Verifying Modal Formulas over I/O-Automata by means of Type Theory

M.P.A. Sellink
Department of Philosophy, Utrecht University
P.O. Box 80.126, 3508 TC Utrecht, The Netherlands
email: alex@phil.ruu.nl

Abstract

We introduce the notion of an I/O-automaton over a signature Σ. Beside we introduce a modal logic to reason about such an I/O-automaton. The semantics of the logic is defined in terms of a given Σ-algebra. We illustrate how the question whether or not an execution in a given I/O-automaton is a model for formula ϕ can be reduced to an inhabitation problem in the Calculus of Inductive Constructions. Furthermore we present a proof for soundness and completeness.

1 Introduction

There are several ways to reason about distributed systems. One way is to describe the behaviour of distributed systems in terms of transition systems (I/O-automata [12]). Understanding of the behaviour of the distributed system is obtained by deriving properties (invariants) of the transition system. In [9] these properties are expressed in first order predicate logic. An interesting idea is the addition of modal operators to this logic. This idea is not new [10, 8, 14]. The alluring thing about this modal operators is the expressive power of the formalism. It enables us to describe all kinds of properties like for instance 'a read action $r(d)$ will eventually be followed by a send action $s(d)$' in a very straightforward way. It can be very difficult to describe such a property in for instance an algebra based formalism like ACP [2]. The drawback of expressive power in general, however, is that expressive power of a formalism and complexity of its meta theory are usually exchangeable.

Proof theories, however, do not play a role in this paper. We focus on the problem of *model-checking*, i.e. the problem whether a transition system T, is a model for a given formula ϕ. It turns out to be possible to define a mapping $[\![\ ]\!]$ that maps logical expressions of the form $T \models \phi$ to types of the *Calculus of Inductive Constructions* [13] in such a way that inhabitation of that type implies the validity of $T \models \phi$. The construction of inhabitants can be carried out in the interactive proof construction program Coq [6], which is an implementation of this Calculus of Inductive Constructions. Doing so, we obtain

computer-checked — and hence much more reliable — proofs of properties of transition systems.

In order to give a precise description of the mapping $[\![\ ]\!]$ we have to formalise the assumption that the state space of an I/O-automaton is the cartesian product of a finite number of free data structures (like naturals, booleans, lists). This assumption is not a serious restriction. In many practical examples the state space actually is the cartesian product of a number of very simple data structures. Even in larger examples the underlying data structures remain simple, only the number of components in the cartesian product increases.

The formalisation of this adapted notion of I/O-automaton is done in Section 2. A modal logic, to express properties of such I/O-automata is also introduced in this section. Section 3 contains a brief explanation of inductive types. We assume that the reader is familiar with the notion of *Pure Type Systems* (PTS) as introduced by Terlouw and Berardi (independently) and restrict ourselves to some standard examples of inductive types. For an introduction to PTSs, see [1]. The embedding $[\![\ ]\!]$ is introduced in sections 4 and 5. A proof for soundness and completeness, is given in Section 6.

Acknowledgements. Thanks go to Inge Bethke, Marc Bezem and Jan Springintveld for comments on a draft version of this paper and to Jaco van de Pol, Marco Hollenberg, Jan Friso Groote, Marc Bezem and Inge Bethke and Jan Springintveld for helpful discussions.

2 A Modal Logic for I/O-Automata

A *sort* (set symbol) is a syntactic object. Let S be a non-empty finite set of sorts. We write S^* for the set of finite sequences of elements of S.

Definition 2.1. *An S-sorted signature Σ is a finite family of finite sets*

$$\langle\, \Sigma_{w_1,w_2} \mid w_1, w_2 \in S^* \,\rangle$$

such that the Σ_{w_1,w_2} are pairwise disjoint. An element $f \in \Sigma_{w_1,w_2}$ with $w_2 \neq \varepsilon$ (the empty word) is called a function symbol with domain w_1 and co-domain w_2. *When $w_1 \equiv \varepsilon$ then f is called a constant.*

We write Σ_w for $\bigcup_{w' \in S^*} \Sigma_{w',w}$. To each S-sorted signature Σ we relate a so-called *Sort Dependency Graph* (SDG). The set of nodes of the SDG equals the set of sorts. There is an edge from s_1 to s_2 iff $s_1 \neq s_2$ and there is a $w \in S^*$ and an $f \in \Sigma_{w,s_1}$ such that s_2 occurs in w. This notion will be used in Section 4 (Definition 4.1).

Definition 2.2. *Let Σ be an S-sorted signature. A Σ-algebra is a pair $\langle\, A, F \,\rangle$ with*

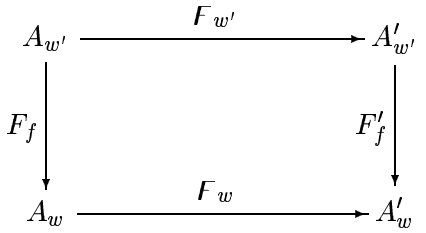
- *A is a family of carrier sets $\langle\, A_s \mid s \in S \,\rangle$,*
- *F is a family of functions $\langle\, F_f : A_{w_1} \longrightarrow A_{w_2} \mid f \in \Sigma_{w_1,w_2} \,\rangle$,*

where $A_x = \prod_{i=1}^{n} A_{x_i}$ if $x \equiv x_1 \cdots x_n$.

Furthermore we assume that we have a family $\mathbb{V} = \langle\, \mathbb{V}_w \mid w \in S^* \,\rangle$ of sets of variables. $\mathbb{V}_w$ is the set of variables of type w.

Definition 2.3. *Let Σ, Σ' be S-sorted signatures with $\Sigma \subseteq \Sigma'$. Let $\mathcal{A} = \langle\, A, F \,\rangle$ be a Σ-algebra and let $\mathcal{A}' = \langle\, A', F' \,\rangle$ be a Σ'-algebra. A collection of mappings $\mathit{F} = \langle\, \mathit{F}_s \mid s \in S \,\rangle$ with $\mathit{F}_s : A_s \longrightarrow A'_s$ for all $s \in S$ is called an isomorphism from $\mathcal{A}$ to $\mathcal{A}'$ iff*

- $\mathit{F}_s : A_s \longrightarrow A'_s$ *is a bijection for all $s \in S$*
- *The following diagram commutes for all $f \in \Sigma_{w',w}$.*



i.e. $\mathit{F}_w \circ F_f = F'_f \circ \mathit{F}_{w'}$ for all $f \in \Sigma_{w',w}$. Here $\mathit{F}_{s_1 \cdots s_n} \stackrel{\text{def}}{=} \prod_{i=1}^n \mathit{F}_{s_i}$ where $\prod_{i=1}^n g_i$ is defined by $(x_1, \ldots, x_n) \longmapsto (g_1(x_1), \ldots, g_n(x_n))$.

In the literature, isomorphisms are usually mappings between two algebras over the same signature. When F is an isomorphism from $\mathcal{A}$ to $\mathcal{A}'$ then F is an isomorphism in the traditional sense between $\mathcal{A}$ and the restriction of $\mathcal{A}'$ to Σ.

Definition 2.4. *The set $\mathcal{T}_w(\Sigma)$ of Σ-terms of type $w = s_1 \cdots s_n \in S^*$ is defined inductively as follows:*

- $\mathbb{V}_w \subseteq \mathcal{T}_w(\Sigma)$,
- $(t_1, \ldots, t_n) \in \mathcal{T}_w(\Sigma)$ *whenever $t_i \in \mathcal{T}_{s_i}(\Sigma)$ for all $i = 1, \ldots, n$,*
- $f(t) \in \mathcal{T}_w(\Sigma)$ *whenever $f \in \Sigma_{w',w}$ and $t \in \mathcal{T}_{w'}(\Sigma)$.*

Furthermore we define $\mathcal{T}(\Sigma) = \bigcup_{w \in \mathcal{S}^*} \mathcal{T}_w(\Sigma)$.

The set $\mathrm{Fv}(t)$ of free variables of t is defined in the usual way. A function $\xi_w : \mathbb{V}_w \longrightarrow A_w$ which assigns values to variables is called a *valuation*. We write $\xi_w[x \mapsto a]$ for the valuation defined by

$$\xi_w[x \mapsto a](y) = \begin{cases} a & \text{when } y \equiv x \\ \xi_w(y) & \text{otherwise} \end{cases}$$

Let $\xi = \langle\, \xi_w \mid w \in S^* \,\rangle$ with $\xi_w : V_w \longrightarrow A_w$ be a family of valuations. Given a valuation ξ, every term of type $\mathcal{T}_w(\Sigma)$ can be mapped (interpreted) into A_w in a canonical way. This interpretation $[\![\ ]\!]_{\mathcal{A},\xi}$ under valuation ξ is defined inductively:

$$\begin{array}{rcl} x_w & \longmapsto & \xi_w(x_w) \\ (t_1, \ldots, t_n) & \longmapsto & ([\![t_1]\!]_{\mathcal{A},\xi}, \ldots, [\![t_n]\!]_{\mathcal{A},\xi}) \\ f(t) & \longmapsto & F_f([\![t]\!]_{\mathcal{A},\xi}). \end{array}$$

Next we inductively define the set of formulas in the presence of modal operators. We follow the approach of [4] where only the binary operators $\mathcal{U}$ (*until*) and $\mathcal{S}$ (*since*) are primitive. All other modal operators are defined in terms of these two.

Definition 2.5. *Let $w, \varpi \in S^*$. The set $\mathcal{F}(\Sigma, \varpi)$ of Σ-formulas over ϖ is defined inductively as follows:*

- $\bot \in \mathcal{F}(\Sigma, \varpi)$.
- $t_1 \stackrel{w}{=} t_2 \in \mathcal{F}(\Sigma, \varpi)$ *for all* $t_1, t_2 \in \mathcal{T}_w(\Sigma)$.
- $\phi \Rightarrow \psi \in \mathcal{F}(\Sigma, \varpi)$ *for all* $\phi, \psi \in \mathcal{F}(\Sigma, \varpi)$.
- $\downarrow x \,.\, \phi \in \mathcal{F}(\Sigma, \varpi)$ *for all* $\phi \in \mathcal{F}(\Sigma, \varpi)$ *and for all* $x \in \mathbb{V}_\varpi$.
- $\phi \,\mathcal{U}\, \psi \in \mathcal{F}(\Sigma, \varpi)$ *for all* $\phi, \psi \in \mathcal{F}(\Sigma, \varpi)$.
- $\phi \,\mathcal{S}\, \psi \in \mathcal{F}(\Sigma, \varpi)$ *for all* $\phi, \psi \in \mathcal{F}(\Sigma, \varpi)$.

In a well-formed $\mathcal{F}(\Sigma, \varpi)$-formula the $\downarrow$-binder (which is pronounced as 'here') is not allowed to bind variables of type w with $w \not\equiv \varpi$. This $\downarrow$-binder is used to construct so-called *state dependent formulas*, i.e. formulas that depend on specific points in time. A similar idea can be found in [3], where the collection of atomic formulas is enriched with so-called *nominals* (formulas that hold in exactly one time-point) in order to increase the expressive power of the formalism. We will say more about the $\downarrow$-binder later.

The following abbreviations are used (unary operators bind stronger than binary ones):

$\neg\phi$	:=	$\phi \Rightarrow \bot$	not ϕ
$\top$	:=	$\neg\bot$	the true proposition
$\phi \vee \psi$	:=	$\neg\phi \Rightarrow \psi$	ϕ or ψ
$\phi \wedge \psi$	:=	$\neg(\phi \Rightarrow \neg\psi)$	ϕ and ψ
$\Diamond\phi$	:=	$\top \,\mathcal{U}\, \phi$	eventually ϕ
$\Box\phi$	:=	$\neg\Diamond\neg\phi$	always in the future ϕ
$\bigcirc\phi$	:=	$\bot \,\mathcal{U}\, \phi$	next ϕ
$\blacklozenge\phi$	:=	$\top \,\mathcal{S}\, \phi$	sometimes in the past ϕ
$\blacksquare\phi$	:=	$\neg\blacklozenge\neg\phi$	always in the past ϕ
$\bullet\phi$	:=	$\bot \,\mathcal{S}\, \phi$	previous ϕ
$\nabla\phi$	:=	$\phi \wedge (\neg\phi \,\mathcal{S}\, \neg\phi)$	just ϕ

As usual in modal logics, we define the semantics of a formula in terms of a collection of so-called worlds or *states* and an accessibility relation on these states. In our case the accessibility relation will be a linear order. These linear ordered collection of states will precisely correspond to the executions of an I/O-automaton. Different from e.g. [4] the validity of atomic formulas is not simply postulated by a valuation function.

Definition 2.6. *Suppose that Σ is an S-sorted signature. A Σ-automaton is a tuple $\langle \varpi, \textsc{Act}, \Im, \longrightarrow\!\!\triangleright \rangle$ where*

- $\varpi \in S^*$ *represents the state space,*
- ACT *is a set of actions,*
- $\Im \subseteq \mathcal{T}_\varpi(\Sigma)$ *represents the set of initial states,*
- $\longrightarrow\!\triangleright \subseteq \mathcal{T}_\varpi(\Sigma) \times \text{ACT} \times \mathcal{T}_\varpi(\Sigma)$ *represents the a transition relation.*

Usually an I/O-automaton is also equipped with an equivalence relation used to define *fairness*. Since we do not consider fairness in this paper we dropped this equivalence relation.

We write $x \xrightarrow{a}\!\triangleright\, y$ for $\longrightarrow\!\triangleright(x, a, y)$. Closed terms of type ϖ are called *states*.

An *execution* σ of an Σ-automaton is a possibly infinite sequence

$$\sigma = [s_0\ a_1\ s_1\ a_2\ s_2 \cdots\cdots a_n\ s_n \cdots\cdots]$$

such that $s_0 \in \Im$ and $s_{i-1} \xrightarrow{a_i}\!\triangleright\, s_i$ for all $i = 1, 2, \ldots$. Let $\ell(\sigma) \leq \omega$ denote the length of σ. When we drop the requirement $s_0 \in \Im$ then σ is called an *execution fragment*. Let

$$\sigma' = [s_{\ell(\sigma)}\ a'_1\ s'_1\ a'_2\ s'_2 \cdots a_{\ell(\sigma')}\ s_{\ell(\sigma')}]$$

be an execution fragment. The execution

$$[s_0\ a_1\ s_1\ a_2\ s_2 \cdots a_{\ell(\sigma)}\ s_{\ell(\sigma)}\ a'_1\ s'_1\ a'_2\ s'_2 \cdots a_{\ell(\sigma')}\ s_{\ell(\sigma')}]$$

is denoted as $\sigma\hat{}\sigma'$. Furthermore we define projection functions π_n for $n \geq 0$ such that $\pi_n(\sigma) = s_n$ for all $n = 0, \ldots, \ell(\sigma)$. We write $\mathcal{E}(\mathsf{A})$ for the set of all executions of Σ-automaton A. Furthermore $\Re \subseteq \mathcal{T}_\varpi(\Sigma)$ is the set of *reachable states*, i.e. the least set, closed under $\longrightarrow\!\triangleright$, such that $\Im \subseteq \Re$.

Validity is defined with respect to a position in an execution. Let $\mathcal{A}$ be a Σ-algebra, A a Σ-automaton, $\sigma \in \mathcal{E}(\mathsf{A})$ and $n \in \mathbb{N}$. Then we write $\mathsf{A}, \sigma, n \models_{\mathcal{A}} \phi$ for 'ϕ holds on the n-th position of σ with respect to $\mathcal{A}$'.

Definition 2.7. *Let* A *be a* Σ_{A}*-automaton for some* $\Sigma_{\mathsf{A}} \subseteq \Sigma$ *and let* $\sigma \in \mathcal{E}(\mathsf{A})$ *be an execution. Let* $[\![\ ]\!]_{\mathcal{A},\xi}$ *interpret* $\mathcal{T}(\Sigma)$ *in some* Σ*-algebra* $\mathcal{A}$ *with valuation* ξ *and let* ϖ *represent the state space of* A*. Then validity of* $\phi \in \mathcal{F}(\Sigma, \varpi)$ *is defined by induction over the structure of* ϕ*.*

$$
\begin{array}{lll}
\mathsf{A}, \sigma, n, \xi \not\models_{\mathcal{A}} \text{\L} & & \\
\mathsf{A}, \sigma, n, \xi \models_{\mathcal{A}} t_1 \stackrel{s}{=} t_2 & \textit{iff} & [\![t_1]\!]_{\mathcal{A},\xi} = [\![t_2]\!]_{\mathcal{A},\xi} \\
\mathsf{A}, \sigma, n, \xi \models_{\mathcal{A}} \phi \Rightarrow \psi & \textit{iff} & \mathsf{A}, \sigma, n, \xi \models_{\mathcal{A}} \phi \ \textit{implies}\ \mathsf{A}, \sigma, n, \xi \models_{\mathcal{A}} \psi \\
\mathsf{A}, \sigma, n, \xi \models_{\mathcal{A}} \downarrow x_\varpi \,.\, \phi & \textit{iff} & \mathsf{A}, \sigma, n, \xi \models \phi[x_\varpi := \pi_n(\sigma)] \\
\mathsf{A}, \sigma, n, \xi \models_{\mathcal{A}} \phi \,\mathcal{U}\, \psi & \textit{iff} & \textit{there exists an } m > n \textit{ such that } \quad \mathsf{A}, \sigma, m, \xi \models_{\mathcal{A}} \psi \\
 & & \textit{and} \quad \mathsf{A}, \sigma, i, \xi \models_{\mathcal{A}} \phi \quad \textit{for all } n < i < m \\
\mathsf{A}, \sigma, n, \xi \models_{\mathcal{A}} \phi \,\mathcal{S}\, \psi & \textit{iff} & \textit{there exists an } m < n \textit{ such that } \quad \mathsf{A}, \sigma, m, \xi \models_{\mathcal{A}} \psi \\
 & & \textit{and} \quad \mathsf{A}, \sigma, i, \xi \models_{\mathcal{A}} \phi \quad \textit{for all } m < i < n
\end{array}
$$

Note that $\pi_n(\sigma)$ indeed is an object in the language, so the substitution $[x_\varpi := \pi_n(\sigma)]$ in the semantics for $\downarrow x_\varpi \,.\, \phi$ is syntactically correct. We could also have written

$$\mathsf{A}, \sigma, n, \xi \models_{\mathcal{A}} \downarrow x_\varpi \,.\, \phi \;\text{ iff }\; \mathsf{A}, \sigma, n, \xi[x_\varpi \mapsto [\![\pi_n(\sigma)]\!]_{\mathcal{A},\xi}] \models \phi$$

When one or more components of the expression $\mathsf{A}, \sigma, n, \xi \models_{\mathcal{A}} \phi$ are left out, we assume by convention that the expression is universally quantified over them. When ϕ is a closed formula then we can omit ξ since the validity holds iff it holds for every valuation ξ. When n is omitted it means that validity holds for all $n \leq \ell(\sigma)$ and we say that the formula holds for σ. Usually we are interested in the question whether or not a closed formula ϕ holds for every $\sigma \in \mathcal{E}(\mathsf{A})$. In this case we call A a *model* for ϕ and we will write $\mathsf{A} \models_{\mathcal{A}} \phi$. Finally, $\models_{\mathcal{A}} \phi$ means that $\mathsf{A} \models_{\mathcal{A}} \phi$ for all Σ-automata of the form $\langle\, \varpi, \_, \_, \_ \,\rangle$.

A formula ϕ such that $\mathsf{A}, \sigma, n \models_{\mathcal{A}} \phi$ iff $n \in M$ for some $\varnothing \subset M \subset \mathbb{N}$ is called a *state dependent formula*. ($\subset$ as opposed to $\subseteq$ stands for strict inclusion.) The typical use of the $\downarrow$-binder is to build state dependent formulas. It is, however, completely obvious that the use of the $\downarrow$-binder does not guarantee state dependentness. E.g. consider $\models_{\mathcal{A}} \downarrow x \,.\, \mathtt{T}$. The converse also does not hold, i.e. absence of the $\downarrow$-binder does not guarantee state independentness. For instance $\mathsf{A}, \sigma, n \models_{\mathcal{A}} \mathtt{T}\, \mathcal{S}\, \mathtt{T}$ iff $n > 0$.

Example 2.8. Let $S = \{N\}$, $\Sigma_{\varepsilon,N} = \{o\}$, $\Sigma_{N,N} = \{s\}$, $\Sigma_{NN,N} = \{sum\}$ and $\Sigma_{w,N} = \varnothing$ otherwise. Define $\Sigma = \langle\, \Sigma_{w,N} \mid w \in N^* \,\rangle$ and let $\mathsf{A} = \langle\, \varpi, \textsc{Act}, \Im, \longrightarrow\!\!\triangleright \,\rangle$ be a Σ_{A}-automaton, where Σ_{A} is the signature that is obtained from Σ by removing the function symbol sum, $\textsc{Act} = \{a\}$, $\Im = \{o\}$ and $\longrightarrow\!\!\triangleright \;= \{(w, a, s(w)) \mid w \in N\}$. Define

$$\begin{aligned}
\phi_1 &\equiv sum\,(o, n) \stackrel{N}{=} n \\
\phi_2 &\equiv sum\,(s(m), n) \stackrel{N}{=} s(sum\,(m, n)) \\
\psi &\equiv \Diamond \downarrow x \,.\, x \stackrel{N}{=} s(s(o))
\end{aligned}$$

Let $\mathcal{A} = \langle\, \mathbb{N}, F \,\rangle$ with $F_o = 0$, $F_s : x \longmapsto x + 1$ and $F_{sum} = +$ then $\mathcal{A}$ is a Σ-algebra and $\mathsf{A}, \sigma, n \models_{\mathcal{A}} \phi_i$ for $i = 1, 2$ and for all σ, n. We have $\mathsf{A}, \sigma, n \models_{\mathcal{A}} \phi_1$ since

$$\begin{array}{ll}
\mathsf{A}, \sigma, n, \xi \models_{\mathcal{A}} sum\,(o, n) \stackrel{N}{=} n & \Longleftrightarrow \\
[\![sum\,(o, n)]\!]_{\mathcal{A},\xi} = [\![n]\!]_{\mathcal{A},\xi} & \Longleftrightarrow \\
[\![sum\,]\!]_{\mathcal{A},\xi}([\![o]\!]_{\mathcal{A},\xi}, [\![n]\!]_{\mathcal{A},\xi}) = [\![n]\!]_{\mathcal{A},\xi} & \Longleftrightarrow \\
F_{sum}(F_o, \xi(n)) = \xi(n) & \Longleftrightarrow \\
0 + \xi(n) = \xi(n) &
\end{array}$$

for all valuations ξ, for all $\sigma \in \mathcal{E}(\mathsf{A})$ and for all $n \in \mathbb{N}$. Similarly, $\mathsf{A}, \sigma, n \models_{\mathcal{A}} \phi_2$ holds. Furthermore we have that $\mathsf{A}, \sigma, 0 \models_{\mathcal{A}} \psi$ for all σ such that $\ell(\sigma) \geq 2$ since

$$\begin{array}{ll}
\mathsf{A}, \sigma, 0, \xi \models_{\mathcal{A}} \Diamond \downarrow x \,.\, x \stackrel{N}{=} s(s(o)) & \Longleftrightarrow \\
\mathsf{A}, \sigma, 0, \xi \models_{\mathcal{A}} \mathtt{T}\, \mathcal{U} \downarrow x \,.\, x \stackrel{N}{=} s(s(o)) & \Longleftrightarrow \\
\text{there exists an } m > 0 \text{ such that} & \\
\quad \text{(i)}: \; \mathsf{A}, \sigma, i, \xi \models_{\mathcal{A}} \mathtt{T} \quad \text{for all } 0 < i < m \text{ and} & \\
\quad \text{(ii)}: \; \mathsf{A}, \sigma, m, \xi \models_{\mathcal{A}} \downarrow x \,.\, x \stackrel{N}{=} s(s(o)) &
\end{array}$$

Take $m = 2$ then (i) is trivial and (ii) holds since $\mathsf{A}, \sigma, 2, \xi \models_{\mathcal{A}} \downarrow x \,.\, x \stackrel{N}{=} s(s(o))$ iff $\mathsf{A}, \sigma, 2, \xi \models_{\mathcal{A}} x \stackrel{N}{=} s(s(o))[x := \pi_2(\sigma)]$ which holds by reflexivity because $\pi_2(\sigma) \equiv s(s(o))$ for all $\sigma \in \mathcal{E}(\mathsf{A})$ with $\ell(\sigma) \geq 2$. This easy fact can be proved formally by a tedious induction over $\mathcal{E}(\mathsf{A})$ and $\longrightarrow\!\!\triangleright$.

Definition 2.9. *Let Σ be an S-sorted signature. The closed term algebra $\mathtt{TERM}(\Sigma)$ for Σ is the Σ-algebra $\langle\, \mathtt{A}, \mathtt{F} \,\rangle$ with*

- $\mathtt{A} = \langle\, \mathtt{A}_s \mid s \in S \,\rangle$ *and $\mathtt{A}_s$ is the set of closed terms of type s,*
- $\mathtt{F} = \langle\, \mathtt{F}_f \mid f \in \Sigma_{w,w'} \,\rangle$ *with $\mathtt{F}_f(t) = f(t)$ for all $t \in \mathcal{T}_w(\Sigma)$, $f \in \Sigma_{w,w'}$.*

Definition 2.10. *Let $\mathcal{A}$ be a Σ'-algebra. A signature $\Sigma \subseteq \Sigma'$ is called a constructor signature of Σ' with respect to $\mathcal{A}$ iff there is an isomorphism $\mathit{F} : \mathtt{TERM}(\Sigma) \longrightarrow \mathcal{A}$*

In Example 2.8, Σ_{A} is a constructor signature of Σ with respect to $\mathcal{A} = \langle\, \mathbb{N}, F \,\rangle$, since $\mathit{F}_N : \mathtt{A}_N \longrightarrow \mathbb{N}$ via $\underbrace{s(\cdots\cdots(s}_{n}(o))\cdots) \longmapsto n$ satisfies the conditions of Definition 2.3.

Example 2.11. Let $S = \{Z\}$, $\Sigma_{\varepsilon,Z} = \{o\}$, $\Sigma_{Z,Z} = \{p, s\}$ and $\Sigma_{w,w'} = \varnothing$ otherwise, then $\Sigma = \langle\, \Sigma_{w,w'} \mid w, w' \in Z^* \,\rangle$ is an S-sorted signature. Let $\mathcal{A} = \langle\, A, F \,\rangle$ with

$$\begin{array}{lcl} A_Z & := & \mathbb{Z} \\ F_o & := & 0 \\ F_p & := & z \longmapsto z - 1 \\ F_s & := & z \longmapsto z + 1 \end{array}$$

then $\mathcal{A}$ is a Σ-algebra. Now there is no constructor signature of Σ with respect to $\mathcal{A}$ since on the one hand $\mathit{F} : \mathtt{TERM}(\Sigma') \longrightarrow \mathcal{A}$ is not injective when $p, s \in \Sigma'$ and on the other hand the surjectivity of F is lost as soon as p or s is removed from Σ'.

3 Inductive Type Theory

The Calculus of Inductive Constructions [13] extends the well-known system λC [5], which is the strongest system of the Barendregt-cube [1], in three ways:

1. The type $*$ of all types is divided into $*^s$ (the type of all sets) and $*^p$ (the type of all propositions).
2. The top sorts are replaced by hierarchies of sorts, like in Luo's ECC [11].
3. So-called *inductive types* are added.

These inductive types enable the user to build all kinds of data structures as well as notions like cartesian products, logical conjunction and disjunction, Leibniz equality, the existential quantifier etc. The basic idea of inductive type theory is that one can form a term

$$\mathsf{Ind}(X : A)\{C_1 \mid \cdots \mid C_n\} \quad , n \in \mathbb{N} \tag{1}$$

of type A under the meta-condition that $C_1, \ldots, C_n$ are so-called *constructor types* for X. Here 'meta-condition' means that the condition is not derivable in the type system itself. The intuitive meaning of 'being a constructor type for X' is that inhabitants of such type are usable to construct inhabitants of X. Inductive types are introduced via the introduction rule

$$\frac{\Gamma \vdash I : A}{\Gamma \vdash \mathsf{Constr}(i, I) : C_i[X := I]}\, 1 \leq i \leq n$$

where $I \equiv \mathsf{Ind}(X : A)\{C_1 \mid \cdots \mid C_n\}$ is an inductive term. In order to derive the premise (introduction of inductive terms) of this rule we have to check the meta-properties $C_1, \ldots, C_n \in \mathrm{CT}(X)$ (where $\mathrm{CT}(X)$ is the set of constructor types of X).

Example 3.1. Abbreviate $\mathsf{Ind}(X : *^s)\{X \mid X \to X\}$ by $\mathbf{N}$ then

$$\frac{\dfrac{\langle\, X : *^* \,\rangle \vdash X : *^s \quad \langle\, X : *^* \,\rangle \vdash X \to X : *^s}{\langle\ \rangle \vdash \mathbf{N} : *^s}\, \{X, X \to X\} \subseteq \mathrm{CT}(X)}{\langle\ \rangle \vdash \mathsf{Constr}(1, \mathbf{N}) : \mathbf{N} \quad \text{and} \quad \langle\ \rangle \vdash \mathsf{Constr}(2, \mathbf{N}) : \mathbf{N} \to \mathbf{N}}$$

and now for instance

$$\mathsf{Constr}(2, \mathbf{N})\ (\mathsf{Constr}(2, \mathbf{N})\ (\mathsf{Constr}(2, \mathbf{N})\ (\mathsf{Constr}(1, \mathbf{N})))) \tag{2}$$

represents the natural number 3. In the sequel we will abbreviate $\mathsf{Constr}(1, \mathbf{N})$ by $\underline{0}$ and $(\mathsf{Constr}(2, \mathbf{N})\ \underline{n})$ by $\underline{n+1}$. For instance, (2) will be abbreviated by $\underline{3}$. The elimination rule for inductive types provides the required induction principles. In case of the inductive type $\mathbf{N}$ this elimination rule says that

$$\Gamma \vdash \mathsf{Elim}(n, P)\{f_o \mid f_s\} : P\ n$$

whenever Γ proves (for some sort ς)

$$\begin{array}{lcl} n & : & \mathbf{N} \\ P & : & \mathbf{N} \to \varsigma \\ f_o & : & P\ \mathsf{Constr}(1, \mathbf{N}) \\ f_s & : & \Pi x{:}\mathbf{N}\,.\, P\,x \to P\ (\mathsf{Constr}(2, \mathbf{N})\ x) \end{array}$$

An extra reduction, called *ι-reduction*, is added. Similar to the fact that β-reduction corresponds to an (*introduction*, *elimination*)-pair of a Π-type, the ι-reduction corresponds to an (*introduction*, *elimination*)-pair of an inductive type. Example:

$$\frac{(\cdots) \quad \Gamma \vdash f_o : P\ \mathsf{Constr}(1, \mathbf{N}) \qquad \dfrac{\Gamma \vdash \mathbf{N} : *^s}{\Gamma \vdash \mathsf{Constr}(1, \mathbf{N}) : \mathbf{N}}\ \textit{introduction}}{\Gamma \vdash \mathsf{Elim}(\mathsf{Constr}(1, \mathbf{N}), P)\{f_o \mid f_s\} : P\ \mathsf{Constr}(1, \mathbf{N})}\ \textit{elimination}$$

so in this case $\mathsf{Elim}(\mathsf{Constr}(1, \mathbf{N}), P)\{f_o \mid f_s\} \to_\iota f_o$. Note that this reduction preserves types.

We can build a large variety of mathematical concepts with this machinery. In particular, by choosing $P \equiv \lambda z{:}\mathbf{N}\,.\,\mathbf{N}$ of type $\mathbf{N} \to *^s$ we can construct primitive recursive functions. For instance

$$\oplus := \lambda x, y{:}\mathbf{N}\,.\,\mathsf{Elim}(x, P)\{y \mid \lambda x, y{:}\mathbf{N}\,.\,y\}$$

satisfies $\oplus\ \underline{n}\ \underline{m} \twoheadrightarrow_{\beta\iota} \underline{n+m}$. Similarly, we can define a λ-term $\ominus$ representing 'cut-off-subtraction', i.e. $\ominus\ \underline{n}\ \underline{m} \twoheadrightarrow_{\beta\iota} \underline{n \dot{-} m}$ where

$$n \dot{-} m = \begin{cases} n - m & \text{if } n \geq m \\ 0 & \text{if } n < m \end{cases}$$

In the sequel we will use infix notations for $\oplus$ and $\ominus$.

The rest of this section consists of a list of inductively defined types that are frequently used in this paper.

The generalised cartesian product

Let $\times_n : \mathsf{SET}^n \longrightarrow \mathsf{SET}$ be the generalised cartesian product operator, i.e.

$$\times_n : (A_1, \ldots, A_n) \longmapsto \prod_{i=1}^{n} A_i.$$

This operator can be represented by

$$\mathsf{x}_n \equiv \lambda A_1, \ldots, A_n{:}*^s\,.\,\mathsf{Ind}(X : *^s)\{A_1 \to \cdots \to A_n \to X\} \tag{3}$$

The term $\mathsf{Constr}(1, \mathsf{Ind}(X : *^s)\{A_1 \to \cdots \to A_n \to X\})\, a_1 \cdots a_n$ which lives in context $\langle\, A_i : *^s\ ,\ a_i : A_i \,\rangle_{i=1,\ldots,n}$ will be denoted as $< a_1, \ldots, a_n >$. The (closed) term

$$\begin{array}{l} \lambda A_1, \ldots, A_n{:}*^s\,. \\ \quad \lambda q{:}\mathsf{Ind}(X : *^s)\{A_1 \to \cdots \to A_n \to X\}\,. \\ \qquad \mathsf{Elim}(q, \lambda z{:}\mathsf{Ind}(X : *^s)\{A_1 \to \cdots \to A_n \to X\}\,.\,A_i)\{\lambda x_1{:}A_1\,.\ldots.\,\lambda x_n{:}A_n\,.\,x_i\} \end{array}$$

behaves like the polymorphic i-th projection function.

We write $(A_1 \times \cdots \times A_n)$ for $(\mathsf{x}_n\ A_1 \cdots A_n)$ so

$$\langle\, A_i : *^s\ ,\ a_i : A_i \,\rangle_{i=1,\ldots,n} \vdash < a_1, \ldots, a_n > : A_1 \times \cdots \times A_n.$$

Conjunction and disjunction

Define

$$\curlywedge := \lambda A, B{:}*^p\,.\,\mathsf{Ind}(X : *^p)\{A \to B \to X\}.$$

Closed inhabitants of $(\curlywedge\ A\ B)$ are necessarily of the form $(\mathsf{Constr}(1, \curlywedge\ A\ B)\ a\ b)$ where a and b are inhabitants of A and B respectively. In other words: The only way to build proofs for $(\curlywedge\ A\ B)$ is to build proofs for A and B and apply $\mathsf{Constr}(1, \curlywedge\ A\ B)$ to these proofs. We'll write $(A \curlywedge B)$ for $(\curlywedge\ A\ B)$.

Similarly, we write $(A \curlyvee B)$ for $(\curlyvee\ A\ B)$, where

$$\curlyvee \equiv \lambda A, B{:}*^p\,.\,\mathsf{Ind}(X : *^p)\{A \to X \mid B \to X\}.$$

The false proposition

The number of constructors in (1) is a natural number, so it is perfectly allowed to have no constructors at all. For instance

$$\bot := \mathsf{Ind}(X : *^p)\{\ \} \tag{4}$$

is the term of type $*^p$ such that closed inhabitants are necessarily of the form

$$\mathsf{Constr}(i, \bot)\ \ldots$$

for some $1 \leq i \leq 0$. This is equivalent to saying that there are no closed inhabitants of $\bot$. The name $\bot$ is justified by the fact that we can construct a proof object for an arbitrary inhabitant of $*^p$, from an inhabitant of $\bot$:

$$\langle\ \rangle \vdash \lambda\alpha{:}*^p\,.\,\lambda f{:}\bot\,.\,\mathsf{Elim}(f, P)\{\ \} : \Pi\alpha{:}*^p\,.\,\bot \to (P\,f)$$

so when we substitute $\lambda z{:}\bot\,.\,\alpha$ for P, we obtain a proofterm for $\Pi\alpha{:}*^p\,.\,\bot \to \alpha$. We will abbreviate $\bot \to \bot$ by $\top$. Let $\mathrm{id}_\bot := \lambda x{:}\bot\,.\,x$ then $\langle\ \rangle \vdash \mathrm{Id}_\bot : \top$.

The $\leq$-relation on natural numbers

We can, for every $n \in \mathbb{N}$, represent the predicate $\geq_n$ on natural numbers by an inductive type, for $\geq_n$ is the weakest predicate that holds in n and that is closed under successor: $\geq_n (x) \implies \geq_n (x+1)$. The binary relation $\lambda\!\!\lambda n.\, \geq_n$ defined by

$$\lambda\!\!\lambda n.\, \geq_n (x, y) \iff \geq_x (y),$$

which is of course just the ordinary $\leq$-relation, can be represented by

$$\preceq := \lambda n{:}\mathbf{N}\,.\,\mathsf{Ind}(X : \mathbf{N} \to *^p)\{X\ n \mid \Pi m{:}\mathbf{N}\,.\,X\ m \to X\ (\mathsf{Constr}(2, \mathbf{N})\ m)\}.$$

The 'meta-λ' symbol $\lambda\!\!\lambda$ is used to avoid confusion with the λ-binder of the type system. The meaning is the same: $\lambda\!\!\lambda x\,.\,x+y$ is the function 'add y' and $\lambda\!\!\lambda y\,.\,x+y$ is the function 'add x'. So e.g. $(\lambda\!\!\lambda x\,.\,x+y)(z) = z+y$, which is similar to β-reduction. We write $(a_1 \preceq a_2)$ for $(\preceq\ a_1\ a_2)$.

Leibniz equality

Define

$$=_{\mathcal{L}}\ := \lambda A{:}*^s\,.\,\lambda a{:}A\,.\,\mathsf{Ind}(X : A \to *^s)\{X\,a\} \tag{5}$$

We write $(a_1 =_{\mathcal{L}} a_2)$ for $(=_{\mathcal{L}}\ A\ a_1\ a_2)$. Inductive types generate *free structures*. This means that inhabitants of an inductive type, that are constructed from different constructors, are not Leibniz equal. For instance the predicate

$$\mathcal{Z} := \lambda x{:}\mathbf{N}\,.\,\mathsf{Elim}(x, \lambda z{:}\mathbf{N}\,.\,*^p)\{\top \mid \lambda y{:}\mathbf{N}\,.\,\lambda P{:}*^p\,.\,\bot\}$$

distinguishes between natural numbers constructed from the first and from the second constructor, $\mathcal{Z}\ \underline{0} =_{\beta\iota} \top$ and $\mathcal{Z}\ \underline{n+1} =_{\beta\iota} \bot$. This predicate $\mathcal{Z}$ can be used to build an

inhabitant of an arbitrary φ of type $*^p$ in the presence of a proofterm $H : \underline{0} =_{\mathcal{L}} \underline{n+1}$. Almost by construction the equality $=_{\mathcal{L}}$ satisfies the substitution property

$$\frac{a_1 =_{\mathcal{L}} a_2 \qquad \phi(a_1)}{\phi(a_2)}$$

since $\Gamma \vdash \mathsf{Elim}(H, P)\{f\} : P\ a_2$ whenever $\Gamma \vdash f : P\ a_1$ and $\Gamma \vdash H : a_1 =_{\mathcal{L}} a_2$. The constructor of $=_{\mathcal{L}}$ is a proofterm for polymorphic reflexivity:

$$\langle\ \rangle \vdash \lambda A{:}*^s\,.\,\lambda a{:}A\,.\,\mathsf{Constr}(1, =_{\mathcal{L}}\ A\ a) : \Pi A{:}*^s\,.\,\Pi a{:}A\,.\,a =_{\mathcal{L}} a.$$

The existential quantifier

Define

$$\mathbf{E} := \lambda A{:}*^s\,.\,\lambda P{:}A \to *^p\,.\,\mathsf{Ind}(X : *^p)\{\Pi x{:}A\,.\,(P\ x) \to X\} \tag{6}$$

and let ϕ be a term of type $*^p$ possibly containing a variable x of type A for some A of type $*^s$. An inhabitant of $(\mathbf{E}\ A\ \lambda x{:}A\,.\,\phi)$ is necessarily of the form $(\mathsf{Constr}(1, \mathbf{E}\ A\ \lambda x{:}A\,.\,\phi)\ a\ p)$ where a is of type A and p is of type $\phi[x := a]$. Consequently we can (from the intuitionistic point of view) see $(\mathbf{E}\ A\ \lambda x{:}A\,.\,\phi)$ as a representation of $\exists x \in A\,.\,\phi$.

4 The Implementation of Terms and Formulas

We are interested in the validity of judgements of the form $\mathsf{A}, \sigma, n \models_{\mathcal{A}} \phi$. Our strategy is to associate such judgements with terms of type $*^p$ in such a way that inhabitance of the associated term implies validity of the judgement and vice versa. In this section we define a mapping $[\![\ ]\!]_{\mathcal{A},\mathsf{A}}$, parameterised by $\mathcal{A}$ and A, that maps all concepts from Section 2 to terms of the Calculus of Inductive Constructions. This mapping will be such that

$$[\![\mathsf{A}, \sigma, n \models_{\mathcal{A}} \phi]\!]_{\mathcal{A},\mathsf{A}} := [\![\phi]\!]_{\mathcal{A},\mathsf{A}}\ \ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}.$$

The interpretation $[\![\phi]\!]_{\mathcal{A},\mathsf{A}}$ will be defined with induction over the structure of ϕ. Therefore we need interpretations $[\![\to]\!]_{\mathcal{A},\mathsf{A}}$, $[\![\Box]\!]_{\mathcal{A},\mathsf{A}}$ etc. for the operators and interpretations $[\![\bot]\!]_{\mathcal{A},\mathsf{A}}$ and $[\![t_1 \stackrel{s}{=} t_2]\!]_{\mathcal{A},\mathsf{A}}$ for the atomic formulas.

To begin, we define a mapping $[\![\ ]\!]$ that maps the sorts of a signature Σ with a well-founded SDG to terms of type $*^s$, in such a way that

$$\text{closed normal forms of type } [\![\mathcal{T}_w(\Sigma)]\!] \quad \cong \quad \text{closed terms of } \mathcal{T}_w(\Sigma). \tag{7}$$

This mapping does not depend on any Σ-algebra or I/O-automaton.

Definition 4.1. *Let Σ be an S-sorted signature with a well-founded* SDG. *Let $s \in S$ and let $\Sigma_s = \{f_1, \ldots, f_k\}$ with $f_i \in \Sigma_{w_i,s}$ and $w_i = s_{i,1} \cdots s_{i,n(i)}$ for $i = 1, \ldots, k$. Now*

$$[\![\mathcal{T}_s(\Sigma)]\!] := \mathsf{Ind}(X : *^s)\{C_1 \mid \cdots \mid C_k\}$$

where $C_i \equiv [\![s_{i,1}]\!] \to \cdots \to [\![s_{i,n(i)}]\!] \to [\![s]\!]$ *with*

$$[\![s']\!] = \begin{cases} X & \text{when } s' \equiv s \\ [\![\mathcal{T}_{s'}(\Sigma)]\!] & \text{otherwise} \end{cases}$$

for all $i = 1, \ldots, k$. *Furthermore* $[\![f_i]\!] := \mathsf{Constr}(i, [\![\mathcal{T}_s(\Sigma)]\!])$ *for all* $i = 1, \ldots, k$ *and* $[\![\varepsilon]\!] \to A$ *is identified with* A.

The well-foundedness of this definition follows immediately from the well-foundedness of the SDG. By construction we realised (7) for $w \in S$ (words of length one). When we define

$$[\![\mathcal{T}_{s_1 \cdots s_n}(\Sigma)]\!] := [\![\mathcal{T}_{s_1}(\Sigma)]\!] \times \cdots \times [\![\mathcal{T}_{s_n}(\Sigma)]\!]$$

then (7) is satisfied for all $w \in S^*$ because the closed normal forms of type $[\![\mathcal{T}_{s_1}(\Sigma)]\!] \times \cdots \times [\![\mathcal{T}_{s_n}(\Sigma)]\!]$ are exactly those of the form $< [\![t_1]\!], \ldots, [\![t_n]\!] >$.

When $\mathsf{A} = \langle\, \varpi, \textsc{Act}, \Im, \longrightarrow\!\!\triangleright \,\rangle$ is an I/O-automaton over Σ_A then $[\![\mathcal{T}_\varpi(\Sigma_\mathsf{A})]\!]$ represents the set of states. In general, a set of formulas Φ that we want to check is defined over a larger signature Σ, i.e. $\Phi \subseteq \mathcal{F}(\Sigma)$ with $\Sigma \supseteq \Sigma_\mathsf{A}$. In order to interpret $\phi \in \mathcal{F}(\Sigma)$ we have to extend the mapping $[\![\ ]\!]$ to the full signature Σ. Let $f \in \Sigma \setminus \Sigma_\mathsf{A}$, say $f \in \Sigma_{w, s_1 \cdots s_n}$. The interpretation for f we choose, depends on the Σ-algebra. Let $\mathcal{A}$ be a Σ-algebra such that Σ_A is a constructor signature with respect to $\mathcal{A}$, witnessed by isomorphism $\digamma$. Now we can find terms $t_i^a \in \mathcal{T}_{s_i}(\Sigma_\mathsf{A})$, for $i = 1, \ldots, n$, such that $\digamma(t_1^a, \ldots, t_n^a) = f(a)$.

$$\begin{array}{ccc} & \Sigma_\mathsf{A} & \qquad\qquad \Sigma \\ & [\![\ ]\!] \downarrow & \qquad\quad [\![\ ]\!] \downarrow \\ (t_1^a, \ldots, t_n^a) \in \prod_{i=1}^{n} \mathcal{T}_{s_i}(\Sigma_\mathsf{A}) & \xrightarrow{\digamma_{s_1 \cdots s_n}} & A_{s_1 \cdots s_n} \ni [\![f(a)]\!] \end{array}$$

By induction we assume that we have $[\![a]\!]_\mathcal{A}$. Moreover, interpretations $[\![t_i^a]\!]$ are given by Definition 4.1 since $t_i^a \in \mathcal{T}(\Sigma_\mathsf{A})$. Now let $[\![f]\!]_\mathcal{A}$ be the λ-abstraction term such that

$$[\![f]\!]_\mathcal{A}\ [\![a]\!]_\mathcal{A} \twoheadrightarrow_\beta [\![(t_1^a, \ldots, t_n^a)]\!]_\mathcal{A} \quad \text{for all } a \in \mathcal{T}_w(\Sigma).$$

and define $[\![f\, a]\!]_\mathcal{A} = [\![f]\!]_\mathcal{A}\ [\![a]\!]_\mathcal{A}$. Furthermore put

$$[\![\,(t_1, \ldots, t_n)\,]\!]_\mathcal{A} = < [\![t_1]\!]_\mathcal{A}, \ldots, [\![t_n]\!]_\mathcal{A} >$$

then we have inductively defined $[\![t]\!]$ for all $t \in \mathcal{T}(\Sigma)$. Note that (7) is satisfied now for all $w \in S^*$ and for the extended signature Σ.

Open terms are represented by open terms. A term t containing a free variable x_w lives only in contexts extending $\langle\ [\![x_w]\!] : [\![\mathcal{T}_w(\Sigma)]\!]\ \rangle$.

We need interpretations for formulas, executions, etc. Things will be set up such that $\langle\ \rangle \vdash [\![\phi]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n} : *^P$ is derivable for every σ, n and closed ϕ. The term $[\![\sigma]\!]_{\mathcal{A},\mathsf{A}}$ and its type $[\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}}$ will be introduced at the end of this section. In the sequel we abbreviate

$[\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \to \mathbf{N} \to *^p$, which is the type of $[\![\phi]\!]_{\mathcal{A},\mathsf{A}}$, by $\mathbf{F}$. The interpretation $[\![\phi]\!]_{\mathcal{A},\mathsf{A}}$ for atomic ϕ is defined by using (4) on page 10 in case $\phi \equiv \bot\!\!\!\bot$ and (5) on page 10 in case $\phi \equiv t_1 \stackrel{s}{=} t_2$.

$$\begin{array}{lcl} [\![\bot\!\!\!\bot]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\, [\![\sigma]\!]_{\mathcal{A},\mathsf{A}} : [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \,.\, \lambda n{:}\mathbf{N} \,.\, \bot \\ [\![t_1 \stackrel{s}{=} t_2]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\, [\![\sigma]\!]_{\mathcal{A},\mathsf{A}} : [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \,.\, \lambda n{:}\mathbf{N} \,.\, [\![t_1]\!]_{\mathcal{A}} =_{\mathcal{L}} [\![t_2]\!]_{\mathcal{A}} \end{array}$$

When ϕ is not an atomic formula, say $\phi \equiv \phi_1 \star \phi_2$, for some $\star \in \{\Rightarrow, \mathcal{U}, \mathcal{S}\}$, then $[\![\phi]\!]_{\mathcal{A},\mathsf{A}} := [\![\star]\!]_{\mathcal{A},\mathsf{A}}\ [\![\phi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\phi_2]\!]_{\mathcal{A},\mathsf{A}}$ for some $[\![\star]\!]_{\mathcal{A},\mathsf{A}} : \mathbf{F} \to \mathbf{F} \to \mathbf{F}$. The interpretation of $\star$ is derived from the semantics. We illustrate this with an example.

Example 4.2. We define $[\![\Rightarrow]\!]_{\mathcal{A},\mathsf{A}} : \mathbf{F} \to \mathbf{F} \to \mathbf{F}$. From the semantics we conclude that we want

$$[\![\sigma, n \models_{\mathcal{A}} \phi \Rightarrow \psi]\!]_{\mathcal{A},\mathsf{A}} =_{\beta\iota} [\![\sigma, n \models_{\mathcal{A}} \phi]\!]_{\mathcal{A},\mathsf{A}} \to [\![\sigma, n \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}}$$

to hold for all $\sigma \in \mathcal{E}(\mathsf{A})$ and for all $n \in \mathbb{N}$. Furthermore we want $[\![\Rightarrow]\!]_{\mathcal{A},\mathsf{A}}$ to be such that $[\![\Rightarrow]\!]_{\mathcal{A},\mathsf{A}}\ [\![\phi]\!]_{\mathcal{A},\mathsf{A}}\ [\![\psi]\!]_{\mathcal{A},\mathsf{A}} =_{\beta\iota} [\![\phi \Rightarrow \psi]\!]_{\mathcal{A},\mathsf{A}}$. So we want

$$\begin{array}{lll} (\,[\![\Rightarrow]\!]_{\mathcal{A},\mathsf{A}}\ [\![\phi]\!]_{\mathcal{A},\mathsf{A}}\ [\![\psi]\!]_{\mathcal{A},\mathsf{A}}\,)\,[\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\,\underline{n} & =_{\beta\iota} & [\![\phi \Rightarrow \psi]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\,\underline{n} \\ & = & [\![\sigma, n \models_{\mathcal{A}} \phi \Rightarrow \psi]\!]_{\mathcal{A},\mathsf{A}} \\ & =_{\beta\iota} & [\![\sigma, n \models_{\mathcal{A}} \phi]\!]_{\mathcal{A},\mathsf{A}} \to [\![\sigma, n \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}} \\ & = & (\,[\![\phi]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\,\underline{n}) \to (\,[\![\psi]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\,\underline{n}). \end{array}$$

Obviously, this can be obtained by defining

$$[\![\Rightarrow]\!]_{\mathcal{A},\mathsf{A}} := \lambda\phi, \psi{:}\mathbf{F} \,.\, \lambda\sigma{:}\, [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \,.\, \lambda n{:}\mathbf{N} \,.\, (\phi\ \sigma\ n) \to (\psi\ \sigma\ n)$$

The other connectives are defined analogously. In the definitions of $[\![\mathcal{U}]\!]_{\mathcal{A},\mathsf{A}}$ and $[\![\mathcal{S}]\!]_{\mathcal{A},\mathsf{A}}$ we make use of auxiliary λ-terms ϱ of type $\mathbf{N} \to [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \to [\![\mathcal{T}_\varpi(\Sigma)]\!]_{\mathcal{A}}$ satisfying

$$\varrho\,\underline{n}\,[\![\sigma]\!]_{\mathcal{A},\mathsf{A}} \twoheadrightarrow_{\beta\iota} [\![\pi_n(\sigma)]\!]_{\mathcal{A},\mathsf{A}}$$

and tr of type $\mathbf{F} \to [\![\mathcal{E}(\mathsf{A})]\!] \to \mathbf{N} \to \mathbf{N} \to *^p$. The intuitive meaning of $(tr\ \ [\![\phi]\!]\,\sigma\ n\ m)$ is that ϕ holds on the trace $\pi_{n+1}(\sigma) \cdots \pi_{m-1}(\sigma)$ of intermediate states. In formula: $\sigma, k \models \phi$ for all $n < k < m$. When the number $m - n - 1$ of intermediate states is negative (i.e. when $n \geq m$) then $(tr\ \ [\![\phi]\!]\,\sigma\ n\ m)$ evaluates to $\bot$. The definitions for ϱ and tr are given in the next section.

$$\begin{array}{lcl} [\![\downarrow]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda P{:}\, [\![\mathcal{T}_\varpi(\Sigma)]\!]_{\mathcal{A},\mathsf{A}} \to \mathbf{F} \,.\, \lambda\sigma{:}\, [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \,.\, \lambda n{:}\mathbf{N} \,.\, P\,(\varrho\ n\ \sigma)\ \sigma\ n \\ [\![\mathcal{U}]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\phi, \psi{:}\mathbf{F} \,.\, \lambda\sigma{:}\, [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \,.\, \lambda n{:}\mathbf{N} \,. \\ & & \quad \mathbf{E}\ \mathbf{N}\ \lambda m{:}\mathbf{N} \,.\, ((\psi\ \sigma\ m) \curlywedge (tr\ \phi\ \sigma\ n\ m) \\ [\![\mathcal{S}]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\phi, \psi{:}\mathbf{F} \,.\, \lambda\sigma{:}\, [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \,.\, \lambda n{:}\mathbf{N} \,. \\ & & \quad \mathbf{E}\ \mathbf{N}\ \lambda m{:}\mathbf{N} \,.\, ((\psi\ \sigma\ m) \curlywedge (tr\ \phi\ \sigma\ m\ n)) \end{array}$$

Now $[\![\downarrow x_\varpi \,.\, \phi]\!]_{\mathcal{A},\mathsf{A}} := [\![\downarrow]\!]_{\mathcal{A},\mathsf{A}}\,(\lambda\, [\![x_\varpi]\!] : [\![\mathcal{T}_\varpi(\Sigma)]\!] \,.\, [\![\phi]\!]_{\mathcal{A},\mathsf{A}})$.

The abbreviations, introduced in Section 2, can simply be 'copied':

$$
\begin{array}{lcl}
[\![\neg]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\phi{:}\mathbf{F}\,.\,[\![\to]\!]_{\mathcal{A},\mathsf{A}}\,\phi\,[\![\bot]\!]_{\mathcal{A},\mathsf{A}} \\
[\![\top]\!]_{\mathcal{A},\mathsf{A}} & := & [\![\neg]\!]_{\mathcal{A},\mathsf{A}}\,[\![\bot]\!]_{\mathcal{A},\mathsf{A}} \\
[\![\vee]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\phi,\psi{:}\mathbf{F}\,.\,[\![\to]\!]_{\mathcal{A},\mathsf{A}}\,([\![\neg]\!]_{\mathcal{A},\mathsf{A}}\,\phi)\,\psi \\
[\![\wedge]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\phi,\psi{:}\mathbf{F}\,.\,[\![\neg]\!]_{\mathcal{A},\mathsf{A}}\,([\![\to]\!]_{\mathcal{A},\mathsf{A}}\,\phi\,([\![\neg]\!]_{\mathcal{A},\mathsf{A}}\,\psi)) \\
[\![\Diamond]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\phi{:}\mathbf{F}\,.\,[\![\mathcal{U}]\!]_{\mathcal{A},\mathsf{A}}\,[\![\top]\!]_{\mathcal{A},\mathsf{A}}\,\phi \\
[\![\Box]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\phi{:}\mathbf{F}\,.\,[\![\neg]\!]_{\mathcal{A},\mathsf{A}}\,([\![\Diamond]\!]_{\mathcal{A},\mathsf{A}}\,([\![\neg]\!]_{\mathcal{A},\mathsf{A}}\,\phi)) \\
[\![\bigcirc]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\phi{:}\mathbf{F}\,.\,[\![\mathcal{U}]\!]_{\mathcal{A},\mathsf{A}}\,[\![\bot]\!]_{\mathcal{A},\mathsf{A}}\,\phi \\
[\![\blacklozenge]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\phi{:}\mathbf{F}\,.\,[\![\mathcal{S}]\!]_{\mathcal{A},\mathsf{A}}\,[\![\top]\!]_{\mathcal{A},\mathsf{A}}\,\phi \\
[\![\blacksquare]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\phi{:}\mathbf{F}\,.\,[\![\neg]\!]_{\mathcal{A},\mathsf{A}}\,([\![\blacklozenge]\!]_{\mathcal{A},\mathsf{A}}\,([\![\neg]\!]_{\mathcal{A},\mathsf{A}}\,\phi)) \\
[\![\bullet]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\phi{:}\mathbf{F}\,.\,[\![\mathcal{S}]\!]_{\mathcal{A},\mathsf{A}}\,[\![\bot]\!]_{\mathcal{A},\mathsf{A}}\,\phi \\
[\![\nabla]\!]_{\mathcal{A},\mathsf{A}} & := & \lambda\phi{:}\mathbf{F}\,.\,[\![\wedge]\!]_{\mathcal{A},\mathsf{A}}\,\phi\,([\![\mathcal{S}]\!]_{\mathcal{A},\mathsf{A}}\,([\![\neg]\!]_{\mathcal{A},\mathsf{A}}\,\phi)([\![\neg]\!]_{\mathcal{A},\mathsf{A}}\,\phi))
\end{array}
$$

In order to define $[\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}}$ we introduce an inductive type $[\![\Re]\!]_{\mathcal{A},\mathsf{A}}$ representing the reachability predicate $\Re$. The set $\mathcal{E}(\mathsf{A})$ can then be represented by

$$[\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} := \mathsf{Ind}(X : *^s)\{\Pi x{:}\,[\![\mathcal{T}_\varpi(\Sigma)]\!]_{\mathcal{A}}\,.\,([\![\Re]\!]_{\mathcal{A},\mathsf{A}}\,x) \to X\}.$$

Inhabitants of $[\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}}$ are pairs (x, y) with x representing a state in the state space and y representing a proof of the reachability of x. Such a proof is just a construction

$$
\begin{array}{l}
\mathsf{Constr}(2,\,[\![\Re]\!]_{\mathcal{A},\mathsf{A}})\,[\![s_{n-1}]\!]_{\mathcal{A}}\,[\![x]\!]_{\mathcal{A}}\,[\![a_{n-1}]\!]_{\mathcal{A},\mathsf{A}}\;\;p_{n-1} \\
\quad(\mathsf{Constr}(2,\,[\![\Re]\!]_{\mathcal{A},\mathsf{A}})\,[\![s_{n-2}]\!]_{\mathcal{A}}\,[\![s_{n-1}]\!]_{\mathcal{A}}\,[\![a_{n-2}]\!]_{\mathcal{A},\mathsf{A}}\;\;p_{n-2} \\
\qquad\qquad\vdots \\
\quad(\mathsf{Constr}(2,\,[\![\Re]\!]_{\mathcal{A},\mathsf{A}})\,[\![s_1]\!]_{\mathcal{A}}\,[\![s_2]\!]_{\mathcal{A}}\,[\![a_2]\!]_{\mathcal{A},\mathsf{A}}\;\;p_2 \\
\quad\;(\mathsf{Constr}(2,\,[\![\Re]\!]_{\mathcal{A},\mathsf{A}})\,[\![s_0]\!]_{\mathcal{A}}\,[\![s_1]\!]_{\mathcal{A}}\,[\![a_1]\!]_{\mathcal{A},\mathsf{A}}\;\;p_1 \\
\quad\;(\mathsf{Constr}(1,\,[\![\Re]\!]_{\mathcal{A},\mathsf{A}})\,[\![s_0]\!]_{\mathcal{A}}\;\;p_0)))\cdots)
\end{array}
$$

of an execution $s_0 \xrightarrow{a_1}\!\triangleright\; s_1 \xrightarrow{a_2}\!\triangleright\; \cdots \xrightarrow{a_{n-1}}\!\triangleright\; s_{n-1} \xrightarrow{a_n}\!\triangleright\; x$ from an initial state to x. The λ-term p_0 represents a proof for $s_0 \in \Im$ and p_{i+1} represents a proof for $s_i \xrightarrow{a_{i+1}}\!\triangleright\; s_{i+1}$. Thus, there is a bijection between $\mathcal{E}(\mathsf{A})$ and $[\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}}$ mapping the dependent pair mentioned above to $[s_0\ s_1 \cdots s_{n-1}\ x] \in \mathcal{E}(\mathsf{A})$.

The predicate $\Re$ is represented by the inductive type

$$
\begin{array}{l}
[\![\Re]\!]_{\mathcal{A},\mathsf{A}} := \mathsf{Ind}(X : [\![\mathcal{T}_\varpi(\Sigma)]\!]_{\mathcal{A}} \to *^s) \\
\qquad\qquad\{\Pi x{:}\,[\![\mathcal{T}_\varpi(\Sigma)]\!]_{\mathcal{A}}\,.\,([\![\Im]\!]_{\mathcal{A},\mathsf{A}}\,x) \to (X\ x) \\
\qquad\qquad\mid \Pi x_1, x_2{:}\,[\![\mathcal{T}_\varpi(\Sigma)]\!]_{\mathcal{A}}\,.\,\Pi a{:}\,[\![\textsc{Act}]\!]_{\mathcal{A},\mathsf{A}}\,. \\
\qquad\qquad\quad([\![\longrightarrow\!\triangleright]\!]_{\mathcal{A},\mathsf{A}}\;\;a\;x_1\;x_2) \to (X\;x_1) \to (X\;x_2)\}
\end{array}
$$

Here $[\![\Im]\!]_{\mathcal{A},\mathsf{A}}$ represents $\Im$. The proofterm p_0 is provided by this term $[\![\Im]\!]_{\mathcal{A},\mathsf{A}}$. Furthermore $[\![\longrightarrow\!\triangleright]\!]_{\mathcal{A},\mathsf{A}}$ represents $\longrightarrow\!\triangleright$ and $[\![\textsc{Act}]\!]_{\mathcal{A},\mathsf{A}}$ represents Act. One might expect $*^p$ instead of $*^s$ in the type of $[\![\Re]\!]_{\mathcal{A},\mathsf{A}}$ since a term of type $[\![\Re]\!]_{\mathcal{A},\mathsf{A}}\,[\![x]\!]_{\mathcal{A}}$ intuitively represents a proof

of the reachability of x. However, the induction principle that we require for our purposes is not allowed in Coq when we replace $*^s$ by $*^p$.

The representation $[\![\longrightarrow\!\triangleright]\!]_{\mathcal{A},\mathsf{A}}$ of the transition relation is an inductive type with a constructor for every action of Act. Its constructors provide the proofterms $p_1, \ldots, p_{n-1}$ mentioned in the reachability proof above. The representation $[\![\textsc{Act}]\!]_{\mathcal{A},\mathsf{A}}$ is straightforward.

5 The Construction of ϱ and tr

In this section we introduce the λ-terms, needed to build the λ-terms ϱ and tr used on page 13. To begin, we need λ-terms for standard notions like length (ℓ), head and tail. Here these notions are not defined on lists but on executions, which slightly complicates their construction. Therefore we worked out, as an example, the details in the construction of a representation for ℓ.

Example 5.1. We build a λ-term $\boldsymbol{\ell}$ of type $[\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \to \mathbf{N}$ that calculates the length (number of steps) of an execution. Let $P \equiv \lambda z\colon [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} . \mathbf{N}$ of type $\mathbf{N} \to *^s$ be the (constant) function, mapping each term to $\mathbf{N}$. The expression

$$\boldsymbol{\ell} := \lambda\sigma\colon [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} . \mathsf{Elim}(\sigma, P)\{g\}$$

is well-typed iff g is of type

$$\Pi x\colon [\![\mathcal{T}_{\varpi}(\Sigma)]\!]_{\mathcal{A}} . ([\![\Re]\!]_{\mathcal{A},\mathsf{A}}\, x) \to \underbrace{(P\ \sigma)}_{=_\beta \mathbf{N}}.$$

Furthermore $\boldsymbol{\ell}\,(\mathsf{Constr}(1, [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}})\ x\ y)$ ι-reduces to $g\ x\ y$. The definition of 'length' depends only on y. (x does not contain any information about the length of the execution.) Hence, we take $g \equiv \lambda x\colon [\![\mathcal{T}_{\varpi}(\Sigma)]\!]_{\mathcal{A}} . \lambda y\colon [\![\Re]\!]_{\mathcal{A},\mathsf{A}}\, x . g'(y)$ where $g'(y)$ of type $\mathbf{N}$ is given by

$$g'(y) := \mathsf{Elim}(y, \lambda z\colon [\![\mathcal{T}_{\varpi}(\Sigma)]\!]_{\mathcal{A}} . \mathbf{N})\{g_1 \mid g_2\}$$

with

$$g_1 \ : \ \Pi x_0\colon [\![\mathcal{T}_{\varpi}(\Sigma)]\!]_{\mathcal{A}} . ([\![\Im]\!]_{\mathcal{A},\mathsf{A}}\, x_0) \to \underbrace{(P\ x_0)}_{=_\beta \mathbf{N}}$$

$$g_2 \ : \ \Pi x_1, x_2\colon [\![\mathcal{T}_{\varpi}(\Sigma)]\!]_{\mathcal{A}} . \Pi a\colon [\![\textsc{Act}]\!]_{\mathcal{A},\mathsf{A}} . \\ ([\![\longrightarrow\!\triangleright]\!]_{\mathcal{A},\mathsf{A}}\ a\ x_1\ x_2) \to ([\![\Re]\!]_{\mathcal{A},\mathsf{A}}\, x_1) \to \underbrace{(P\ x_1)}_{=_\beta \mathbf{N}} \to \underbrace{(P\ x_2)}_{=_\beta \mathbf{N}}.$$

Now

$$\begin{array}{lll} g'(\mathsf{Constr}(1, [\![\Re]\!]_{\mathcal{A},\mathsf{A}})\ x\ y) & \twoheadrightarrow_\iota & g_1\ x\ y \\ g'(\mathsf{Constr}(2, [\![\Re]\!]_{\mathcal{A},\mathsf{A}})\ x_1\ x_2\ a\ h\ p) & \twoheadrightarrow_\iota & g_2\ x_1\ x_2\ a\ h\ p\ g'(p). \end{array}$$

The length of $[x]$ should be 0, so

$$g_1 := \lambda x_0\colon [\![\mathcal{T}_{\varpi}(\Sigma)]\!]_{\mathcal{A}} . \lambda y_0\colon [\![\Im]\!]_{\mathcal{A},\mathsf{A}}\, x_0 . \underline{0}$$

and the length of $[\cdot \cdots\cdots x_1\ a\ x_2]$ should be $1 + \ell([\cdot \cdots\cdots x_1])$ so

$$\begin{array}{lcl} g_2 & := & \lambda x_1, x_2 \colon [\![\mathcal{T}_\varpi(\Sigma)]\!]_{\mathcal{A}}\ . \\ & & \quad \lambda a \colon [\![\textsc{Act}]\!]_{\mathcal{A},\mathsf{A}}\ . \lambda h \colon [\![\longrightarrow\!\!\triangleright]\!]_{\mathcal{A},\mathsf{A}}\ \ a\ x_1\ x_2\ .\ \lambda p \colon [\![\Re]\!]_{\mathcal{A},\mathsf{A}}\ x_1\ .\ \lambda n \colon \mathbf{N}\ .\ n \oplus \underline{1}. \end{array}$$

The next step is to construct λ-terms *head* of type $[\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \to [\![\mathcal{T}_\varpi(\Sigma)]\!]_{\mathcal{A}}$ and *tail* of type $[\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \to [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}}$ with the usual meaning. The construction of these terms is analogously to the construction of ℓ. The details are left to the reader.

Now we can define a function

$$\begin{array}{l} \mathit{aux} := \lambda n \colon \mathbf{N}\ .\ \mathsf{Elim}(n, \lambda z \colon \mathbf{N}\ .\ [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \to [\![\mathcal{T}_\varpi(\Sigma)]\!]_{\mathcal{A},\mathsf{A}}) \\ \qquad \{\mathit{head} \\ \qquad \mid \lambda m \colon \mathbf{N}\ .\ \lambda f \colon [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}} \to [\![\mathcal{T}_\varpi(\Sigma)]\!]_{\mathcal{A}}\ .\ \lambda \sigma \colon [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}}\ .\ f\ (\mathit{tail}\ \sigma)\} \end{array}$$

which projects the states in reversed order, i.e.

$$\begin{array}{lcl} \mathit{aux}\ 0\ [s_0 \cdots s_n] & = & s_n \\ \qquad\qquad \vdots & & \\ \mathit{aux}\ n\ [s_0 \cdots s_n] & = & s_0 \end{array}$$

and define

$$\varrho := \lambda n \colon \mathbf{N}\ .\ \lambda \sigma \colon [\![\mathcal{E}(\mathsf{A})]\!]_{\mathcal{A},\mathsf{A}}\ .\ \mathit{aux}((\ell\ \sigma) \ominus n)\ \sigma.$$

Note that this λ-term ϱ actually represents the extension of $\lambda\!\!\lambda n.\pi_n$ to the total function on $\mathbb{N} \times \mathcal{E}(\mathsf{A})$ satisfying $\pi_n(\sigma) = \pi_m(\sigma)$ for all $n, m \geq \ell(\sigma)$.

We conclude this section with the definition of the λ-term tr.

$$\begin{array}{lcl} tr & := & \lambda \phi \colon \mathbf{F}\ .\ \lambda \sigma \colon [\![\mathcal{E}(\mathsf{A})]\!]\ .\ \lambda n, m \colon \mathbf{N}\ . \\ & & \quad \mathsf{Elim}(m \ominus n, \lambda z \colon \mathbf{N}\ .\ *^p) \\ & & \qquad \{\bot \\ & & \qquad \mid \lambda x \colon \mathbf{N}\ .\ \lambda U \colon *^p\ . \\ & & \qquad\quad \mathsf{Elim}(x, \lambda z \colon \mathbf{N}\ .\ *^p) \\ & & \qquad\qquad \{\top \\ & & \qquad\qquad \mid \lambda y \colon \mathbf{N}\ .\ \lambda U' \colon *^p\ .\ U' \wedge (\phi\ \sigma\ (m \oplus y \oplus \underline{1}))\}\} \end{array}$$

Note that there exist terms $H_{n+1}, \ldots, H_{m-1}$ such that $\langle\ \rangle \vdash H_i : [\![\sigma, i \models_{\mathcal{A}} \phi]\!]_{\mathcal{A},\mathsf{A}}$ whenever there exists a term H such that $\langle\ \rangle \vdash H : (tr\ [\![\phi]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ \underline{m})$.

6 Justification of the Approach

In this section we prove soundness and completeness of the implementation. That is

$$\sigma, n \models_{\mathcal{A}} \phi \iff \exists H.\ \langle\ \rangle \vdash H : [\![\sigma, n \models_{\mathcal{A}} \phi]\!]_{\mathcal{A},\mathsf{A}}\ .$$

The proof uses induction to the structure of ϕ. The $\Rightarrow$-case of this induction proof (i.e. the case $\phi \equiv \phi_1 \Rightarrow \phi_2$) uses a lemma (6.2) which establishes that we can 'push' the meta-implication into the type system. We start with a substitution lemma.

Lemma 6.1. *For all $t, t' \in \mathcal{T}(\Sigma)$ and for all $\phi \in \mathcal{F}(\Sigma)$ we have:*

- $[\![t'[x := t]]\!] = [\![t']\!][x := [\![t]\!]]$
- $[\![\phi[x := t]]\!] = [\![\phi]\!][x := [\![t]\!]]$

Proof. A straightforward induction to the structure of t' and ϕ. ❑

The following lemma states the equivalence of two expressions (i) and (ii) of the form

(i) $\exists H . \langle\ \rangle \vdash H : P \implies \exists H' . \langle\ \rangle \vdash H' : Q$
(ii) $\exists H'' . \langle\ \rangle \vdash H'' : P \to Q$

for certain types P and Q. Assuming that it is consistent to add an axiom for classical reasoning to the calculus of *inductive* constructions[1], we can easily verify that (i) $\Longrightarrow$ (ii) does not hold for general P and Q. For instance take $P \equiv \Pi\alpha{:}*^p . \alpha \curlyvee (\alpha \to \bot)$ and $Q \equiv \bot$. However, it holds for interpretations of judgements.

Lemma 6.2. *Let $\phi, \psi \in \mathcal{F}(\Sigma)$, $\sigma \in \mathcal{E}(\mathsf{A})$ and $m, n \in \mathbb{N}$. Then*

$$\exists H . \langle\ \rangle \vdash H : [\![\sigma, n \models_{\mathcal{A}} \phi]\!]_{\mathcal{A},\mathsf{A}} \implies \exists H' . \langle\ \rangle \vdash H' : [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}}$$

iff

$$\exists H'' . \langle\ \rangle \vdash H'' : [\![\sigma, n \models_{\mathcal{A}} \phi]\!]_{\mathcal{A},\mathsf{A}} \to [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}} .$$

Proof. The proof from right to left is trivial. The proof from left to right uses induction to the structure of ϕ. Note that the lemma (from left to right) holds trivially when $\langle\ \rangle \vdash H' : [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}}$ for some H'. Choose $H'' \equiv \lambda x : [\![\sigma, n \models_{\mathcal{A}} \phi]\!]_{\mathcal{A},\mathsf{A}} . H'$. Assume that H' does not exist. When there is a λ-term H such that $\langle\ \rangle \vdash [\![\sigma, n \models_{\mathcal{A}} \phi]\!]_{\mathcal{A},\mathsf{A}}$ the lemma holds vacuously since we assumed the non-existence of H'. Hence we assume that H does also not exist.

- $\phi \equiv \text{\Lightning} . \ [\![\sigma, n \models_{\mathcal{A}} \text{\Lightning}]\!]_{\mathcal{A},\mathsf{A}} =_{\beta\iota} \bot$. Choose $H'' \equiv \lambda x{:}\bot . \mathsf{Elim}(x, [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}})\{\ \}$.
- $\phi \equiv t_1 \stackrel{w}{=} t_2$. We assumed that H does not exist, i.e. $[\![t_1]\!]_{\mathcal{A}} \neq_{\beta\iota} [\![t_2]\!]_{\mathcal{A}}$. Since Leibniz equality defines free structures, this means that there exists a λ-term h such that

  $$\langle\ \rangle \vdash h : ([\![t_1]\!]_{\mathcal{A}} =_{\mathcal{L}} [\![t_2]\!]_{\mathcal{A}}) \to \bot .$$

  Now take $H'' \equiv \lambda x : [\![\sigma, n \models_{\mathcal{A}} \phi]\!]_{\mathcal{A},\mathsf{A}} . \mathsf{Elim}(h\ x, [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}})\{\ \}$.

[1] Geuvers showed [7] that such an axiom can be added to the calculus of constructions without losing the consistency of that system

- $\phi \equiv \phi_1 \Rightarrow \phi_2$. We assumed that H and H' do not exist. This means that there exists a λ-term h such that $\langle\ \rangle \vdash h : [\![\sigma, n \models_{\mathcal{A}} \phi_1]\!]_{\mathcal{A},\mathsf{A}}$ and that there does not exist a λ-term h' such that $\langle\ \rangle \vdash h' : [\![\sigma, n \models_{\mathcal{A}} \phi_2]\!]_{\mathcal{A},\mathsf{A}}$. For all other cases would validate

$$\exists h\,.\,\langle\ \rangle \vdash h : [\![\sigma, n \models_{\mathcal{A}} \phi_1]\!]_{\mathcal{A},\mathsf{A}} \implies \exists h'\,.\,\langle\ \rangle \vdash h' : [\![\sigma, n \models_{\mathcal{A}} \phi_2]\!]_{\mathcal{A},\mathsf{A}}$$

which implies, by IH, that

$$\langle\ \rangle \vdash H : [\![\sigma, n \models_{\mathcal{A}} \phi_1]\!]_{\mathcal{A},\mathsf{A}} \to [\![\sigma, n \models_{\mathcal{A}} \phi_2]\!]_{\mathcal{A},\mathsf{A}} \equiv [\![\sigma, n \models_{\mathcal{A}} \phi_1 \Rightarrow \phi_2]\!]_{\mathcal{A},\mathsf{A}}$$

for some H, contradicting the assumption that H does not exist. Again by IH we conclude that there exists a λ-term h'' such that

$$\langle\ \rangle \vdash h'' : [\![\sigma, n \models_{\mathcal{A}} \phi_2]\!]_{\mathcal{A},\mathsf{A}} \to [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}}\,.$$

Now take $H'' \equiv \lambda x\!:\![\![\sigma, n \models_{\mathcal{A}} \phi]\!]_{\mathcal{A},\mathsf{A}}\ .\, h''(x\ h)$.

- $\phi \equiv \downarrow x_\varpi\,.\,\phi'$. We assumed that there is no λ-term H of type

$$[\![\sigma, n \models_{\mathcal{A}} \downarrow x_\varpi\,.\,\phi']\!]_{\mathcal{A},\mathsf{A}} =_{\beta\iota} [\![\sigma, n \models_{\mathcal{A}} \phi'[x_\varpi := \pi_n(\sigma)]]\!]_{\mathcal{A},\mathsf{A}}\,.$$

By IH we have a λ-term H'' of type

$$[\![\sigma, n \models_{\mathcal{A}} \phi'[x_\varpi := \pi_n(\sigma)]]\!]_{\mathcal{A},\mathsf{A}} \to [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}}$$

so we are done.

- $\phi \equiv \phi_1\,\mathcal{U}\,\phi_2$. We assumed that there is no H of type $[\![\sigma, n \models_{\mathcal{A}} \phi_1\,\mathcal{U}\,\phi_2]\!]_{\mathcal{A},\mathsf{A}}$, i.e. no H of type

$$\mathbf{E}\ \mathbf{N}\ \lambda y\!:\!\mathbf{N}\,.\,([\![\phi_2]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ y) \curlywedge (tr\ [\![\phi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ y).$$

This means that there does not exist a λ-term $\underline{k}$, with $k \in \mathbb{N}$, such that $[\![\sigma, k \models_{\mathcal{A}} \phi_2]\!]_{\mathcal{A},\mathsf{A}}$ and $(tr\ [\![\phi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ \underline{k})$ are both inhabited in the empty context. We have to show that we can find a λ-term h of type

$$\Pi y\!:\!\mathbf{N}\,.\,((\,[\![\phi_2]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ y) \curlywedge (tr\ [\![\phi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ y)) \to [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}} \qquad (8)$$

in the empty context, since

$$\begin{array}{rcl} H'' & \equiv & \lambda x\!:\![\![\sigma, n \models_{\mathcal{A}} \phi_1\,\mathcal{U}\,\phi_2]\!]_{\mathcal{A},\mathsf{A}}\ . \\ & & \quad \mathsf{Elim}(x, \lambda z\!:\![\![\sigma, n \models_{\mathcal{A}} \phi_1\,\mathcal{U}\,\phi_2]\!]_{\mathcal{A},\mathsf{A}}\ .\ [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}}\,)\{h\} \end{array}$$

is of the right type in that case. We show that we can find λ-terms h_κ of type

$$((\,[\![\phi_2]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \kappa) \curlywedge (tr\ [\![\phi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ \kappa)) \to [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}}$$

for each closed term κ of type $\mathbf{N}$. Let κ be an arbitrary closed term of type $\mathbf{N}$, say $\kappa =_{\beta\iota} \underline{k}$ ($k \in \mathbb{N}$). We prove the existence of λ-term h_κ by a case distinction.

- $n \leq k$. Take $h_\kappa \equiv \lambda p\,.\,\mathsf{Elim}(\mathsf{Elim}(p, \bot)\{\lambda x\, y.y\},\, [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}}\,)\{\ \}$.
- $n > k$. Distinguish between two cases:
  * If $[\![\sigma, k \models_{\mathcal{A}} \phi_2]\!]_{\mathcal{A},\mathsf{A}}$ is not inhabited in the empty context then, by IH, we have a λ-term h'_κ such that
    $$\langle\ \rangle \vdash h'_\kappa : [\![\sigma, k \models_{\mathcal{A}} \phi_2]\!]_{\mathcal{A},\mathsf{A}} \to [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}}$$
    and hence $\lambda p\,.\,h'_\kappa\ \mathsf{Elim}(p,\, [\![\sigma, k \models_{\mathcal{A}} \phi_2]\!]_{\mathcal{A},\mathsf{A}}\,)\{\lambda xy\,.\,x\}$ is of the right type.
  * If $(tr\ [\![\phi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ \underline{k})$ is not inhabited in the empty context then there is an $i \in \mathbb{N}$ such that $n < i < k$ and $[\![\sigma, i \models_{\mathcal{A}} \phi_1]\!]_{\mathcal{A},\mathsf{A}}$ is not inhabited in the empty context. Again by IH we have a λ-term h''_κ such that
    $$\langle\ \rangle \vdash h_\kappa : [\![\sigma, i \models_{\mathcal{A}} \phi_1]\!]_{\mathcal{A},\mathsf{A}} \to [\![\sigma, m \models_{\mathcal{A}} \psi]\!]_{\mathcal{A},\mathsf{A}}$$
    Now take $\lambda p\,.\,h''_\kappa\ (q_i\ \mathsf{Elim}(p, (tr\ [\![\phi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ \underline{k}))\{\lambda xy\,.\,y\})$ where q_i of type
    $$(tr\ [\![\phi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ \underline{k}) \to [\![\sigma, i \models_{\mathcal{A}} \phi_1]\!]_{\mathcal{A},\mathsf{A}}\,.$$
    It is an easy exercise that such a λ-term q_i exists.

In order to inhabit (8) it is not sufficient that we have the collection

$$\{h_\kappa \mid \kappa \text{ a closed term of type } \mathbf{N}\}.$$

There must be a uniform construction of $h_{\kappa \oplus \underline{1}}$ from h_κ. Obviously, the two constructions above (labelled with *) are uniform. However, the decision in which case we are must also be decidable. It is easy to see that this is the case: if there exist λ-terms h'''_κ of type

$$[\![\sigma, k \models_{\mathcal{A}} \phi_1]\!]_{\mathcal{A},\mathsf{A}} \tag{9}$$

for all closed terms κ of type $\mathbf{N}$, then we are always in the first case and otherwise we are always in the second case when are we are beyond the smallest counterexample for (9).

- $\phi \equiv \phi_1\, \mathcal{S}\, \phi_2$. Same kind of arguments. Actually, this case is less complicated then the previous one since looking to the past is bounded. (0 is the earliest point in time).

❑

Theorem 6.3. *(Soundness and Completeness)*

$$\sigma, n \models \phi \iff \exists H\,.\,\langle\ \rangle \vdash H : [\![\sigma, n \models \phi]\!]_{\mathcal{A},\mathsf{A}}\,.$$

Proof. By induction over the structure of ϕ

- $\phi \equiv \bot$. Both the left-hand-side and the right-hand-side of the theorem never hold since there is no term H such that $\langle\ \rangle \vdash H : \bot$.

- $\phi \equiv t_1 \stackrel{s}{=} t_2$. Let Σ' be the constructor signature of Σ with respect to $\mathcal{A}$ and let $F : \mathtt{TERM}(\Sigma') \longrightarrow \mathcal{A}$ be an isomorphism. Furthermore let $t'_i \in \mathcal{T}_w(\Sigma')$ satisfy $F_w(t'_i) = [\![t_i]\!]$ for $i = 1, 2$. Assume that $\sigma, n \models t_1 \stackrel{s}{=} t_2$ then $[\![t_1]\!] = [\![t_2]\!]$ by definition. Now $F_w(t'_1) = [\![t_1]\!] = [\![t_2]\!] = F_w(t_2)$ so $t'_1 = t'_2$ because F_w is bijective. $[\![\ ]\!]_{\mathcal{A}}$ is a bijection between $\mathcal{T}(\Sigma')$ and $[\![\mathcal{T}(\Sigma')]\!]_{\mathcal{A}}$ so $[\![t'_1]\!]_{\mathcal{A}} = [\![t'_2]\!]_{\mathcal{A}}$. By construction we have that $[\![t_i]\!]_{\mathcal{A}} \twoheadrightarrow_{\beta\iota} [\![t'_i]\!]_{\mathcal{A}}$ so $[\![t'_1]\!]_{\mathcal{A}} = [\![t'_2]\!]_{\mathcal{A}}$ implies $[\![t_1]\!]_{\mathcal{A}} =_{\beta\iota} [\![t_2]\!]_{\mathcal{A}}$. This proves the lemma from left to right because, by *$\beta\iota$-Conversion* we can derive

$$\langle\ \rangle \vdash \mathsf{Constr}(1, =_{\mathcal{L}}\ [\![\mathcal{T}_w(\Sigma)]\!]_{\mathcal{A}}\ [\![t_1]\!]_{\mathcal{A}}) : [\![t_1]\!]_{\mathcal{A}} =_{\mathcal{L}} [\![t_2]\!]_{\mathcal{A}}$$

For the proof from right to left we use that inductive types define free structures with respect to Leibniz equality, i.e.

$$[\![t'_1]\!]_{\mathcal{A}} \neq_{\beta\iota} [\![t'_2]\!]_{\mathcal{A}} \implies \langle\ \rangle \vdash H : ([\![t'_1]\!]_{\mathcal{A}} =_{\mathcal{L}} [\![t'_2]\!]_{\mathcal{A}}) \to \bot$$

for some proofterm H. From the assumption that the type system is consistent we derive that we can not inhabit $[\![t'_1]\!]_{\mathcal{A}} =_{\mathcal{L}} [\![t'_2]\!]_{\mathcal{A}}$ in the empty context whenever $[\![t'_1]\!]_{\mathcal{A}} \neq_{\beta\iota} [\![t'_2]\!]_{\mathcal{A}}$.

- $\phi \equiv \downarrow x\,.\,\psi$.

$$\begin{array}{ll}
[\![\sigma, n \models \downarrow x\,.\,\psi]\!]_{\mathcal{A},\mathsf{A}} & = \\
[\![\downarrow x\,.\,\psi]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n} & =_{\beta\iota} \\
[\![\downarrow]\!]_{\mathcal{A},\mathsf{A}}\ (\lambda\, [\![x]\!]_{\mathcal{A}} : [\![\mathcal{T}_{\varpi}(\Sigma)]\!]_{\mathcal{A}}\ .\ [\![\psi]\!]_{\mathcal{A},\mathsf{A}})\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n} & =_{\beta\iota} \\
(\lambda\, [\![x]\!]_{\mathcal{A}} : [\![\mathcal{T}_{\varpi}(\Sigma)]\!]_{\mathcal{A}}\ .\ [\![\psi]\!]_{\mathcal{A},\mathsf{A}})\ ([\![\pi]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}})\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n} & =_{\beta\iota} \\
[\![\psi]\!]_{\mathcal{A},\mathsf{A}}\ [\ [\![x]\!]_{\mathcal{A}} := [\![\pi]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ ]\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n} & =_{\beta\iota} \\
[\![\psi]\!]_{\mathcal{A},\mathsf{A}}\ [\ [\![x]\!]_{\mathcal{A}} := [\![\pi_n(\sigma)]\!]_{\mathcal{A},\mathsf{A}}\ ]\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n} & =_{\beta\iota} \\
[\![\psi[x := \pi_n(\sigma)]]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n} & = \\
[\![\sigma, n \models \psi[x := \pi_n(\sigma)]]\!]_{\mathcal{A},\mathsf{A}} &
\end{array}$$

so $\sigma, n \models \downarrow x\,.\,\psi$ iff $\sigma, n \models \psi[x := \pi_n(\sigma)]$ iff (induction hypothesis)

$$\langle\ \rangle \vdash H : [\![\sigma, n \models \psi[x := \pi_n(\sigma)]]\!]_{\mathcal{A},\mathsf{A}}$$

for some H iff $\langle\ \rangle \vdash H : [\![\sigma, n \models \downarrow x\,.\,\psi]\!]_{\mathcal{A},\mathsf{A}}$ by *$\beta\iota$-Conversion*.

- $\phi \equiv \psi_1 \Rightarrow \psi_2$. We recall that

$$[\![\sigma, n \models \psi_1 \Rightarrow \psi_2]\!]_{\mathcal{A},\mathsf{A}} =_{\beta\iota} [\![\sigma, n \models \psi_1]\!]_{\mathcal{A},\mathsf{A}} \to [\![\sigma, n \models \psi_2]\!]_{\mathcal{A},\mathsf{A}}$$

(Example 4.2). Let $\langle\ \rangle \vdash H : [\![\sigma, n \models \psi_1 \Rightarrow \psi_2]\!]_{\mathcal{A},\mathsf{A}}$. Furthermore assume that $\sigma, n \models \psi_1$ then by the induction hypothesis $\langle\ \rangle \vdash H' : [\![\sigma, n \models \psi_1]\!]_{\mathcal{A},\mathsf{A}}$. Now

$$\langle\ \rangle \vdash H\ H' : [\![\sigma, n \models \psi_2]\!]_{\mathcal{A},\mathsf{A}}$$

so (by using the induction hypothesis again) it follows that $\sigma, n \models \psi_2$. This proves the theorem from right to left.

Now assume that $\sigma, n \models \psi_1 \Rightarrow \psi_2$. When we combine this with the induction hypothesis for ψ_1 and ψ_2 we find that

$$\exists H \,.\, \langle\ \rangle \vdash H : [\![\sigma, n \models \psi_1]\!]_{\mathcal{A},\mathsf{A}} \quad \Longrightarrow \quad \exists H' \,.\, \langle\ \rangle \vdash H' : [\![\sigma, n \models \psi_2]\!]_{\mathcal{A},\mathsf{A}} \,.$$

From Lemma 6.2 it follows that there exists a λ-term H'' of type

$$[\![\sigma, n \models \psi_1]\!]_{\mathcal{A},\mathsf{A}} \;\to\; [\![\sigma, n \models \psi_2]\!]_{\mathcal{A},\mathsf{A}} \;\equiv\; [\![\sigma, n \models \psi_1 \Rightarrow \psi_2]\!]_{\mathcal{A},\mathsf{A}} \,.$$

- $\phi \equiv \psi_1 \,\mathcal{U}\, \psi_2$.

$$\begin{array}{ll} [\![\sigma, n \models \psi_1 \,\mathcal{U}\, \psi_2]\!]_{\mathcal{A},\mathsf{A}} & =_{\beta\iota} \\ [\![\psi_1 \,\mathcal{U}\, \psi_2]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n} & =_{\beta\iota} \\ [\![\mathcal{U}]\!]_{\mathcal{A},\mathsf{A}}\ [\![\psi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\psi_2]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n} & =_{\beta\iota} \\ \mathbf{E}\ \mathbf{N}\ \lambda m{:}\mathbf{N} \,.\, (\,[\![\psi_2]\!]_{\mathcal{A},\mathsf{A}}\ \sigma\ m) \curlywedge (tr\ \ [\![\psi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ m) & \end{array}$$

so by $\beta\iota$-conversion

$$\begin{array}{ll} \langle\ \rangle \vdash H : [\![\sigma, n \models \psi_1 \,\mathcal{U}\, \psi_2]\!]_{\mathcal{A},\mathsf{A}} & \Longleftrightarrow \\ \langle\ \rangle \vdash H : \mathbf{E}\ \mathbf{N}\ \lambda m{:}\mathbf{N} \,.\, (\,[\![\psi_2]\!]_{\mathcal{A},\mathsf{A}}\ \sigma\ m) \curlywedge (tr\ \ [\![\psi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ m) & \Longleftrightarrow \\ \text{the following three conditions are satisfied:} & \end{array}$$

(i): $H =_{\beta\iota} \mathsf{Constr}(1,\ [\![\phi_1 \,\mathcal{U}\, \phi_2]\!]_{\mathcal{A},\mathsf{A}}\,)\ \kappa\ H'$
(ii): $\langle\ \rangle \vdash \kappa : \mathbf{N}$
(iii): $\langle\ \rangle \vdash H' : (\,[\![\psi_2]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \ \kappa) \curlywedge (tr\ \ [\![\psi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ \kappa)$

Note that we use here that $[\![\psi_1]\!]_{\mathcal{A},\mathsf{A}}$, $[\![\psi_2]\!]_{\mathcal{A},\mathsf{A}}$, $[\![\sigma]\!]_{\mathcal{A},\mathsf{A}}$ and $\underline{n}$ do not contain a free variable m since they are closed terms. The term H' can be found iff we can find terms H'' and h (in the empty context) of type $(\,[\![\psi_2]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \ \kappa)$ and type $(tr\ \ [\![\psi_1]\!]_{\mathcal{A},\mathsf{A}}\ [\![\sigma]\!]_{\mathcal{A},\mathsf{A}}\ \underline{n}\ \kappa)$ respectively. Let $k \in \mathbb{N}$ such that $\underline{k} =_{\beta\iota} \kappa$. From the induction hypothesis it follows that H'' can be found iff $\sigma, k \models \psi_2$. The term h can be found iff $n > k$ and there are terms $h_{k+1}, \ldots, h_{n-1}$ such that

$$\langle\ \rangle \vdash h_i : [\![\psi_1]\!]_{\mathcal{A},\mathsf{A}}\ \sigma\ \underline{i} \qquad \text{for all } i = k+1, \ldots, n-1. \tag{10}$$

From the induction hypothesis it follows that (10) holds iff $\sigma, i \models \psi_1$ for all $i = k+1, \ldots, n-1$. When we combine the results that we obtained from the induction hypothesis than we find that we can construct H iff $\sigma, n \models \psi_1 \,\mathcal{U}\, \psi_2$.

❑

References

[1] H.P. Barendregt. Lambda calculi with types. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 117–309. Oxford Science Publications, 1992.

[2] J.A. Bergstra and J.W. Klop. The algebra of recursively defined processes and the algebra of regular processes. In *Proceedings of the* 11^{th} *ICALP,* Antwerp, volume 172 of *Lecture Notes in Computer Science*, pages 82–95. Springer-Verlag, 1984.

[3] P. Blackburn. Nominal tense logic. *Notre Dame Journal of Formal Logic*, 34(1):56–83, 1993.

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

[5] T. Coquand and G. Huet. The calculus of constructions. *Information and Control*, 76:95–120, 1988.

[6] G. Dowek, A. Felty, H. Herbelin, G. Huet, C. Murthy, C. Parent, C. Paulin-Mohring, and B. Werner. The Coq proof assistant user's guide. Version 5.8. Technical report, INRIA – Rocquencourt, May 1993.

[7] H. Geuvers. *Logics and Type Systems.* Phd thesis, University of Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, September 1993.

[8] B.T. Hailpern and S. Owicki. Modular verification of computer communication protocols. *IEEE Transactions on Computers*, 31(1):56–68, 1993.

[9] L. Helmink, M.P.A. Sellink, and F.W. Vaandrager. Proof-checking a data link protocol. In H.P. Barendregt and T. Nipkow, editors, *Proceedings of the* 1^{st} *Annual Workshop "Types for Proofs and Programs" TYPES '93,* Nijmegen, the Netherlands, volume 806 of *Lecture Notes in Computer Science*, pages 127–165. Springer-Verlag, 1993.

[10] L. Lamport. Specifying concurrent program modules. *ACM Transactions on Programming Languages and Systems*, 5(2):190–222, 1983.

[11] Z. Luo. ECC, the Extended Calculus of Constructions. In *Proceedings of the* 4^{th} *Annual Symposium on Logic in Computer Science,* Asilomar, California, pages 386–395. IEEE, 1989.

[12] N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. Technical Report MIT/LCS/TR-387, MIT Laboratory of Computer Science, April 1987.

[13] C. Paulin-Mohring. Inductive definitions in the system Coq. Rules and properties. In M. Bezem and J.F. Groote, editors, *Proceedings of the* 1^{st} *International Conference on Typed Lambda Calculi and Applications, TLCA '93,* Utrecht, The Netherlands, volume 664 of *Lecture Notes in Computer Science*, pages 328–345. Springer-Verlag, 1993.

[14] D.E. Shasha, A. Pnueli, and W. Ewald. Temporal verification of carrier-sense local area network protocols. In *Principles of Programming Languages*, pages 54–65. ACM, 1984.