivable in the setting of [21] for all closed terms. These extra axioms are needed to prove correctness of the linearization.

Timed μCRL Specifications For the purpose of this paper we restrict to the timed μCRL specifications that do not contain left merge ($\between$), communication ($|$), ultimate delay ($\partial\mathcal{U}$), and before ($\ll$) operators explicitly. These operators were introduced to allow the finite axiomatization of parallel composition ($\|$) and timing constructs in the bisimulation setting, and they are hardly used explicitly in timed μCRL specifications.

We consider *systems* of *process equations* with the right hand sides from the following subset of timed μCRL terms

$$p ::= \mathsf{a}(\overrightarrow{t}) \mid \delta \mid \mathsf{Y}(\overrightarrow{t}) \mid p + p \mid p \cdot p \mid p \parallel p \mid \textstyle\sum_{d:D} p \mid p \lhd c \rhd p \mid \partial_H(p) \mid p \cdot t \mid t \gg p \mid t \ggg p$$

For a system of process equations G containing a process equation for X, $(\mathsf{X}(\overrightarrow{t}), G)$ is a *process definition* if $\overrightarrow{t}$ is a list of data terms that corresponds to the type of process X. The combination of the given data specification with a process definition $(\mathsf{X}(\overrightarrow{t}), G)$ of process equations determines a timed μCRL *specification*. Such a specification depends on a finite subset Act of $ActLab$ and on $Comm$, an enumeration of γ restricted to the labels in Act.

2.1 Example: Bottle Filling System

This example is taken from [15, Section 3] and [3, Section 4.2.5]. We start from the informal specification from [3, page 153]: "Bottles on a conveyor belt are filled with 10 liters of liquid poured from a container with a capacity of m liters. The container is filled at a constant rate of r liters per second. When a bottle is under the container, a tap is opened and the bottle is filled at a rate of 3 liters per second until the container becomes empty. From that moment, the bottle is filled at the same rate as the container. When the bottle is full, the tap is closed and the conveyor belt starts moving to put the next bottle under the container which takes 1 second. Obviously, it is highly preferable that overflow (of the container) never occurs. Of course, it is also preferable that the container does not get empty during the filling of each bottle."

Specification in Timed μCRL The time domain used in this specification is nonnegative rational numbers. For the specification of the conveyor belt we distinguish three 'modes of operation': mv – moving, nf – normal filling, and sf – slow filling.

$$\begin{aligned}
\mathsf{CB}^{\mathsf{mv}}(t{:}\,Time) &= !\mathsf{start} \triangleleft (t+1) \cdot \mathsf{CB}^{\mathsf{nf}}(t+1) \\
\mathsf{CB}^{\mathsf{nf}}(t{:}\,Time) &= \sum_{t'{:}\,Time} \big(?\mathsf{empty} \triangleleft (t+t') \cdot \mathsf{CB}^{\mathsf{sf}}(t+t', 3t') \lhd t' < 10/3 \rhd \\
&\qquad !\mathsf{stop} \triangleleft (t+10/3) \cdot \mathsf{CB}^{\mathsf{mv}}(t+10/3) \big) \\
\mathsf{CB}^{\mathsf{sf}}(t{:}\,Time, l{:}\,\mathbb{Q}) &= !\mathsf{stop} \triangleleft (t+(10-l)/r) \cdot \mathsf{CB}^{\mathsf{mv}}(t+(10-l)/r)
\end{aligned}$$

The process $\mathsf{CB}^{\mathsf{mv}}$ executes the $!\mathsf{start}$ action and then behaves as the process $\mathsf{CB}^{\mathsf{nf}}$. The latter process can synchronize with the container process by $?\mathsf{empty}$ at time period $t+t'$, where $t' \in [\mathbf{0}, 10/3)$, or it can synchronize by $!\mathsf{stop}$ action at time $t+10/3$. The further behavior of $\mathsf{CB}^{\mathsf{nf}}$ depends on which action it synchronized.

For the specification of the container also three modes are distinguished: inc – increasing the amount of liquid, dec – decreasing, and dry – liquid goes through the empty container directly into the bottle.

$$\begin{aligned}
\mathsf{C}^{\mathsf{inc}}(t{:}\,Time, h{:}\,\mathbb{Q}) &= \sum_{t'{:}\,Time} \big(?\mathsf{start} \triangleleft (t+t') \cdot \mathsf{C}^{\mathsf{dec}}(t+t', h+rt') \lhd t' < (m-h)/r \rhd \\
&\qquad !\mathsf{overflow} \triangleleft (t+(m-h)/r) \cdot \delta \triangleleft (t+(m-h)/r) \big) \\
\mathsf{C}^{\mathsf{dec}}(t{:}\,Time, h{:}\,\mathbb{Q}) &= \sum_{t'{:}\,Time} ?\mathsf{stop} \triangleleft (t+t') \cdot \mathsf{C}^{\mathsf{inc}}(t+t', h-(3-r)t') \lhd t' \le h/(3-r) \rhd \delta \triangleleft \mathbf{0} \\
&\qquad + !\mathsf{empty} \triangleleft (t+h/(3-r)) \cdot \mathsf{C}^{\mathsf{dry}}(t+h/(3-r)) \\
\mathsf{C}^{\mathsf{dry}}(t{:}\,Time) &= \sum_{t'{:}\,Time} ?\mathsf{stop} \triangleleft t' \cdot \mathsf{C}^{\mathsf{inc}}(t', 0)
\end{aligned}$$

The process $\mathsf{C}^{\mathsf{dec}}$ behaves nondeterministically at time $t+h/(3-r)$. Depending on the parallel process it can either perform $?\mathsf{stop}$ or $!\mathsf{empty}$ in order to synchronize with that process.

Using the above descriptions of the conveyor belt and the container, the system can be given as:

$$\mathsf{T}(t{:}\,Time, h{:}\,\mathbb{Q}) = \partial_H(\mathsf{CB}^{\mathsf{mv}}(t) \parallel \mathsf{C}^{\mathsf{inc}}(t,h))$$

where $\gamma(?s, !s) = \gamma(!s, ?s) = s$ for $s \in \{\mathsf{start}, \mathsf{stop}, \mathsf{empty}\}$ and γ is undefined otherwise, and $H = \{?s, !s \mid s \in \{\mathsf{start}, \mathsf{stop}, \mathsf{empty}\}\}$. In words, the system is a parallel composition of the conveyor belt and the container processes, that are forced to synchronize on all actions except $!\mathsf{overflow}$.

Let G contain all of the above equations. Then the process definition $(\mathsf{T}(0,h), G)$ forms the specification of the bottle-filling system.

3 Linearization

The problem of linearization of a timed μCRL specification defined by $(\mathsf{X}(\overrightarrow{t}), G)$ consists of generation of a new timed μCRL specification which

- depends on the same *Act* and *Comm*,
- contains all data definitions of the original one, and, possibly, definitions of the auxiliary data types,
- is defined by $(\mathsf{Z}(\mathrm{m}_{\mathsf{X}}(\overrightarrow{t})), L)$, where L contains exactly one process equation for Z in linear form (defined later), and m_{X} is a mapping from the parameters of X to the parameters of X.

such that all processes that are solutions of $(\mathsf{X}(\overrightarrow{t}), G)$ are also solutions of $\mathsf{Z}(\mathrm{m}_{\mathsf{X}}(\overrightarrow{t})), L)$.

It is not possible to linearize a timed μCRL specification which is unguarded, e.g. $\mathsf{X} = \mathsf{X}$ cannot be brought to the linear form. The exact notion of guardedness in μCRL is rather complicated. In a nutshell, in a guarded process every occurrence of a recursive call is preceded (with sequential composition) by an action. We refer to [22, Section 3.6] for a precise definition.

We define Timed Parallel Greibach Normal Form (TPEGNF) and Timed Linear Process Equation (TLPE) as special forms of process equations in timed μCRL. TPEGNF and TLPE are similar to the Greibach Normal Form [10] for context-free languages. A timed μCRL process equation is in TPEGNF if it is of the form:

$$\begin{aligned}\mathsf{X}(\overrightarrow{d{:}D}) = &\sum_{i\in I}\sum_{\overrightarrow{e_i:E_i}} \mathsf{a}_i(\overrightarrow{f_i}(\overrightarrow{d,e_i}))\vartriangleleft t_i(\overrightarrow{d,e_i})\cdot p_i(\overrightarrow{d,e_i}) \lhd c_i(\overrightarrow{d,e_i}) \rhd \delta\vartriangleleft\mathbf{0}\\ &+\sum_{j\in J}\sum_{\overrightarrow{e_j:E_j}} \mathsf{a}_j(\overrightarrow{f_j}(\overrightarrow{d,e_j}))\vartriangleleft t_j(\overrightarrow{d,e_j}) \lhd c_j(\overrightarrow{d,e_j}) \rhd \delta\vartriangleleft\mathbf{0}\\ &+\sum_{\overrightarrow{e_\delta:E_\delta}} \delta\vartriangleleft t_\delta(\overrightarrow{d,e_\delta}) \lhd c_\delta(\overrightarrow{d,e_\delta}) \rhd \delta\vartriangleleft\mathbf{0}\end{aligned}$$

where I and J are disjoint, and all $p_i(\overrightarrow{d,e_i})$ have the following syntax:

$$p{::=}\mathsf{a}(\overrightarrow{t}) \mid \delta \mid \mathsf{Y}(\overrightarrow{t}) \mid p\cdot p \mid p\parallel p \mid \partial_H(p\|p) \mid \partial_H(\mathsf{Y}(\overrightarrow{t})) \mid p\vartriangleleft t \mid t\gg p \mid t\ggg p$$

A timed μCRL process equation is called *Timed Linear Process Equation (TLPE)* if it is of the same form as above, but the terms $p_i(\overrightarrow{d,e_i})$ are recursive calls of the form $\mathsf{X}(\overrightarrow{g_i}(\overrightarrow{d,e_i}))$ for some function vectors $\overrightarrow{g_i}$.

The equation is explained as follows. The process X, being in a state vector $\overrightarrow{d}$, can for any $\overrightarrow{e_i}$, that satisfy the condition $c_i(\overrightarrow{d,e_i})$, perform an action a_i parameterized by $\overrightarrow{f_i}(\overrightarrow{d,e_i})$ at the absolute time $t_i(\overrightarrow{d,e_i})$, and then proceed to the state $\overrightarrow{g_i}(\overrightarrow{d,e_i})$. Moreover, it can for any $\overrightarrow{e_j}$, that satisfy the condition $c_j(\overrightarrow{d,e_j})$, perform an action a_j parameterized by $\overrightarrow{f_j}(\overrightarrow{d,e_j})$, and then terminate successfully. The last summand indicates that for any $\overrightarrow{e_\delta{:}E_\delta}$, that satisfies $c_\delta(\overrightarrow{d,e_\delta})$, the process can wait till the absolute time $t_\delta(\overrightarrow{d,e_\delta})$.

As input for the linearization procedure we take a timed μCRL process definition $(\mathsf{X}(\overrightarrow{t}), G)$. Further on, the process goes through a number of intermediate forms, including TPEGNF, and finally we get to TLPE. All the steps are described in [22, Chapter 6] and are proved to transform a system of process equations in timed μCRL to an equivalent one.

3.1 Linearization of the Example

In this subsection we illustrate some of the linearization steps on our example. The example does not contain double bound variables, so we can start with reducing right hand sides of the equations

with the help of conventional term rewriting [2]. This step is described in [22, Section 6.2.2]. By doing this step the equations for $\mathsf{CB}^{\mathsf{nf}}$ and $\mathsf{C}^{\mathsf{inc}}$ change to the following:

$$\begin{aligned}\mathsf{CB}^{\mathsf{nf}}(t{:}Time) = &\sum_{t':Time} ?\mathsf{empty} \triangleleft (t+t') \cdot \mathsf{CB}^{\mathsf{sf}}(t+t', 3t') \lhd t' < 10/3 \rhd \delta \triangleleft \mathbf{0} \\ &+ !\mathsf{stop} \triangleleft (t+10/3) \cdot \mathsf{CB}^{\mathsf{mv}}(t+10/3) \\ \mathsf{C}^{\mathsf{inc}}(t{:}Time, h{:}\mathbb{Q}) = &\sum_{t':Time} ?\mathsf{start} \triangleleft (t+t') \cdot \mathsf{C}^{\mathsf{dec}}(t+t', h+rt') \lhd t' < (m-h)/r \rhd \delta \triangleleft \mathbf{0} \\ &+ !\mathsf{overflow} \triangleleft (t+(m-h)/r) \cdot \delta \triangleleft (t+(m-h)/r)\end{aligned}$$

In both equations we move the alternative composition operator $(+)$ outside the sum operator and eliminate the sum in the second summand.

At this point all our equations, except the one for T are in TPEGNF. We proceed by guarding [22, Section 6.2.4] the equation for T. We have to consider the term $\partial_H(\mathsf{CB}^{\mathsf{mv}}(t) \parallel \mathsf{C}^{\mathsf{inc}}(t,h))$ and apply the guarding procedure to it. We get the following:

$$\begin{aligned}&\partial_H(\mathsf{CB}^{\mathsf{mv}}(t) \parallel \mathsf{C}^{\mathsf{inc}}(t,h)) = \\ &\quad \mathsf{start} \triangleleft (t+1) \cdot \partial_H(\mathsf{CB}^{\mathsf{nf}}(t+1) \parallel \mathsf{C}^{\mathsf{dec}}(t+1, h+r)) \lhd h < m-r \rhd \delta \triangleleft \mathbf{0} \\ &+ !\mathsf{overflow} \triangleleft (t+(m-h)/r) \cdot \delta \triangleleft (t+(m-h)/r) \lhd m-r \leq h \rhd \delta \triangleleft \mathbf{0}\end{aligned}$$

Here we use the fact that only $!\mathsf{overflow}$ action is not forced to synchronize, and the rest of the actions have to. It is interesting to see how the $!\mathsf{overflow}$ action gets its condition. It has to happen before time $t+1$, otherwise the action $!\mathsf{start} \triangleleft (t+1)$ should occur first. This means that $t+(m-h)/r \leq t+1$ should hold, which is equivalent to $m-r \leq h$.

Now we consider the term $\partial_H(\mathsf{CB}^{\mathsf{nf}}(t) \parallel \mathsf{C}^{\mathsf{dec}}(t,h))$ and apply the guarding procedure to it. Here we again use the fact that the all the actions of the two processes have to synchronize. We get the following:

$$\begin{aligned}&\partial_H(\mathsf{CB}^{\mathsf{nf}}(t) \parallel \mathsf{C}^{\mathsf{dec}}(t,h)) = \\ &\quad \mathsf{stop} \triangleleft (t+10/3) \cdot \partial_H(\mathsf{CB}^{\mathsf{mv}}(t+10/3) \parallel \mathsf{C}^{\mathsf{inc}}(t+10/3, h-10(3-r)/3)) \\ &\qquad \lhd 10(3-r)/3 \leq h \rhd \delta \triangleleft \mathbf{0} \\ &+ \mathsf{empty} \triangleleft (t+h/(3-r)) \cdot \partial_H(\mathsf{CB}^{\mathsf{sf}}(t+h/(3-r), 3h/(3-r)) \parallel \mathsf{C}^{\mathsf{dry}}(t+h/(3-r))) \\ &\qquad \lhd h < 10(3-r)/3 \rhd \delta \triangleleft \mathbf{0}\end{aligned}$$

Now we consider the term $\partial_H(\mathsf{CB}^{\mathsf{sf}}(t,l) \parallel \mathsf{C}^{\mathsf{dry}}(t))$ and apply the guarding procedure to it. We get the following:

$$\begin{aligned}&\partial_H(\mathsf{CB}^{\mathsf{sf}}(t,l) \parallel \mathsf{C}^{\mathsf{dry}}(t)) = \\ &\mathsf{stop} \triangleleft (t+(10-l)/r) \cdot \partial_H(\mathsf{CB}^{\mathsf{mv}}(t+(10-l)/r) \parallel \mathsf{C}^{\mathsf{inc}}(t+(10-l)/r, 0))\end{aligned}$$

At this point we are ready to make a single equation for the whole system (cf. [22, Section 4.3]). Here we use the fact that only the parallel processes are reachable. We define a new sort

$State = \{\mathsf{mv\_inc}, \mathsf{nf\_dec}, \mathsf{sf\_dry}, \mathsf{dl}\}$ and use it as a parameter of the resulting process T:

$$
\begin{aligned}
&\mathsf{T}(s{:}State, t{:}Time, h, l{:}\mathbb{Q}) = \\
&\quad \mathsf{start} \vartriangleleft (t+1) \cdot \mathsf{T}(\mathsf{nf\_dec}, t+1, h+r, 0) \lhd s = \mathsf{mv\_inc} \ \wedge \ h < m - r \rhd \delta \vartriangleleft \mathbf{0} \\
&+ \mathsf{!overflow} \vartriangleleft (t + (m-h)/r) \cdot \mathsf{T}(\mathsf{dl}, t + (m-h)/r, 0, 0) \\
&\qquad \lhd s = \mathsf{mv\_inc} \ \wedge \ m - r \leq h \rhd \delta \vartriangleleft \mathbf{0} \\
&+ \mathsf{stop} \vartriangleleft (t + 10/3) \cdot \mathsf{T}(\mathsf{mv\_inc}, t + 10/3, h - 10(3-r)/3, 0) \\
&\qquad \lhd s = \mathsf{nf\_dec} \ \wedge \ 10(3-r)/3 \leq h \rhd \delta \vartriangleleft \mathbf{0} \\
&+ \mathsf{empty} \vartriangleleft (t + h/(3-r)) \cdot \mathsf{T}(\mathsf{sf\_dry}, t + h/(3-r), 0, 3h/(3-r)) \\
&\qquad \lhd s = \mathsf{nf\_dec} \ \wedge \ h < 10(3-r)/3 \rhd \delta \vartriangleleft \mathbf{0} \\
&+ \mathsf{stop} \vartriangleleft (t + (10-l)/r) \cdot \mathsf{T}(\mathsf{mv\_inc}, t + (10-l)/r, 0, 0) \lhd s = \mathsf{sf\_dry} \rhd \delta \vartriangleleft \mathbf{0} \\
&+ \delta \vartriangleleft t \lhd s = \mathsf{dl} \rhd \delta \vartriangleleft \mathbf{0}
\end{aligned}
$$

This equation is in TLPE format. Let system of equations L contain the above equation only. Then $(\mathsf{T}(\mathsf{mv\_inc}, 0, h, 0), L)$ is the linearized specification of the bottle-filling system.

4 Time-free Abstraction and Analysis

An important notion of timed μCRL processes is *well-timedness*. A term $\mathsf{a}(\overrightarrow{t}) \vartriangleleft t \cdot p$ is *well-timed* if $p \approx t \gg p$. If t is such that $c(t) \approx \mathbf{t}$ implies $p \approx t \gg p$, then $\mathsf{a}(\overrightarrow{t}) \vartriangleleft t \cdot p \lhd c(t) \rhd \delta \vartriangleleft \mathbf{0}$ is also well-timed. Terms $\mathsf{a}(\overrightarrow{t}) \vartriangleleft t$ and $\delta \vartriangleleft t$ are also well-timed. If p and q are well-timed terms, then $p+q$, $\sum_{d:D} p$ and $p \lhd c \rhd \delta \vartriangleleft \mathbf{0}$ are also well-timed terms.

An equation in TPEGNF is well-timed if for all $i \in I$ the terms $\mathsf{a}_i(\overrightarrow{t_i}) \vartriangleleft t_i \cdot p_i \lhd c_i \rhd \delta \vartriangleleft \mathbf{0}$ are well-timed. The linearization method for timed μCRL ensures that the resulting TLPE is well-timed, e.g., in our example, $t \gg \mathsf{T}(s,t,h,l) \approx \mathsf{T}(s,t,h,l)$.

The *time-free abstraction* (cf. [21, Section 4.2]) of well-timed TLPEs can be used for further analysis with methods that are designed for untimed μCRL. For instance, strong bisimilarity of time-free abstractions of two well-timed TLPEs is equivalent to the timed bisimilarity of them. In the initial timed μCRL specification time has a direct influence on the specified behavior, for instance on the interleavings of parallel components (for example $\mathsf{a} \vartriangleleft 1 \parallel \mathsf{b} \vartriangleleft 2 \approx \mathsf{a} \vartriangleleft 1 \cdot \mathsf{b} \vartriangleleft 2$ in timed μCRL). This is why performing the time-free abstraction on the initial specification will not work (because $\mathsf{a}(1) \parallel \mathsf{b}(2) \not\approx \mathsf{a}(1) \cdot \mathsf{b}(2)$ in μCRL). However, after linearization the influence of time on the specified behavior is encoded in the parameters and conditions of resulting TLPE, i.e. time becomes just a conventional data type in untimed μCRL.

Applying time-free abstraction to our example gives us the following μCRL equation:

$$
\begin{aligned}
&\mathsf{T}(s{:}State, t{:}Time, h, l{:}\mathbb{Q}) = \\
&\quad \mathsf{start}(t+1) \cdot \mathsf{T}(\mathsf{nf\_dec}, t+1, h+r, 0) \lhd s = \mathsf{mv\_inc} \ \wedge \ h < m - r \rhd \delta \\
&+ \mathsf{!overflow}(t + (m-h)/r) \cdot \mathsf{T}(\mathsf{dl}, t + (m-h)/r, 0, 0) \\
&\qquad \lhd s = \mathsf{mv\_inc} \ \wedge \ m - r \leq h \rhd \delta \\
&+ \mathsf{stop}(t + 10/3) \cdot \mathsf{T}(\mathsf{mv\_inc}, t + 10/3, h - 10(3-r)/3, 0) \\
&\qquad \lhd s = \mathsf{nf\_dec} \ \wedge \ 10(3-r)/3 \leq h \rhd \delta \\
&+ \mathsf{empty}(t + h/(3-r)) \cdot \mathsf{T}(\mathsf{sf\_dry}, t + h/(3-r), 0, 3h/(3-r)) \\
&\qquad \lhd s = \mathsf{nf\_dec} \ \wedge \ h < 10(3-r)/3 \rhd \delta \\
&+ \mathsf{stop}(t + (10-l)/r) \cdot \mathsf{T}(\mathsf{mv\_inc}, t + (10-l)/r, 0, 0) \lhd s = \mathsf{sf\_dry} \rhd \delta \\
&+ \Delta(t) \lhd s = \mathsf{dl} \rhd \delta
\end{aligned}
$$

Analysis We try to prove some properties of the bottle-filling system. For this we assume that in the initial state $m > r > 0$. It is easy to see that both r and m do not change in T, and

therefore these properties are invariants of T. It is also easy to see that $h \geq 0$ is an invariant of the system.

Having assumed that, we see that if $h \geq m - r$ in the initial state, then the overflow is eminent at time $(m-h)/r$. It is interesting to see what happens if $h < m - r$ in the initial state. To this end, we can see that the following formula is an *invariant* (cf. [7]) of the LPE:

$$r \leq 30/13 \ \wedge \ (s = \mathsf{mv\_inc} \implies h < m - r) \ \wedge \ (s = \mathsf{nf\_dec} \implies h < m)$$

This gives us the fact that if $r \leq 30/13$, then overflow is not reachable provided $h < m - r$ holds in the initial state. In case $r > 30/13$, the process T is equal to

$$\begin{aligned}
&\mathsf{T}(s{:}\mathit{State}, t{:}\mathit{Time}, h, l{:}\mathbb{Q}) = \\
&\quad \mathsf{start}(t+1) \cdot \mathsf{stop}(t + 13/3) \cdot \mathsf{T}(\mathsf{mv_inc}, t + 13/3, h + r - 10(3-r)/3, 0) \\
&\qquad \lhd s = \mathsf{mv_inc} \ \wedge \ h < m - r \rhd \delta \\
&\quad + \mathsf{!overflow}(t + (m-h)/r) \cdot \mathsf{T}(\mathsf{dl}, t + (m-h)/r, 0, 0) \\
&\qquad \lhd s = \mathsf{mv_inc} \ \wedge \ m - r \leq h \rhd \delta \\
&\quad + \Delta(t) \lhd s = \mathsf{dl} \rhd \delta
\end{aligned}$$

This is because $r > 30/13$ and $h \geq 0$ implies that $h + r < 10(3-r)/3$ is always false. It is clear that the value of h increases with every sequence of start, stop actions by a constant, so the overflow is eminent.

The next question is whether the container may become empty. From the previous analysis follows that this can only happen if $r \leq 30/13$ and $h < m - r$ holds in the initial state. If $r < 30/13$, the value of h will decrease with each sequence of start, stop actions by a constant. So, eventually, its value will become smaller than $10(3-r)/3$ and the container will become empty. In case $r = 30/13$, the value of h is constant in state $\mathsf{nf\_dec}$ and equal to the initial value of h plus $10(3-r)/3 = 30/13$, so, the container does not become empty in this case.

5 Conclusions and Future Work

We presented the language of timed μCRL with an example specification. We outlined the method of simplifying transformations based on equational axiomatization, and illustrated it on the example. This transformation allows for a time-free abstraction of the specification, which in turn enables the use of tools and techniques for verification of untimed systems. For proving properties of the presented example we used known invariant [7] techniques.

An interesting direction for future work is in adopting efficient real-time abstraction techniques similar to the regions and zones methods [1] for timed automata. Another interesting approach is to make use of model checking techniques, similar to the ones available for timed automata in tools like UPPAAL [18].

A symbolic model checking approach for untimed μCRL has been recently proposed in [17]. It looks more applicable to the time setting than the explicit model checking [20] of modal mu-calculus formulas. In order to apply any of these methods for timed setting a well-thought extension of modal mu-calculus (or another action-based temporal logic) to real-time is needed.

References

[1] R. Alur. Timed automata. In *Proc. CAV'99*, LNCS 1633, pages 8–22, 1999.

[2] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, August 1999.

[3] J. C. M. Baeten and C. A. Middelburg. *Process Algebra with Timing.* Monographs in TCS. Springer, 2002.

[4] J. C. M. Baeten and W. P. Weijland. *Process Algebra*. Cambridge Tracts in TCS 18. Cambridge University Press, 1990.

[5] J. A. Bergstra, J. Heering, and P. Klint, editors. *Algebraic Specification*. ACM Press, ACM Press Frontier Series, 1989.

[6] J. A. Bergstra and J. W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60(1/3):109–137, 1984.

[7] M. A. Bezem and J. F. Groote. Invariants in process algebra with data. In B. Jonsson and J. Parrow, editors, *Proc. CONCUR'94*, LNCS 836, pages 401–416. Springer, 1994.

[8] S. N. Burris and H. P. Sankappanavar. *A Course in Universal Algebra*. Number 78 in Graduate Texts in Mathematics. Springer-Verlag, 1981.

[9] W. J. Fokkink. *Introduction to Process Algebra*. Texts in TCS. An EATCS Series. Springer, 2000.

[10] S. A. Greibach. A new normal-form theorem for context-free phase structure grammars. *JACM*, 12(1):42–52, 1965.

[11] J. F. Groote. The syntax and semantics of timed μCRL. Report SEN-R9709, CWI, Amsterdam, 1997.

[12] J. F. Groote and S. P. Luttik. Undecidability and completeness results for process algebras with alternative quantification over data. Report SEN-R9806, CWI, Amsterdam, July 1998. Available from `http://www.cwi.nl/~luttik/`.

[13] J. F. Groote and M. A. Reniers. Algebraic process verification. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebra*, chapter 17, pages 1151–1208. Elsevier, 2001.

[14] J. F. Groote and J. J. v. Wamel. Algebraic data types and induction in μCRL. Report P9409, University of Amsterdam, Programming Research Group, 1994.

[15] J. F. Groote and J. J. v. Wamel. Analysis of three hybrid systems in timed μCRL. *SCP*, 39:215–247, 2001.

[16] J. F. Groote and J. J. v. Wamel. The parallel composition of uniform processes with data. *TCS*, 266(1-2):631–652, 2001.

[17] J. F. Groote and T. A. C. Willemse. A checker for modal formulae for processes with data. In F. S. de Boer, M. M. Bonsangue, S. Graf, and W.-P. de Roever, editors, *Proc. FMCO'04*, LNCS 3188, pages 223–239, 2004.

[18] 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.

[19] J. Loeckx, H.-D. Ehrich, and M. Wolf. Algebraic specification of abstract data types. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Vol 5*, chapter 4, pages 217–316. Oxford University Press, 2000.

[20] R. Mateescu and M. Sighireanu. Efficient on-the-fly model-checking for regular alternation-free mu-calculus. *SCP*, 2002.

[21] M. A. Reniers, J. F. Groote, J. J. v. Wamel, and M. B. v. d. Zwaag. Completeness of Timed μCRL. *Fund. Inf.*, 50(3-4):361–402, 2002.

[22] Y. S. Usenko. *Linearization in μCRL*. PhD thesis, Eindhoven University of Technology, December 2002.

[23] T. A. C. Willemse. *Semantics and Verification in Process Algebras with Data and Timing*. PhD thesis, Eindhoven University of Technology, 2003.

A Axioms of Timed μCRL

We assume that

- x, y, z are variables of sort *Proc*; c, c_1, c_2 are variables of sort *Bool*; $d, d^1, d^n, d', \ldots$ are data variables (but d in $\sum_{d:D}$ is not a variable); and t, u, w are variables of sort *Time*.
- b stands for either $\mathsf{a}(\overrightarrow{d})$, or δ;
- $\overrightarrow{d} = \overrightarrow{d'}$ is an abbreviation for $eq(d^1, d'^1) \wedge \cdots \wedge eq(d^n, d'^n)$, where $\overrightarrow{d} = d^1, \ldots, d^n$ and $\overrightarrow{d'} = d'^1, \ldots, d'^n$;
- the axioms where p and q occur are schemata ranging over all terms p and q of sort *Proc*, *including* those in which d occurs freely;
- the axiom (SUM2) is a scheme ranging over all terms r of sort *Proc* in which d *does not* occur freely.

$x + y \approx y + x$ (A1)

$x + (y + z) \approx (x + y) + z$ (A2)

$x + x \approx x$ (A3)

$(x + y) \cdot z \approx x \cdot z + y \cdot z$ (A4)

$(x \cdot y) \cdot z \approx x \cdot (y \cdot z)$ (A5)

$x + \partial_{\mathcal{U}}(x) \approx x$ (A6T)

$\delta + \partial_{\mathcal{U}}(x) \approx \delta$ (A6T′)

$\delta \cdot x \approx \delta$ (A7)

$x \lhd \mathbf{t} \rhd y \approx x$ (Cond1)

$x \lhd \mathbf{f} \rhd y \approx y$ (Cond2)

$x \lhd c \rhd y \approx x \lhd c \rhd \delta \fatsemi \mathbf{0} + y \lhd \neg c \rhd \delta \fatsemi \mathbf{0}$ (Cond3T)

$(x \lhd c_1 \rhd \delta \fatsemi \mathbf{0}) \lhd c_2 \rhd \delta \fatsemi \mathbf{0} \approx (x \lhd c_1 \wedge c_2 \rhd \delta \fatsemi \mathbf{0})$ (Cond4T)

$(x \lhd c_1 \rhd \delta \fatsemi \mathbf{0}) + (x \lhd c_2 \rhd \delta \fatsemi \mathbf{0}) \approx x \lhd c_1 \vee c_2 \rhd \delta \fatsemi \mathbf{0}$ (Cond5T)

$(x \lhd c \rhd \delta \fatsemi \mathbf{0}) \cdot y \approx (x \cdot y) \lhd c \rhd \delta \fatsemi \mathbf{0}$ (Cond6T)

$(x + y) \lhd c \rhd \delta \fatsemi \mathbf{0} \approx x \lhd c \rhd \delta \fatsemi \mathbf{0} + y \lhd c \rhd \delta \fatsemi \mathbf{0}$ (Cond7T)

$(x \lhd c \rhd \delta \fatsemi \mathbf{0}) \between y \approx (x \between y) \lhd c \rhd \delta \fatsemi \mathbf{0}$ (Cond8T)

$(x \lhd c \rhd \delta \fatsemi \mathbf{0}) \mid y \approx (x \mid y) \lhd c \rhd \delta \fatsemi \mathbf{0}$ (Cond9T)

$(x \lhd c \rhd \delta \fatsemi \mathbf{0}) \cdot (y \lhd c \rhd \delta \fatsemi \mathbf{0}) \approx (x \cdot y) \lhd c \rhd \delta \fatsemi \mathbf{0}$ (ScaT)

$p \lhd eq(d, e) \rhd \delta \fatsemi \mathbf{0} \approx p[d := e] \lhd eq(d, e) \rhd \delta \fatsemi \mathbf{0}$ (PET)

$\sum_{d:D} x \approx x$ (SUM1)

$\sum_{e:D} r \approx \sum_{d:D} (r[e := d])$ (SUM2)

$\sum_{d:D} p \approx \sum_{d:D} p + p$ (SUM3)

$\sum_{d:D} (p + q) \approx \sum_{d:D} p + \sum_{d:D} q$ (SUM4)

$\sum_{d:D} (p \cdot x) \approx (\sum_{d:D} p) \cdot x$ (SUM5)

$\sum_{d:D} (p \between x) \approx (\sum_{d:D} p) \between x$ (SUM6)

$\sum_{d:D} (p \mid x) \approx (\sum_{d:D} p) \mid x$ (SUM7)

$\sum_{d:D} (\partial_H(p)) \approx \partial_H(\sum_{d:D} p)$ (SUM8)

$\sum_{d:D} (p \lhd c \rhd \delta \fatsemi \mathbf{0}) \approx (\sum_{d:D} p) \lhd c \rhd \delta \fatsemi \mathbf{0}$ (SUM12T)

$\partial_H(b) \approx b$ if $b = \mathsf{a}(\overrightarrow{d})$ and $\mathsf{a} \notin H$ (D1)

$\partial_H(b) \approx \delta$ otherwise (D2)

$\partial_H(x + y) \approx \partial_H(x) + \partial_H(y)$ (D3)

$\partial_H(x \cdot y) \approx \partial_H(x) \cdot \partial_H(y)$ (D4)

$\partial_H(x \lhd c \rhd \delta \fatsemi \mathbf{0}) \approx \partial_H(x) \lhd c \rhd \delta \fatsemi \mathbf{0}$ (D5T)

$\partial_H(x \fatsemi t) \approx \partial_H(x) \fatsemi t$ (D7)

$x \parallel y \approx (x \between y + y \between x) + x \mid y$ (CM1)

$b \fatsemi t \between y \approx (b \fatsemi t \ll y) \cdot y$ (CM2T)

$(b \fatsemi t \cdot x) \between y \approx (b \fatsemi t \ll y) \cdot ((t \gg x) \parallel y)$ (CM3T)

$(x + y) \between z \approx x \between z + y \between z$ (CM4)

$(b \cdot x) \mid b' \approx (b \mid b') \cdot x$ (CM5)

$(b \cdot x) \mid (b' \cdot y) \approx (b \mid b') \cdot (x \parallel y)$ (CM7)

$(x + y) \mid z \approx x \mid z + y \mid z$ (CM8)

$(x \mid y) \fatsemi t \approx x \fatsemi t \mid y$ (ATA7)

$(x \mid y) \fatsemi t \approx x \mid y \fatsemi t$ (ATA8)

$\mathsf{a}(\overrightarrow{d}) \mid \mathsf{a}'(\overrightarrow{d'}) \approx \gamma(\mathsf{a}, \mathsf{a}')(\overrightarrow{d}) \lhd \overrightarrow{d} = \overrightarrow{d'} \rhd \delta$ if $\gamma(\mathsf{a}, \mathsf{a}')$ is defined (CF1)

$\mathsf{a}(\overrightarrow{d}) \mid \mathsf{a}'(\overrightarrow{d'}) \approx \delta$ otherwise (CF2)

$x \mid y \approx y \mid x$ (SC3)

$(x \between y) \between z \approx x \between (y \parallel z)$ (SC1)

$(x \mid y) \mid z \approx x \mid (y \mid z)$ (SC4)

$x \mid (y \between z) \approx (x \mid y) \between z$ (SC5)

$x \between \delta \approx x \cdot \delta$ (SCD1)

$x \mid \delta \approx \partial_{\mathcal{U}}(x)$ (SCDT2)

$x \fatsemi t \between y \approx (x \between y) \fatsemi t$ (SCT1)

$(x \fatsemi t \between u \ggg y) \lhd u \leq t \rhd \delta \fatsemi \mathbf{0} \approx (x \fatsemi t \between y) \lhd u \leq t \rhd \delta \fatsemi \mathbf{0}$ (SCT2)

$x \approx \sum_{t:Time} x \fatsemi t$ (AT1)

$b \fatsemi t \cdot y \approx b \fatsemi t \cdot t \gg y$ (AT2)

$x \fatsemi t \fatsemi u \approx x \fatsemi t \lhd t = u \rhd \delta \fatsemi \mathbf{0} + \partial_{\mathcal{U}}(x) \fatsemi min(t,u)$ (ATA1′)

$(x + y) \fatsemi t \approx x \fatsemi t + y \fatsemi t$ (ATA2)

$(x \cdot y) \fatsemi t \approx x \fatsemi t \cdot y$ (ATA3)

$(\sum_{d:D} p) \fatsemi t \approx \sum_{d:D} p \fatsemi t$ (ATA4)

$(x \lhd c \rhd \delta \fatsemi \mathbf{0}) \fatsemi t \approx x \fatsemi t \lhd c \rhd \delta \fatsemi \mathbf{0}$ (ATA5′)

$t \gg x \approx t \ggg x + \delta \fatsemi t$ (ATB0)

$t \ggg x \approx \sum_{u:Time} x \fatsemi u \lhd t \leq u \rhd \delta \fatsemi \mathbf{0}$ (ATD0)

$x \ll b \approx x$ (ATC1′)

$x \ll (y + z) \approx x \ll y + x \ll z$ (ATC2)

$x \ll (y \cdot z) \approx x \ll y$ (ATC3)

$x \ll \sum_{d:D} p \approx \sum_{d:D} x \ll p$ (ATC4)

$x \ll (y \lhd c \rhd \delta \fatsemi \mathbf{0}) \approx (x \ll y) \lhd c \rhd \delta \fatsemi \mathbf{0} + x \fatsemi \mathbf{0}$ (ATC5′)

$x \ll (y \between z) \approx x \ll (y \ll z)$ (ATC6)

$x \ll (y \mid z) \approx x \ll (y \ll z)$ (ATC7)

$x \ll (\partial_H(y)) \approx x \ll y$ (ATC8)

$x \ll (y \fatsemi t) \approx \sum_{u:Time} (x \ll y) \fatsemi u \lhd u \leq t \rhd \delta \fatsemi \mathbf{0}$ (ATC11)