# Modal $\mu$-calculus (version 1.1)

Jeroen J.A. Keiren
VU University Amsterdam

December 13, 2013

In this document we will discuss methods for describing properties of systems. One could describe the high-level behaviour of a system, and then verify whether the behaviour of the implementation adheres to this specification using techniques like branching bisimulation. In practice, the high-level behaviour of a system is often quite complex, and it is hard to establish a concise, evidently correct specification. Furthermore, in earlier stages of the development, the behaviour of a system is often not yet known completely. We therefore study a language that can be used to express properties *outside* the behavioural description.

Typically, a property makes a statement about part of the behaviour of the system, and the conjunction of all these properties gives a verdict of the correctness of the system. Properties can be expressed in a large number of different logics; the most popular of which are linear temporal logic (LTL), and computation tree logic (CTL). The logic CTL$^*$ encompasses both LTL and CTL. The most expressive logic that is commonly used for expressing properties is the modal $\mu$-calculus, in which a proper superset of CTL$^*$ can be expressed.

This document was written as course material for the course *System Validation* at TU Delft. In this course, we use the mCRL2 toolset [GM13] for verifying our systems. In mCRL2 the modal $\mu$-calculus is extended to allow reasoning with data. This extension is called the first-order $\mu$-calculus. We incrementally build this $\mu$-calculus through simpler logics.

## 1 Labelled transition systems

Throughout this document we assume that the systems we verify are described as a labelled transition system (LTS). This is a system description consisting of states and edge-labelled transitions.

**Definition 1.1** (Labelled transition system (LTS))**.** A labelled transition system is a five-tuple $(S, Act, \rightarrow, s_0, T)$, where

- $S$ is a set of states,
- $Act$ is a set of (multi-)actions,
- $\rightarrow \subseteq S \times Act \times S$ is the transition relation,
- $s_0 \in S$ is the initial state, and

- $T \subseteq S$ is the set of terminal states.

We generally write $t \xrightarrow{a} t'$ instead of $(t, a, t') \in \rightarrow$. If every state in an LTS has a finite number of outgoing transitions we say that it is *finitely branching*. LTSs can be compared using, for example, strong bisimulation. The definition of strong bisimulation is recalled here to later illustrate the correspondence with modal logics.

**Definition 1.2** (Strong bisimulation)**.** A relation $R \subseteq S \times S$ on a labelled transition system $(S, Act, \rightarrow, s_0, T)$ is a strong bisimulation relation if, whenever $s \; R \; t$ then:

- if $s \rightarrow s'$ then there exists a $t'$ s.t. $t \rightarrow t'$ and $s' \; R \; t'$;
- if $t \rightarrow t'$ then there exists an $s'$ s.t. $s \rightarrow s'$ and $s' \; R \; t'$;
- if $s \in T$ then $t \in T$;
- if $t \in T$ then $s \in T$.

We say that $s$ is *strongly bisimilar* to $t$, denoted $s \underline{\leftrightarrow} t$, if there exists a strong bisimulation relation $R$ such that $s \; R \; t$.

# 2 Hennessy-Milner Logic

The modal logic underlying our $\mu$-calculus is *Hennessy-Milner logic* [HM85]. All formulas in this logic express properties about *states*.

**Definition 2.1** (Hennessy-Milner logic)**.** A *Hennessy-Milner logic (HML)* formula over a set of actions $Act$ is defined using the following syntax:

$$\varphi, \psi ::= \mathit{true} \mid \mathit{false} \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \implies \psi \mid \langle a \rangle \varphi \mid [a]\varphi$$

where $a \in Act$.

Here *true* is the formula that holds universally, *false* holds nowhere, negation, conjunction, disjunction and implication have their usual meaning, and $\langle a \rangle \varphi$ and $[a]\varphi$ are the modalities that denote that there is an $a$-successor in which $\varphi$ holds, and that for all $a$-successors $\varphi$ holds, respectively.

An HML formula either holds in a state, or does not hold in that state. We say that an LTS satisfies an HML formula if that formula holds in its initial state. More formally, given a labelled transition system $L = (S, Act, \rightarrow, s, T)$, a state $s$, and an HML formulae $\varphi$ we can say that $s$ *satisfies* $\varphi$, denoted $L, s \vDash \varphi$, if:

- $L, s \vDash \mathit{true}$ holds in all states,
- $L, s \vDash \mathit{false}$ holds in no state,
- $L, s \vDash \neg\varphi$ iff not $L, s \vDash \varphi$,
- $L, s \vDash \varphi \wedge \psi$ iff $L, s \vDash \varphi$ and $L, s \vDash \psi$,
- $L, s \vDash \varphi \vee \psi$ iff either $L, s \vDash \varphi$ or $L, s \vDash \psi$,

- $L, s \vDash \varphi \implies \psi$ iff $L, s \vDash \varphi$ implies $L, s \vDash \psi$,
- $L, s \vDash \langle a \rangle \varphi$ iff there is a $t \in S$ s.t. $s \xrightarrow{a} t$ and $L, t \vDash \varphi$, and
- $L, s \vDash [a]\varphi$ iff for all $t \in S$ if $s \xrightarrow{a} t$ then $L, t \vDash \varphi$.

Combining the connectives in HML, more interesting properties can be formulated. We give a few examples below.

**Example 2.2.** The following are properties that can be expressed in HML:

- $\langle a \rangle \langle b \rangle \mathit{true}$ it is possible to do an $a$ step followed by a $b$ step, alternatively, on can say an $a$ step is possible to a state in which $b$ is enabled;
- $\langle a \rangle (\langle b \rangle \mathit{true} \vee \langle c \rangle \mathit{true})$ an $a$ step is possible to a state in which either a $b$ or a $c$ action is enabled;
- $[a]\langle b \rangle \mathit{true}$ after every $a$ step a $b$ is enabled (although an a need not be enabled in the intial state);
- $[a]\mathit{false}$ whenever an $a$ is done, a state is reached in which *false* is valid. This only holds whenever no $a$ can be done.

▲

An alternative way to view the semantics of an HML formula is to think of it as a set of states that satisfies the given formula. Below we give the so called *denotational semantics* of HML, denoted as $[\![\varphi]\!]_L \subseteq S$. Intuitively, the correspondence to the previous definition is that $s \in [\![\varphi]\!]_L$ iff $L, s \vDash \varphi$.

**Definition 2.3** (HML denotational semantics)**.** Let $L$ be an LTS and $\varphi$ an HML formula. We define $[\![\varphi]\!] \subseteq S$ inductively as follows:

$$
\begin{aligned}
[\![\mathit{true}]\!]_L &= S \\
[\![\mathit{false}]\!]_L &= \emptyset \\
[\![\neg\varphi]\!]_L &= S \setminus [\![\varphi]\!]_L \\
[\![\varphi \wedge \psi]\!]_L &= [\![\varphi]\!]_L \cap [\![\psi]\!]_L \\
[\![\varphi \vee \psi]\!]_L &= [\![\varphi]\!]_L \cup [\![\psi]\!]_L \\
[\![\varphi \implies \psi]\!]_L &= (S \setminus [\![\varphi]\!]_L) \cup [\![\psi]\!]_L \\
[\![\langle a \rangle \varphi]\!]_L &= \{s \in S \mid \exists_{t \in S} s \xrightarrow{a} t \wedge t \in [\![\varphi]\!]_L\} \\
[\![[a]\varphi]\!]_L &= \{s \in S \mid \forall_{t \in S} s \xrightarrow{a} t \implies t \in [\![\varphi]\!]_L\}
\end{aligned}
$$

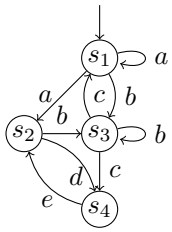
For simplifying formulas the standard rules for propositional logic as well as the following rules for HML are valid:

$$
\begin{aligned}
\varphi \implies \psi &= \neg\varphi \vee \psi \\
\langle a \rangle \varphi &= \neg[a]\neg\varphi \\
\langle a \rangle \mathit{false} &= \mathit{false} \\
[a]\mathit{true} &= \mathit{true} \\
\langle a \rangle (\varphi \vee \psi) &= \langle a \rangle \varphi \vee \langle a \rangle \psi \\
[a](\varphi \wedge \psi) &= [a]\varphi \wedge [a]\psi
\end{aligned}
$$

There is also some interplay between both modalities; if there is an $a$ transition to a state in which $\varphi$ holds, and after all $a$ transitions $\psi$ holds, then there is an $a$ transition to a state in which $\varphi$ and $\psi$ hold. This is formalised by the following rule:

$$\langle a\rangle\varphi \wedge [a]\psi \implies \langle a\rangle(\varphi \wedge \psi).$$

**Exercise 2.4.** Consider the following LTS.




For each of the following HML properties, decide whether it holds in the initial state.

- $\langle a\rangle true$
- $\langle c\rangle true$
- $[c]false$
- $[a]\langle b\rangle true$
- $[a]\langle d\rangle true$
- $[a](\langle d\rangle true \vee \langle a\rangle\langle d\rangle true)$
- $[a][b]\langle c\rangle true$
- $[a][b]([b]false \implies \langle d\rangle true)$
- $[b][c]([b]false \implies \langle e\rangle true)$

**Exercise 2.5.** Give an LTS that satisfies the following three formulae:

- $\langle a\rangle(\langle b\rangle true \wedge \langle c\rangle true)$
- $[a][b](\langle d\rangle true \wedge [d]\langle e\rangle true)$
- $\langle a\rangle\langle c\rangle\langle d\rangle true$

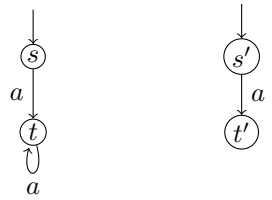
Can this be done in three states?

When Hennessy and Milner introduced their modal logic, they showed that it could exactly distinguish bisimilarity on finite LTSs in the sense that bisimilar states satisfy exactly the same HML formulae. This is formalised in the following theorem.

**Theorem 2.6.** *Let $L = (S, Act, \rightarrow, s, T)$ be a finite LTS, and let $t, t' \in S$, then $t \underline{\leftrightarrow} t'$ if and only if for all HML formulae $\varphi$, $L, t \vDash \varphi \iff L, t' \vDash \varphi$.*

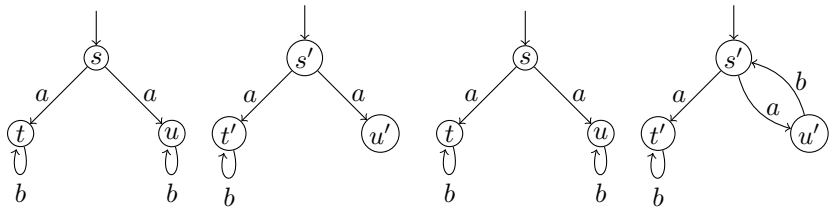
A direct consequence of this theorem is that for non-bisimilar states an HML formula exists that distinguishes these states.

**Example 2.7.** Consider the following two LTSs.




States $s$ and $s'$ are not bisimilar, which is witnessed by the formula $\langle a\rangle\langle a\rangle \mathit{true}$ which holds in $s$ but not in $s'$. ▲

**Exercise 2.8.** For each of the following pairs of LTSs, give a distinguishing formula.




The first extension we make to Hennessy-Milner logic is to reason about sets of actions in the modalities, instead of over single actions. To facilitate this, we introduce action formulae. These are formulae that describe a set of actions.

**Definition 2.9** (Action formula)**.** An action formula over a set of actions $Act$ is defined using the following syntax:

$$A, B ::= \mathit{false} \mid \mathit{true} \mid a \mid \overline{A} \mid A \cup B \mid A \cap B$$

where $a \in Act$. Note that we have the following equivalences:

$$\mathit{true} = \overline{\mathit{false}}$$
$$A \cap B = \overline{\overline{A} \cup \overline{B}}$$

Formally a set of actions is associated to an action formula as follows:

$$\begin{aligned} [\![\mathit{false}]\!] &= \emptyset \\ [\![\mathit{true}]\!] &= Act \\ [\![a]\!] &= \{a\} \\ [\![\overline{A}]\!] &= Act \setminus [\![A]\!] \\ [\![A \cup B]\!] &= [\![A]\!] \cup [\![B]\!] \\ [\![A \cap B]\!] &= [\![A]\!] \cap [\![B]\!] \end{aligned}$$

Intuitively, *false* matches the empty set of actions, *true* matches all actions, $a \in Act$ is an action, $\overline{A}$ matches any action in $Act \setminus A$, $A \cup B$ matches all actions

that are either in $A$ or in $B$, and $A \cap B$ denotes those actions occurring in both $A$ and $B$.

Using such action formulae, we can write $[A]\varphi$ and $\langle A \rangle \varphi$. These are defined as follows:

$$\langle A \rangle \varphi = \bigvee_{a \in [\![A]\!]} \langle a \rangle \varphi \qquad\qquad [A]\varphi = \bigwedge_{a \in [\![A]\!]} [a]\varphi$$

Using action formulae we can compactly write properties, such as in the following example.

**Example 2.10.** The following properties use action formulae:

- $\langle \mathit{true} \rangle \mathit{true}$ an action is possible;
- $\langle a \cup b \rangle [a \cup b] \mathit{false}$ after an $a$ or a $b$ action we end up in a state in which $a$ and $b$ are not possible.

▲

Especially the first property in this example is cumbersome to express in plain HML, since it requires explicitly writing a disjunction over all actions in $\mathit{Act}$. *Deadlock* states do not satisfy the first formula; symmetrically the formula $[\mathit{true}]\mathit{false}$ is *only* satisfied by deadlock states.

**Exercise 2.11.** Explain what the formula $[a \cup b]\mathit{false}$ means, and give an LTS in which it holds in the initial state.

**Exercise 2.12.** Assume that $\mathit{Act} = \{a, b, c\}$. Write a property equivalent to $\langle \mathit{true} \rangle \mathit{true}$ is plain HML (*i.e.* without action formulae).

# 3 Regular Hennessy-Milner Logic

HML can only express properties of behaviour with a finite depth. Properties like the following two cannot be expressed directly in HML without resorting to infinite formulae.

- all reachable states satisfy $\varphi$ (*i.e.* $\varphi$ hold invariantly)

$$\mathit{Inv}(\varphi) = \varphi \wedge [\mathit{true}]\varphi \wedge [\mathit{true}][\mathit{true}]\varphi \wedge \cdots$$

- there is a reachable state which satisfies $\varphi$ (*i.e.* possibly $\varphi$ holds)

$$\mathit{Pos}(\varphi) = \varphi \vee \langle \mathit{true} \rangle \varphi \vee \langle \mathit{true} \rangle \langle \mathit{true} \rangle \varphi \vee \cdots$$

We extend Hennessy-Milner logic such that instead of sets of actions in the modalities we may write $\langle \beta \rangle$ and $[\beta]$ where $\beta$ are regular expressions. This allows for a more concise formulation of properties, and allows us to express properties over *infinite* behaviours.

**Definition 3.1** (Regular expressions)**.** Regular expressions $\beta$ are defined according to the following grammar:

$$\beta_1, \beta_2 ::= \varepsilon \mid A \mid \beta_1 \cdot \beta_2 \mid \beta_1 + \beta_2 \mid \beta_1^*$$

where $A$ is an action formula as defined previously. We abbreviate $\beta_1^+ = \beta_1 \cdot \beta_1^*$ for the non-empty repetition. In the above $\varepsilon$ denotes the empty sequence of actions. The formula $\beta_1 \cdot \beta_2$ denotes the concatenation of the sequences of actions in $\beta_1$ and $\beta_2$; $\beta_1 + \beta_2$ represents the choice between the sequences of actions in $\beta_1$ and $\beta_2$; finally, $\beta_1^*$ represents the, possibly empty, repetition of the sequence in $\beta_1$.

The definitions of $+$ and $\cdot$ can be removed using the following rules:

$$\begin{aligned} \langle\varepsilon\rangle\varphi &= \varphi & [\varepsilon]\varphi &= \varphi \\ \langle\beta_1 \cdot \beta_2\rangle\varphi &= \langle\beta_1\rangle\langle\beta_2\rangle\varphi & [\beta_1 \cdot \beta_2]\varphi &= [\beta_1][\beta_2]\varphi \\ \langle\beta_1 + \beta_2\rangle\varphi &= \langle\beta_1\rangle\varphi \vee \langle\beta_2\rangle\varphi & [\beta_1 + \beta_2]\varphi &= [\beta_1]\varphi \wedge [\beta_2]\varphi \\ \langle\beta^*\rangle\varphi &= \varphi \vee \langle\beta\rangle\langle\beta^*\rangle\varphi & [\beta^*]\varphi &= \varphi \wedge [\beta][\beta^*]\varphi \end{aligned}$$

Repetition can be characterised using infinitary conjunctions and disjunctions respectively. This way, one can think of $\langle\beta^*\rangle\varphi$ as $\varphi \vee \langle\beta\varphi\rangle \vee \langle\beta\rangle\langle\beta\rangle\varphi \vee \cdots$. The must modality $[\beta^*]\varphi$ can similarly be characterised as $\varphi \wedge [\beta]\varphi \wedge [\beta][\beta]\varphi \wedge \cdots$. In the next section we will see how this can be made more compact using recursion.

The two properties that we started this section with can be expressed as follows in regular HML:

- $Inv(\varphi) = [true^*]\varphi$
- $Pos(\varphi) = \langle true^*\rangle\varphi$

**Example 3.2.** We next show some typical examples expressible in the regular HML, and explain their structure in more detail.

$[true^*]\langle true\rangle true$ deadlock freedom

Here $true^*$ contains all traces in the system, so every execution ends up in a state in which $\langle true\rangle true$ holds. Again $true$ matches all actions, so $\langle true\rangle true$ expresses that a step is possible. Summarising, every execution ends up in a state in which a step is possible, so the system contains no deadlock.

$[true^* \cdot error]false$ error does not occur

The first part again says any execution ends up in state in which $[error]false$ holds. This means in every reachable state, the $error$ action is not possible.

$\langle a^*\rangle[b]false$ there is an $a$ path along which eventually $b$ is disabled

In this property $a^*$ contains all traces consisting of only $a$ actions. The property hence says there exists a path consisting of only $a$ actions, that ends up in a state in which $[b]false$ holds, *i.e.* in which no $b$ action is possible.

$[a^*]\langle b\rangle true$ along every $a$ path, $b$ is always enabled

The first part says that in every state that is reachable by only doing $a$ actions, the formula $\langle b\rangle true$ holds, and thus in every such state $b$ is always enabled.

$[true^*](\langle b\rangle true \implies \langle c\rangle true)$ invariantly, if $b$ is enabled then $c$ is also enabled.

If a property holds in every reachable state, expressed through $[true^*]$ we also say that it holds invariantly. In this case, in every reachable state, $\langle b\rangle true \implies \langle c\rangle true$ holds, which means that if $b$ is enabled, then also $c$ is enabled.

> $[true^* \cdot open \cdot \overline{close}^* \cdot open]false$ invariantly it is impossible to do two consecutive *open* actions without an intermediate *close* action.

The expression $true^* \cdot open \cdot \overline{close}^* \cdot open$ contains all executions on which at some point an *open* transition occurs, follows by a sequence of actions other than *close*, and then another *open*. The property than states that such an execution cannot occur, so a *close* must always happen inbetween two *open*s.

> $[true^* \cdot receive \cdot \overline{receive \cup send}^* \cdot send \cdot \overline{receive}^* \cdot send]false$ invariantly after a receive, no two consecutive sends can happen, unless another receive occurs before the send.

This property is similar to the one before, however, in the previous property the 'first' action could not happen twice without the 'second' action happening in the mean time. In this case the 'second' action cannot happen twice without the 'first' one happening in the mean time. If we dissect this property, the explanation is as follows. Every execution of which the last transition is a *receive* action, ends in a state in which $[\overline{receive \cup send}^* \cdot send \cdot \overline{receive}^* \cdot send]false$ holds. So after such an execution, all executions that do a sequence of actions other than *receive* or *send*, followed by a *send*, followed by anything other than *receive*, ending with another *send* is not allowed. ▲

**Exercise 3.3.** Express each of the following properties in regular HML.

- The system contains a path to a deadlock state;
- along every $a$ path, if $b$ is disabled, then $c$ is enabled;
- invariantly, after taking an $a$ action, a $b$ action can eventually happen, unless an error occurs.

**Exercise 3.4.** Give an LTS in which the property $[a]\langle true \cdot b\rangle true$ holds, but $[true^* \cdot a]\langle true \cdot b\rangle true$ does not hold. What is the essential difference between these two properties?

# 4 Modal $\mu$-calculus

Regular HML already is a powerful logic. Many of the properties that occur in practice can be expressed in it. However, there are properties that cannot be expressed in regular HML. Typical examples of such properties are the following:

- $Inev(\varphi)$ all computations inevitably reach a state which satisfies $\varphi$;
- $Safe(\varphi)$ for some execution $\varphi$ holds everywhere.

To overcome this we add recursion, with explicit minimal and maximal fixed points. This way, an extremely powerful logic, the modal $\mu$-calculus is obtained. This expressivity comes at the price of comprehension though; to express properties in the $\mu$-calculus, and assess whether a formula indeed expresses the intended property, requires experience.

**Partial orders, monotonic functions and fixed points** A partial order is a relation that is reflexive, antisymmetric and transitive. Let $U$ be a set with partial order $\leq$. Given a function $f\colon U \to U$, an element $x \in U$ is a fixed point for $f$ if $f(s) = s$. Moreover, $s$ is a least fixed point or greatest fixed point if for all fixed points $t$, $s \leq t$ or $t \leq s$ respectively. The least and greatest fixed point do not necessarily exist, but if they do, they are unique and denoted $\mu X.f(X)$ and $\nu X.f(X)$.

A function $f\colon U \to U$ is called monotonic if for all $s, t$, $s \leq t \implies f(s) \leq f(t)$. If $f$ is monotonic, and $U$ has a least element $s_0$ (*i.e.* $s_0 \leq s$ for all $s \in U$) and a greatest element $t_0$ (*i.e.* $t \leq t_0$ for all $t \in U$) then the least- and greatest fixed points exist, and can be computed iteratively as follows. Define $s_{i+1} = f(s_i)$ and $t_{i+1} = f(t_i)$. Since $s_0 \leq s_1$ and $t_1 \leq t_0$, and using monotonicity it follows that $s_i \leq s_{i+1}$ and $t_{i+1} \leq t_i$ for all $i \geq 0$. If $U$ is finite, there are $k$ and $\ell$ such that $s_k = s_{k+1}$ and $t_\ell = t_{\ell+1}$. It is not hard to see that $\mu X.f(X) = s_k$ and $\nu X.f(X) = t_k$. In other words, for a finite set $U$ with least and greatest elements, the least- and greatest fixed points of a monotonic function can be computed by repeated application of the function to the least element and the greatest element respectively.

**The $\mu$-calculus** We now leave these formal basics behind, and define the syntax of the modal $\mu$-calculus as an extension of regular HML. We assume a set of variables $\mathsf{X}$ with elements $X, Y, Z, X_1, X_2, \ldots$.

**Definition 4.1** (Modal $\mu$-calculus)**.** The syntax of the modal $\mu$-calculus is defined as follows:

$$\varphi, \psi ::= \mathit{true} \mid \mathit{false} \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \implies \psi \mid \langle\beta\rangle\varphi \mid [\beta]\varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid X$$

Observe that this is regular HML extended with the constructs $\mu X.\varphi$ and $\nu X.\varphi$, and in a formula we may refer to the variables in $\mathsf{X}$. We restrict ourselves to *closed* formulas, which means that every variable that occurs in a subformula must be bound by some fixed point operator. In $\mu X.X$, the $X$ occurs bound, and the formula is closed, whereas in $\mu X.Y$, the $Y$ occurs *free* (*i.e.* not bound), and the formula is open (not closed). Furthermore, every recursion variable must occur in the scope of an *even* number of negations, where for counting negations the formula $\varphi \implies \psi$ is interpreted as $\neg\varphi \vee \psi$. This is a technicality to ensure *monotonicity*, which allows us to properly give semantics to the $\mu$-calculus.

The least and the greatest fixed points are each others' dual. This is formalised in the following equations:

$$\nu X.\varphi = \neg\mu X.\neg\varphi[X := \neg X]$$
$$\mu X.\varphi = \neg\nu X.\neg\varphi[X := \neg X]$$

here $\varphi[X := \neg X]$ is the formula $\varphi$ in which every occurrence of $X$ is syntactically replaced by $\neg X$.

As promised before, we come back to the meaning of iteration in regular HML. Iteration can be expressed in the $\mu$-calculus as follows:

$$\langle\beta^*\rangle\varphi = \mu X.(\varphi \vee \langle\beta\rangle X) \qquad [\beta^*]\varphi = \nu X.(\varphi \wedge [\beta]X)$$

**Exercise 4.2.** To enhance your understanding of regular HML and the modal $\mu$-calculus, write the regular HML formulas from Example 3.2 in the $\mu$-calculus without using regular expressions.

Assuming a finite state space, $\mu X.\varphi$ and $\nu X.\phi$ denote least- and greatest fixed points, but of what? Recall that the denotational semantics of HML gave us a set of states in which a formula holds; we want something similar for the $\mu$-calculus. Therefore, we interpret $\varphi$ as a function that takes a set of states and returns a set of states, *i.e.* we assume that $X \subseteq S$. More formally, we are computing fixed points over the powerset of states (*i.e.* the set of subsets of $S$), in which we use $\subseteq$ as the ordering. The least element of the powerset of states is the empty set, denoted $\emptyset$, and its greatest element is the set of all states, $S$.

To accomodate reasoning with these fixed points, $\mu$-calculus formulae are interpreted in an environment $\eta \colon \mathsf{X} \to 2^S$, that assigns sets of states to proposition variables. We write $\eta[X := S']$ to denote the environment $\eta$ in which $X$ receives the value $S'$, and all other variables get the value from $\eta$ such that $\eta[X := S'](Y) = S'$ if $X = Y$, and $\eta[X := S'](Y) = \eta(Y)$ otherwise. For closed formulas, the environment does not influence the solution, *i.e.* if $\varphi$ is closed, then $[\![\varphi]\!]^\eta_L \equiv [\![\varphi]\!]^{\eta'}_L$ for all environments $\eta$ and $\eta'$. For closed formulas we therefore sometimes omit the environment. Next we first give the semantics of the $\mu$-calculus and then get back to the fixed points and how they can be computed.

The semantics of the $\mu$-calculus can be defined inductively as follows:

**Definition 4.3** ($\mu$-calculus semantics)**.** Let $L$ be an LTS, and let $\eta \colon X \to 2^S$ be an environment.

$$
\begin{aligned}
[\![\mathit{true}]\!]^\eta_L &= S \\
[\![\mathit{false}]\!]^\eta_L &= \emptyset \\
[\![X]\!]^\eta_L &= \eta(X) \\
[\![\neg\varphi]\!]^\eta_L &= S \setminus [\![\varphi]\!]^\eta_L \\
[\![\varphi \wedge \psi]\!]^\eta_L &= [\![\varphi]\!]^\eta_L \cap [\![\psi]\!]^\eta_L \\
[\![\varphi \vee \psi]\!]^\eta_L &= [\![\varphi]\!]^\eta_L \cup [\![\psi]\!]^\eta_L \\
[\![\varphi \implies \psi]\!]^\eta_L &= (S \setminus [\![\varphi]\!]^\eta_L) \cup [\![\psi]\!]^\eta_L \\
[\![\langle A \rangle \varphi]\!]^\eta_L &= \{s \in S \mid \exists_{t \in S, a \in [\![A]\!]}\, s \xrightarrow{a} t \wedge t \in [\![\varphi]\!]^\eta_L\} \\
[\![[A]\varphi]\!]^\eta_L &= \{s \in S \mid \forall_{t \in S, a \in [\![A]\!]}\, s \xrightarrow{a} t \implies t \in [\![\varphi]\!]^\eta_L\} \\
[\![\mu X.\varphi]\!]^\eta_L &= \mu T \subseteq S.\Phi_\eta(T) \\
[\![\nu X.\varphi]\!]^\eta_L &= \nu T \subseteq S.\Phi_\eta(T)
\end{aligned}
$$

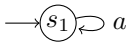
where $\Phi_\eta \colon 2^S \to 2^S$ is the functional defined as $\Phi_\eta(T) = [\![\varphi]\!]^{\eta[X:=T]}_L$.

If $\varphi$ is closed, we omit the environment from $\Phi$. For the semantics of the fixed points to be well-defined, we require monotonicity of $\Phi$, which is guaranteed using the syntactic restriction we have made before. Since $\Phi$ is monotonous, and $S$ is finite, we can compute the fixed points using iteration. Let $\Phi^0(T) = T$, and $\Phi^{n+1}(T) = \Phi(\Phi^n(T))$, then the fixed points can be redefined as

$$
[\![\mu X.\varphi]\!]^\eta_L = \bigcup_i \Phi^i_\eta(\emptyset) \qquad\qquad [\![\nu X.\varphi]\!]^\eta_L = \bigcap_i \Phi^i_\eta(S)
$$

In fact it suffices to start computing with $\Phi^0$ in both cases, and apply the function to its own result until the computation stabilises. This immediately gives us an algorithm for computing the states in which a $\mu$-calculus formula holds. We illustrate this using a number of examples.

**Example 4.4.** Consider the LTS depicted below. We will show whether $\mu X.\langle a\rangle X$ and $\nu X.\langle a\rangle X$ hold.




We first show whether $\mu X.\langle a\rangle X$ holds, *i.e.* we compute the set $[\![\mu X.\langle a\rangle X]\!]_L$. Let $\Phi$ be the functional such that $\Phi(T) = [\![\langle a\rangle X]\!]_L^{[X:=T]}$. The solution that we are interested in now is the least fixed point of this functional. Since we are dealing with a least fixed point, we start the approximation at $\emptyset$, and we proceed as follows:

$$\begin{aligned}
\Phi^0(\emptyset) &= \emptyset \\
\Phi^1(\emptyset) &= [\![\langle a\rangle X]\!]_L^{[X:=\emptyset]} \\
&= \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \wedge t \in [\![X]\!]_L^{[X:=\emptyset]}\} \\
&= \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \wedge t \in [X := \emptyset](X)\} \\
&= \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \wedge t \in \emptyset\} \\
&= \emptyset
\end{aligned}$$

so we immediately find the fixed point, and we have shown that the property does not hold in any state.

Next consider $\nu X.\langle a\rangle X$. Let $\Phi$ be the same functional as before. For $[\![\nu X.\langle a\rangle X]\!]_L$ we have to compute the *greatest fixed point* of the functional instead, and we start approximating at $S$:

$$\begin{aligned}
\Phi^0(S) &= S \\
\Phi^1(S) &= [\![\langle a\rangle X]\!]_L^{[X:=S]} \\
&= \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \wedge t \in [\![X]\!]_L^{[X:=S]}\} \\
&= \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \wedge t \in [X := S](X)\} \\
&= \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \wedge t \in S\} \\
&= S
\end{aligned}$$

Again the iteration stops after a single step, and we show that all states are in the greatest fixed point, hence the property holds in all states. ▲

From this example we can learn that, effectively, the least fixed point only holds if we can traverse through the minimal fixed point variables a finite number of times, whereas greatest fixed points may be traversed infinitely often.

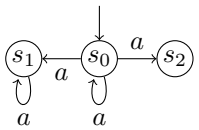
If we look at $\mu X.\varphi$ and $\nu X.\varphi$, then the least fixed point is valid for the smallest set satisfying $X = \varphi$, whereas the greatest fixed point is valid for the largest such set. As a consequence, we immediately get the following relationship between them:

$$\mu X.\varphi \implies \nu X.\varphi$$

Let us consider the following less trivial example.

**Example 4.5.** We want to check, in the following LTS, whether there is a computation along which eventually $a$ will be disabled. The question is, in the following formula, whether we should use a least or a greatest fixed point

$$\{\mu, \nu\}X.[a]\mathit{false} \vee \langle \mathit{true}\rangle X$$




We again compute both. Let $\Phi$ be the functional such that $\Phi(T) = [\![[a]\mathit{false} \vee \langle \mathit{true}\rangle X]\!]_L^{[X:=T]}$. Now, $[\![\mu X.[a]\mathit{false} \vee \langle \mathit{true}\rangle X]\!]_L = \mu T \subseteq S.\Phi(T)$, which we can compute using iteration as follows:

$$
\begin{aligned}
\Phi^0(\emptyset) &= \emptyset \\
\Phi^1(\emptyset) &= [\![[a]\mathit{false} \vee \langle \mathit{true}\rangle X]\!]_L^{[X:=\emptyset]} \\
&= [\![[a]\mathit{false}]\!]_L^{[X:=\emptyset]} \cup [\![\langle \mathit{true}\rangle X]\!]_L^{[X:=\emptyset]} \\
&= \{s \in S \mid \forall_{t\in S, a\in\{a\}} s \xrightarrow{a} t \implies t \in \emptyset\} \\
&\quad \cup \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \wedge t \in [\![X]\!]_L^{[X:=\emptyset]}\} \\
&= \{s_2\} \cup \emptyset \\
&= \{s_2\} \\
\Phi^2(\emptyset) &= [\![[a]\mathit{false} \vee \langle \mathit{true}\rangle X]\!]_L^{[X:=\{s_2\}]} \\
&= [\![[a]\mathit{false}]\!]_L^{[X:=\{s_2\}]} \cup [\![\langle \mathit{true}\rangle X]\!]_L^{[X:=\{s_2\}]} \\
&= \{s_2\} \cup \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \wedge t \in [\![X]\!]_L^{[X:=\{s_2\}]}\} \\
&= \{s_2\} \cup \{s_0\} \\
&= \{s_0, s_2\} \\
\Phi^3(\emptyset) &= [\![[a]\mathit{false} \vee \langle \mathit{true}\rangle X]\!]_L^{[X:=\{s_0,s_2\}]} \\
&= [\![[a]\mathit{false}]\!]_L^{[X:=\{s_0,s_2\}]} \cup [\![\langle \mathit{true}\rangle X]\!]_L^{[X:=\{s_0,s_2\}]} \\
&= \{s_2\} \cup \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \wedge t \in [\![X]\!]_L^{[X:=\{s_0,s_2\}]}\} \\
&= \{s_2\} \cup \{s_0\} \\
&= \{s_0, s_2\}
\end{aligned}
$$

so the property holds in $s_0$ and $s_2$.

Likewise we can compute the greatest fixed point, using the same functional.

$$
\begin{aligned}
\Phi^0(S) &= S \\
\Phi^1(S) &= [\![[a]\mathit{false} \vee \langle \mathit{true}\rangle X]\!]_L^{[X:=S]} \\
&= [\![[a]\mathit{false}]\!]_L^{[X:=S]} \cup [\![\langle \mathit{true}\rangle X]\!]_L^{[X:=S]} \\
&= \{s_2\} \cup \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \wedge t \in [\![X]\!]_L^{[X:=S]}\} \\
&= \{s_2\} \cup \{s_0, s_1\} \\
&= S
\end{aligned}
$$

So all states satisfy the greatest fixed point version of the property. If we look at the property that we want to check, *i.e.* there is a computation along which eventually $a$ is disabled, this should not hold in $s_1$, since always an $a$ is enabled from that state, however, in $s_2$ it holds trivially, since $a$ is disabled in that state, and from $s_0$ we can take the $a$ step to $s_2$, again satisfying the property.

Therefore, for this eventuality property, we should use the least fixed point in expressing the property. ▲

In general, safety properties express that something invariantly holds in the system, and typically they are formulated using greatest fixed points. Liveness properties generally express that something must happen eventually (*i.e.* within a finite number of steps), and it is therefore expressed using least fixed points. The two properties at the start of this section are typical examples of liveness and safety properties. Using the modal $\mu$-calculus we can express these properties, that were not expressible in regular HML, as follows:

- $\mathit{Inev}(\varphi) = \mu X.\varphi \vee [\mathit{true}]X$;
- $\mathit{Safe}(\varphi) = \nu X.\varphi \wedge \langle \mathit{true}\rangle X$.

Intuitively, the first property expresses that either $\varphi$ holds in the current state, or in each of the following states, $X$ must hold. Since the property has a least fixed point, $X$ may only be traversed finitely many times, which means that along each execution, within a finite number of steps, $\varphi$ must hold. Beware that this property trivially holds in a deadlock state. Likewise, in the second property, in the current state $\varphi$ must hold, and there is a successor state in which $X$ holds. Since this is a greatest fixed point, this may be traversed infinitely often, and we find that there exists and execution, along which in every state $\varphi$ holds.

These examples do not take specific actions into account. We can, however, restrict the executions that are considered in a property. In the previous section we have seen that the property $\langle a^*\rangle[b]\mathit{false}$ expresses that there is an $a$ path along which eventually $b$ is disabled. A typical liveness property is that along *every* $a$ path, $b$ must eventually be disabled. Using fixed points we can express this as

$$\mu X.[a]X \vee [b]\mathit{false}.$$

**Exercise 4.6.** Use the semantics to check that $\mu X.[\mathit{true}]X \vee \varphi$ holds in a deadlock state, for all formulae $\varphi$.

If you want to make sure that a state in which $\varphi$ holds is really reached along every path, you can modify the formula by requiring that a step is possible as follows.

$$\mu X.([\mathit{true}]X \wedge \langle \mathit{true}\rangle \mathit{true}) \vee \varphi$$

Effectively this says that, if $\varphi$ does not hold, then a step must be possible, hence $[\mathit{true}]X$ is not trivially satisfied.

**Example 4.7.** We next give some more examples of $\mu$-calculus properties.

> $\nu X.[\overline{\mathit{receive}}]X \wedge [\mathit{send}]\mathit{false}$ as long as no *receive* action has happened, *send* may not occur.

If we look closely at this property, for it to become true, both $[\overline{\mathit{receive}}]X$ and $[\mathit{send}]\mathit{false}$ must hold. The latter is the case if no *send* action is possible; the first part holds trivially in a state in which no *receive* action is possible. So, along all executions along which *receive* has not happened, *send* is not possible. This could equivalently be expressed as $[\overline{\mathit{receive}}^*][\mathit{send}]\mathit{false}$.

$[true^* \cdot receive]\mu X.[\overline{send}]X \wedge \langle true \rangle true$ after every receive eventually a send must happen.

We first look at the fixed point subformula. This expresses that inevitably *send* must happen in a finite number of steps along every execution. The first part tells us this is the case for all executions that end in a *receive* transition.

$\nu X.\mu Y.(\langle r \rangle X \vee \langle \overline{r} \rangle Y)$ it is possible to infinitely often do an $r$ action.

This property is more complicated, since it alternates between greatest- and least fixed points. Recall the intuition that we may pass through a greatest fixed point infinitely often, whereas a least fixed point may only be traversed finitely many times. In this case, we pass through $X$ infinitely many times, whereas we may only pass through $Y$ a finite number of times. That means that there is an execution on which $r$ is allowed infinitely often, whereas between consecutive $r$ transitions, only a finite number of other transitions is allowed. This is understood easier by looking at the following example. Consider the two labelled transition system depicted below.


We compute the solution simultaneously for both LTSs. Let $\Phi_X(T) = [\![\mu Y.(\langle r \rangle X \vee \langle \overline{r} \rangle Y)]\!]_L^{[X:=T]}$ and $\Phi_{Y,\eta}(T) = [\![\langle r \rangle X \vee \langle \overline{r} \rangle Y]\!]_L^{\eta[Y:=T]}$ be two functionals. To solve this fixed point, we need to nest the iterations, and we proceed as follows:

$$
\begin{aligned}
&\Phi^0_X(S) = S\\
&\Phi^1_X(S) = [\![\mu Y.(\langle r \rangle X \vee \langle \overline{r} \rangle Y)]\!]_L^{[X:=S]}\\
&\qquad = \mu T \subseteq \Phi_{Y,[X:=S]}\\
&\qquad\quad \Phi^0_{Y,[X:=S]}(\emptyset) && = \emptyset\\
&\qquad\quad \Phi^1_{Y,[X:=S]}(\emptyset) && = [\![\langle r \rangle X \vee \langle \overline{r} \rangle Y]\!]_L^{[X:=S][Y:=\emptyset]}\\
&\qquad\qquad && = [\![\langle r \rangle X]\!]_L^{[X:=S][Y:=\emptyset]}\\
&\qquad\qquad && \quad \cup [\![\langle \overline{r} \rangle Y]\!]_L^{[X:=S][Y:=\emptyset]}\\
&\qquad\qquad && = \{s_1, t_1\} \cup \emptyset\\
&\qquad\qquad && = \{s_1, t_1\}\\
&\qquad\quad \Phi^2_{Y,[X:=S]}(\emptyset) && = [\![\langle r \rangle X \vee \langle \overline{r} \rangle Y]\!]_L^{[X:=S][Y:=\{s_1,t_1\}]}\\
&\qquad\qquad && = [\![\langle r \rangle X]\!]_L^{[X:=S][Y:=\{s_1,t_1\}]}\\
&\qquad\qquad && \quad \cup [\![\langle \overline{r} \rangle Y]\!]_L^{[X:=S][Y:=\{s_1,t_1\}]}\\
&\qquad\qquad && = \{s_1, t_1\} \cup \{s_2\}\\
&\qquad\qquad && = \{s_1, s_2, t_1\}\\
&\qquad\quad \Phi^3_{Y,[X:=S]}(\emptyset) && = [\![\langle r \rangle X \vee \langle \overline{r} \rangle Y]\!]_L^{[X:=S][Y:=\{s_1,s_2,t_1\}]}\\
&\qquad\qquad && = [\![\langle r \rangle X]\!]_L^{[X:=S][Y:=\{s_1,s_2,t_1\}]}\\
&\qquad\qquad && \quad \cup [\![\langle \overline{r} \rangle Y]\!]_L^{[X:=S][Y:=\{s_1,s_2,t_1\}]}
\end{aligned}
$$

$$
\begin{aligned}
&&&= \{s_1, t_1\} \cup \{s_2\} \\
&&&= \{s_1, s_2, t_1\} \\
&= \{s_1, s_2, t_1\} \\
\Phi^2_X(S) &= [\![\mu Y.(\langle r\rangle X \vee \langle \overline{r}\rangle Y)]\!]_L^{[X:=\{s_1,s_2,t_1\}]} \\
&= \mu T \subseteq \Phi_{Y,[X:=\{s_1,s_2,t_1\}]} \\
&\quad \Phi^0_{Y,[X:=\{s_1,s_2,t_1\}]}(\emptyset) && = \emptyset \\
&\quad \Phi^1_{Y,[X:=\{s_1,s_2,t_1\}]}(\emptyset) && = [\![\langle r\rangle X \vee \langle \overline{r}\rangle Y]\!]_L^{[X:=\{s_1,s_2,t_1\}][Y:=\emptyset]} \\
&&&= [\![\langle r\rangle X]\!]_L^{[X:=\{s_1,s_2,t_1\}][Y:=\emptyset]} \\
&&&\quad \cup [\![\langle \overline{r}\rangle Y]\!]_L^{[X:=\{s_1,s_2,t_1\}][Y:=\emptyset]} \\
&&&= \{s_1\} \cup \emptyset \\
&\quad \Phi^2_{Y,[X:=\{s_1,s_2,t_1\}]}(\emptyset) && = [\![\langle r\rangle X \vee \langle \overline{r}\rangle Y]\!]_L^{[X:=\{s_1,s_2,t_1\}][Y:=\{s_1\}]} \\
&&&=^* \{s_1\} \cup \{s_2\} \\
&\quad \Phi^3_{Y,[X:=\{s_1,s_2\}]}(\emptyset) && = [\![\langle r\rangle X \vee \langle \overline{r}\rangle Y]\!]_L^{[X:=\{s_1,s_2,t_1\}][Y:=\{s_1,s_2\}]} \\
&&&=^* \{s_1, s_2\} \\
&= \{s_1, s_2\} \\
\Phi^3_X(S) &= [\![\mu Y.(\langle r\rangle X \vee \langle \overline{r}\rangle Y)]\!]_L^{[X:=\{s_1,s_2\}]} \\
&= \mu T \subseteq \Phi_{Y,[X:=\{s_1,s_2\}]} \\
&\quad \Phi^0_{Y,[X:=\{s_1,s_2\}]}(\emptyset) && = \emptyset \\
&\quad \Phi^1_{Y,[X:=\{s_1,s_2\}]}(\emptyset) && = [\![\langle r\rangle X \vee \langle \overline{r}\rangle Y]\!]_L^{[X:=\{s_1,s_2\}][Y:=\emptyset]} \\
&&&=^* \{s_1\} \cup \emptyset \\
&\quad \Phi^2_{Y,[X:=\{s_1,s_2\}]}(\emptyset) && = [\![\langle r\rangle X \vee \langle \overline{r}\rangle Y]\!]_L^{[X:=\{s_1,s_2\}][Y:=\{s_1\}]} \\
&&&=^* \{s_1, s_2\} \\
&\quad \Phi^3_{Y,[X:=\{s_1,s_2\}]}(\emptyset) && = [\![\langle r\rangle X \vee \langle \overline{r}\rangle Y]\!]_L^{[X:=\{s_1,s_2\}][Y:=\{s_1,s_2\}]} \\
&&&=^* \{s_1, s_2\} \\
&=^* \{s_1, s_2\} \\
&= \Phi^2_X(S)
\end{aligned}
$$

In the computation above, at $=^*$ multiple steps have been combined. Also observe that each time we need to recompute the innermost fixed point again using the new approximation of the outermost fixed point.

It is important to note that the order of the fixed points in this formula matters. Compare the above to the formula

> $\mu X.\nu Y.(\langle r\rangle X \vee \langle \overline{r}\rangle Y)$ there is an execution in which only finitely many $r$ steps occur.

Now we may only pass through $X$ finitely many times, hence we can only do finitely many $r$ steps, but we can, and in fact should, pass through $Y$ infinitely often. We can compute the solution to this using the same functionals.

$\Phi^0_X(\emptyset) = \emptyset$

$$
\begin{aligned}
\Phi^1_X(\emptyset) &= [\![\mu Y.(\langle r\rangle X \vee \langle \overline{r}\rangle Y)]\!]_L^{[X:=\emptyset]} \\
&= \nu T \subseteq \Phi_{Y,[X:=\emptyset]} \\
&\quad \Phi^0_{Y,[X:=\emptyset]}(S) && = S \\
&\quad \Phi^1_{Y,[X:=\emptyset]}(S) && = [\![\langle r\rangle X \vee \langle \overline{r}\rangle Y]\!]_L^{[X:=\emptyset][Y:=S]} \\
&\quad && =^* \{s_2, t_2\} \\
&\quad \Phi^2_{Y,[X:=\emptyset]}(S) && = [\![\langle r\rangle X \vee \langle \overline{r}\rangle Y]\!]_L^{[X:=\emptyset][Y:=\{s_2,t_2\}]} \\
&\quad && =^* \{t_2\} \\
&\quad \Phi^3_{Y,[X:=\emptyset]}(S) && = [\![\langle r\rangle X \vee \langle \overline{r}\rangle Y]\!]_L^{[X:=\emptyset][Y:=\{t_2\}]} \\
&\quad && =^* \{t_2\} \\
&= \{t_2\} \\
\Phi^2_X(\emptyset) &= [\![\mu Y.(\langle r\rangle X \vee \langle \overline{r}\rangle Y)]\!]_L^{[X:=\{t_2\}]} \\
&= \nu T \subseteq \Phi_{Y,[X:=\{t_2\}]} \\
&\quad \Phi^0_{Y,[X:=\{t_2\}]}(S) && = S \\
&\quad \Phi^1_{Y,[X:=\{t_2\}]}(S) && = [\![\langle r\rangle X \vee \langle \overline{r}\rangle Y]\!]_L^{[X:=\{t_2\}][Y:=S]} \\
&\quad && =^* \{t_1\} \cup \{s_2, t_2\} \\
&\quad \Phi^2_{Y,[X:=\{t_2\}]}(S) && = [\![\langle r\rangle X \vee \langle \overline{r}\rangle Y]\!]_L^{[X:=\{t_2\}][Y:=\{s_2,t_1,t_2\}]} \\
&\quad && =^* \{t_1\} \cup \{t_2\} \\
&\quad \Phi^3_{Y,[X:=\{t_2\}]}(S) && = [\![\langle r\rangle X \vee \langle \overline{r}\rangle Y]\!]_L^{[X:=\{t_2\}][Y:=\{t_1,t_2\}]} \\
&\quad && =^* \{t_1, t_2\} \\
&= \{t_1, t_2\} \\
\Phi^3_X(\emptyset) &= [\![\mu Y.(\langle r\rangle X \vee \langle \overline{r}\rangle Y)]\!]_L^{[X:=\{t_1,t_2\}]} \\
&= \nu T \subseteq \Phi_{Y,[X:=\{t_1,t_2\}]} \\
&=^{*,\dagger} \{t_1, t_2\}
\end{aligned}
$$

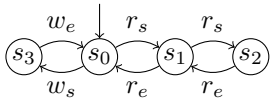
The final approximation step at † is left to the reader.

Finally we show a property that involves fairness.

> $\nu X.\mu Y.\nu Z.([r]X \wedge ([r]\mathit{false} \vee [\overline{r}]Y) \wedge [\overline{r}]Z)$ if $r$ is enabled infinitely often, than it is taken infinitely often.

This is a complex property, expressing a fairness requirement. Dissecting the property, the $[r]X$ says that after every $r$ transition the property must hold again. The second part says that either $r$ is not enabled, or, if $r$ is enabled, every other step leads to a state in which $Y$ holds; since $Y$ is a least fixed point, this can only be the case if we are forced to do $r$ after finitely many steps, in case $r$ is always enabled. Finally, if an $r$ is never possible, we allow infinite executions of steps other than $r$ using the last constraint. ▲

**Exercise 4.8.** Consider the following LTS, modelling mutual exclusion between two readers and a single writer. The identity of the readers is immaterial to the validity of the mutual exclusion property, and is therefore excluded. Reading is started using the action $r_s$ and ended using $r_e$. Likewise for writing.




Verify using the semantics whether the following properties hold in the initial state:

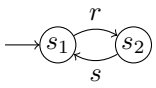
- on some path, a reader can infinitely often start reading

$$\nu X.\mu Y.\langle r_s\rangle X \vee \langle \overline{r_s}\rangle Y$$

- on some path, eventually a writer can never start writing:

$$\mu X.\langle true\rangle X \vee (\nu Y.[w_s]false \wedge \langle true\rangle Y)$$

**Exercise 4.9** (*)**.** Check whether the property $\nu X.\mu Y.\nu Z.([r]X \wedge ([r]false \vee [\overline{r}]Y) \wedge [\overline{r}]Z)$ holds in the following LTS using approximation.




# 5 First order modal $\mu$-calculus

For use in mCRL2, the modal $\mu$-calculus has been extended one step further. The fixed point variables can be parameterised by *data*, and data can be used in regular expressions; effectively this allows one to concisely express, for example, counters. Using this feature properties can, in fact, sometimes be expressed exponentially more compactly. This is witnessed by the encoding of CTL$^*$ in the first-order modal $\mu$-calculus [CGR11].

First order modal $\mu$-calculus formulae are interpreted in a context of abstract data types. Since this is the full version of the modal $\mu$-calculus that we will use, we repeat all of its aspects here. First of all, along transitions we allow multiple actions to happen at the same time. This is called a *multi-action*, in which the empty multi-action represents the silent action $\tau$, and individual actions can be parameterised by a number of data values.

**Definition 5.1** (Multi-action)**.** Multi-actions $\alpha$ adhere to the following syntax:

$$\alpha_1, \alpha_2 ::= \tau \mid a(t_1, \ldots, t_n) \mid \alpha_1 \vdash \alpha_2$$

here $\tau$ denotes the empty multi-action, which is equivalent to the *silent* or *internal* action, and $a(t_1, \ldots, t_n)$ is an action $a$ with data parameters $t_1$ to $t_n$.

Action formulae are defined as before, but also here we add constructs to deal with data.

**Definition 5.2** (Action formula)**.** An action formula over a set of actions $Act$ is defined using the following syntax:

$$A, B ::= b \mid \alpha \mid \overline{A} \mid A \cup B \mid A \cap B \mid \exists d\colon D.A \mid \forall d\colon D.A$$

here $b$ is an expression of sort $\mathbb{B}$.

If $b$ is *true*, then it represents the set of all (multi-)actions, and if *false* it represents the empty set. Instead of actions, we allow multi-actions $\alpha$, and the existential quantification represents a union over all elements $d$, *i.e.* $\exists d\colon D.A$ represents the set $\bigcup_{d\colon D} A$, likewise universal quantification represents generalised intersection. We have the following equivalences between action formulas:

$$A \cap B = \overline{\overline{A} \cup \overline{B}}$$
$$\forall d\colon D.A = \overline{\exists d\colon D.\overline{A}}$$

Regular expressions are as before.

**Definition 5.3** (Regular expression)**.** Regular expressions $\beta$ are defined according to the following grammar:

$$\beta_1, \beta_2 ::= \varepsilon \mid A \mid \beta_1 \cdot \beta_2 \mid \beta_1 + \beta_2 \mid \beta_1^*$$

Finally, first-order modal $\mu$-calculus formulae are defined as follows.

**Definition 5.4** (First-order modal $\mu$-calculus)**.** The first-order modal $\mu$-calculus adheres to the following syntax:

$$\begin{aligned}\varphi, \psi ::= & b \mid \neg\varphi \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \exists d\colon D.\varphi \mid \forall d\colon D.\varphi \mid \langle\beta\rangle\varphi \mid [\beta]\varphi \mid X(t_1, \ldots, t_n) \\ & \mid \mu X(d_1\colon D_1 := t_1, \ldots, d_n\colon D_n := t_n).\varphi \\ & \mid \nu X(d_1\colon D_1 := t_1, \ldots, d_n\colon D_n := t_n).\varphi\end{aligned}$$

here $b$ again is an arbitrary expression of type $\mathbb{B}$.

Using the extension with data, we can now express properties that also concern, *e.g.*, values of the parameters of actions. Furthermore, we can aggregate information in the parameters of variables.

**Example 5.5.** We describe a number of properties in which data is used.

> $\nu X.[\overline{\exists d\colon D.receive(d)}] \wedge [\exists e\colon D.send(e)]false$ as long as no message has been received, none can be sent.

This property is similar to the first property in Example 4.7. In this case, $\exists d\colon D.receive(d)$ matches any action $receive(v)$, for all values $v$. Likewise for *send*.

The above property allows message $w$ to be sent, after only $v$ has been received. To prevent this, we can formulate the following stronger property.

> $\forall d\colon D.\nu X.[\overline{receive(d)}] \wedge [send(d)]false$ as long as message $d$ has not been received, it cannot be sent, for all messages $d$.

Note that here we need to reason about the *same* message in multiple modalities; we therefore need to extract the quantifier outside of the modalities.

Staying in the domain of communication protocols, we can also count messages.

> $\nu X(n\colon \mathbb{N} := 0).[receive]X(n+1) \wedge [send](n > 0 \wedge X(n-1)) \wedge [\overline{receive \cup send}]X(n)$ the number of messages sent is at most the number of messages received.

In this property, the parameter $n$ keeps track of the difference between the number of received and the number of sent messages. If a message is received, $n$ is incremented and the same property needs to hold again. If a message is sent, then $n$ must be positive, to denote that the message was received first, and then the property has to hold for $n - 1$. We also allow any other action using the last conjunct, and this does not influence the number of pending messages. ▲

**Acknowledgements** In composing this note I benefited from reading *Modelling Distributed Systems: Protocol Verification with $\mu$CRL* by Wan Fokkink [Fok11], and *Modelling and Analysis of Communicating Systems* by Jan Friso Groote and MohammadReza Mousavi [GM13]. Furthermore, I would like to thank Hans van den Bogert, Mark Janssen, Andrew Polonski and David Williams for helpful comments on earlier versions of these notes.

# References

[CGR11] Sjoerd Cranen, Jan Friso Groote, and Michel Reniers. A linear translation from CTL* to the first-order modal -calculus. *Theoretical Computer Science*, 412(28):3129–3139, June 2011.

[Fok11] W.J. Fokkink. *Modelling Distributed Systems: Protocol Verification with $\mu$CRL*. Springer, 2 edition, 2011.

[GM13] J.F. Groote and M.R. Mousavi. *Modelling and Analysis of Communicating Systems*. MIT Press, 2013. to appear.

[HM85] Matthew C B Hennessy and A.J.R.G (Robin) Milner. Algebraic laws for non-determinism and concurrency. *Journal of the ACM*, 32(1):137161, 1985.