A Modal Logic for μCRL

J.F. Groote
S.F.M. van Vlijmen
Utrecht University, Faculty of Philosophy
Heidelberglaan 8, 3584 CS Utrecht, the Netherlands
E-mail: jfg@phil.ruu.nl, vlijmen@phil.ruu.nl

Abstract

The language μCRL allows to specify processes with data and to reason with them in an algebraic vein. This allows to express and reason about global correctness of systems. Sometimes, one only needs to analyse particular properties of μCRL-processes. Modal logics are very convenient to express and verify such properties. Therefore, we define a modal logic for μCRL. It is a branching time modal logic based on actions. It has future and past operators and it allows for reasoning about data, for instance using first order quantification over data variables. It is shown that these modal formulae cannot distinguish between divergence sensitive branching bisimilar processes.

The first author is partly supported by the Netherlands Computer Science Research Foundation (SION) with financial support of the Netherlands Organisation for Scientific Research (NWO).

1 Introduction

μCRL is a language to describe and analyse processes containing data based on an algebraic tradition [10, 11, 12]. The main idea behind application of process algebra is the following. A distributed system $p = p_1 \parallel p_2 \parallel \cdots \parallel p_n$ is provided. A specification of the external behaviour q is defined. Then, abstracting from internal activity in p using $\tau_I \partial_H$, the correctness of the distributed system is expressed using the equation $\tau_I \partial_H(p) = q$. Algebraic process theory offers a whole range of techniques to prove such an equation. For many cases this is sufficient, and beneficial. It stimulates the development of distributed systems with a 'nice' external behaviour.

However, for some other purposes the algebraic approach is not adequate. Sometimes it is only necessary to know one single property of a distributed system, which may be inconvenient or even impossible to express using an equation. Temporal or modal logic turns out to be adequate for such purposes. Therefore, we propose a syntax and a semantics for a modal logic for μCRL.

The question what the form of a modal logic must be is not an easy one to answer. In the literature there are very many different logics. Many are state based [3, 17]. Some are action based [14]. As μCRL is an action based formalism, a modal logic for μCRL is naturally action based. Many are based on linear time [17] and some advocate branching time [3, 14]. Linear time logics reason about a particular run of the protocol. This allows to express fairness properties, such as 'if a process reads an element d in a particular run, then during that

run, it will be delivered'. On the other hand, branching time modal logics allow to express that 'once in the future there is a moment where an a and a b action can be performed'. As we feel both properties should be expressible, both kind of operators are included in the logic. A last feature is the use of backward modalities that occur in some modal logics. When experimenting with the logic it turned out to be very convenient to be able to say for instance that 'when it happens that a process signals that there is no problem, there has never been a problematic situation before that time'. So, backward modalities are also included.

A feature that has not been extensively studied in existing logics (as far as we know) is data (but see [13]). μCRL has especially been designed to conveniently handle data in specifications. It is natural that a modal logic for μCRL is also capable to express properties of data. Therefore, quantifiers over data variables, and an equality predicate have been included in the logic.

This leads to a logic with 28 primitives. This is rather large compared to elementary modal logics. But we think that the current logic is fully adequate for its purpose, which is to express properties over distributed systems in a completely formal, but still convenient way. At the same time the requirement that the logics should be close to existing logics and have a straightforward semantics should be satisfied.

Acknowledgements. We thank Doeko Bosscher and Alex Sellink for their valuable comments. Furthermore, we thank a large number of people for discussions about the particular form of the logic.

2 Overview of the language μCRL

In this section we give a compact introduction to μCRL. It is meant to provide the context for definitions to follow. These definitions are given in such a way that little knowledge of μCRL is necessary. For instance, the syntax of the modal logic is given, based on the signature of a μCRL specification only. The semantics is given, assuming that an operational interpretation of μCRL processes exists. The detailed definitions of the signature and operational behaviour for μCRL can be found in [10]. The remainder of this section is an adapted version of the introduction appearing in [12].

First, we assume the existence of a set $\mathcal{N}$ of *names* that are used to denote sorts, variables, functions, processes and labels of actions. The names in $\mathcal{N}$ are words over an alphabet not containing

$$\bot, +, \|, \between, |, \lhd, \rhd, \cdot, \delta, \tau, \partial, \rho, \Sigma, \surd, \times, \rightarrow, :, =,), (, \}, \{, \text{,}, \text{ a space and a newline.}$$

The space and the newline serve as separators between names and are used for the layout of specifications. The other symbols have special functions. Moreover, $\mathcal{N}$ does not contain the reserved keywords **sort**, **proc**, **var**, **act**, **func**, **comm**, **rew** and **from**.

Data types are specified as the standard abstract data types [8], using sorts, functions and axioms. Sorts are declared using the keyword **sort** and functions are declared using the keyword **func**. Axioms are declared using the keyword **rew**, referring to the possibility to use rewriting technology for evaluation of terms. The variables that are used in the axioms must be declared directly before the axioms. Their scope only extends to the next single **rew** declaration.

As an example we define the Booleans. The Booleans must be included in each μCRL specification.

sort **Bool**
func $\mathsf{t}, \mathsf{f} :\rightarrow$ **Bool**

The following example shows how natural numbers with a zero, a successor, addition and multiplication can be declared.

Example 2.1.

sort **Bool**, $\mathbb{N}$
func t, f : **Bool**
$0 :\rightarrow \mathbb{N}$
$S : \mathbb{N} \rightarrow \mathbb{N}$
$add, times : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$
var $x, y : \mathbb{N}$
rew $add(x, 0) = x$
$add(x, S(y)) = S(add(x, y))$
$times(x, 0) = 0$
$times(x, S(y)) = add(x, times(x, y))$

Processes may contain actions representing elementary activities that can be performed. These actions must be explicitly declared using the keyword **act**. Actions may be parameterised by data. In the following lines an action declaration is displayed.

act a, b, c
$a, d : \mathbb{N}$

Here parameterless actions a, b, c and actions a, d depending on natural numbers are declared. Note that overloading is allowed, as long as this cannot lead to confusion (see [10] for details). In this case the actions a and $a(n)$ (with n of sort $\mathbb{N}$) are different actions.

In μCRL parallel processes communicate via synchronisation of actions. A communication specification, declared using the keyword **comm**, prescribes which actions may synchronise on the level of the labels of actions. For instance, in

comm $in|out = com$

each action $in(t_1, \ldots, t_k)$ can communicate with $out(t'_1, \ldots, t'_m)$ to $com(t_1, \ldots, t_k)$ provided $k = m$ and t_i and t'_i denote the same data element for $i = 1, \ldots, k$.

Processes are declared using the keyword **proc**. An example is

proc $counter(x{:}\mathbb{N}) = p$
$buffer = q$

In the first line a counter is declared. It is a process with one parameter x of sort $\mathbb{N}$. The parameter x may be used in the process term p that specifies its behaviour. In the second line a parameterless process *buffer* is declared. Its behaviour is given by the process term q.

Definition 2.2 *(Process terms).* An expression p is called a *process term* iff p has the following syntax:

$$\begin{array}{rcl} p & ::= & (p+p) \mid (p \cdot p) \mid (p \parallel p) \mid (p \between p) \mid (p \lhd t \rhd p) \mid \sum_{d:D}(p) \mid \\ & & \partial_{\{n_1,\ldots,n_m\}}(p) \mid \tau_{\{n_1,\ldots,n_m\}}(p) \mid \rho_{\{n_1 \to n'_1,\ldots,n_m \to n'_m\}}(p) \mid \\ & & \delta \mid \tau \mid n \mid n(t_1,\ldots,t_m). \end{array}$$

where the n, n_i, n'_i are *names*, the t, t_i stand for data terms, d is a variable and D denotes a sort name.

Most operators stem from ACP [1]. Only the conditional construct $p \lhd t \rhd p$ is taken from [16]. In process terms we omit brackets according to the convention that $\cdot$ binds strongest, the conditional construct binds stronger than the parallel operators which in turn bind stronger than $+$.

We give a short description of the behaviour represented by closed process terms.

- The $+$ denotes the alternative composition. The process $p + q$ has the same behaviour as the argument that performs the first step.
- The $\cdot$ represents the sequential composition operator. The process $p \cdot q$ behaves as p, and in case p terminates, it continues to behave as q.
- The merge (or parallel composition operator) $\parallel$ denotes the interleaving of its arguments, except that actions from both arguments may communicate if explicitly allowed in a communication specification.
- The left merge $\between$ and the communication merge $\mid$ are auxiliary operators, to be used for analytical purposes. The left merge is as the merge, except that the first step of $p \between q$ must originate from p. The communication merge $\mid$ is also as the merge, except that $p|q$ has a communication action between p and q as its first step.
- The *conditional* construct $p \lhd t \rhd q$ is an alternative way to write an **if - then - else**-expression and is introduced by HOARE cs. [16]. The data term t is supposed to be of the standard sort of the Booleans (**Bool**). The process $p \lhd t \rhd q$ behaves as p if the data term t evaluates to true (t) and it behaves as q if t evaluates to false (f).
- The sum operator is used to declare a variable d of a specific sort D for use in a process term p. The scope of the variable d is exactly the process term mentioned in the sum operator. The behaviour associated to $\sum_{d:D}(p)$ is a choice between the instantiations of the process term p with values of the sort of the variable d.
- The encapsulation operator (∂) and the hiding operator (τ) are used to rename the action labels $n_1, \ldots, n_m$ to δ, resp. τ. The renaming operator ρ renames action labels according to the scheme in its first argument.
- The constants δ and τ describe two basic types of behaviour. The constant δ describes the process that cannot do anything, in particular it cannot terminate. The constant τ can be used to represent internal activity that cannot be observed.

- The terms n and $n(t_1, \ldots, t_m)$ represent either process instantiations or actions: n refers to a declared process (or to an action) without parameters and $n(t_1, \ldots, t_m)$ contains the arguments (i.e., the data terms) of the identifier.

A complete μCRL-*specification* consists of an interleaving of sort, function, axiom, action, communication and process declarations.

As an example we give a specification of a data transfer process TR. Data elements of sort D are transferred from in to out.

sort	**Bool**
func	$\mathsf{t}, \mathsf{f} :\rightarrow$ **Bool**
sort	D
func	$d1, d2, d3 :\rightarrow D$
act	$in, out : D$
proc	$TR = \sum_{x:D}(in(x) \cdot out(x) \cdot TR)$

2.1 The signature of a specification

From a specification we can determine its signature, i.e., the sorts, functions, actions and processes it defines. The signature of a specification provides us with enough information to define the structure of modal formulae about it. The following definition tells the form of a signature. The precise recipe of how a specification generates the signature can be found in [10], but is completely straightforward.

Definition 2.3. The *signature* $Sig = (Sort, Fun, Act)$ contains sets of the following form:

- $Sort = \{S_1, \ldots, S_m\}$ where $S_1, \ldots, S_m$ are the sort names. **Bool** $\in Sort$ is the predefined set of booleans.
- Fun is a set containing expressions that are either of the form $n{:} \rightarrow S'$ ($S' \in Sort$), or of the form $n{:}S_1 \times \cdots \times S_l \rightarrow S'$ ($S_1, \ldots, S_l, S' \in Sort$). The elements of Fun represent the many-sorted function symbols. $\mathsf{t}{:} \rightarrow$ **Bool** $\in Fun$ and $\mathsf{f}{:} \rightarrow$ **Bool** $\in Fun$ are predefined constants representing true and false.
- Act is a set with elements of the form n or $n{:}S_1 \times \cdots \times S_l$ ($S_1, \ldots, S_l \in Sort$) representing many-sorted actions.

If $Sig = (Sort, Fun, Act)$ is a signature, then we write $Sig.Sort$ for $Sort$, $Sig.Fun$ for Fun and $Sig.Act$ for Act. It is assumed that actions and process names do not overlap, and that all sorts exist. Overloading is allowed, but the resulting sort of a function must be determined by the name of a function and the sorts and number of its arguments. For a very detailed and precise list of restrictions on signatures, see [10].

In the full definition of a signature for μCRL also communications and processes play a role. We do not need these here, and therefore they are omitted.

Given a signature Sig, a set of variables $\mathcal{V}$ is a set containing elements $\langle x{:}S \rangle$ with x a name and S a sort in $Sig.Sort$. The name x may not overlap with a function, action, process name or sort in Sig. Furthermore, the variable names are not overloaded.

The set of terms $\mathbb{T}_S(Sig, \mathcal{V})$ of sort S is inductively defined, simultaneously over all sorts by

- If $\langle n{:}S\rangle \in \mathcal{V}$ a variable, or $n{:} \rightarrow S \in Sig.Fun$ then $n \in \mathbb{T}_S(Sig, \mathcal{V})$.
- If $f{:}S_1 \times \ldots \times S_l \rightarrow S \in Sig.Fun$ and $t_1 \in \mathbb{T}_{S_1}(Sig, \mathcal{V}), \ldots, t_l \in \mathbb{T}_{S_l}(Sig, \mathcal{V})$, then $f(t_1, \ldots, t_l) \in \mathbb{T}_S(Sig, \mathcal{V})$.

Given a set of variables $\mathcal{V}$ and a signature Sig, the set of actions $Act(Sig, \mathcal{V})$ is defined as:

$$Act(Sig, \mathcal{V}) = \{a(t_1, \ldots, t_l) | a{:}S_1 \times \cdots \times S_l \in Sig.Act \text{ and } t_1 \in \mathbb{T}_{S_1}(Sig, \mathcal{V}), \ldots, t_l \in \mathbb{T}_{S_l}(Sig, \mathcal{V})\}.$$

We write $\mathbb{T}_S(Sig)$ for $\mathbb{T}_S(Sig, \emptyset)$ and $Act(Sig)$ for $Act(Sig, \emptyset)$.

2.2 Semantics of the data types

First we adapt the standard definitions of algebras etc. to μCRL (see e.g. [8] for these definitions). We assume, but do not define it here, that a Sig-algebra satisfies the axioms under the rewrite keyword in some given specification.

Definition 2.4. Let Sig be a signature. A *Sig-algebra* $\mathbb{A}_{Sig}$ is a structure containing

- for each $S \in Sig.Sort$ a non-empty domain $D(\mathbb{A}_{Sig}, S)$,
- for each $n{:} \rightarrow S \in Sig.Fun$ a constant $C(\mathbb{A}_{Sig}, n) \in D(\mathbb{A}_{Sig}, S)$,
- for each $n{:}S_1 \times \cdots \times S_m \rightarrow S \in Sig.Fun$ a function $F(\mathbb{A}_{Sig}, n{:}S_1 \times \cdots \times S_m)$ from $D(\mathbb{A}_{Sig}, S_1) \times \cdots \times D(\mathbb{A}_{Sig}, S_m)$ to $D(\mathbb{A}_{Sig}, S)$.

Definition 2.5. Let Sig be a signature and let $\mathbb{A}_{Sig}$ be a Sig-algebra. We define the interpretation $[\![\cdot]\!]_{\mathbb{A}_{Sig}}$ from terms to the domains of $\mathbb{A}_{Sig}$ as follows:

- if $t \equiv n$, then $[\![t]\!]_{\mathbb{A}_{Sig}} \stackrel{\text{def}}{=} C(\mathbb{A}_{Sig}, n)$,
- if $t \equiv n(t_1, \ldots, t_m)$ for some $m \geq 1$ with $t_i \in \mathbb{T}_{S_i}(Sig)$, then $[\![t]\!]_{\mathbb{A}_{Sig}} \stackrel{\text{def}}{=} F(\mathbb{A}_{Sig}, n{:}S_1 \times \cdots \times S_m)([\![t_1]\!]_{\mathbb{A}_{Sig}}, \ldots, [\![t_m]\!]_{\mathbb{A}_{Sig}})$.

For terms $t_1, t_2 \in \mathbb{T}_S(Sig)$ we write $\mathbb{A}_{Sig} \models t_1 = t_2$ iff $[\![t_1]\!]_{\mathbb{A}_{Sig}} = [\![t_2]\!]_{\mathbb{A}_{Sig}}$. For actions $a(t_1, \ldots, t_l)$ and $b(u_1, \ldots, u_k)$ we write $\mathbb{A}_{Sig} \models a(t_1, \ldots, t_l) = b(u_1, \ldots, u_k)$ iff $a \equiv b$, $k \equiv l$ and $\mathbb{A}_{Sig} \models t_i = u_i$ for all $1 \leq i \leq l$.

Definition 2.6. Let Sig be a signature and $\mathcal{V}$ a set of variables. A *substitution* ζ over Sig and $\mathcal{V}$ is a mapping such that for each $\langle x{:}S\rangle \in \mathcal{V}$ it holds that $\zeta(\langle x{:}S\rangle) \in D(\mathbb{A}_{Sig}, S)$. Substitutions are extended to terms by:

$$\begin{aligned}
&\zeta(x) \stackrel{\text{def}}{=} \zeta(\langle x{:}S\rangle) \quad \text{if } \langle x{:}S\rangle \in \mathcal{V} \text{ for some } \textit{name}\ S,\\
&\zeta(n) \stackrel{\text{def}}{=} C(\mathbb{A}_{Sig}, n) \quad \text{if } n{:} \rightarrow S \in Sig.Fun,\\
&\zeta(n(t_1, \ldots, t_m)) \stackrel{\text{def}}{=} F(\mathbb{A}_{Sig}, n{:}S_1 \times \cdots \times S_m)(\zeta(t_1), \ldots, \zeta(t_m)).
\end{aligned}$$

If $\langle x, S\rangle \in \mathcal{V}$ and $d \in D(\mathbb{A}_{Sig}, S)$, then

$$\zeta[x := d](\langle y, S'\rangle) = \begin{cases} d & \text{if } x = y \text{ and } S = S',\\ \zeta(\langle y, S'\rangle) & \text{otherwise.}\end{cases}$$

Definition 2.7. Let Sig be a signature. A Sig-algebra $\mathbb{A}_{Sig}$ is called *boolean preserving* iff

- it is not the case that $\mathbb{A}_{Sig} \models \mathsf{t} = \mathsf{f}$,
- $|D(\mathbb{A}_{Sig}, \mathbf{Bool})| = 2$, i.e., t and f are exactly the two elements of sort **Bool**.

For μCRL we only consider Sig-algebras that are boolean preserving, and satisfy a specification under consideration [10].

The next definition introduces the notion of a transition system. Each μCRL process generates a transition system. For the definition of the semantics of the modal logic, it is not necessary to know how a particular μCRL specification generates a particular transition system. Therefore, we do not provide this translation, but instead refer to [10].

Definition 2.8. Let Sig be a signature. A transition system $\mathcal{A}_{Sig}$ is a structure $(S, \longrightarrow, s)$ where

- S is a set of *states*; $\surd \in S$,
- $\longrightarrow \subseteq S \times (Act(Sig) \cup \{\tau\}) \times S$ is a *transition relation*,
- $s \in S$ is the *initial state*.

Elements $(s', l, s'') \in \longrightarrow$ are generally written as $s' \stackrel{l}{\longrightarrow} s''$. A *path* σ of a transition system $\mathcal{A}_{Sig}$ is a sequence

$$\sigma = s_0, a_0, \ldots, s_n, a_n, s_{n+1} \ldots$$

with $s_0 = s$, $s_i \in S$, $a_i \in Act(Sig) \cup \{\tau\}$ and $s_i \stackrel{a_i}{\longrightarrow} s_{i+1}$. We write σ^a_k for a_k and σ^s_k for s_k. If the sequence is finite, it ends in $s_n \in S$ and we define $len(\sigma) = n$. Otherwise $len(\sigma) = \infty$. For $k \leq len(\sigma)$ we write $\sigma\restriction_k$ for $s_0, a_0, \ldots, s_{k-1}, a_{k-1}, s_k$.

We say that a path σ extends a path ρ iff $\sigma\restriction_{len(\rho)} = \rho$. A path σ is called a run iff σ is infinite or there is a no path ρ that extends σ.

We extend the notion of transitions to (indexed) runs in the following way. If $k \leq k' \leq k''$, $a \in Act(Sig)$ we write

$$\langle \sigma, k \rangle \stackrel{a}{\Longrightarrow} \langle \sigma, k'' \rangle \text{ iff } \sigma^a_k = \tau, \sigma^a_{k+1} = \tau, \ldots, \sigma^a_{k'} = a, \ldots, \sigma^a_{k''-1} = \tau, \sigma^a_{k''} = \tau$$

and

$$\langle \sigma, k \rangle \Rightarrow \langle \sigma, k' \rangle \text{ iff } \sigma^a_k = \tau,\ \sigma^a_{k+1} = \tau, \ldots, \sigma^a_{k'} = \tau.$$

We say that a state s is *divergent* iff there is an infinite path $s \stackrel{\tau}{\longrightarrow} s_1 \stackrel{\tau}{\longrightarrow} s_2 \stackrel{\tau}{\longrightarrow} \cdots \stackrel{\tau}{\longrightarrow} \cdots$.

3 Syntax of the modal logic for μCRL

In this section we formulate a modal logic, as a mixture of features of Hennessy-Milner Logic [14], Computational Tree Logic [3] and Temporal Logic [17], extended with aspects to handle data.

$\top$ and $\bot$ represent true and false. We have the ordinary propositional connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$. The modal connectives are always viewed with respect to a current run. However, this run is not necessarily fixed; the operators $\exists$ and $\forall$ express properties of branches of the current run. The formula $\exists\phi$ expresses that there is a run that extends the current run from now on and that satisfies ϕ. $\forall\phi$ expresses that every run extending the current run from now on satisfies ϕ.

There are two constants δ and $\surd$ to detect the two different kinds of termination. The constant $\surd$ says that after a finite number of internal steps the terminating state $\surd$ is reached on the current path. The constant δ expresses that a *dead* state will be found on the current path after a finite number of internal steps. A state is dead if it has no outgoing transitions, and if it is not the terminating state $\surd$.

The formula $\Diamond\phi$ expresses that ϕ holds now or will once hold in the current run; when ϕ eventually holds it will not necessarily hold for the rest of the run. The formula $\Box\phi$ expresses that ϕ holds now and will from now on always hold in the current run. The formula $\blacklozenge\phi$ says that ϕ holds now or held once in the past, and $\blacksquare\phi$ expresses that up to and including now ϕ has always been valid.

The formula $@\phi$ expresses that now an action a can be performed on the current path and ϕ must hold in a state reached after a. As we consider τ actions as transparent, this a action may be pre- and succeeded by any number of τ steps.

The formula $\overline{@}\phi$ expresses that if in the current state an a action can be performed along the current path, then ϕ must hold afterwards.

There is also an operator to go one visible action back. This is the $\mathcal{J}$ operator that occurs in the work of [5] and resembles the ones in [15, 6]. We have two operators of this kind: $\overline{\mathcal{J}}\phi$ and $\mathcal{J}\phi$. The first, $\overline{\mathcal{J}}\phi$, holds if ϕ holds, in every reachable state, one visible step back. The second, $\mathcal{J}\phi$, holds if there is a reachable state, one visible step back, in which ϕ holds. Dually, we have also the operators $\overline{\mathcal{N}}$ and $\mathcal{N}$ that look one visible step forward. Furthermore, there are the until operator $\phi\mathcal{U}\psi$ and a since operator $\phi\mathcal{S}\psi$. The until operator says that from now on ϕ holds, until ψ holds (and ψ will once hold). The since operator $\phi\mathcal{S}\psi$ expresses that from the moment ϕ held, ψ must have held, up till now. Also for $\mathcal{U}$ and $\mathcal{S}$ there are two duals. The $\phi\overline{\mathcal{U}}\psi$ operator says that if ψ will hold once in the future, ϕ will have been valid between now and that particular moment. $\phi\overline{\mathcal{S}}\psi$ says that if ψ has been valid, once in the past, then ϕ has since that time once been valid.

In order to reason about data, we have added the possibility to express equations $t_1 = t_2$ and to use quantifiers. For instance, we can say that whenever we read a number over channel a this number is 5:

$$\Box\forall x{:}\mathbb{N}.(\overline{r(x)}\top \rightarrow x = 5).$$

Quantified variables may occur in equations $t_1 = t_2$ and they may occur in the formulas $\overline{n(t_1, \ldots, t_k)}$ and $\overline{n(t_1, \ldots, t_k)}$ where n is an action name. A number of examples are provided in section 4, after the definition of the semantics.

Definition 3.1. Let Sig be a signature and $\mathcal{V}$ be a set of variables as defined in Section 2.1. The set of $\mathcal{MF}_{Sig,\mathcal{V}}$ of *modal formulae* is inductively defined as follows.

- $\top, \bot, \surd, \delta \in \mathcal{MF}_{Sig,\mathcal{V}}$;

- if $t_1, t_2 \in \mathbb{T}_S(Sig, \mathcal{V})$ for some sort $S \in Sig.Sort$, then $t_1 = t_2 \in \mathcal{MF}_{Sig,\mathcal{V}}$;
- if $\phi \in \mathcal{MF}_{Sig,\mathcal{V}}$, then $\neg\phi, \Diamond\phi, \Box\phi, \blacksquare\phi, \blacklozenge\phi, \exists\phi, \forall\phi, \overline{\mathcal{J}}\phi, \mathcal{J}\phi, \overline{\mathcal{N}}\phi, \mathcal{N}\phi \in \mathcal{MF}_{Sig,\mathcal{V}}$;
- if $\phi \in \mathcal{MF}_{Sig,\mathcal{V}[x:S]}$ and x is a name which is not being used for a sort, action, function or a process in Sig and $S \in Sig.Sort$, then $\exists x{:}S.\phi, \forall x{:}S.\phi \in \mathcal{MF}_{Sig,\mathcal{V}}$. The notation $\mathcal{V}[x{:}S]$ is defined as $(\mathcal{V} \setminus \{\langle x{:}S\rangle | S \text{ a name}\}) \cup \langle x{:}S\rangle$.
- if $\phi, \psi \in \mathcal{MF}_{Sig,\mathcal{V}}$, then $\phi \vee \psi, \phi \wedge \psi, \phi \rightarrow \psi, \phi \leftrightarrow \psi, \phi\mathcal{U}\psi, \phi\mathcal{S}\psi, \phi\overline{\mathcal{U}}\psi, \phi\overline{\mathcal{S}}\psi \in \mathcal{MF}_{Sig,\mathcal{V}}$;
- if $\phi \in \mathcal{MF}_{Sig,\mathcal{V}}$ and $a \in Act(Sig, \mathcal{V})$, then $@\phi, \overline{@}\phi \in \mathcal{MF}_{Sig,\mathcal{V}}$.

When omitting brackets, we assume $=$ binds strongest, then all unary operators, then $\mathcal{U}$, $\overline{\mathcal{U}}$, $\mathcal{S}$, $\overline{\mathcal{S}}$, then $\wedge$, then $\vee$, then $\rightarrow$ and $\leftrightarrow$ binds weakest.

4 Semantics of the modal logic for μCRL

Definition 4.1. Let Sig be a signature and $\mathcal{V}$ be a set of variables. Let $\mathbb{A}_{Sig}$ be a boolean preserving Sig-algebra. Let $\mathcal{A}_{Sig}$ be a transition system and let $\sigma \in run(\mathcal{A}_{Sig})$. Let ζ be a substitution and $k \in \mathbb{N}$ a natural number. We define $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi$ for a modal formula $\phi \in \mathcal{MF}_{Sig,\mathcal{V}}$ inductively on the structure of ϕ:

- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \top$ holds;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \surd$ iff $\sigma^s_{k'} = \surd$ and $\langle \sigma, k\rangle \Rightarrow \langle \sigma, k'\rangle$ for some $k' \geq k$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \delta$ iff $len(\sigma) \neq \infty$, $\langle \sigma, k\rangle \Rightarrow \langle \sigma, len(\sigma)\rangle$ and $\sigma^s_{len(\sigma)} \neq \surd$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models t = u$ iff $\mathbb{A}_{Sig} \models \zeta(t) = \zeta(u)$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi \wedge \psi$ iff $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta \models \psi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi \vee \psi$ iff $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi$ or $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \psi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi \rightarrow \psi$ iff
  $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi$ does not hold or $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \psi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \neg\phi$ iff not $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi \leftrightarrow \psi$ iff
  both $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi$ hold, or do not hold;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \mathcal{J}\phi$ iff there is some $0 \leq k' < k$ and an action $a \in Act(Sig)$ such that $\langle \sigma, k'\rangle \stackrel{a}{\Rightarrow} \langle \sigma, k\rangle$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \overline{\mathcal{J}}\phi$ iff for all $0 \leq k' < k$ and action $a \in Act(Sig)$ if $\langle \sigma, k'\rangle \stackrel{a}{\Rightarrow} \langle \sigma, k\rangle$, then $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \mathcal{N}\phi$ iff there is some $k' > k$ and an action $a \in Act(Sig)$ such that $\langle \sigma, k\rangle \stackrel{a}{\Rightarrow} \langle \sigma, k'\rangle$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \phi$;

- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \overline{\mathcal{N}}\phi$ iff for all $k' > k$ and action $a \in Act(Sig)$ if $\langle \sigma, k \rangle \stackrel{a}{\Longrightarrow} \langle \sigma, k' \rangle$, then $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \Diamond\phi$ iff for some $k \leq k' \leq len(\sigma)$ $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \Box\phi$ iff for all $k \leq k' \leq len(\sigma)$ $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \blacklozenge\phi$ iff for some $0 \leq k' \leq k$ $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \blacksquare\phi$ iff for all $0 \leq k' \leq k$ $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \exists\phi$ iff
  there is a run σ' that extends $\sigma\restriction_k$ such that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma', \zeta, k \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \forall\phi$ iff
  for each run σ' that extends $\sigma\restriction_k$ it holds that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma', \zeta, k \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi\mathcal{U}\psi$ iff there is some $k \leq k' \leq len(\sigma)$ such that
  $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \psi$ and for all $k \leq k'' < k'$ $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k'' \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi\overline{\mathcal{U}}\psi$ iff for all $k \leq k' \leq len(\sigma)$ if $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \psi$, then there is a $k \leq k'' < k'$ such that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k'' \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi\mathcal{S}\psi$ iff there is some $0 \leq k' \leq k$ such that
  $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \psi$ and for all $k' < k'' \leq k$ $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k'' \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \phi\overline{\mathcal{S}}\psi$ iff for all $0 \leq k' \leq k$ if $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \psi$, then there is a $k < k'' \leq k'$ such that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k'' \models \psi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \forall x{:}S.\phi$ iff for each element $d \in D(\mathbb{A}_{Sig}, S)$
  $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta[x := d], k \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \exists x{:}S.\phi$ iff for some element $d \in D(\mathbb{A}_{Sig}, S)$
  $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta[x := d], k \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models @\phi$ iff there is some $k' > k$ such that
  $\langle \sigma, k \rangle \stackrel{b}{\Longrightarrow} \langle \sigma, k' \rangle$, $\mathbb{A}_{Sig} \models b = \zeta(a)$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \phi$;
- $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k \models \overline{@}\phi$ iff if there is some $k' > k$ such that $\langle \sigma, k \rangle \stackrel{b}{\Longrightarrow} \langle \sigma, k' \rangle$ and $\mathbb{A}_{Sig} \models b = \zeta(a)$, then $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, k' \models \phi$.

We write $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta \models \phi$ for $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, 0 \models \phi$. We write $\mathcal{A}_{Sig}, \mathbb{A}_{Sig} \models \phi$ iff for every run $\sigma \in run(\mathcal{A}_{Sig})$ and every substitution ζ it is the case that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta \models \phi$. We write $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \zeta, s \models \phi$ iff for every run $\sigma \in run(\mathcal{A}_{Sig})$ with $\sigma_0^s = s$ it holds that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, 0 \models \phi$. We write $\mathcal{A}_{Sig} \models \phi$ iff for every Sig-algebra $\mathbb{A}_{Sig}$ it holds that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig} \models \phi$. We write $\models \phi$ iff for every transition system $\mathcal{A}_{Sig}$ it holds that $\mathcal{A}_{Sig} \models \phi$. Whenever $\mathcal{A}_{Sig}$ and $\mathbb{A}_{Sig}$ are clear from the context, we sometimes drop them.

Example 4.2. The following formula distinguishes between $p = a(b + c)$ and $q = a\,b + a\,c$. Note that the existential quantor is absolutely necessary. Without such a branching time operator p and q cannot be distinguished, i.e., if $\mathcal{A}_p$ and $\mathcal{A}_q$ are the (obvious) transition systems belonging to p and q, respectively, we find:

$\mathcal{A}_p \models \Diamond(\exists ⓑ\top \wedge \exists ⓒ\top)$ and
$\mathcal{A}_q \not\models \Diamond(\exists ⓑ\top \wedge \exists ⓒ\top)$.

Example 4.3. The correctness of an alternating bit protocol, which must repeatedly read a datum $d{:}D$ via channel $r(d)$ and then write it via $s(d)$, can be specified by postulating:

- The protocol cannot do anything else than reading or writing a datum:

$$\Box((\exists d{:}D.\textcircled{r(d)}\top) \vee (\exists d{:}D.\textcircled{s(d)}\top)).$$

- If a datum d is read, the next activity is to deliver it:

$$\Box\forall d{:}D.(\textcircled{r(d)}\top \rightarrow \mathcal{N}\textcircled{s(d)}\top).$$

- If a datum d is written, it must previously have been read:

$$\Box\forall d{:}D.(\textcircled{s(d)}\top \rightarrow \mathcal{J}\textcircled{r(d)}\top).$$

Example 4.4. In [6] it was remarked that modal logics with backward modalities can distinguish between the processes $p = a(\tau\,b + c)$ and $q = a(\tau\,b + c) + a\,c$. The processes p and q are obtained by instantiating Milner's third τ-law, which is valid in weak bisimulation semantics [18]. In our setting the formula $\forall(ⓐⓒ\top \rightarrow ⓐⓒ\mathcal{J}\exists ⓑ\top)$ holds for p, but not for q.

Example 4.5. Consider the following three modal formulae.

$\phi_1 = \Box(ⓐ\top \rightarrow \Diamond ⓑ\top)$,
$\phi_2 = \Box(ⓐ\top \rightarrow \exists\Diamond ⓑ\top)$,
$\phi_3 = \Box(ⓐ\top \rightarrow \Diamond\exists ⓑ\top)$.

Formula ϕ_1 expresses that there are no infinite a-paths. Ultimately, i.e., after finitely many a's, a b must happen. The formula ϕ_2 expresses that after each a there is a possibility to reach a b via some path. The formula ϕ_3 expresses that after an a via the current run, there must once be the direct option to do a b step. But b need not be on the current run.

Example 4.6. The Hennessy-Milner modalities $\langle a\rangle.\phi$ and $[a].\phi$ can be expressed in the logic as follows:

$\langle a\rangle.\phi \equiv \exists ⓐ\phi$,
$[a].\phi \equiv \forall\overline{ⓐ}\phi$.

Example 4.7. Deadlock freedom can be expressed in subtly different ways,

(a) $\forall\Box\mathcal{N}\top$,
(b) $\forall\Box(\mathcal{N}\top \vee \surd)$,
(c) $\forall\Box\neg\delta$.

(a) says that there is always a next visible action, independently of the choices that are being made. (b) says that there is always a next visible action, or the process terminates. (c) expresses that there are always next actions, but these may be internal. In particular divergences (infinite traces of internal actions) are not considered a deadlock in (c), but these are in (a) and (b).

We now provide a number of theorems explaining some of the structure present in the logic. The main consequence of the following theorem is that $\neg$ can be pushed to the end of formulas. So, each formula is equivalent to a formula where $\neg$ only occurs directly in front of a $\surd$, δ or a $t = u$.

Theorem 4.8.

$\models \neg\bot \leftrightarrow \top$ $\quad$ $\models \neg\top \leftrightarrow \bot$ $\quad$ $\models \neg\neg\phi \leftrightarrow \phi$
$\models \neg\Diamond\phi \leftrightarrow \Box\neg\phi$ $\quad$ $\models \neg\Box\phi \leftrightarrow \Diamond\neg\phi$ $\quad$ $\models \neg\blacksquare\phi \leftrightarrow \blacklozenge\neg\phi$
$\models \neg\blacklozenge\phi \leftrightarrow \blacksquare\neg\phi$ $\quad$ $\models \neg\exists\phi \leftrightarrow \forall\neg\phi$ $\quad$ $\models \neg\forall\phi \leftrightarrow \exists\neg\phi$
$\models \neg\mathcal{J}\phi \leftrightarrow \overline{\mathcal{J}}\neg\phi$ $\quad$ $\models \neg\overline{\mathcal{J}}\phi \leftrightarrow \mathcal{J}\neg\phi$ $\quad$ $\models \neg\mathcal{N}\phi \leftrightarrow \overline{\mathcal{N}}\neg\phi$
$\models \neg\overline{\mathcal{N}}\phi \leftrightarrow \mathcal{N}\neg\phi$ $\quad$ $\models \neg\exists x{:}S.\phi \leftrightarrow \forall x{:}S.\neg\phi$ $\quad$ $\models \neg\forall x{:}S.\phi \leftrightarrow \exists x{:}S.\neg\phi$
$\models \neg(\phi \vee \psi) \leftrightarrow (\neg\phi \wedge \neg\psi)$ $\quad$ $\models \neg(\phi \wedge \psi) \leftrightarrow (\neg\phi \vee \neg\psi)$ $\quad$ $\models \neg(\phi \rightarrow \psi) \leftrightarrow (\phi \wedge \neg\psi)$
$\models \neg(\phi \leftrightarrow \psi) \leftrightarrow (\neg\phi \leftrightarrow \psi)$ $\quad$ $\models \neg(\phi\mathcal{U}\psi) \leftrightarrow \neg\phi\overline{\mathcal{U}}\psi$ $\quad$ $\models \neg(\phi\overline{\mathcal{U}}\psi) \leftrightarrow \neg\phi\mathcal{U}\psi$
$\models \neg(\phi\overline{\mathcal{U}}\psi) \leftrightarrow \neg\phi\mathcal{U}\psi$ $\quad$ $\models \neg(\phi\overline{\mathcal{S}}\psi) \leftrightarrow \neg\phi\mathcal{S}\psi$ $\quad$ $\models \neg@\phi \leftrightarrow \overline{@}\neg\phi$
$\models \neg\overline{@}\phi \leftrightarrow @\neg\phi$

Proof. Straightforward, using the semantics. □

The next theorem and corollary provide a functionally complete set of 12 primitives for the logic.

Theorem 4.9.

$\models \bot \leftrightarrow (\delta \wedge \neg\delta)$ $\quad$ $\models \top \leftrightarrow \neg(\delta \wedge \neg\delta)$ $\quad$ $\models \Box\phi \leftrightarrow \neg\Diamond\neg\phi$
$\models \blacksquare\phi \leftrightarrow \neg\blacklozenge\neg\phi$ $\quad$ $\models \forall\phi \leftrightarrow \neg\exists\neg\phi$ $\quad$ $\models \overline{\mathcal{J}}\phi \leftrightarrow \neg\mathcal{J}\neg\phi$
$\models \overline{\mathcal{N}}\phi \leftrightarrow \neg\mathcal{N}\neg\phi$ $\quad$ $\models \forall x{:}S.\phi \leftrightarrow \neg\exists x{:}S.\neg\phi$ $\quad$ $\models \phi \vee \psi \leftrightarrow \neg(\neg\phi \wedge \neg\psi)$
$\models (\phi \rightarrow \psi) \leftrightarrow \neg(\phi \wedge \neg\psi)$ $\quad$ $\models \Diamond\phi \leftrightarrow \top\mathcal{U}\phi$ $\quad$ $\models \blacklozenge\phi \leftrightarrow \top\mathcal{S}\phi$
$\models \phi\overline{\mathcal{U}}\psi \leftrightarrow \neg(\neg\phi\mathcal{U}\psi)$ $\quad$ $\models \phi\overline{\mathcal{S}}\psi \leftrightarrow \neg(\neg\phi\mathcal{S}\psi)$ $\quad$ $\models \overline{@}\phi \leftrightarrow \neg@\neg\phi$
$\models (\psi \leftrightarrow \psi) \leftrightarrow (\phi \wedge \psi) \vee (\neg\phi \wedge \neg\psi)$

Proof. Straightforward. □

Corollary 4.10. *$\surd$, δ, $t = u$, $\neg$, $\wedge$, $\exists$, $\mathcal{J}$, $\mathcal{N}$, $\exists x{:}S.$, $\mathcal{U}$, $\mathcal{S}$ and $@$ are a functionally complete set of primitives.*

Theorem 4.11.

$\models \forall\top \leftrightarrow \top$	$\models \forall\bot \leftrightarrow \bot$	$\models \forall t = u \leftrightarrow t = u$
$\models \exists\top \leftrightarrow \top$	$\models \exists\bot \leftrightarrow \bot$	$\models \exists t = u \leftrightarrow t = u$
$\models \forall\phi \rightarrow \exists\phi$	$\models \exists\phi \rightarrow \forall\exists\phi$	
$\models \exists\forall\phi \leftrightarrow \forall\phi$	$\models \forall\forall\phi \leftrightarrow \forall\phi$	$\models \exists\exists\phi \leftrightarrow \exists\phi$
$\models \mathcal{J}\forall\phi \rightarrow \forall\mathcal{J}\phi$	$\models \exists\mathcal{J}\phi \rightarrow \mathcal{J}\exists\phi$	
$\models \forall\mathcal{N}\phi \rightarrow \mathcal{N}\forall\phi$	$\models \mathcal{N}\exists\phi \rightarrow \exists\mathcal{N}\phi$	

Proof. Straightforward using the semantics. □

Theorem 4.12.

$\models \forall\exists\phi \rightarrow \exists\phi$	$\models \forall\mathcal{J}\phi \rightarrow \mathcal{J}\exists\phi$	$\models \mathcal{N}\forall\phi \rightarrow \exists\mathcal{N}\phi$
$\models \exists\overline{\mathcal{J}}\phi \rightarrow \overline{\mathcal{J}}\exists\phi$	$\models \forall\overline{\mathcal{J}}\phi \rightarrow \overline{\mathcal{J}}\exists\phi$	$\models \overline{\mathcal{J}}\forall\phi \rightarrow \forall\overline{\mathcal{J}}\phi$
$\models \overline{\mathcal{N}}\exists\phi \rightarrow \exists\overline{\mathcal{N}}\phi$	$\models \overline{\mathcal{N}}\forall\phi \rightarrow \exists\overline{\mathcal{N}}\phi$	$\models \forall\overline{\mathcal{N}}\phi \rightarrow \overline{\mathcal{N}}\forall\phi$

Proof. Using Theorem 4.9 and 4.11. □

5 Divergence sensitive branching bisimulation

We have seen that the modal formulae can detect divergences (Example 4.5) and distinguish between weakly bisimilar processes (Example 4.4). However, it would be unpleasant if there would be formulae that could distinguish between processes that are generally considered equivalent, such as p and $p + p$ (see [1, 18]). In this section we show that this is not the case. Actually, we show that we cannot distinguish between processes that are divergence sensitive branching bisimilar. This has some pleasant consequences. See Remark 5.6. The divergence sensitive version of branching bisimularity seems to be new. However, divergence sensitive stuttering equivalence is introduced in [7].

The results in this section are inspired by [7] where it was shown that weak bisimulation where backward steps must also be mimicked is exactly branching bisimulation. Moreover, it was shown that until operators are also connected with branching bisimulation. Inspection of the proofs below indeed reveals that the properties that distinguish branching bisimulation from, say, weak bisimulation are especially related to the characteristics of the operators $\mathcal{U}$ and $\mathcal{J}$. Note again that the notion of branching bisimulation as presented here, differs slightly from those presented in [9, 7].

Definition 5.1 *(Divergence sensitive branching bisimulation).* Let Sig be a signature and let $\mathbb{A}_{Sig}$ be a boolean preserving Sig-algebra and let $\mathcal{A}_{Sig} = (S, \rightarrow, s)$ be a transition system. A relation $R \subseteq S \times S$ is called a *divergence sensitive branching bisimulation* iff R is symmetric and for all pairs $p, q \in S$, with pRq it holds that

1. if p is divergent, then q is divergent.
2. if $p \equiv \surd$ then $q \equiv \surd$.

3. if $p \xrightarrow{a} p'$, then
   - either $a \equiv \tau$ and $p'Rq$,
   - or there are $q_0, q_1, \ldots, q_m, \ldots, q_n \in S$ such that $q = q_0 \xrightarrow{\tau} q_1 \xrightarrow{\tau} \cdots q_m \xrightarrow{a} q_{m+1} \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_n$ for all $0 \leq i \leq m$ it is the case that pRq_i and for all $m+1 \leq j \leq n$ it holds that $p'Rq_j$.

We write $p \leftrightarrows_{dsb} q$ iff there is a divergence sensitive branching bisimulation R such that pRq.

The following definition provides an auxiliary notion to prove Theorem 5.5.

Definition 5.2 *(Path divergence sensitive branching bisimulation).* Let Sig be a signature, $\mathbb{A}_{Sig}$ a boolean preserving Sig-algebra and let $\mathcal{A}_{Sig} = (S, \rightarrow, s)$ be a transition system. Two paths σ, ρ are path divergence sensitive branching bisimilar iff there is a divergence sensitive branching bisimulation relation R and a relation $\mathcal{R} \subseteq \mathbb{N} \times \mathbb{N}$ such that

- $0\mathcal{R}0$;
- for all $i, j \in \mathbb{N}$ $i\mathcal{R}j$ implies $\sigma_i^s R \rho_j^s$;
- if $n\mathcal{R}m$ and $\langle \sigma, n \rangle \xrightarrow{a} \langle \sigma, n+1 \rangle$, then
  - either $a \equiv \tau$ and $n+1\mathcal{R}m$,
  - or $\langle \rho, m \rangle \Rightarrow \langle \rho, m+k \rangle \xrightarrow{a} \langle \rho, m+k+1 \rangle \Rightarrow \langle \rho, m+k+l \rangle$, for all $0 \leq i \leq k$ it holds that $n\mathcal{R}m+i$ and for all $0 < j \leq l$ it is the case that $n+1\mathcal{R}m+k+j$.
- vice versa.
- if $n > 0$, $n\mathcal{R}m$ and $\langle \sigma, n-1 \rangle \xrightarrow{a} \langle \sigma, n \rangle$, then
  - either $a \equiv \tau$ and $n-1\mathcal{R}m$,
  - or $\langle \rho, m-k-l \rangle \Rightarrow \langle \rho, m-k \rangle \xrightarrow{a} \langle \rho, m-k+1 \rangle \Rightarrow \langle \rho, m \rangle$, for all $0 \leq i \leq l$ it holds that $n-1\mathcal{R}m-k-i$ and for all $0 \leq j < k$ it is the case that $n\mathcal{R}m-j$.
- vice versa.

Lemma 5.3. *Let Sig be a signature, let $\mathbb{A}_{Sig}$ be a boolean preserving Sig algebra and let $\mathcal{A}_{Sig} = (S, \rightarrow, s)$ be a transition system. Let σ be a path such that $\sigma_0^s \leftrightarrows_{dsb} s$. Then there is a path ρ such that $\rho_0^s \equiv s$ and σ, ρ are path divergence sensitive branching bisimilar.*

Lemma 5.4. *Let Sig be a signature, let $\mathbb{A}_{Sig}$ be a boolean preserving Sig-algebra and let $\mathcal{A}_{Sig} = (S, \rightarrow, s)$ be a transition system. Let σ and ρ be path divergence sensitive branching bisimilar via relation $\mathcal{R}$ such that $n\mathcal{R}m$. Then for all $\phi \in \mathcal{MF}_{Sig,\mathcal{V}}$ and all ζ:*

$$\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n \models \phi \quad \Leftrightarrow \quad \mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, m \models \phi$$

Proof. We prove this lemma with induction on ϕ (using corollary 4.10).

- Assume $\phi \equiv \surd$. Assume also that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n \models \surd$. This means that $\langle \sigma, n \rangle \Rightarrow \langle \sigma, n' \rangle$ and $\sigma^s_{n'} \equiv \surd$. Hence, $\langle \rho, m \rangle \Rightarrow \langle \rho, m' \rangle$ and $n' \mathcal{R} m'$. So, $\rho^s_{m'} \equiv \surd$. Hence, $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n \models \surd$.

- Assume $\phi \equiv \delta$. Assume also that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n \models \delta$. This means that $len(\sigma) \neq \infty$, $\langle \sigma, n \rangle \Rightarrow \langle \sigma, len(\sigma) \rangle$ and $\sigma^s_{len(\sigma)} \not\equiv \surd$. Hence, $\langle \rho, m \rangle \Rightarrow \langle \rho, m' \rangle$ and $len(\sigma) \mathcal{R} m'$.

  Now suppose $\langle \rho, m' \rangle \xrightarrow{a}$ for some $a \not\equiv \tau$. This cannot be mimicked by $\langle \sigma, len(\sigma) \rangle$. Hence, $len(\sigma) \mathcal{R} m'$ cannot hold. Contradiction.

  Suppose $\rho^s_{m'}$ is divergent. This also contradicts $len(\sigma) \mathcal{R} m'$. Hence, there must be some $m' \leq m'' < \infty$ such that $len(\rho) = m''$ and $\langle \rho, m' \rangle \Rightarrow \langle \rho, len(\rho) \rangle$. So, $\langle \rho, m \rangle \Rightarrow \langle \rho, len(\rho) \rangle$ and $len(\sigma) \mathcal{R} len(\rho)$. Clearly, $\rho^s_{len(\rho)} \not\equiv \surd$ and therefore, $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, m \models \delta$.

- The cases $\phi \equiv t = u$, $\phi \equiv \neg\psi$, $\phi \equiv \psi_1 \wedge \psi_2$ are trivial.

- Assume $\phi \equiv \exists \psi$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n \models \exists \psi$. Hence, there is a σ' extending $\sigma \restriction_n$ such that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma', \zeta, n \models \psi$. Clearly, there is an extension ρ' of $\rho \restriction_{m'}$ that is divergence sensitive branching bisimilar to σ'. Hence, using the induction hypothesis, $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho', \zeta, m \models \psi$. Clearly, $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, m \models \exists \psi$.

- Assume $\phi \equiv \mathcal{J}\psi$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n \models \mathcal{J}\psi$. This means that there is some $0 \leq n' < n$ and an action $a \in Act(Sig)$ such that $\langle \sigma, n' \rangle \stackrel{a}{\Rightarrow} \langle \sigma, n \rangle$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n' \models \psi$. Hence, there is some $m' \leq m$ such that $\langle \rho, m' \rangle \stackrel{a}{\Rightarrow} \langle \rho, m \rangle$ and $n' \mathcal{R} m'$. So, via the induction hypothesis $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, m' \models \psi$ and hence $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, m \models \mathcal{J}\phi$.

- Assume $\phi \equiv \mathcal{N}\psi$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n \models \mathcal{N}\psi$. This means that there is some $n' > n$ and an action $a \in Act(Sig)$ such that $\langle \sigma, n \rangle \stackrel{a}{\Rightarrow} \langle \sigma, n' \rangle$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n' \models \psi$. Hence, There is some $m' > m$ such that $\langle \rho, m \rangle \stackrel{a}{\Rightarrow} \langle \rho, m' \rangle$ and $n' \mathcal{R} m'$. By induction $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, m' \models \psi$. Hence, $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, m \models \phi$.

- Assume $\phi \equiv \exists x{:}S.\psi$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n \models \phi$. This means that for some element $v \in D(\mathbb{A}_{Sig}, S)$ it holds that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta[x{:=}v], n \models \psi$. Hence, it follows by induction $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta[x{:=}v], m \models \psi$. So, $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, m \models \phi$.

- Assume $\phi \equiv \psi_1 \mathcal{U} \psi_2$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n \models \phi$. Hence, there is an $n \leq n' \leq len(\sigma)$ such that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n' \models \psi_2$ and for all $n \leq n'' < n'$ $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n'' \models \psi_1$. There is some sequence tr of actions such that $\langle \sigma, n \rangle \stackrel{tr}{\Rightarrow} \langle \sigma, n' \rangle$. Hence, there is an m' such that $\langle \rho, m \rangle \stackrel{tr}{\Rightarrow} \langle \rho, m' \rangle$ and $n' \mathcal{R} m'$. So, $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, m' \models \psi_2$. Moreover, for every $m \leq i < m'$ there is some $n \leq j < n'$ such that $j \mathcal{R} i$. Therefore for each $m \leq i < m'$ $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, i \models \psi_1$. So, $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, m \models \phi$.

- Assume $\phi \equiv \psi_1 \mathcal{S} \psi_2$. This case is symmetric to the previous one.

- Assume $\phi \equiv @.\psi$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n \models \phi$. This means that there is some $n' > n$ such that $\langle \sigma, n \rangle \stackrel{b}{\Rightarrow} \langle \sigma, n' \rangle$, $\mathbb{A}_{Sig} \models b = \zeta(a)$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, n' \models \psi$. Hence, there is some $m' > m$ such that $\langle \rho, m \rangle \stackrel{b}{\Rightarrow} \langle \rho, m' \rangle$ and $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, m' \models \psi$. Clearly, $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, m \models @.\psi$.

□

Theorem 5.5. *Let Sig be a signature, let $\mathbb{A}_{Sig}$ be a boolean preserving Sig algebra and $\mathcal{A}_{Sig} = (S, \rightarrow, s)$ be a transition system. Let s and t be divergence sensitive branching bisimilar. Then*

$$\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \zeta, s \models \phi \quad \Leftrightarrow \quad \mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \zeta, t \models \phi$$

Proof. Due to symmetry, we only show '$\Rightarrow$'. Suppose $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \zeta, s \models \phi$. We must show that for all ρ, with $\rho^s_0 = t$, it holds that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, 0 \models \phi$. Fix ρ. By lemma 5.3, there is a σ with $\sigma^s_0 = s$ such that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \sigma, \zeta, 0 \models \phi$. By lemma 5.4 it follows that $\mathcal{A}_{Sig}, \mathbb{A}_{Sig}, \rho, \zeta, 0 \models \phi$. □

Remark 5.6. Theorem 5.5 has an important consequence. Suppose the modal formula ϕ must be checked for a parallel system M. Using equations and principles valid in divergence sensitive branching bisimulation semantics M can be rewritten to the following linear form [2]:

$$\begin{aligned} X(\vec{d}) = & \textstyle\sum_{i \in I} \sum_{\vec{e}:\vec{D_i}} a_i(f_i(\vec{d},\vec{e}))\, X(g_i(\vec{d},\vec{e})) \triangleleft c_i(\vec{d},\vec{e}) \triangleright \delta + \\ & \textstyle\sum_{j \in J} \sum_{\vec{e}:\vec{D_j}} a_j(f_j(\vec{d},\vec{e})) \triangleleft c_j(\vec{d},\vec{e}) \triangleright \delta. \end{aligned}$$

Here $\sum_{i \in I}$ is a meta notation representing iterated summation. It is rather straightforward to transform μCRL specifications to this linear form [4]. Because such a linear form is achieved using equations and principles valid in divergence sensitive bisimulation semantics, the linear form maintains the validity of modal formulae. Checking ϕ wrt. this linear form is – as we expect – substantially easier than checking ϕ wrt. M.

Remark 5.7. An important and useful concept are distinguishing formulae. If two transition systems are not equivalent, then there is a modal formula that indicates a reason why; the formula is valid in one transition system and invalid in the other. For strong and weak bisimulation distinguishing formulae are often formulated in Hennessy-Milner Logic [14]. For branching bisimulation a variant of Hennessy-Milner Logic with until operator is capable to express the distinguishing formulae [7]. We think that the modal logic presented here is capable of expressing the distinguishing formulae for divergence sensitive branching bisimulation inequivalent transition systems expressed in μCRL. However, proving such a theorem may be difficult as μCRL transition systems contain infinite branching. All results about distinguishing formulae mentioned above assume finitely branching transition systems.

References

[1] J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.

[2] M.A. Bezem and J.F. Groote. Invariants in process algebra with data. Technical Report 98, Logic Group Preprint Series, Utrecht University, 1993.

[3] M.C. Browne, E.M. Clarke, and O. Grümberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59(1,2):115–131, 1988.

[4] J.J. Brunekreef. Process specification in a UNITY format. Report P9329, Programming Research Group, University of Amsterdam, 1993.

[5] J.J. Brunekreef, J.P. Katoen, R.L.C. Koymans, and S. Mauw. Design and analysis of dynamic leader election protocols in broadcast networks. Technical Report P9324, Programming Research Group, University of Amsterdam, 1993.

[6] R. De Nicola, U. Montanari, and F.W. Vaandrager. Back and forth bisimulations. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*, pages 152–165. Springer-Verlag, 1990.

[7] R. De Nicola and F.W. Vaandrager. Three logics for branching bisimulation (extended abstract). In *Proceedings* 5^{th} *Annual Symposium on Logic in Computer Science,* Philadelphia, USA, pages 118–129. IEEE Computer Society Press, 1990. Full version available as Rapporto di Ricerca SI-92/07, Dipartimento di Scienze dell'Informazione, Università degli Studi di Roma "La Sapienza", November 1992.

[8] H. Ehrig and B. Mahr. *Fundamentals of algebraic specifications I*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1985.

[9] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In G.X. Ritter, editor, *Information Processing 89*, pages 613–618. North-Holland, 1989. Full version available as Report CS-R9120, CWI, Amsterdam, 1991.

[10] J.F. Groote and A. Ponse. The syntax and semantics of μCRL. Report CS-R9076, CWI, Amsterdam, 1990.

[11] J.F. Groote and A. Ponse. Proof theory for μCRL. Report CS-R9138, CWI, Amsterdam, 1991.

[12] J.F. Groote and A. Ponse. Proof theory for μCRL: a language for processes with data. In D.J. Andrews, J.F. Groote, and C.A. Middelburg, editors, *Proceedings of the International Workshop on Semantics of Specification Languages*, pages 232–251. Workshops in Computing, Springer Verlag, 1994.

[13] M. Hennessy and A. Ingólfsdóttir. A theory of communicating processes with value-passing. In M. Paterson, editor, *Proceedings* 17^{th} *ICALP,* Warwick, volume 443 of *Lecture Notes in Computer Science*, pages 209–219. Springer-Verlag, July 1990.

[14] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[15] M. Hennessy and C. Stirling. The power of the future perfect in program logics. *Information and Computation*, 67:23–52, 1985.

[16] C.A.R. Hoare, I.J. Hayes, He Jifeng, C.C. Morgan, A.W. Roscoe, J.W. Sanders, I.H. Sorensen, J.M. Spivey, and B.A. Sufrin. Laws of programming. *Communications of the ACM*, 30(8):672–686, August 1987.

[17] R.L.C. Koymans. *Specifying Message Passing and Time-Critical Systems with Temporal Logic*. PhD thesis, Technische Universiteit Eindhoven, 1989.

[18] R. Milner. *Communication and Concurrency*. Prentice-Hall International, Englewood Cliffs, 1989.