Full Abstraction for a Recursively Typed Lambda Calculus with Parallel Conditional *

Fritz Müller

email: `(λx.muellerxcs.uni-sb.de)@`

URL: `rw4.cs.uni-sb.de/~mueller/recursive.ps.gz`

1993

Abstract

We define the syntax and reduction relation of a recursively typed lambda calculus with a parallel case-function (a parallel conditional). The reduction is shown to be confluent. We interpret the recursive types as information systems in a restricted form, which we call *prime systems*. A denotational semantics is defined with this interpretation. We define the syntactical normal form approximations of a term and prove the Approximation Theorem: The semantics of a term equals the limit of the semantics of its approximations. The proof uses inclusive predicates (logical relations). The semantics is adequate with respect to the observation of Boolean values. It is also fully abstract in the presence of the parallel case-function.

Keywords: lambda calculus, recursive type, parallel conditional, parallel or, confluence, denotational semantics, information system, approximation theorem, limiting completeness, inclusive predicates, adequacy, full abstraction

1 Introduction

In his seminal paper [Plo77], Gordon Plotkin explores the relationship between the operational (reduction) semantics and the denotational semantics of the functional programming language PCF. PCF is a call-by-name typed lambda calculus with the ground types boolean and integer, and any functional type. In order to compare operational and denotational semantics, one defines a notion of operational observation and a preorder on terms induced by this notion. In the case of PCF, the observation is of integer values only, and the preorder is defined by observation of arbitrary terms

*revised version of Report 12/1993 of SFB 124, FB 14 - Informatik, Universität des Saarlandes, Saarbrücken 1993 through integer contexts. The closed terms of ground type integer are singled out as *programs*. Programs are regarded as the only terms whose syntactical values (integers) can be observed directly. If the semantics of a program M is an integer value i, then M can be reduced to i. This result is called the *adequacy* of the semantics. (The denotational semantics is simply called the semantics here and in the following.)

A more general result about terms of any type is the *Approximation Theorem* or limiting completeness, as proved in [Wad78] for the untyped lambda calculus and in [Ber79] for PCF. The approximations of a term M are defined, roughly, as the normal form prefixes of the reducts of M. The Approximation Theorem states that the semantics of a term equals the limit of the semantics of its approximations.

Plotkin's programme proceeds as follows: The operational preorder on terms is defined as $M \sqsubseteq N$ iff for all contexts $C[\,]$ such that $C[M]$ and $C[N]$ are programs: if $C[M]$ reduces to a value i, then also $C[N]$. If $\mathcal{S}[\![M]\!] \sqsubseteq \mathcal{S}[\![N]\!]$, where $\mathcal{S}$ is the semantics function, then $M \sqsubseteq N$; this follows from adequacy. The converse, if $M \sqsubseteq N$ then $\mathcal{S}[\![M]\!] \sqsubseteq \mathcal{S}[\![N]\!]$, is not true for PCF with only sequential operations. This is due to the fact that there are parallel functions in the semantic model, like the parallel or, that cannot be defined syntactically. But when a parallel if-operation, or the parallel or, is added to the syntax, then "if $M \sqsubseteq N$ then $\mathcal{S}[\![M]\!] \sqsubseteq \mathcal{S}[\![N]\!]$" holds. This is called the *full abstraction* of the semantics; the operational and denotational preorders on terms coincide.

We elaborate the programme above for a call-by-name recursively typed lambda calculus and establish similar results : Approximation Theorem and adequacy for the sequential or parallel calculus and full abstraction for the parallel calculus only.

Chapter 2 defines the syntax and the reduction relation of our calculus. Types are built up from the separated sum $+$, the cartesian separated product $\times$, the function space $\rightarrow$, and recursion. Every recursive type denotes a possibly infinite type tree. Recursive types with the same type tree are regarded as equivalent. Terms are built up from variables, λ-abstraction, application, and constants for the type constructors $+$ and $\times$. Among the constants is a parallel case operation `pcase`. The operational semantics is defined by the one-step reduction $\rightarrow$ of a redex in any context. We prove that reduction is confluent. For the proof we use the confluence theorem of [Mül92] which says roughly: The combination of the lambda calculus with a confluent, left-linear and not variable-applying algebraic term rewriting system is confluent.

The subsequent chapters explore the semantics. We use information systems to give the semantics of recursive types [LW91, Win93]. Chapter 3 introduces a specialized form of information systems that we call *prime systems*: Here the predicates of consistency and entailment are given by binary relations on the set of *primes* (= tokens). Prime systems were first introduced for different purposes under the name event structures in [NPW81] and shown to be equivalent to prime algebraic coherent partial orders. We transfer the results of [LW91] to our prime systems: The class of prime systems is a complete partial order under the substructure relation $\trianglelefteq$. We define operations on prime systems corresponding to our type constructors $+$, $\times$, $\to$ and show that they are continuous.

This enables us, in Chapter 4, to give a semantic interpretation of type trees and recursive types as prime systems. The interpretation of finite prefixes of a type tree gives a $\trianglelefteq$-chain of prime systems; the interpretation of the whole type tree is the limit of this chain. Note that the primes at one level of the chain are directly contained in the following levels; there is no need for embedding-projection pairs as in the inverse limit solution of recursive domain equations. This is an advantage of the concrete representation of domains by information systems or prime systems. Anyway, this concrete representation of domain elements by sets of primes will be needed to prove full abstraction. Chapter 4 also gives the semantics function $\mathcal{S}$ on terms and proves its soundness: Reduction does not change the semantics of terms.

Chapter 5 proves the Approximation Theorem. We define a prefix order $\prec$ on terms where the constant Ω is the least term. A normal form A is an *approximation* of a term M iff there is a reduct N of M such that $A \prec N'$ for all reducts N' of N. The set $\mathcal{A}(M)$ of approximations of M is an ideal and can be seen as the syntactic value or Böhm tree of M. For the parallel calculus, it is not possible to define approximations by an analogue of head normal forms. But for the sequential calculus (without pcase), we give two analogues of head normal forms to define alternative sets of approximations. The Approximation Theorem says that the semantics of a term equals the limit of the semantics of its approximations. This is proved by the inclusive predicate technique, as it was used in [MP87] to prove the analogous theorem for the untyped lambda calculus. We adapt the technique to prime systems: We give an inductive definition of the inclusive predicates (logical relations) on the primes of our prime system interpretation of types.

Chapter 6 proves adequacy and full abstraction of the semantics. We have to define a notion of observation and the corresponding operational preorder on terms. We choose to observe the values $\mathsf{0}$ and $\mathsf{1}$ of type $\mathsf{bool} = \mathsf{void} + \mathsf{void}$, where void is the type of just one bottom element. So our *programs* are the closed terms of type bool. For a program M we define the operational value $\mathcal{O}[\![M]\!]$ as 0 or 1 if M reduces to $\mathsf{0}$ or $\mathsf{1}$ respectively, and as $\perp$ otherwise. The Adequacy Theorem says that $\mathcal{O}[\![M]\!] = \mathcal{S}[\![M]\!]\perp$ for every program M; it is a consequence of the Approximation Theorem.

The operational preorder on terms is defined as $M \sqsubseteq N$ iff for all contexts $C[\,]$ such that $C[M]$ and $C[N]$ are programs, $\mathcal{O}[\![C[M]]\!] \subseteq \mathcal{O}[\![C[N]]\!]$ holds. Again we have: If $\mathcal{S}[\![M]\!] \subseteq \mathcal{S}[\![N]\!]$, then $M \sqsubseteq N$, as a consequence of adequacy. Full abstraction, $M \sqsubseteq N$ iff $\mathcal{S}[\![M]\!] \subseteq \mathcal{S}[\![N]\!]$, is proved for the parallel calculus. As in [Plo77] the proof is based on the Definability Lemma: For all finite elements d of a semantic domain there is a term M with $\mathcal{S}[\![M]\!]\perp = d$. The proof uses the representation of elements as sets of primes.

The last Chapter 7 proves that the pcase-function is definable from the parallel and function.

Related work

Recently, [Win93] gave two recursively typed λ-calculi with their denotational semantics, by information systems, and proved the adequacy by the inclusive predicate (logical relation) technique. The first calculus has an eager (call-by-value) operational semantics. The second one has lazy (call-by-name) operational semantics like ours, but a different notion of observation is chosen: For every type certain terms are singled out as *canonical forms*. For product types these are the terms (M, N), for sum types $\mathsf{inl}(M)$ and $\mathsf{inr}(M)$, and for function types the terms $\lambda x.M$. The observation that is made of terms is the convergence to a canonical form. The given denotational semantics is adequate with respect to this notion of observation. This means that a term converges to a canonical form iff its semantics is not bottom. Especially, the semantics of every term $\lambda x.M$ is not bottom, whereas we have $\mathcal{S}[\![\lambda x.\Omega]\!]\varepsilon = \bot$.

Finally some remarks on coalesced sums and the observation of termination for all types. We did not include the coalesced sum in our type system, only separated sums. The coalesced sum of two domains is the disjoint union of the domains, with the two bottom elements identified. A coalesced sum would demand *strict* constructors inl: $\tau \to \tau \oplus \varrho$ and inr: $\varrho \to \tau \oplus \varrho$. These constructors have to evaluate their arguments to a non-bottom value before they can be used by a case-operation. (In contrast our corresponding constructors $\mathsf{0}$ and $\mathsf{1}$ are non-strict; they can be used without evaluated argument.) But the detection of non-bottom values is a complicated task for functional types, when we assume our denotational semantics of functions. On the other side I see no use for coalesced sums of functional types. Therefore I think that coalesced sums should be restricted to non-functional types, so that e.g. the recursive definition of the flat cpo of integers becomes possible. The check for non-bottomness of functional values, if it is desired, should be programmed using special functions incorporated in the language, e.g. Plotkin's "exists" operator.

[Cos89] constructs evaluators for a recursively typed lambda calculus with coalesced sums and strict, coalesced products of any type. The notion of observation for these evaluators is the observation of termination for terms of all types. The relation of operational and denotational semantics is given by the property of "complete adequacy": The semantics of any term is non-bottom iff its evaluation terminates. This ensures the detection of non-bottomness for coalesced sums. The work succeeds with a trick: The semantic domains are lattices; top elements (that are not syntactically definable) are added to the domains. Thus a term like $\lambda x.\mathsf{if}\, x\, (\mathsf{if}\, x\, \Omega\, \mathsf{0})\, \Omega$, whose normal semantics is $\bot$, now becomes non-bottom. For the normal cpo semantics only a vague sketch of an evaluator is given.

There has been later work proving adequacy for a lazy functional language with recursive and *polymorphic* types, also using information systems [BC94].

2 Syntax and reduction

2.1 Types

We adopt the syntax of the recursive type system of [CC90, CC91]. Especially, recursive types are considered equivalent if they have the same unfoldings as regular trees. But instead of type constants we have some more type constructors besides $\rightarrow$. The *type expressions* are given by the following grammar, where t stands for elements of a denumerable set V_T of type variables:

$$\tau ::= t \mid \tau + \tau \mid \tau \times \tau \mid \tau \rightarrow \tau \mid \mu t.\tau \mid \mathsf{void}$$

T_μ is the set of all type expressions. T_μ^c is the set of all closed type expressions, called *types*.

We give the informal meaning of types in terms of domains:

$\sigma + \tau$ is the separated sum of σ and τ,

$\sigma \times \tau$ is the cartesian separated product of σ and τ,

$\sigma \rightarrow \tau$ is the space of continuous functions from σ to τ,

$\mu t.\tau$ is the fixed point of the mapping $t \mapsto \tau$, the solution of the recursive domain equation $t = \tau$,

void is the canonical notation of the undefined type; it has the same meaning as $\mu t.t$. In [CC90] it is called Ω. The corresponding domain has just one element $\perp$.

We define the *simple types* by the grammar:

$$\tau ::= \mathsf{void} \mid \tau + \tau \mid \tau \times \tau \mid \tau \rightarrow \tau$$

T is the set of all simple types. It is $T \subseteq T_\mu^c$.

Definition 2.1 The void-prefix order $\prec \subseteq T \times T_\mu^c$ is the least partial order satisfying:

1) $\mathsf{void} \prec \tau$ for all $\tau \in T_\mu^c$,

2) $\sigma \prec \sigma', \tau \prec \tau' \;\Rightarrow\; \sigma \mathbin{@} \tau \prec \sigma' \mathbin{@} \tau'$
for $@ \in \{+, \times, \rightarrow\}$, $\sigma, \tau \in T$, and $\sigma', \tau' \in T_\mu^c$.

$\prec$ is a partial order on T. For every $\sigma, \tau \in T$ with an upper bound there is a least upper bound $\sigma \sqcup \tau \in T$. T_∞ denotes the ideal completion of T, i.e. the set of ideals of simple types, ordered by $\subseteq$. Here ideals are sets I of simple types that are non-empty, downward closed: $\tau \in I \wedge \sigma \prec \tau \;\Rightarrow\; \sigma \in I$, and directed: for all $\sigma, \tau \in I$ there is $\varrho \in I$ with $\sigma \prec \varrho$ and $\tau \prec \varrho$. The elements of T_∞ are called *type trees* and are also denoted by σ, τ, ϱ.

We define $\mathsf{void} \in T_\infty$ as $\mathsf{void} = \{\mathsf{void}\}$. For $@ = +, \times, \rightarrow$ and $\sigma, \tau \in T_\infty$ we define

$$\sigma \;@\; \tau = \{\mathsf{void}\} \cup \{\sigma' \;@\; \tau' \mid \sigma' \in \sigma \;\wedge\; \tau' \in \tau\}$$

Every type tree of T_∞ has one of the forms void, $\sigma + \tau$, $\sigma \times \tau$, $\sigma \rightarrow \tau$ with unique $\sigma, \tau \in T_\infty$.

Definition 2.2 The *unfolding* $\rightsquigarrow \subseteq T_\mu^c \times T_\mu^c$ is the least relation satisfying:

1) $\mu t.\tau \rightsquigarrow \tau[\mu t.\tau/t]$
   The right term is the replacement of $\mu t.\tau$ for all free occurrences of t in τ; it is also closed. Note that $\mu t.\tau$ does not contain free variables that could be bound after the replacement.

2) $\tau \rightsquigarrow \tau' \;\Rightarrow\; (\tau \;@\; \sigma) \rightsquigarrow (\tau' \;@\; \sigma)$ and $(\sigma \;@\; \tau) \rightsquigarrow (\sigma \;@\; \tau')$ for $@ \in \{+, \times, \rightarrow\}$, $\tau, \tau', \sigma \in T_\mu^c$.

$\rightsquigarrow$ reduces only one *outermost* redex $\mu t.\tau$. The outermost redexes are disjoint, therefore $\rightsquigarrow$ fulfills the diamond property: If $\tau \rightsquigarrow \sigma$ and $\tau \rightsquigarrow \varrho$, then there is ψ with $\sigma \rightsquigarrow \psi$ and $\varrho \rightsquigarrow \psi$.

$\rightsquigarrow^*$ is the reflexive, transitive closure of $\rightsquigarrow$. It is confluent: If $\tau \rightsquigarrow^* \sigma$ and $\tau \rightsquigarrow^* \varrho$, then there is ψ with $\sigma \rightsquigarrow^* \psi$ and $\varrho \rightsquigarrow^* \psi$.
If $\sigma \prec \tau$ and $\tau \rightsquigarrow^* \tau'$, then also $\sigma \prec \tau'$, for all $\sigma \in T$ and $\tau, \tau' \in T_\mu^c$.
For every $\tau \in T_\mu^c$ we define the unfolding

$$\tau^* = \{\sigma \in T \mid \exists \tau' \in T_\mu^c.\; \tau \rightsquigarrow^* \tau' \text{ and } \sigma \prec \tau'\}$$

Proposition 2.3 $\tau^* \in T_\infty$.

Proof: We have to show that τ^* is an ideal. It is non-empty, $\mathsf{void} \in \tau^*$, and downward closed. It is also directed: Let $\sigma, \varrho \in \tau^*$. Then there is τ' with $\tau \rightsquigarrow^* \tau'$, $\sigma \prec \tau'$ and τ'' with $\tau \rightsquigarrow^* \tau''$, $\varrho \prec \tau''$. As $\rightsquigarrow$ is confluent, there is ψ with $\tau' \rightsquigarrow^* \psi$ and $\tau'' \rightsquigarrow^* \psi$. It follows $\sigma \prec \psi$ and $\varrho \prec \psi$, therefore $\sigma \sqcup \varrho \prec \psi$ and $\sigma \sqcup \varrho \in \tau^*$. ■

Definition 2.4 We define an equivalence relation $\approx$ on types by: $\sigma \approx \tau$ iff $\sigma^* = \tau^*$.

$\approx$ is decidable [AC90].

2.2 Terms

For every type $\tau \in T_\mu^c$ there is a denumerable set V^τ of variables of type τ. The sets V^τ are mutually disjoint. Their members are denoted by $x^\tau, y^\tau, \ldots$ There is a set $\mathcal{C}$ of constants with types $ctype : \mathcal{C} \rightarrow T_\mu^c$.

General untyped terms are built from variables and constants by application MN and (λ-)abstraction $\lambda x.M$, without regarding the types. Λ is the set of all untyped terms.
We give rules for the formation of typed terms; $M : \sigma$ means: M has type σ, $\sigma \in T_\mu^c$:

(const) $c : ctype(c)$ for $c \in \mathcal{C}$

(var) $x^\sigma : \sigma$

($\rightarrow$ I) $M : \tau \;\Rightarrow\; \lambda x^\sigma.M : \sigma \rightarrow \tau$

$(\rightarrow \mathrm{E})\quad M:\sigma\rightarrow\tau, N:\sigma \;\Rightarrow\; MN:\tau$

$(\approx)\qquad M:\sigma, \sigma\approx\tau \;\Rightarrow\; M:\tau$

Terms are considered equal modulo α-conversion. We abbreviate $\lambda x.\lambda y.M$ as $\lambda xy.M$. Often type superscripts of variables will be omitted. $\mathcal{T}$ is the set of all typed terms. The type of a typed term is unique up to $\approx$, so the inference rules could be given for type trees instead of types. $\mathcal{T}_\sigma$ is the set of all terms with type $\sigma \in T_\mu^c$ or with type tree $\sigma \in T_\infty$. $\mathcal{T}_\sigma^c$ is the corresponding set of all closed terms. In the following chapters terms will always be understood to be typed.

For every type σ we can define a fixed point combinator:

$$Y_\sigma = \lambda y^{\sigma\rightarrow\sigma}.(\lambda x^{\mu t.(t\rightarrow\sigma)}.y(xx))(\lambda x^{\mu t.(t\rightarrow\sigma)}.y(xx)) : (\sigma\rightarrow\sigma)\rightarrow\sigma$$

Remark: We have given a type system with rule $(\approx)$ instead of explicit conversion operators between the types $\mu t.\sigma$ and $\sigma[\mu t.\sigma/t]$, called rep/abs, unfold/fold or elim/intro in [Win93, Cos89, AC90, Gun92]. There are untyped terms that can be typed in our system, but not in a system with explicit conversion, even with the introduction of arbitrary rep/abs in the term. E.g. let $M = Y(\lambda fx.f)$ and $N = Y(\lambda fxy.f)$ in (vM, vN). In this term, M and N must have the same type, which is impossible in an abs/rep-system. In our system the types of $M : \mu t.\sigma \rightarrow t$ and $N : \mu t.\sigma \rightarrow \sigma \rightarrow t$ are equivalent. Moreover our type system with rule $(\approx)$ has principle type schemes. A system with the weaker congruence $\sim$, as the smallest congruence (w.r.t. type constructors) such that $\mu t.\sigma \sim \sigma[\mu t.\sigma/t]$, lacks this property [CC90, CC91].

Our special set of constants consists of the following symbols for all types σ, τ, ϱ:

$\mathsf{0}_{\sigma,\tau}$: $\sigma\rightarrow(\sigma+\tau)$, also called “inleft” in the literature

$\mathsf{1}_{\sigma,\tau}$: $\tau\rightarrow(\sigma+\tau)$, also called “inright”

$\mathsf{case}_{\sigma,\tau,\varrho}$: $(\sigma+\tau)\rightarrow(\sigma\rightarrow\varrho)\rightarrow(\tau\rightarrow\varrho)\rightarrow\varrho$, sequential conditional

$\mathsf{pcase}_{\sigma,\tau,\varrho}$: $(\sigma+\tau)\rightarrow\varrho\rightarrow\varrho\rightarrow\varrho$, parallel conditional. Note the type different from case's type.

$\mathsf{pair}_{\sigma,\tau}$: $\sigma\rightarrow\tau\rightarrow(\sigma\times\tau)$, $\mathsf{pair}\,x\,y$ is also written (x,y)

$\mathsf{fst}_{\sigma,\tau}$: $(\sigma\times\tau)\rightarrow\sigma$

$\mathsf{snd}_{\sigma,\tau}$: $(\sigma\times\tau)\rightarrow\tau$

Ω_σ : σ, the canonical undefined term of type σ. Ω_σ has the same denotational semantics as $Y_\sigma(\lambda x^\sigma.x)$. There are no reduction rules for Ω.

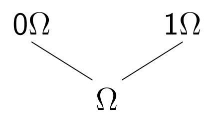
We will frequently omit the type subscripts of the constants. The term rewriting system will treat them as single symbols. Notice that we do not introduce these operators by special term formation rules for the types $\sigma+\tau$ and $\sigma\times\tau$, as it is often done, but as constants of higher order types that can be applied by normal application. $\mathsf{0}$, $\mathsf{1}$, pair are the constructors for building up the canonical terms of type $\sigma+\tau$, $\sigma\times\tau$ respectively. case, pcase, fst, snd are the corresponding evaluators. We will usually write $\mathsf{0}$ instead of $\mathsf{0}\Omega$ and $\mathsf{1}$ instead of $\mathsf{1}\Omega$.

We could also include in our calculus separated sum types with a different number of components than two. A special case would be the type constructor lift with just one type argument. It adds a new bottom element to the domain of the type. The constants for this type constructor would be $\ell_\sigma : \sigma \to (\mathsf{lift}\sigma)$ and $\mathsf{lcase}_{\sigma,\tau} : (\mathsf{lift}\sigma) \to (\sigma \to \tau) \to \tau$, corresponding to $\mathsf{0}$ and case. We omit this type constructor as it can be treated analogously to $+$.

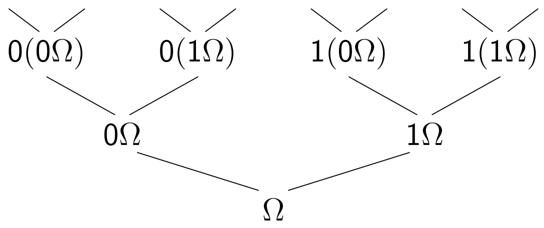
Examples of common types and their canonical terms:

$\mathsf{void} \approx \mu t.t$ has just one element, denoted by Ω_{void}.

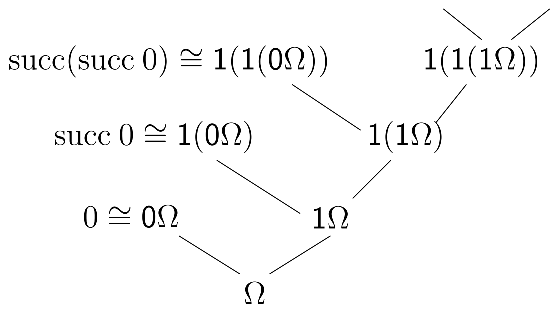
$\mathsf{bool} =_{\mathrm{def}} \mathsf{void} + \mathsf{void}$



$\mathsf{bitstream} =_{\mathrm{def}} \mu t.t + t$



$\mathsf{nat} =_{\mathrm{def}} \mu t.\mathsf{void} + t$,
the lazy natural numbers:



$\mathsf{boollist} =_{\mathrm{def}} \mu t.\mathsf{void} + (\mathsf{bool} \times t)$
$\mathsf{boollist}$ is the type of lists of elements of bool,
e.g. $\mathsf{0}_{\mathsf{void},\mathsf{bool}\times\mathsf{boollist}}\Omega_{\mathsf{void}} : \mathsf{boollist}$, simply written as $\mathsf{0}$ without type subscripts and undefined term Ω , the empty list,
e.g. $\mathsf{1}_{\mathsf{void},\mathsf{bool}\times\mathsf{boollist}}(\mathsf{1}_{\mathsf{void},\mathsf{void}}\Omega_{\mathsf{void}}, \mathsf{0}_{\mathsf{void},\mathsf{bool}\times\mathsf{boollist}}\Omega_{\mathsf{void}}) : \mathsf{boollist}$, simply written as $\mathsf{1}(\mathsf{1},\mathsf{0})$, the list of one element $\mathsf{1}$.

Note that "infinitely branching" domains, like the flat domain of natural numbers of PCF, cannot be defined in our type system because the type constructor of coalesced sums is missing.

2.3 Reduction

We define a reduction relation $\to$ on terms. It performs a one-step reduction of a single redex in any context. It is the least relation satisfying:

(β) the β-reduction rule:
$(\lambda x.M)N \to M[x{:=}N]$ for any terms M,N and variable x, where $M[x{:=}N]$ is the substitution of N for the free occurrences of x in M, with appropriate renaming of bound variables of M,

three context rules:

(app) $M \to M' \Longrightarrow MN \to M'N$,
$N \to N' \Longrightarrow MN \to MN'$,

(λ) $M \to M' \Longrightarrow \lambda x.M \to \lambda x.M'$,

and a set of applicative term rewriting rules for the constants, where the variables x, y, z, w denote arbitrary terms:

(case0)	$\mathsf{case}\ (0x)\ y\ z$	$\to$	$y\ x$
(case1)	$\mathsf{case}\ (1x)\ y\ z$	$\to$	$z\ x$
(pair1)	$\mathsf{fst}\ (\mathsf{pair}\ x\ y)$	$\to$	x
(pair2)	$\mathsf{snd}\ (\mathsf{pair}\ x\ y)$	$\to$	y
(pcase0)	$\mathsf{pcase}\ (0x)\ y\ z$	$\to$	y
(pcase1)	$\mathsf{pcase}\ (1x)\ y\ z$	$\to$	z
(pcase00)	$\mathsf{pcase}_{\sigma,\tau,\varrho_0+\varrho_1}\ x\ (0y)\ (0z)$	$\to$	$0\ (\mathsf{pcase}_{\sigma,\tau,\varrho_0}\ x\ y\ z)$
(pcase11)	$\mathsf{pcase}_{\sigma,\tau,\varrho_0+\varrho_1}\ x\ (1y)\ (1z)$	$\to$	$1\ (\mathsf{pcase}_{\sigma,\tau,\varrho_1}\ x\ y\ z)$
(pcase$\times\times$)	$\mathsf{pcase}_{\sigma,\tau,\varrho_1\times\varrho_2}\ x\ (y_1, y_2)\ (z_1, z_2)$	$\to$	$(\mathsf{pcase}_{\sigma,\tau,\varrho_1}\ x\ y_1\ z_1, \mathsf{pcase}_{\sigma,\tau,\varrho_2}\ x\ y_2\ z_2)$
(pcase $\to$)	$(\mathsf{pcase}_{\sigma,\tau,\varrho_1\to\varrho_2}\ x\ y\ z)\ w$	$\to$	$\mathsf{pcase}_{\sigma,\tau,\varrho_2}\ x\ (y\ w)\ (z\ w)$

$\to^*$ is the reflexive, transitive closure of $\to$.

Note the order of parameters of case: y is the 0-part, z is the 1-part. The functionality of case permits the definition of the usual evaluators "outleft" and "outright", so that we need not introduce them with reduction rules:

$$\begin{array}{lcl}
\mathsf{out0}_{\sigma,\tau} & : & (\sigma+\tau)\to\sigma \\
\mathsf{out0} & =_{\mathrm{def}} & \lambda x.\mathsf{case}\ x\ (\lambda y.y)\ \Omega \\
\mathsf{out1}_{\sigma,\tau} & : & (\sigma+\tau)\to\tau \\
\mathsf{out1} & =_{\mathrm{def}} & \lambda x.\mathsf{case}\ x\ \Omega\ (\lambda y.y)
\end{array}$$

pcase is not a sequential function, as it forces its three arguments to be reduced in parallel. As soon as the "boolean value" of its first argument appears, a reduction with rule (pcase0) or (pcase1) can be made. As soon as the second and the third argument convey the same piece of information, namely a constructor 0, 1 or pair, this piece of information can be drawn out of the pcase-expression according to rule (pcase00), (pcase11) or (pcase$\times\times$). If the second and the third argument are of functional type, then the argument w of the pcase-expression can be drawn in according to rule ($\mathsf{pcase}{\rightarrow}$), so that $(y\,w)$ and $(z\,w)$ can deliver constructor information before the evaluation of x is finished. Note that pcase appears on the right sides of its rules ($\mathsf{pcase00}$)–($\mathsf{pcase}{\rightarrow}$). It performs a recursion on the type tree of its second and third argument. We could think of a parallel conditional with the same type as case. But for such a conditional it is more difficult to implement this recursion by rewrite rules; in fact we would need conditioned rewrite rules with λ-abstractions and $\mathsf{out0}$, $\mathsf{out1}$ in the right sides.

Proposition 2.5 *Our reduction relation $\rightarrow$ fulfills the subject reduction property: If $M : \sigma$ and $M \rightarrow^* N$, then also $N : \sigma$.*

Proof: The property can be checked for each reduction rule.

Theorem 2.6 (Confluence) *$\rightarrow$ is confluent (Church-Rosser) on typed terms:*
For any typed term $M \in \mathcal{T}$ with $N \leftarrow^* M \rightarrow^* P$ there is a term Q with $N \rightarrow^* Q \leftarrow^* P$. ($N, P, Q$ are also typed with equivalent types due to the subject reduction property.)

Note that the restriction of M to typed terms is essential, as can be seen with the term $\mathsf{pcase}\ x\ (\mathsf{0}y)\ (\mathsf{0}z)\ w$. This term is not typable, as $(\mathsf{0}y)$ is not of function type. It reduces to $\mathsf{pcase}\ x\ (\mathsf{0}y\ w)\ (\mathsf{0}z\ w)$ by rule ($\mathsf{pcase}{\rightarrow}$), and to $\mathsf{0}(\mathsf{pcase}\ x\ y\ z)\ w$ by rule ($\mathsf{pcase00}$). This critical pair does not converge to a common reduct.

Proof: We will use the confluence theorem of [Mül92]: For every left-linear, not variable-applying ATRS (applicative term rewriting system) with reduction relation $\rightarrow$ and every $\rightarrow$-closed set T of terms: If $\rightarrow$ is confluent on the applicative terms of T then $\rightarrow$ is confluent on T. We explain the notions of this theorem in our context:

The applicative terms are the terms without any λ-abstraction, i.e. they are built only from variables, constants and application. An ATRS is a set of pairs $\langle L{\rightarrow}R\rangle$ of applicative terms, where L is no variable and all variables of R appear in L, too. In our case, the ATRS is the set of reduction rules ($\mathsf{case0}$) ... ($\mathsf{pcase}{\rightarrow}$). Together with β-reduction and the context rules (app) and (λ) it determines the reduction relation $\rightarrow$ on terms of Λ. It is left-linear, i.e. every variable has at most one occurrence in each left side of the rules. It is not variable-applying, i.e. no left side of any rule contains a subterm of the form (xM), where x is a variable. In our case, T will be the set $\mathcal{T}$ of typed terms. $\mathcal{T}$ is $\rightarrow$-closed, i.e. for every $M \in \mathcal{T}$ the following hold:
1) $M \rightarrow M' \Rightarrow M' \in \mathcal{T}$, the subject reduction property,
2) every subterm of M is in $\mathcal{T}$,
3) for every occurrence u of an abstraction in M, $M/u = \lambda \ldots$, there is a variable x not occurring in M with $M[u \leftarrow x] \in \mathcal{T}$.
We use the same notations for occurrences of subterms and replacement at an occurrence as [Hue80, Mül92]. In condition 3 we chose a new variable of the appropriate type.

Now it remains to prove the confluence of $\rightarrow$ on the set $\mathcal{A}$ of applicative terms of $\mathcal{T}$, i.e. the confluence of the ATRS alone, without β-reduction. Our theorem, the confluence of $\rightarrow$ on all terms of $\mathcal{T}$, follows by the cited theorem.

From now on, $\rightarrow$ is the reduction relation on applicative terms of Λ. We will first prove that $\rightarrow$ is locally confluent on $\mathcal{A}$ via convergence of critical pairs, then prove that $\rightarrow$ is noetherian (terminating, strongly normalizing) and conclude the confluence of $\rightarrow$ on $\mathcal{A}$ by Newman's Lemma (Lemma 2.4 of [Hue80]). *Local (or weak) confluence* of $\rightarrow$ on a set T of terms means: For any $M \in T$ with $N \leftarrow M \rightarrow P$ there is a term Q with $N \rightarrow^* Q \leftarrow^* P$.

Notice that the sufficient conditions for confluence in [Hue80] that check only convergence of critical pairs, without termination, are not applicable here: Huet's Lemma 3.3 is almost applicable (Corollary: Any left-linear parallel closed term rewriting system is confluent), but it demands of the critical pair:
$y\ w \leftarrow (\mathsf{pcase}\ (0x)\ y\ z)\ w \rightarrow \mathsf{pcase}\ (0x)\ (y\ w)\ (z\ w)$ that there should be a parallel reduction step: $y\,w \rightarrow \mathsf{pcase}\,(0x)\,(y\,w)\,(z\,w)$. Note that the right term of a critical pair is defined by a reduction *at the root*. The lemma demands a parallel reduction step from the left to the right term, not an arbitrary reduction. But in our example there is only a reduction in the opposite direction. [Toy88, Corollary 3.2] gives a sufficient condition more general than Huet's Lemma 3.3; it is also not applicable here by the same reason.

For the proof of local confluence of $\rightarrow$ on $\mathcal{A}$ we will apply a generalized version of Lemma 3.1 of [Hue80]: "For any term rewriting system $\mathcal{R}$: The relation $\rightarrow_{\mathcal{R}}$ is locally confluent iff for every critical pair (P, Q) of $\mathcal{R}$ we have $P \downarrow Q$, i.e. P and Q have a common reduct." This lemma cannot be applied directly, as the non-typable, non-convergent critical pair given before this proof shows us. It should state local confluence on certain subsets of terms which resemble sets of well-typed terms, similar to the $\rightarrow$-closed sets of terms above. This leads us to:

Definition 2.7 A subset T of terms is called $\rightarrow_{\mathcal{R}}$*-complete* for a term rewriting system with reduction relation $\rightarrow_{\mathcal{R}}$ if for every $M \in T$ the following hold:

1) $M \rightarrow_{\mathcal{R}} M' \Rightarrow M' \in T$,
2) every subterm of M is in T,
3) for every set of occurrences $u_1, \ldots, u_n$ of the same subterm N in M, i.e. $M/u_i = N$ for all i, there is a variable x not occurring in M with $M[u_1 \leftarrow x] \ldots [u_n \leftarrow x] \in T$.

Let us recall the definition of critical pairs of a term rewriting system.

Definition 2.8 Let $\langle S {\rightarrow} T\rangle$,$\langle L {\rightarrow} R\rangle$ be two rules whose variables are renamed such that L and S have disjoint variable sets. Let u be an occurrence in L such that L/u is no variable and L/u and S are unifiable with substitution μ as the most general unifier. The superposition of $\langle S {\rightarrow} T\rangle$ on $\langle L {\rightarrow} R\rangle$ in u determines the *critical pair* (P, Q) defined by $P = (\mu L)[u \leftarrow \mu T]$, $Q = \mu R$. It is $P \leftarrow \mu L \rightarrow Q$. We call μL an *overlap* of the critical pair (P, Q).

Our generalization of Huet's Lemma 3.1 is now:

Lemma 2.9 *For any term rewriting system $\mathcal{R}$ and $\to_{\mathcal{R}}$-complete subset T of terms: The reduction relation $\to_{\mathcal{R}}$ is locally confluent on T iff for every critical pair (P, Q) of $\mathcal{R}$ with an overlap in T we have $P \downarrow Q$.*

Proof: (sketch) The proof is essentially the proof of Lemma 3.1 in [Hue80]. The "only if" part is trivial again. For the "if" part we add the assumption $M \in T$. Case 1 (disjoint redexes) and case 2a (prefix redexes that do not overlap) are the same as in [Hue80]. Case 2b deals with overlapping redexes: An overlap of the critical pair is obtained from the subterm M/u_1 by replacing some subterms by variables. It is $M/u_1 \in T$ according to condition 2 of $\to_{\mathcal{R}}$-completeness. The replacement of subterms by variables is possible according to condition 3 of $\to_{\mathcal{R}}$-completeness, so that the overlap is in T. Thus $P \downarrow Q$ by hypothesis, and the proof proceeds as in [Hue80]. ■

We use the lemma to show local confluence of $\to$ on $\mathcal{A}$. $\mathcal{A}$ is $\to$-complete. Eight critical pairs with an overlap in $\mathcal{A}$ remain to be checked for convergence:

$$
\begin{array}{rcl}
(\mathsf{0}y) \leftarrow & \mathsf{pcase}\ (\mathsf{0}x)\ (\mathsf{0}y)\ (\mathsf{0}z) & \to \mathsf{0}(\mathsf{pcase}\ (\mathsf{0}x)\ y\ z) \\
(\mathsf{1}y) \leftarrow & \mathsf{pcase}\ (\mathsf{0}x)\ (\mathsf{1}y)\ (\mathsf{1}z) & \to \mathsf{1}(\mathsf{pcase}\ (\mathsf{0}x)\ y\ z) \\
(\mathsf{0}z) \leftarrow & \mathsf{pcase}\ (\mathsf{1}x)\ (\mathsf{0}y)\ (\mathsf{0}z) & \to \mathsf{0}(\mathsf{pcase}\ (\mathsf{1}x)\ y\ z) \\
(\mathsf{1}z) \leftarrow & \mathsf{pcase}\ (\mathsf{1}x)\ (\mathsf{1}y)\ (\mathsf{1}z) & \to \mathsf{1}(\mathsf{pcase}\ (\mathsf{1}x)\ y\ z) \\
(y_1, y_2) \leftarrow & \mathsf{pcase}\ (\mathsf{0}x)\ (y_1, y_2)\ (z_1, z_2) & \to (\mathsf{pcase}\ (\mathsf{0}x)\ y_1\ z_1, \mathsf{pcase}\ (\mathsf{0}x)\ y_2\ z_2) \\
(z_1, z_2) \leftarrow & \mathsf{pcase}\ (\mathsf{1}x)\ (y_1, y_2)\ (z_1, z_2) & \to (\mathsf{pcase}\ (\mathsf{1}x)\ y_1\ z_1, \mathsf{pcase}\ (\mathsf{1}x)\ y_2\ z_2) \\
y\ w \leftarrow & \mathsf{pcase}\ (\mathsf{0}x)\ y\ z\ w & \to \mathsf{pcase}\ (\mathsf{0}x)\ (y\ w)\ (z\ w) \\
z\ w \leftarrow & \mathsf{pcase}\ (\mathsf{1}x)\ y\ z\ w & \to \mathsf{pcase}\ (\mathsf{1}x)\ (y\ w)\ (z\ w)
\end{array}
$$

We prove now that $\to$ is noetherian on applicative terms. (This will also be used in the proof of Lemma 5.3.) We define a mapping φ from applicative terms to $\{2, 3, \ldots\}$ inductively by the following equations:

$$
\begin{array}{lcl}
\varphi M & = & 2, \text{ if } M \text{ is a variable or a constant} \\
\varphi(\mathsf{0}M) & = & 2 \cdot \varphi M \\
\varphi(\mathsf{1}M) & = & 2 \cdot \varphi M \\
\varphi(\mathsf{pcase}\ M) & = & 2 \cdot \varphi M \\
\varphi(\mathsf{pcase}\ MN) & = & 2 \cdot \varphi M \cdot \varphi N \\
\varphi(\mathsf{pcase}\ MNP) & = & 2 \cdot \varphi M \cdot \varphi N \cdot \varphi P \\
\varphi(\mathsf{pair}\ M) & = & 2 + \varphi M \\
\varphi(\mathsf{pair}\ MN) & = & 2 + \varphi M + \varphi N \\
\varphi(MN) & = & (\varphi M)^{\varphi N}, \text{ for all other applications } MN
\end{array}
$$

By simple computations we show for every reduction rule $\langle L \rightarrow R \rangle$ that $\varphi L > \varphi R$, where variables of the rule stand for arbitrary terms. The two interesting rules are:

$(\mathsf{pcase} \times \times)$ $\mathsf{pcase}\, x\, (\mathsf{pair}\, y_1\, y_2)\, (\mathsf{pair}\, z_1\, z_2) \;\rightarrow\; \mathsf{pair}\, (\mathsf{pcase}\, x\, y_1\, z_1)\, (\mathsf{pcase}\, x\, y_2\, z_2)$

$\varphi L = 2 \cdot \varphi x \cdot (2 + \varphi y_1 + \varphi y_2) \cdot (2 + \varphi z_1 + \varphi z_2)$

$\varphi R = 2 + 2 \cdot \varphi x \cdot \varphi y_1 \cdot \varphi z_1 + 2 \cdot \varphi x \cdot \varphi y_2 \cdot \varphi z_2$

$(\mathsf{pcase} \rightarrow)$ $(\mathsf{pcase}\, x\, y\, z)\, w \;\rightarrow\; \mathsf{pcase}\, x\, (y\, w)\, (z\, w)$

$\varphi L = 2^{\varphi w} \cdot (\varphi x)^{\varphi w} \cdot (\varphi y)^{\varphi w} \cdot (\varphi z)^{\varphi w}$

$\varphi R = 2 \cdot \varphi x \cdot \varphi (y\, w) \cdot \varphi (z\, w)$

For the last rule (and some other) we need the fact that $(\varphi M)^{\varphi N} \geq \varphi(MN)$ for all terms M, N, which we prove by a case analysis over the term M.

It remains to show that a reduction at any position decreases the φ-value of a term. We prove that

$$\varphi N > \varphi N' \Rightarrow \varphi(MN) > \varphi(MN')$$

and that

$$\varphi M > \varphi M' \text{ and } M \rightarrow M' \Rightarrow \varphi(MN) > \varphi(M'N)$$

for all terms M, N, M', N' by a case analysis over M.

We have now proved that $M \rightarrow N \Rightarrow \varphi M > \varphi N$. Thus there are no infinite reduction chains. From this and the local confluence of $\rightarrow$ on $\mathcal{A}$ follows by Newman's Lemma the confluence of $\rightarrow$ on $\mathcal{A}$. As explained above, the confluence of $\rightarrow$ on all typed terms follows from the theorem of [Mül92]. ■

3 Prime systems

We introduce prime systems as concrete representations of domains, together with operations on them corresponding to the type constructors $+, \times, \rightarrow$. The results of this chapter are taken from [LW91] , where they were given for the more general information systems.

Definition 3.1 A *prime system* $\mathcal{A} = (A, \uparrow, \leq)$ consists of
a set A (the *primes*, denoted by a, b, c),
a reflexive and symmetric binary relation $\uparrow$ on A (the *consistency*),
and a partial order $\leq$ on A (the *entailment*),
such that for all $a, b, c \in A$: If $a \uparrow b$ and $c \leq b$, then $a \uparrow c$.
PSys is the class of all prime systems.

Prime systems were first introduced in [NPW81] under the name "event structures", where the elements of A were interpreted as events of a computation process. (Instead of consistency there was the dual conflict relation.) Here we chose a different name because we do not interpret the elements of A as events, but as pieces of information, as in information systems. A prime is an elementary, indivisible piece of information about data elements. The relation $a \leq b$ means that whenever b is valid of an element, then so is a. $a \uparrow b$ means that both primes a and b may be valid of an element.

Every prime system determines an information system in the sense of [LW91]: The set of tokens is A. A finite subset X of A is consistent ($X \in \mathrm{Con}$) iff for all $a, b \in X$, $a \uparrow b$. For $X \in \mathrm{Con}$ and $a \in A$ we define $X \vdash a$ iff $\exists b \in X.\ a \leq b$. We use the simpler prime systems instead of information systems as they are just suited for our data types.

Definition 3.2 The *elements* of a prime system $\mathcal{A} = (A, \uparrow, \leq)$ are the subsets $d \subseteq A$ that are downward closed: $a \leq b \wedge b \in d \Rightarrow a \in d$, and consistent: $a \uparrow b$ for all $a, b \in d$.

$|\mathcal{A}|$ is the set of elements of $\mathcal{A}$. We call $|\mathcal{A}|$, ordered by $\subseteq$, the *domain* of $\mathcal{A}$. The least element $\emptyset$ is also denoted by $\perp$.

For $X \subseteq A$ we write $X{\downarrow} = \{a \in A \mid \exists b \in X.\ a \leq b\}$, also $a{\downarrow}$ for $\{a\}{\downarrow}$. The *finite elements* of $\mathcal{A}$ are defined as the elements of the form $X{\downarrow}$ for finite $X \subseteq A$.

We will give the characterization of the domains of prime systems from [NPW81]. First some domain theoretic definitions.

Definition 3.3 Let $(D, \sqsubseteq)$ be a partial order. A subset of D is *pairwise consistent* iff any two of its elements have an upper bound in D. $(D, \sqsubseteq)$ is *coherent* iff every pairwise consistent subset of D has a lub.

$p \in D$ is a *complete prime* iff for every $S \subseteq D$, if the lub $\bigsqcup S$ exists and $p \sqsubseteq \bigsqcup S$, then there is $d \in S$ with $p \sqsubseteq d$.
$(D, \sqsubseteq)$ is *prime algebraic* iff for every $d \in D$ the set $\{p \sqsubseteq d \mid p \text{ is a complete prime}\}$ has d as its lub.

Theorem 3.4 *[NPW81] Let $\mathcal{A} = (A, \uparrow, \leq)$ be a prime system. Then $(|\mathcal{A}|, \subseteq)$ is a prime algebraic coherent partial order. Its complete primes are the elements $a{\downarrow}$ for $a \in A$.*
It follows that $(|\mathcal{A}|, \subseteq)$ is also an algebraic cpo. Its isolated (or finite, compact) elements are the finite elements defined above.

Conversely, let $(D, \sqsubseteq)$ be a prime algebraic coherent partial order. Let P be the set of complete primes of D, and $a \uparrow b$ iff $a, b \in P$ have an upper bound. Then $\mathcal{P} = (P, \uparrow, \sqsubseteq)$ is a prime system with $(|\mathcal{P}|, \subseteq)$ isomorphic to $(D, \sqsubseteq)$.

This theorem explains our name for “primes”. From this characterization we only need the fact that the domain of a prime system is a cpo, i.e. has lubs of directed subsets. These lubs are the set unions of the elements.

As in [LW91] we define a complete partial order on the class of prime systems and continuous operations on prime systems.

Definition 3.5 Let $\mathcal{A} = (A, \uparrow_A, \leq_A)$ and $\mathcal{B} = (B, \uparrow_B, \leq_B)$ be prime systems. We define $\mathcal{A} \trianglelefteq \mathcal{B}$ iff $A \subseteq B$ and for all $a, b \in A$: $a \uparrow_A b \Leftrightarrow a \uparrow_B b$ and $a \leq_A b \Leftrightarrow a \leq_B b$.

$\mathcal{A} \trianglelefteq \mathcal{B}$ means that $\mathcal{A}$ is a *subsystem* of $\mathcal{B}$: $A \subseteq B$ and $\uparrow_A, \leq_A$ are the restrictions of $\uparrow_B, \leq_B$ on A. If $\mathcal{A} \trianglelefteq \mathcal{B}$ and $A = B$, then $\mathcal{A} = \mathcal{B}$.

Theorem 3.6 *$\trianglelefteq$ is a partial order with $\perp = (\emptyset, \emptyset, \emptyset)$ as least element. If $\mathcal{A}_0 \trianglelefteq \mathcal{A}_1 \trianglelefteq \ldots$ is an ω-chain of prime systems $\mathcal{A}_i = (A_i, \uparrow_i, \leq_i)$, then*

$$\bigcup_i \mathcal{A}_i = (\bigcup_i A_i, \bigcup_i \uparrow_i, \bigcup_i \leq_i)$$

is the lub of the chain.

Proof: Clearly $\trianglelefteq$ is a partial order, $\perp$ is the least element.
Now for the chain $\mathcal{A}_i$ let $\mathcal{A} = (A, \uparrow, \leq) = (\bigcup_i A_i, \bigcup_i \uparrow_i, \bigcup_i \leq_i)$.

$\mathcal{A}$ is an upper bound of the chain: $A_i \subseteq A$ for all i. Let $a, b \in A_i$. If $a \uparrow_i b$, then $a \uparrow b$. Conversely, if $a \uparrow b$, then $a, b \in A_j$ and $a \uparrow_j b$ for some j. If $j \leq i$, then $\mathcal{A}_j \trianglelefteq \mathcal{A}_i$; if $i \leq j$, then $\mathcal{A}_i \trianglelefteq \mathcal{A}_j$. In either case follows $a \uparrow_i b$. Analogously we show $a \leq_i b \Leftrightarrow a \leq b$.

$\mathcal{A}$ is the least upper bound of the chain: Let $\mathcal{B} = (B, \uparrow_B, \leq_B)$ be an upper bound of the chain. Then $A = \bigcup_i A_i \subseteq B$. Let $a, b \in A$. Then $a, b \in A_i$ for some i. We have $a \uparrow b \Leftrightarrow a \uparrow_i b \Leftrightarrow a \uparrow_B b$ and $a \leq b \Leftrightarrow a \leq_i b \Leftrightarrow a \leq_B b$. ■

We extend $\trianglelefteq$ to n-tuples of prime systems.

Definition 3.7 For $n \geq 1$, $\mathbf{PSys}^n$ are all n-tuples $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ of prime systems. We define

$$(\mathcal{A}_1, \ldots, \mathcal{A}_n) \trianglelefteq (\mathcal{B}_1, \ldots, \mathcal{B}_n) \Leftrightarrow \mathcal{A}_1 \trianglelefteq \mathcal{B}_1 \wedge \ldots \wedge \mathcal{A}_n \trianglelefteq \mathcal{B}_n.$$

.

Proposition 3.8 *$\trianglelefteq$ is a partial order on $\mathbf{PSys}^n$ with $(\perp, \ldots, \perp)$ as least element. All increasing ω-chains in $(\mathbf{PSys}^n, \trianglelefteq)$ have a least upper bound taken coordinate-wise.*

Definition 3.9 Let $F : \mathbf{PSys}^n \to \mathbf{PSys}$ be an operation on prime systems.
F is called *monotonic* iff $\mathcal{A} \trianglelefteq \mathcal{B} \Rightarrow F(\mathcal{A}) \trianglelefteq F(\mathcal{B})$ for all $\mathcal{A}, \mathcal{B} \in \mathbf{PSys}^n$.
F is called *continuous* iff it is monotonic and for any ω-chain of prime systems $\mathcal{A}_0 \trianglelefteq \mathcal{A}_1 \trianglelefteq \ldots$ in $\mathbf{PSys}^n$, $F(\bigcup_i \mathcal{A}_i) = \bigcup_i F(\mathcal{A}_i)$. (Since F is monotonic, $F(\mathcal{A}_i), i \geq 0$, is an ascending chain and $\bigcup_i F(\mathcal{A}_i)$ exists.)

Proposition 3.10 $F : \mathbf{PSys}^n \to \mathbf{PSys}$ *is monotonic (continuous) iff it is monotonic (continuous) in each argument separately (i.e. considered as a function in any of its arguments, holding the others fixed).*

Thus to show that an operation is monotonic or continuous we have to show that some unary operations are monotonic or continuous. The following lemma will help in these proofs.

Definition 3.11 $F : \mathbf{PSys} \to \mathbf{PSys}$ is *continuous on prime sets* iff for any ω-chain of prime systems $\mathcal{A}_0 \trianglelefteq \mathcal{A}_1 \trianglelefteq \ldots$ each prime of $F(\bigcup_i \mathcal{A}_i)$ is a prime of $\bigcup_i F(\mathcal{A}_i)$.

Lemma 3.12 $F : \mathbf{PSys} \to \mathbf{PSys}$ *is continuous iff* F *is monotonic and continuous on prime sets.*

Proof: The "only if" part is obvious.
"if": Let $\mathcal{A}_0 \trianglelefteq \mathcal{A}_1 \trianglelefteq \ldots$ be an ω-chain of prime systems. From $\mathcal{A}_i \trianglelefteq \bigcup_i \mathcal{A}_i$ and monotonicity follows $F(\mathcal{A}_i) \trianglelefteq F(\bigcup_i \mathcal{A}_i)$. Then $\bigcup_i F(\mathcal{A}_i) \trianglelefteq F(\bigcup_i \mathcal{A}_i)$. As F is continuous on prime sets, the primes of $\bigcup_i F(\mathcal{A}_i)$ are the same as those of $F(\bigcup_i \mathcal{A}_i)$. Therefore they are the same prime systems. ■

Operations on prime systems

We give continuous operations on prime systems corresponding to our syntactic type constructors $\mathsf{void}, +, \times, \to$.
Corresponding to void is the prime system $\underline{\perp} = (\emptyset, \emptyset, \emptyset)$. It has the only element $\emptyset = \perp$.

Separated sum +

Definition 3.13 Let $\mathcal{A}_0 = (A_0, \uparrow_0, \leq_0)$ and $\mathcal{A}_1 = (A_1, \uparrow_1, \leq_1)$ be prime systems. Define $\mathcal{A}_0 + \mathcal{A}_1 = (B, \uparrow, \leq)$ by

$$\begin{array}{rcl} B & = & B_0 \cup B_1 \\ & & \quad \text{where } B_0 = \{0\} \cup (\{0\} \times A_0) \text{ and } B_1 = \{1\} \cup (\{1\} \times A_1), \\ a \uparrow b & \Leftrightarrow & (a, b \in B_0 \text{ and if } a = (0, a_0), b = (0, b_0),\ \text{then } a_0 \uparrow_0 b_0) \\ & \text{or} & (a, b \in B_1 \text{ and if } a = (1, a_1), b = (1, b_1),\ \text{then } a_1 \uparrow_1 b_1), \\ a \leq b & \Leftrightarrow & a = 0,\ b \in B_0 \\ & \text{or} & a = 1,\ b \in B_1 \\ & \text{or} & a = (0, a_0),\ b = (0, b_0),\ a_0 \leq_0 b_0 \\ & \text{or} & a = (1, a_1),\ b = (1, b_1),\ a_1 \leq_1 b_1. \end{array}$$

Proposition 3.14 $\mathcal{A}_0 + \mathcal{A}_1$ *is a prime system. Its domain is*

$$|\mathcal{A}_0 + \mathcal{A}_1| = \{\emptyset\} \cup \{\{0\} \cup (\{0\} \times d) \mid d \in |\mathcal{A}_0|\} \cup \{\{1\} \cup (\{1\} \times d) \mid d \in |\mathcal{A}_1|\}.$$

We abbreviate the element $\{0\}$ as 0 and $\{1\}$ as 1.

Theorem 3.15 $+$ *is continuous on* $(\mathbf{PSys}, \trianglelefteq)$.

Proof: It is easy to show that $+$ is continuous in its first and second argument, using Lemma 3.12. ■

Product $\times$

Definition 3.16 Let $\mathcal{A}_0 = (A_0, \uparrow_0, \leq_0)$ and $\mathcal{A}_1 = (A_1, \uparrow_1, \leq_1)$ be prime systems. Define $\mathcal{A}_0 \times \mathcal{A}_1 = (B, \uparrow, \leq)$ by

$$\begin{array}{rcl}
B & = & (\{0\} \times A_0) \cup (\{1\} \times A_1), \\
a \uparrow b & \Leftrightarrow & a = (0, a_0),\ b = (0, b_0),\ a_0 \uparrow_0 b_0 \\
& \text{or} & a = (1, a_1),\ b = (1, b_1),\ a_1 \uparrow_1 b_1 \\
& \text{or} & a = (0, a_0),\ b = (1, b_1) \\
& \text{or} & a = (1, a_1),\ b = (0, b_0), \\
a \leq b & \Leftrightarrow & a = (0, a_0),\ b = (0, b_0),\ a_0 \leq_0 b_0 \\
& \text{or} & a = (1, a_1),\ b = (1, b_1),\ a_1 \leq_1 b_1.
\end{array}$$

Proposition 3.17 *$\mathcal{A}_0 \times \mathcal{A}_1$ is a prime system. Its domain is*

$$|\mathcal{A}_0 \times \mathcal{A}_1| = \{(\{0\} \times d) \cup (\{1\} \times e) \mid d \in |\mathcal{A}_0| \ \wedge\ e \in |\mathcal{A}_1|\}$$

Theorem 3.18 *$\times$ is continuous on $(\mathbf{PSys}, \trianglelefteq)$.*

Proof: It is easy to show that $\times$ is continuous in its first and second argument, using Lemma 3.12. ∎

Function space $\rightarrow$

Definition 3.19 Let $\mathcal{A} = (A, \uparrow_A, \leq_A)$ and $\mathcal{B} = (B, \uparrow_B, \leq_B)$ be prime systems. (We leave out the indexes in the following.)
We define $\mathcal{A} \rightarrow \mathcal{B} = (C, \uparrow, \leq)$:
$C = \overline{A} \times B$, where $\overline{A}$ is the set of all finite subsets of A that are pairwise consistent and incomparable, $\overline{A} = \{X \subseteq A \mid X \text{ finite and } \forall a, b \in X.\ a \uparrow b \ \wedge\ (a \leq b \ \Rightarrow\ a = b)\}$.
Let $(X, a), (Y, b) \in C$.

$$(X, a) \uparrow (Y, b) \ \Leftrightarrow\ (X \uparrow Y \ \Rightarrow\ a \uparrow b),$$

where $X \uparrow Y \ \Leftrightarrow\ \forall a \in X, b \in Y.\ a \uparrow b$.

$$(X, a) \leq (Y, b) \ \Leftrightarrow\ Y \leq X \text{ and } a \leq b,$$

where $Y \leq X \ \Leftrightarrow\ Y \subseteq X\!\downarrow$, i.e. $\forall a \in Y.\ \exists b \in X.\ a \leq b$.

Proposition 3.20 *$\mathcal{A} \rightarrow \mathcal{B}$ is a prime system.*

Proof:
$\uparrow$ is reflexive and symmetric. $\leq$ is reflexive.

$\leq$ is antisymmetric:
Let $(X, a) \leq (Y, b)$ and $(Y, b) \leq (X, a)$. We show $(X, a) = (Y, b)$.
We have $a \leq b$ and $b \leq a$, so $a = b$.
From $X \leq Y$ and $Y \leq X$ we conclude $X \subseteq Y$:
Let $x \in X$. There is $y \in Y$ with $x \leq y$, and $x' \in X$ with $y \leq x'$. So $x \leq x'$, and $x = x'$ by the condition on X. Hence $x = y \in Y$.
Similarly we conclude $Y \subseteq X$.

$\leq$ is transitive:
Let $(X, a) \leq (Y, b) \leq (Z, c)$. We show $(X, a) \leq (Z, c)$.
We have $a \leq b \leq c$, so $a \leq c$. From $Z \leq Y \leq X$ we conclude $Z \leq X$:
Let $z \in Z$. There is $y \in Y$ with $z \leq y$, and $x \in X$ with $y \leq x$.

It remains to show: If $(X, a) \uparrow (Y, b)$ and $(Z, c) \leq (Y, b)$, then $(X, a) \uparrow (Z, c)$.
Suppose $X \uparrow Z$. Then $X \uparrow Y$: Let $x \in X, y \in Y$. $Y \leq Z$, therefore $\exists z \in Z.\ y \leq z$. It is $x \uparrow z$, hence $x \uparrow y$.
We get $a \uparrow b$ and $c \leq b$, therefore $a \uparrow c$. ∎

The elements of $\mathcal{A} \to \mathcal{B}$ correspond to the continuous functions from domain $|\mathcal{A}|$ to $|\mathcal{B}|$.

Proposition 3.21 *Let $r \in |\mathcal{A} \to \mathcal{B}|$. Then $|r| : |\mathcal{A}| \to |\mathcal{B}|$ given by*

$$|r|(d) = \{a \mid \exists X \subseteq d.\ (X, a) \in r\} \text{ for } d \in |\mathcal{A}|$$

is a continuous function from the domain $|\mathcal{A}|$ to $|\mathcal{B}|$.

Proof: We show $|r|(d) \in |\mathcal{B}|$.
$|r|(d)$ is consistent: Let $a, b \in |r|(d)$. There is $X \subseteq d$ with $(X, a) \in r$ and $Y \subseteq d$ with $(Y, b) \in r$. As $(X, a) \uparrow (Y, b)$ and $X \uparrow Y$, we conclude $a \uparrow b$.
$|r|(d)$ is downward closed: Let $b \in |r|(d)$ and $a \leq b$. There is $Y \subseteq d$ with $(Y, b) \in r$. It is $(Y, a) \leq (Y, b)$, so $(Y, a) \in r$ and $a \in |r|(d)$.
$|r|$ is monotonic, obviously.
$|r|$ is continuous: Let D be a directed subset of $|\mathcal{A}|$.

$$\begin{aligned}
\bigcup_{d \in D} |r|(d) &= \{a \mid \exists d \in D.\ \exists X \subseteq d.\ (X, a) \in r\} \\
&= \{a \mid \exists X \subseteq \bigcup D.\ (X, a) \in r\}, \text{ because the } X \text{ are finite} \\
&= |r|(\bigcup D)
\end{aligned}$$

∎

For cpos $(D, \subseteq)$ and $(E, \subseteq)$, let $([D \to E], \subseteq)$ be the cpo of continuous functions from D to E, ordered pointwise by $\subseteq$. We will also write $f : D \to E$ for $f \in [D \to E]$, and $f : D \to E \to F$ for $f \in [D \to [E \to F]]$. For $f : D \to E$ and $d \in D$ we will usually write $f\, d$ instead of $f(d)$, as in the syntax of the lambda calculus. Here also application is associated to the left, i.e. $f\, d\, e = (f\, d)\, e$. We will frequently write $r\, d$ instead of $|r|(d)$. It is clear from the context that the function between domains is meant.

Proposition 3.22 *Let $f : |\mathcal{A}| \to |\mathcal{B}|$ be monotonic and A be the set of primes of $\mathcal{A}$. Then the* prime set of f,

$$Pr(f) = \{(X, a) \mid X \in \overline{A} \wedge a \in f(X{\downarrow})\},$$

is an element of $|\mathcal{A} \to \mathcal{B}|$.

Proof:
$Pr(f)$ is consistent: Let $(X, a), (Y, b) \in Pr(f)$ and assume $X \uparrow Y$. Then $(X \cup Y){\downarrow} \in |\mathcal{A}|$. As $a \in f(X{\downarrow})$ and $b \in f(Y{\downarrow})$, we have $a, b \in f((X \cup Y){\downarrow})$, by monotonicity of f. Therefore $a \uparrow b$.

$Pr(f)$ is downward closed: Let (X, a) and (Y, b) be primes of $\mathcal{A} \to \mathcal{B}$, $(Y, b) \in Pr(f)$ and $(X, a) \leq (Y, b)$. From $Y \leq X$ follows $Y{\downarrow} \subseteq X{\downarrow}$. Then $b \in f(X{\downarrow})$, as $b \in f(Y{\downarrow})$ and f is monotonic. As $a \leq b$, also $a \in f(X{\downarrow})$ and $(X, a) \in Pr(f)$. ■

Theorem 3.23 *For all prime systems $\mathcal{A}, \mathcal{B}$ the map*

$$|.| : (|\mathcal{A} \to \mathcal{B}|, \subseteq) \to ([|\mathcal{A}| \to |\mathcal{B}|], \subseteq)$$

is an isomorphism of cpos. The map Pr is its inverse.
Therefore the complete primes and isolated elements of $[|\mathcal{A}| \to |\mathcal{B}|]$ are the images under $|.|$ of the corresponding elements of $|\mathcal{A} \to \mathcal{B}|$.

Proof: We show that for all $r \in |\mathcal{A} \to \mathcal{B}|$, $Pr(|r|) = r$:

$$\begin{aligned} (X, a) \in Pr(|r|) &\Leftrightarrow a \in |r|(X{\downarrow}) \\ &\Leftrightarrow \exists Y \subseteq X{\downarrow} \,.\, (Y, a) \in r \\ &\Leftrightarrow (X, a) \in r, \text{ because } (X, a) \leq (Y, a) \text{ and } r \text{ is downward closed} \end{aligned}$$

We show that for all $f \in [|\mathcal{A}| \to |\mathcal{B}|]$, $|Pr(f)| = f$:
Let A, B be the set of primes of $\mathcal{A}$ and $\mathcal{B}$, resp. Let $d \in |\mathcal{A}|$ and $a \in B$.

$$\begin{aligned} a \in |Pr(f)|(d) &\Leftrightarrow \exists X \subseteq d.\ X \in \overline{A} \wedge (X, a) \in Pr(f) \\ &\Leftrightarrow \exists X \subseteq d.\ X \in \overline{A} \wedge a \in f(X{\downarrow}) \\ &\Leftrightarrow a \in f(d) \end{aligned}$$

We prove the last equivalence:
$\Rightarrow$: $X{\downarrow} \subseteq d$ and f is monotonic.

$\Leftarrow$: Let $D = \{Y{\downarrow} \mid Y \text{ finite and } Y \subseteq d\}$. D is a directed set in $|\mathcal{A}|$. $\bigcup D = d$. Since f is continuous, there is some finite Y with $Y \subseteq d$ and $a \in f(Y{\downarrow})$. Let X be the set of maximal primes of Y. We get $X \subseteq d$, $X \in \overline{A}$, $Y{\downarrow} = X{\downarrow}$ and $a \in f(X{\downarrow})$.

So the map $|.|$ is one-to-one, Pr is its inverse. It remains to show that $|.|$ and Pr respect the partial order $\subseteq$:

$$\text{For all } r, s \in |\mathcal{A} \to \mathcal{B}| : r \subseteq s \;\Leftrightarrow\; \forall d \in |\mathcal{A}|.\ |r|(d) \subseteq |s|(d)$$

$\Rightarrow$ is obvious.
$\Leftarrow$: Let $(X, a) \in r$. Then $a \in |r|(X{\downarrow})$. As $a \in |s|(X{\downarrow})$, there is $Y \subseteq X{\downarrow}$ with $(Y, a) \in s$. As $Y \leq X$, also $(X, a) \in s$. ■

Theorem 3.24 $\to$ *is continuous on* $(\mathbf{PSys}, \trianglelefteq)$.

Proof:
1) $\to$ is monotonic in its first argument:
Let $\mathcal{A}_0 = (A_0, \uparrow_0, \leq_0) \trianglelefteq \mathcal{A}_0' = (A_0', \uparrow_0', \leq_0')$, $\mathcal{A}_1 = (A_1, \uparrow_1, \leq_1)$ be prime systems and $\mathcal{A}_0 \to \mathcal{A}_1 = (B, \uparrow, \leq)$, $\mathcal{A}_0' \to \mathcal{A}_1 = (B', \uparrow', \leq')$.
We have to prove: $\mathcal{A}_0 \to \mathcal{A}_1 \trianglelefteq \mathcal{A}_0' \to \mathcal{A}_1$.
First we show: $B = \overline{A_0} \times A_1 \subseteq \overline{A_0'} \times A_1 = B'$.
Let $X \in \overline{A_0}$. For all $a, b \in X$: $a \uparrow_0' b$ and $(a \leq_0' b \;\Rightarrow\; a = b)$. Therefore $X \in \overline{A_0'}$.
Now let $(X, a), (Y, b) \in B$.

$$\begin{aligned}
(X, a) \uparrow (Y, b) &\Leftrightarrow (X \uparrow_0 Y \;\Rightarrow\; a \uparrow_1 b) \\
&\Leftrightarrow (X \uparrow_0' Y \;\Rightarrow\; a \uparrow_1 b) \\
&\Leftrightarrow (X, a) \uparrow' (Y, b) \\
(X, a) \leq (Y, b) &\Leftrightarrow Y \leq_0 X \text{ and } a \leq_1 b \\
&\Leftrightarrow Y \leq_0' X \text{ and } a \leq_1 b \\
&\Leftrightarrow (X, a) \leq' (Y, b)
\end{aligned}$$

2) $\to$ is continuous on prime sets in its first argument:
Let $\mathcal{A}_0 \trianglelefteq \mathcal{A}_1 \trianglelefteq \ldots$ be an ω-chain of prime systems with $\mathcal{A}_i = (A_i, \uparrow_i, \leq_i)$, and $\mathcal{B}$ be a prime system.
Let (X, b) be a prime of $(\bigcup_i \mathcal{A}_i) \to \mathcal{B}$. Then $X \in \overline{\bigcup_i A_i}$. Since X is finite, $X \subseteq A_n$ for some n. For all $a, c \in X$, $a \uparrow_n c$ and $(a \leq_n c \;\Rightarrow\; a = c)$, because $\mathcal{A}_n \trianglelefteq \bigcup_i \mathcal{A}_i$. So $X \in \overline{A_n}$ and (X, b) is a prime of $\bigcup_i (\mathcal{A}_i \to \mathcal{B})$.

3) $\to$ is monotonic in its second argument:
Let $\mathcal{A}_0 = (A_0, \uparrow_0, \leq_0)$, $\mathcal{A}_1 = (A_1, \uparrow_1, \leq_1) \trianglelefteq \mathcal{A}_1' = (A_1', \uparrow_1', \leq_1')$ be prime systems and $\mathcal{A}_0 \to \mathcal{A}_1 = (B, \uparrow, \leq)$, $\mathcal{A}_0 \to \mathcal{A}_1' = (B', \uparrow', \leq')$. We have to show: $\mathcal{A}_0 \to \mathcal{A}_1 \trianglelefteq \mathcal{A}_0 \to \mathcal{A}_1'$.
$B = \overline{A_0} \times A_1 \subseteq \overline{A_0} \times A_1' = B'$.

Now let $(X, a), (Y, b) \in B$.

$$\begin{array}{rcl} (X,a) \uparrow (Y,b) & \Leftrightarrow & (X \uparrow_0 Y \ \Rightarrow \ a \uparrow_1 b) \\ & \Leftrightarrow & (X \uparrow_0 Y \ \Rightarrow \ a \uparrow_1' b) \\ & \Leftrightarrow & (X,a) \uparrow' (Y,b) \\ (X,a) \leq (Y,b) & \Leftrightarrow & Y \leq_0 X \text{ and } a \leq_1 b \\ & \Leftrightarrow & Y \leq_0 X \text{ and } a \leq_1' b \\ & \Leftrightarrow & (X,a) \leq' (Y,b) \end{array}$$

4) $\rightarrow$ is continuous on prime sets in its second argument:
Let $\mathcal{A}_0 \trianglelefteq \mathcal{A}_1 \trianglelefteq \dots$ be an ω-chain of prime systems with $\mathcal{A}_i = (A_i, \uparrow_i, \leq_i)$, and $\mathcal{B} = (B, \uparrow, \leq)$ be a prime system.
The set of primes of $\mathcal{B} \rightarrow (\bigcup_i \mathcal{A}_i)$ is $\overline{B} \times (\bigcup_i A_i) = \bigcup_i (\overline{B} \times A_i)$, the set of primes of $\bigcup_i (\mathcal{B} \rightarrow \mathcal{A}_i)$. ■

4 Denotational semantics

4.1 Semantics of types

We give a semantic interpretation of the type trees of T_∞ as prime systems. So we do not solve recursive domain equations directly, but define the semantics of a recursive type $\tau \in T_\mu^c$ by the semantics of its unfolding τ^*.

Definition 4.1 The sequence of maps $\mathcal{P}_n : T_\infty \rightarrow \mathbf{PSys}$, $n \geq 0$, is defined inductively by:

$$\begin{array}{lcl} \mathcal{P}_0(\sigma) & = & \underline{\perp} \text{ for all } \sigma \in T_\infty, \\ \mathcal{P}_{n+1}(\mathsf{void}) & = & \underline{\perp}, \\ \mathcal{P}_{n+1}(\sigma \ @ \ \tau) & = & \mathcal{P}_n(\sigma) \ @ \ \mathcal{P}_n(\tau) \text{ for } @ \in \{+, \times, \rightarrow\} \text{ and } \sigma, \tau \in T_\infty. \end{array}$$

Define $P_i(\sigma)$ as the prime set of $\mathcal{P}_i(\sigma)$.

Proposition 4.2 *For all $\sigma \in T_\infty$, $n \geq 0$: $\mathcal{P}_n(\sigma) \trianglelefteq \mathcal{P}_{n+1}(\sigma)$.*
(This proposition depends only on the monotonicity of the operations $+, \times, \rightarrow$ on prime systems.)

Proof: by induction on n. Trivial for $n = 0$.
Now assume that for some $n \geq 0$: $\forall \sigma \in T_\infty. \ \mathcal{P}_n(\sigma) \trianglelefteq \mathcal{P}_{n+1}(\sigma)$.
We prove $\mathcal{P}_{n+1}(\sigma) \trianglelefteq \mathcal{P}_{n+2}(\sigma)$ for all cases of σ:
$\mathcal{P}_{n+1}(\mathsf{void}) = \underline{\perp} \trianglelefteq \mathcal{P}_{n+2}(\mathsf{void})$.
$\mathcal{P}_{n+1}(\sigma \ @ \ \tau) = \mathcal{P}_n(\sigma) \ @ \ \mathcal{P}_n(\tau) \trianglelefteq \mathcal{P}_{n+1}(\sigma) \ @ \ \mathcal{P}_{n+1}(\tau) = \mathcal{P}_{n+2}(\sigma \ @ \ \tau)$ for $@ \in \{+, \times, \rightarrow\}$.
■

This permits to give the semantics of type trees:

Definition 4.3 Define the map $\mathcal{P} : T_\infty \to \mathbf{PSys}$ by $\mathcal{P}(\sigma) = \bigcup_i \mathcal{P}_i(\sigma)$.
$P(\sigma)$ is the set of primes of $\mathcal{P}(\sigma)$.

Proposition 4.4

$$\begin{array}{rcl} \mathcal{P}(\mathsf{void}) & = & \perp\!\!\!\perp \\ \mathcal{P}(\sigma \mathbin{@} \tau) & = & \mathcal{P}(\sigma) \mathbin{@} \mathcal{P}(\tau) \textit{ for } @ \in \{+, \times, \to\} \textit{ and } \sigma, \tau \in T_\infty \end{array}$$

(This proposition depends on the continuity of the operations $+, \times, \to$ on prime systems.)

Proof: Clearly $\mathcal{P}(\mathsf{void}) = \perp\!\!\!\perp$.

$$\begin{array}{rcl} \mathcal{P}(\sigma \mathbin{@} \tau) & = & \bigcup\limits_i (\mathcal{P}_{i+1}(\sigma \mathbin{@} \tau)) \\ & = & \bigcup\limits_i (\mathcal{P}_i(\sigma) \mathbin{@} \mathcal{P}_i(\tau)) \\ & = & (\bigcup\limits_i \mathcal{P}_i(\sigma)) \mathbin{@} (\bigcup\limits_i \mathcal{P}_i(\tau)) \\ & = & \mathcal{P}(\sigma) \mathbin{@} \mathcal{P}(\tau). \end{array}$$

■

Definition 4.5 The domain for a type tree $\sigma \in T_\infty$ is $D_\sigma = |\mathcal{P}(\sigma)|$,
the domain for a type $\sigma \in T^c_\mu$ is $D_\sigma = |\mathcal{P}(\sigma^*)|$.
For $d \in D_\sigma$, $\sigma \in T_\infty$, we define the n-th *projection* of d as $d|_n = d \cap P_n(\sigma)$.

Note that the primes of $\mathcal{P}(\sigma)$ are expressions of finite size and therefore structural induction may be applied to them. More precisely: For a prime $a \in P(\sigma)$ let $\mathit{level}(a)$ be the least i such that $a \in P_i(\sigma)$.
If $(0, a) \in P(\sigma + \tau)$, then $a \in P(\sigma)$ and $\mathit{level}(a) < \mathit{level}(0, a)$.
If $(1, a) \in P(\sigma + \tau)$, then $a \in P(\tau)$ and $\mathit{level}(a) < \mathit{level}(1, a)$.
The same holds for $\sigma \times \tau$ instead of $\sigma + \tau$.
If $(X, a) \in P(\sigma \to \tau)$, then for all $x \in X$: $x \in P(\sigma)$ and $\mathit{level}(x) < \mathit{level}(X, a)$, and $a \in P(\tau)$ and $\mathit{level}(a) < \mathit{level}(X, a)$.
Therefore definitions and proofs for primes may be given by induction on their parts with smaller level.

4.2 Semantics of terms

We will define the semantics function $\mathcal{S}$ for terms. As usual we need environments: Let $V = \bigcup_{\tau \in T^c_\mu} V^\tau$ be the set of all term variables of any type. An *environment* is a function $\varepsilon : V \to \bigcup_{\sigma \in T^c_\mu} D_\sigma$ such that $\varepsilon(x^\sigma) \in D_\sigma$ for all $x^\sigma \in V$. *Env* is the set of all environments. It is a cpo under the pointwise order $\subseteq$. Its least element is denoted by

$\perp$, $\perp(x) = \perp$ for all x. For any environment ε, $\varepsilon[x \mapsto d]$ is the environment ε' with $\varepsilon'(x) = d$ and $\varepsilon'(y) = \varepsilon(y)$ for $y \neq x$.

For every constant c we will give a continuous function on domains. This function is then transformed by Pr into an element of the prime system corresponding to the type of c. We need versions of Pr for functions with 2 and 3 arguments:

Let $f : |\mathcal{A}| \to (|\mathcal{B}| \to |\mathcal{C}|)$ be continuous for prime systems $\mathcal{A}, \mathcal{B}, \mathcal{C}$. Define $Pr_2(f) \in |\mathcal{A} \to (\mathcal{B} \to \mathcal{C})|$ by $Pr_2(f) = Pr(Pr \circ f)$, where $(f \circ g)x = f(g(x))$. Note that $Pr \circ f$ is continuous since Pr is continuous as an order isomorphism. It is $(Pr_2(f))\, a\, b = |(|Pr_2(f)|\, a)|\, b = f\, a\, b$.

Let $f : |\mathcal{A}| \to (|\mathcal{B}| \to (|\mathcal{C}| \to |\mathcal{D}|))$ be continuous for prime systems $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$. Define $Pr_3(f) \in |\mathcal{A} \to (\mathcal{B} \to (\mathcal{C} \to \mathcal{D}))|$ by $Pr_3(f) = Pr(Pr_2 \circ f)$. Note that $Pr_2 \circ f$ is continuous as Pr_2 is continuous. It is $(Pr_3(f))\, a\, b\, c = |(|(|Pr_3(f)|\, a)|\, b)|\, c = f\, a\, b\, c$.

Definition 4.6 We define the semantics function $\mathcal{S} : \mathcal{T} \to (Env \to \bigcup_{\sigma \in T^c_\mu} D_\sigma)$ by structural induction on the term argument. We write $\mathcal{S}[\![M]\!]$ and $\mathcal{S}[\![M]\!]\varepsilon$, for $M \in \mathcal{T}$, $\varepsilon \in Env$. It is $\mathcal{S}[\![M]\!] \in [Env \to D_\sigma]$ for $M : \sigma$, see the following proposition.

$$
\begin{array}{lcll}
\mathcal{S}[\![\mathsf{0}_{\sigma,\tau}]\!]\varepsilon & = & Pr(0), & \text{with } \; 0 : D_\sigma \to D_{\sigma+\tau} \\
& & & \phantom{\text{with }}\; 0d = \{0\} \cup (\{0\} \times d) \\
\mathcal{S}[\![\mathsf{1}_{\sigma,\tau}]\!]\varepsilon & = & Pr(1), & \text{with } \; 1 : D_\tau \to D_{\sigma+\tau} \\
& & & \phantom{\text{with }}\; 1d = \{1\} \cup (\{1\} \times d) \\
\mathcal{S}[\![\mathsf{case}_{\sigma,\tau\varrho}]\!]\varepsilon & = & Pr_3(case), & \text{with } \; case : D_{\sigma+\tau} \to D_{\sigma\to\varrho} \to D_{\tau\to\varrho} \to D_\varrho \\
& & & \phantom{\text{with }}\; case\, d\, f\, g = \begin{cases} \perp, & \text{if } d = \perp \\ |f|e, & \text{if } d = 0e \\ |g|e, & \text{if } d = 1e \end{cases} \\
\mathcal{S}[\![\mathsf{pcase}_{\sigma,\tau,\varrho}]\!]\varepsilon & = & Pr_3(pcase), & \text{with } \; pcase : D_{\sigma+\tau} \to D_\varrho \to D_\varrho \to D_\varrho \\
& & & \phantom{\text{with }}\; pcase\, a\, b\, c = \begin{cases} b \cap c, & \text{if } a = \perp \\ b, & \text{if } a = 0a' \\ c, & \text{if } a = 1a' \end{cases} \\
\mathcal{S}[\![\mathsf{pair}_{\sigma,\tau}]\!]\varepsilon & = & Pr_2(pair), & \text{with } \; pair : D_\sigma \to D_\tau \to D_{\sigma\times\tau} \\
& & & \phantom{\text{with }}\; pair\, d\, e = (\{0\} \times d) \cup (\{1\} \times e) \\
\mathcal{S}[\![\mathsf{fst}_{\sigma,\tau}]\!]\varepsilon & = & Pr(fst), & \text{with } \; fst : D_{\sigma\times\tau} \to D_\sigma \\
& & & \phantom{\text{with }}\; fst\,(pair\, d\, e) = d \\
\mathcal{S}[\![\mathsf{snd}_{\sigma,\tau}]\!]\varepsilon & = & Pr(snd), & \text{with } \; snd : D_{\sigma\times\tau} \to D_\tau \\
& & & \phantom{\text{with }}\; snd\,(pair\, d\, e) = e
\end{array}
$$

$$\begin{array}{lcl}
\mathcal{S}[\![\Omega_\sigma]\!]\varepsilon & = & \bot \\
\mathcal{S}[\![x]\!]\varepsilon & = & \varepsilon(x) \\
\mathcal{S}[\![\lambda x^\sigma.M]\!]\varepsilon & = & Pr(d \in D_\sigma \mapsto \mathcal{S}[\![M]\!](\varepsilon[x \mapsto d])), \\
& & \text{where } (d \in D \mapsto \exp) \text{ denotes the function} \\
& & \text{that maps each } d \in D \text{ to } \exp \\
\mathcal{S}[\![MN]\!]\varepsilon & = & |\mathcal{S}[\![M]\!]\varepsilon|\,(\mathcal{S}[\![N]\!]\varepsilon)
\end{array}$$

Proposition 4.7 *For all terms $M : \psi$, $\mathcal{S}[\![M]\!] \in [Env \to D_\psi]$.*

Proof: by structural induction on M.
• Let M be a constant:
It is easy to check that the given function on domains is continuous and that the semantics of M is in the appropriate domain. We show this only for $M = \mathsf{pcase}_{\sigma,\tau\varrho}$:

pcase is monotonic (and continuous) in its first argument, since $b \cap c \subseteq b$ and $b \cap c \subseteq c$. *pcase* is continuous in its second (third) argument: This is clear for the cases $a = 0a'$ and $a = 1a'$. In the case $a = \bot$ it follows from the continuity of $\cap$. Now $pcase : D_{\sigma+\tau} \to D_\varrho \to D_\varrho \to D_\varrho$ is continuous, therefore

$$\begin{array}{rcl}
\mathcal{S}[\![\mathsf{pcase}_{\sigma,\tau,\varrho}]\!]\varepsilon = Pr_3(pcase) & \in & |\mathcal{P}((\sigma+\tau)^*) \to \mathcal{P}(\varrho^*) \to \mathcal{P}(\varrho^*) \to \mathcal{P}(\varrho^*)| \\
& = & D_{(\sigma+\tau)\to\varrho\to\varrho\to\varrho}.
\end{array}$$

If $\mathsf{pcase}_{\sigma,\tau,\varrho} : \psi$, then $\psi \approx (\sigma+\tau) \to \varrho \to \varrho$, and $\mathcal{S}[\![\mathsf{pcase}_{\sigma,\tau,\varrho}]\!] \in [Env \to D_\psi]$.

• Let $M = x^\sigma$:
$\mathcal{S}[\![x^\sigma]\!] = (\varepsilon \mapsto \varepsilon(x^\sigma)) : Env \to D_\sigma$ is continuous.

• Let $M = \lambda x^\sigma.N : \sigma \to \tau$:
Then $N : \tau$, and $\mathcal{S}[\![N]\!] \in [Env \to D_\tau]$ follows by induction hypothesis. Let $\varepsilon \in Env$ and $f = (d \in D_\sigma \mapsto \mathcal{S}[\![N]\!](\varepsilon[x \mapsto d]))$. f is continuous, because $\varepsilon[x \mapsto .]$ and $\mathcal{S}[\![N]\!]$ are continuous. So $f \in [D_\sigma \to D_\tau]$, and

$$\mathcal{S}[\![\lambda x.N]\!]\varepsilon = Pr(f) \in |\mathcal{P}(\sigma^*) \to \mathcal{P}(\tau^*)| = D_{\sigma\to\tau}.$$

It remains to show that $\mathcal{S}[\![\lambda x.N]\!]$ is continuous.
It is monotonic: Let $\varepsilon, \varepsilon' \in Env$ and $\varepsilon \subseteq \varepsilon'$. Then

$$\begin{array}{rcl}
\mathcal{S}[\![\lambda x.N]\!]\varepsilon & = & Pr(d \in D_\sigma \mapsto \mathcal{S}[\![N]\!](\varepsilon[x \mapsto d])) \\
& \subseteq & Pr(d \in D_\sigma \mapsto \mathcal{S}[\![N]\!](\varepsilon'[x \mapsto d])),\ \text{as } \mathcal{S}[\![N]\!] \text{ and } Pr \text{ are monotonic} \\
& = & \mathcal{S}[\![\lambda x.N]\!]\varepsilon'
\end{array}$$

Let E be a directed set of environments.

$$\mathcal{S}[\![\lambda x.N]\!](\bigcup_{\varepsilon\in E} \varepsilon) \quad = \quad Pr(d \in D_\sigma \mapsto \mathcal{S}[\![N]\!]((\bigcup_{\varepsilon\in E} \varepsilon)[x \mapsto d]))$$

$$
\begin{aligned}
&= Pr(d \in D_\sigma \mapsto \mathcal{S}[\![N]\!](\bigcup_{\varepsilon \in E}(\varepsilon[x \mapsto d]))) \\
&= Pr(d \in D_\sigma \mapsto \bigcup_{\varepsilon \in E} \mathcal{S}[\![N]\!](\varepsilon[x \mapsto d])), \text{ as } \mathcal{S}[\![N]\!] \text{ is continuous} \\
&= \bigcup_{\varepsilon \in E} Pr(d \in D_\sigma \mapsto \mathcal{S}[\![N]\!](\varepsilon[x \mapsto d])), \text{ as } Pr \text{ is continuous} \\
&= \bigcup_{\varepsilon \in E} \mathcal{S}[\![\lambda x.N]\!]\varepsilon
\end{aligned}
$$

• Let $M = NP$, $N : \sigma \to \tau$, $P : \sigma$:
By induction hypothesis we have $\mathcal{S}[\![N]\!] \in [Env \to D_{\sigma\to\tau}]$ and $\mathcal{S}[\![P]\!] \in [Env \to D_\sigma]$. Let $\varepsilon \in Env$. Then $|\mathcal{S}[\![N]\!]\varepsilon| \in D_\sigma \to D_\tau$ and $\mathcal{S}[\![P]\!]\varepsilon \in D_\sigma$, hence $\mathcal{S}[\![NP]\!]\varepsilon \in D_\tau$. $\mathcal{S}[\![NP]\!]$ is continuous because $\mathcal{S}[\![N]\!]$, $\mathcal{S}[\![P]\!]$ and $|.|$ are continuous. So we get $\mathcal{S}[\![NP]\!] \in [Env \to D_\tau]$. ■

4.3 Soundness of the semantics

We show that reduction does not change the semantics of terms. First we prove the Substitution Lemma.

Lemma 4.8 *(Substitution Lemma)*

$$\mathcal{S}[\![M[x{:=}N]]\!]\varepsilon = \mathcal{S}[\![M]\!](\varepsilon[x \mapsto \mathcal{S}[\![N]\!]\varepsilon]),$$

for all appropriately typed terms M, N, and all $\varepsilon \in Env$.

Proof: by induction on the structure of M, see Lemma 2.12 of [Gun92]. ■

Theorem 4.9 (Soundness) *If $M, N \in \mathcal{T}$ and $M \to^* N$, then $\mathcal{S}[\![M]\!] = \mathcal{S}[\![N]\!]$.*

Proof: It is clear that the semantics of a term is not changed by replacing a subterm by a term with the same semantics. We have the properties:

$$
\begin{aligned}
\mathcal{S}[\![M]\!] = \mathcal{S}[\![M']\!] &\quad\Rightarrow\quad \mathcal{S}[\![MN]\!] = \mathcal{S}[\![M'N]\!] \\
\mathcal{S}[\![N]\!] = \mathcal{S}[\![N']\!] &\quad\Rightarrow\quad \mathcal{S}[\![MN]\!] = \mathcal{S}[\![MN']\!] \\
\mathcal{S}[\![M]\!] = \mathcal{S}[\![M']\!] &\quad\Rightarrow\quad \mathcal{S}[\![\lambda x.M]\!] = \mathcal{S}[\![\lambda x.M']\!]
\end{aligned}
$$

So if $\mathcal{S}[\![M]\!] = \mathcal{S}[\![M']\!]$, then $\mathcal{S}[\![C[M]]\!] = \mathcal{S}[\![C[M']]\!]$ for any context $C[\,]$.
It can be easily checked that each reduction rule does not change the semantics. For the β-rule this follows from the Substitution Lemma. ■

5 Approximation Theorem

For every term M we will define a set $\mathcal{A}(M)$ of normal forms that approximate the reducts of M. $\mathcal{A}(M)$ can be seen as the syntactic value of M or the Böhm tree of M. We will prove the Approximation Theorem: $\mathcal{S}[\![M]\!]\varepsilon = \bigcup_{A\in\mathcal{A}(M)} \mathcal{S}[\![A]\!]\varepsilon$. Thus the semantics of M is entirely determined by the normal form approximations of M.

There are three methods in the literature to prove the Approximation Theorem: [Ber79, Th. 3.1.12] proves it for PCF and [Wad78] for the untyped lambda calculus, both with the aid of a labelled λ-calculus. [MP87] proves it for the untyped λ-calculus by two other methods: by an intermediate semantics and by inclusive predicates. We will give an inclusive predicate proof, modified for the recursively typed λ-calculus and prime systems.

First we use the constant Ω to define the usual Ω-prefix partial order on terms:

Definition 5.1 For every $\sigma \in T_\infty$, $\prec$ is the least relation on $\mathcal{T}_\sigma$ satisfying:
$\Omega \prec M$ for every $M \in \mathcal{T}_\sigma$,
$x \prec x$ for every variable or constant x,
$M \prec M' \;\Rightarrow\; \lambda x.M \prec \lambda x.M'$,
$M \prec M' \wedge N \prec N' \;\Rightarrow\; MN \prec M'N'$.
If $M, N \in \mathcal{T}_\sigma$ have an upper bound under $\prec$, then $M \sqcup N$ is defined as their least upper bound.

It is clearly: $M \prec N \;\Rightarrow\; \mathcal{S}[\![M]\!] \subseteq \mathcal{S}[\![N]\!]$.

Definition 5.2 Let $\sigma \in T_\infty$. $\mathcal{N}_\sigma$ is the set of normal form terms of $\mathcal{T}_\sigma$. Normal forms are denoted by $A, B, \ldots$.
Let $A \in \mathcal{N}_\sigma$, $M \in \mathcal{T}_\sigma$.
A is a *direct approximation* of M, $A \lhd M$, iff $\forall N.\ (M \to^* N \;\Rightarrow\; A \prec N)$.
A is an *approximation* of M, $A \trianglelefteq M$, iff $\exists N.\ M \to^* N$ and $A \lhd N$.
$\mathcal{A}(M)$ denotes the set of approximations of M.
We abbreviate $\overline{\mathcal{S}}[\![M]\!]\varepsilon = \bigcup_{A \trianglelefteq M} \mathcal{S}[\![A]\!]\varepsilon$.

A direct approximation of M conveys a fixed syntactic information about M: It is in normal form and is part of all reducts of M. If $A \lhd M$ and $M \to^* N$, then $A \lhd N$. We want to show that $\mathcal{A}(M)$ is an ideal. Therefore we need the following lemma, which relies on the fact that all applicative terms have a normal form.

Lemma 5.3 *If $A \lhd M$ and $B \lhd M$, then $A \sqcup B$ exists and is a normal form, and $A \sqcup B \lhd M$.*

Proof: $A \sqcup B$ exists because $A \prec M$ and $B \prec M$. Now assume that $A \sqcup B$ is not a normal form. Then there is an occurrence u in $A \sqcup B$ such that $(A \sqcup B)/u$ is a redex.

First assume that it is a β-redex: $(A \sqcup B)/u$ is of the form $(\lambda x.N)P$. Then either A/u is of the form $(\lambda x.N')P'$, or B/u is of this form. This contradicts the assumption that A and B are normal forms.

Now assume that $(A \sqcup B)/u$ is a redex of a constant, corresponding to one of the rules ($\mathsf{case0}$) – ($\mathsf{pcase}\!\rightarrow$). Let $L = M/u$. Let u_i, $1 \leq i \leq n$, be a sequence of all the outermost occurrences of λ-abstractions in L. Let x_i, $1 \leq i \leq n$, be a sequence of distinct variables that do not occur in L. (The type of x_i should be that of L/u_i.) Let $K = L[u_1 \leftarrow x_1, \ldots, u_n \leftarrow x_n]$. K is an applicative term, i.e. it does not contain any λ-abstraction. As $\rightarrow$ is strongly normalizing (noetherian) on applicative terms, there is a normal form K' of K, $K \rightarrow^* K'$. It is $L = K[x_1{:=}(L/u_1), \ldots, x_n{:=}(L/u_n)]$, the result of the replacement of the x_i by L/u_i. Let $L' = K'[x_1{:=}(L/u_1), \ldots, x_n{:=}(L/u_n)]$. Then $L \rightarrow^* L'$. As K' is a normal form and the L/u_i are λ-abstractions, L' is not a redex of a constant.
It is $M \rightarrow^* M[u \leftarrow L']$, as $L \rightarrow^* L'$. As $A \lhd M$ and $B \lhd M$, we have $A \sqcup B \prec M[u \leftarrow L']$. Therefore $(A \sqcup B)/u \prec L'$. This contradicts the fact that L' is not a redex of a constant.

So in every case we deduced a contradiction from the assumption that $A \sqcup B$ is not a normal form. Clearly $A \sqcup B \lhd M$. ■

Theorem 5.4 *$\mathcal{A}(M)$ is an ideal under $\prec$, i.e. it is non-empty, downward closed and directed.*

Proof: We have $\Omega \in \mathcal{A}(M)$.
$\mathcal{A}(M)$ is downward closed: If $A \unlhd M$ and $B \prec A$, then $B \unlhd M$.
$\mathcal{A}(M)$ is directed: Let $A \unlhd M$ and $A' \unlhd M$. There is N with $M \rightarrow^* N \ \wedge\ A \lhd N$, and N' with $M \rightarrow^* N' \ \wedge\ A' \lhd N'$. By confluence there is a term P with $N \rightarrow^* P$ and $N' \rightarrow^* P$. Then $A \lhd P$ and $A' \lhd P$. By the preceding lemma, $A \sqcup A'$ is a normal form and $A \sqcup A' \lhd P$. Hence $A \sqcup A' \unlhd M$. ■

With this proposition $\mathcal{A}(M)$ is an element of the ideal completion of $\mathcal{N}_\sigma$ (under $\prec$); it can be seen as a Böhm tree of M.

Let us first discuss our definition of approximation and compare it with different approaches in the literature:

1) The treatment of PCF in [Ber79] is different: The approximations are obtained by reducing only β- and Y-redexes. The constants are treated like variables; redexes of rules for constants are not reduced. They are only interpreted semantically in the Böhm tree. This approach is only possible because the reduction of constant redexes can be postponed after the reduction of β- and Y-redexes. In our case constants operate on higher order types as well, therefore the reduction of constant redexes is intertwined with β-reduction.

2) $\mathcal{A}(M)$ is not minimal: In many cases there is a proper subset of $\mathcal{A}(M)$ with the same semantics; e.g. for $M = \lambda x.\Omega$ or $M = \Omega N$ the approximation Ω is sufficient. $\mathcal{A}(M)$

was defined to give "all possible" normal form information about M. The questions arise: In which sense is $\mathcal{A}(M)$ maximal? [My conjecture is: For every directed set S of minimum normal forms of M (def. below), if S has the same semantics as $\mathcal{A}(M)$, then $S \subseteq \mathcal{A}(M)$.] Is a smaller set of approximations definable with the same semantics, that gives a substantially stronger Approximation Theorem?

In the presence of parallel operations there is in general no least approximation with the same semantics: Consider

$$M = \lambda x.\mathsf{pcase}\, x\, (\mathsf{case}\, x\, \Omega\, (\lambda y.1))\, 1 : \mathsf{bool} \to \mathsf{bool}.$$

$\mathcal{S}[\![M]\!]\bot$ is the function that maps $1 \mapsto 1$, $0 \mapsto \bot$. Both $\lambda x.\mathsf{pcase}\,\Omega\,(\mathsf{case}\,x\,\Omega\,(\lambda y.1))\,1$ and $\lambda x.\mathsf{pcase}\, x\, \Omega\, 1$ are minimal approximations of M with the same semantics as M.

3) In the presence of pcase it is not possible to define the approximations by an analogue of head normal forms. We will make this statement precise after the proof of the Approximation Theorem. We will also give analogues of head normal forms for the sequential calculus without pcase.

We now prove two useful lemmas about approximations.

Lemma 5.5 *If $M \downarrow N$, then $\mathcal{A}(M) = \mathcal{A}(N)$ and $\overline{\mathcal{S}}[\![M]\!]\varepsilon = \overline{\mathcal{S}}[\![N]\!]\varepsilon$.*

Proof: Let $M \to^* P \leftarrow^* N$. Assume $A \trianglelefteq M$. Then there is M' with $M \to^* M'$ and $A \lhd M'$. By confluence there is L with $M' \to^* L \leftarrow^* P$. Then $A \lhd L$ and $A \trianglelefteq N$. This shows $\mathcal{A}(M) \subseteq \mathcal{A}(N)$. Symmetrically $\mathcal{A}(M) \supseteq \mathcal{A}(N)$. ■

Lemma 5.6 *Let $cM_1 \ldots M_n$ be a term where c is a constant and there are no reducts $M_i \to^* M_i'$, $1 \leq i \leq m \leq n$, with $cM_1' \ldots M_m'$ a redex. Then*

$$\overline{\mathcal{S}}[\![cM_1 \ldots M_n]\!]\varepsilon = (\mathcal{S}[\![c]\!]\bot)\,(\overline{\mathcal{S}}[\![M_1]\!]\varepsilon) \ldots (\overline{\mathcal{S}}[\![M_n]\!]\varepsilon).$$

Proof:

$$\begin{aligned}
\overline{\mathcal{S}}[\![cM_1 \ldots M_n]\!]\varepsilon &= \bigsqcup\{\mathcal{S}[\![A]\!]\varepsilon \mid A \trianglelefteq cM_1 \ldots M_n\} \\
&= \bigsqcup\{\mathcal{S}[\![cA_1 \ldots A_n]\!]\varepsilon \mid A_1 \trianglelefteq M_1 \ \wedge\ \ldots\ \wedge\ A_n \trianglelefteq M_n\} \\
&= (\mathcal{S}[\![c]\!]\bot)\,(\overline{\mathcal{S}}[\![M_1]\!]\varepsilon) \ldots (\overline{\mathcal{S}}[\![M_n]\!]\varepsilon)
\end{aligned}$$

We have used the fact that $A \trianglelefteq cM_1 \ldots M_n$ iff $A = cA_1 \ldots A_n$ with some $A_i \trianglelefteq M_i$; as no $cM_1 \ldots M_m$, $m \leq n$, can be reduced to a redex. ■

Theorem 5.7 (Approximation Theorem) *For all terms M and environments ε:*

$$\mathcal{S}[\![M]\!]\varepsilon = \overline{\mathcal{S}}[\![M]\!]\varepsilon.$$

$\overline{\mathcal{S}}[\![M]\!]\varepsilon \subseteq \mathcal{S}[\![M]\!]\varepsilon$ follows from $\mathcal{S}[\![A]\!]\varepsilon \subseteq \mathcal{S}[\![M]\!]\varepsilon$ for $A \lhd M$. This is a consequence of soundness and of monotonicty of $\mathcal{S}$ w.r.t. $\prec$. We want to prove the remaining inclusion $\mathcal{S}[\![M]\!]\varepsilon \subseteq \overline{\mathcal{S}}[\![M]\!]\varepsilon$ by structural induction on M. Therefore we use inclusive predicates (logical relations), also used in [MP87] to prove the analogous theorem (limiting completeness) for the untyped λ-calculus. We define the inclusive predicates on the sets of primes $P(\sigma)$ of the type interpretations $\mathcal{P}(\sigma)$:

Definition 5.8 For every $\sigma \in T_\infty$ and $\varepsilon \in Env$ we define a relation $<^\sigma_\varepsilon \subseteq P(\sigma) \times \mathcal{T}_\sigma$. $a <^\sigma_\varepsilon M$ is defined by structural induction on a, i.e. in terms of propositions $a' <^\tau_\varepsilon M'$, where a' is a part of a with smaller level.
There are the following cases for σ and the primes:

$$\begin{array}{lll}
\sigma = \tau + \varrho : & 0 <^{\tau+\varrho}_\varepsilon M & \Leftrightarrow\ 0 \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \\
& (0,a) <^{\tau+\varrho}_\varepsilon M & \Leftrightarrow\ (0,a) \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \text{ and } a <^\tau_\varepsilon \mathit{Out0}(M) \\
& 1 <^{\tau+\varrho}_\varepsilon M & \Leftrightarrow\ 1 \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \\
& (1,a) <^{\tau+\varrho}_\varepsilon M & \Leftrightarrow\ (1,a) \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \text{ and } a <^\varrho_\varepsilon \mathit{Out1}(M) \\
& \multicolumn{2}{l}{\text{where } \mathit{Out0}(M) \text{ abbreviates the term } \mathsf{case}\, M\, (\lambda y.y)\, \Omega,} \\
& \multicolumn{2}{l}{\text{and } \mathit{Out1}(M) \text{ the term } \mathsf{case}\, M\, \Omega\, (\lambda y.y).} \\
\sigma = \tau \times \varrho : & (0,a) <^{\tau\times\varrho}_\varepsilon M & \Leftrightarrow\ (0,a) \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \text{ and } a <^\tau_\varepsilon \mathsf{fst}\, M \\
& (1,a) <^{\tau\times\varrho}_\varepsilon M & \Leftrightarrow\ (1,a) \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \text{ and } a <^\varrho_\varepsilon \mathsf{snd}\, M \\
\sigma = \tau \to \varrho : & (X,a) <^{\tau\to\varrho}_\varepsilon M & \Leftrightarrow\ (X,a) \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \text{ and} \\
& & \quad\ \forall N \in \mathcal{T}_\tau.\ (X <^\tau_\varepsilon N\ \Rightarrow\ a <^\varrho_\varepsilon MN)
\end{array}$$

For every set X of primes $X <^\tau_\varepsilon N$ means: $\forall b \in X.\ b <^\tau_\varepsilon N$.

Intuitively $a <^\sigma_\varepsilon M$ means that $a \in \overline{\mathcal{S}}[\![M]\!]\varepsilon$ and that the relation is maintained in all contexts formed by *Out0*, *Out1*, fst, snd and application on related arguments.

We have to prove a few lemmas for the Approximation Theorem.

Lemma 5.9 *If $a \leq b$ and $b <^\sigma_\varepsilon M$, then also $a <^\sigma_\varepsilon M$.*

Proof: by structural induction on b. In every case we have $a \in \overline{\mathcal{S}}[\![M]\!]\varepsilon$.

- $\sigma = \tau + \varrho$:

The case $a = 0$, $b = (0, b')$ is clear.
Now let $a = (0, a')$, $b = (0, b')$. Then $a' \leq b'$ and $b' <^\tau_\varepsilon \mathit{Out0}(M)$. By induction hypothesis follows $a' <^\tau_\varepsilon \mathit{Out0}(M)$.
The cases $a = 1, b = (1, b')$ and $a = (1, a'), b = (1, b')$ are analogous.

- $\sigma = \tau \times \varrho$ is like $\sigma = \tau + \varrho$

• $\sigma = \tau \to \varrho$:
Let $a = (X, a')$, $b = (Y, b')$. It is $Y \leq X$ and $a' \leq b'$.
For all $N \in \mathcal{T}_\tau$ the following implications hold:

$$\begin{aligned} X <^\tau_\varepsilon N &\Rightarrow Y <^\tau_\varepsilon N, \text{ by induction hypothesis} \\ &\Rightarrow b' <^\varrho_\varepsilon MN, \text{ as } (Y, b') <^\sigma_\varepsilon M \\ &\Rightarrow a' <^\varrho_\varepsilon MN, \text{ by induction hypothesis} \end{aligned}$$

Therefore $a = (X, a') <^\sigma_\varepsilon M$. ■

Lemma 5.10 *If $a <^\sigma_\varepsilon M$ and $M \downarrow N$, then also $a <^\sigma_\varepsilon N$.*

Proof: by structural induction on a.
We have $\overline{\mathcal{S}}[\![M]\!]\varepsilon = \overline{\mathcal{S}}[\![N]\!]\varepsilon$ by Lemma 5.5, therefore $a \in \overline{\mathcal{S}}[\![N]\!]\varepsilon$.

• $\sigma = \tau + \varrho$:
Let $a = (0, a')$. Then $a' <^\tau_\varepsilon \mathit{Out0}(M)$. By induction hypothesis follows $a' <^\tau_\varepsilon \mathit{Out0}(N)$, so $a <^\sigma_\varepsilon N$.
$a = (1, a')$ is analogous.

• $\sigma = \tau \times \varrho$ is like $\sigma = \tau + \varrho$.

• $\sigma = \tau \to \varrho$:
Let $a = (X, a')$. For all $P \in \mathcal{T}_\tau$:

$$\begin{aligned} X <^\tau_\varepsilon P &\Rightarrow a' <^\varrho_\varepsilon MP, \text{ as } a <^\sigma_\varepsilon M \\ &\Rightarrow a' <^\varrho_\varepsilon NP, \text{ by induction hyp., as } MP \downarrow NP \end{aligned}$$

Therefore $a <^\sigma_\varepsilon N$. ■

We also need the new notion of passive term:

Definition 5.11 A term M is a *redex part* iff $M = \lambda x.N$ for some x and N, or there is some typed left-hand side L of a rule ($\mathsf{case0}$)…($\mathsf{pcase}{\to}$) and a subterm L' of L such that $L' \neq L$, L' is no variable and M is obtained from L' by replacing variables by terms of the same type.

This means: M is a redex part iff M is of one of the following forms:

$\lambda x.N,\ \mathsf{0},\ \mathsf{0}N,\ \mathsf{1},\ \mathsf{1}N,$
$\mathsf{pair},\ \mathsf{pair}\, N_1,\ \mathsf{pair}\, N_1\, N_2,\ \mathsf{fst},\ \mathsf{snd},$
$\mathsf{case},\ \mathsf{case}\,(\mathsf{0}N),\ \mathsf{case}\,(\mathsf{0}N_1)\, N_2,\ \mathsf{case}\,(\mathsf{1}N),\ \mathsf{case}\,(\mathsf{1}N_1)\, N_2,$
$\mathsf{pcase},\ \mathsf{pcase}\, N_1,\ \mathsf{pcase}\, N_1\,(\mathsf{0}N_2),\ \mathsf{pcase}\, N_1\,(\mathsf{1}N_2),\ \mathsf{pcase}\, N_1\,(N_2, N_3),$
$\mathsf{pcase}\, N_1\, N_2$ with $N_2 : \tau \to \varrho$, $\mathsf{pcase}\, N_1\, N_2\, N_3$ with $N_2, N_3 : \tau \to \varrho$.
(Note the type restrictions of the last two forms: They are parts of the left-hand side of rule ($\mathsf{pcase}{\to}$).)

A term M is called *passive* iff there is no redex part N with $M \to^* N$.

No reduct of a passive term is able to interact with a context in the reduction of a redex. Simple examples of passive terms are the variables. The following two lemmas state the needed properties of passive terms.

Lemma 5.12

1) If M is passive and $MN \rightarrow^* P$, then $P = M'N'$ with $M \rightarrow^* M'$ and $N \rightarrow^* N'$.

2) If M is passive, then MN is also passive for all N.

3) If M is passive, then $\overline{\mathcal{S}}[\![MN]\!]\varepsilon = |\overline{\mathcal{S}}[\![M]\!]\varepsilon|\ (\overline{\mathcal{S}}[\![N]\!]\varepsilon)$ for all N.

Proof:

1) The proof is by induction on the length n of the reduction $MN \rightarrow^* P$.
   It is clear for $n = 0$.
   Induction step: Let $MN \rightarrow^* P \rightarrow Q$ be a reduction of length $n+1$. By induction hypothesis $P = M'N'$ with $M \rightarrow^* M'$ and $N \rightarrow^* N'$. M' is no redex part. Therefore either $Q = M''N'$ with $M' \rightarrow M''$ or $Q = M'N''$ with $N' \rightarrow N''$.

2) Let $MN \rightarrow^* P$. By part 1) we have $P = M'N'$ with $M \rightarrow^* M'$. As M' is not a redex part, P is not a redex part either. (There is no rule with a variable-applying left-hand side $xM_1 \dots M_n$.)

3) For all A we have:

$$\begin{array}{rcl} A \trianglelefteq MN & \Leftrightarrow & \exists P.\ MN \rightarrow^* P \ \wedge\ A \lhd P \\ & \Leftrightarrow & \exists M', N'.\ M \rightarrow^* M' \ \wedge\ N \rightarrow^* N' \ \wedge\ A \lhd M'N', \ \ \Rightarrow \text{ by part 1)} \\ & \Leftrightarrow & \exists M', N', B, C.\ M \rightarrow^* M' \ \wedge\ N \rightarrow^* N' \ \wedge \\ & & A = BC \ \wedge\ B \lhd M' \ \wedge\ C \lhd N', \\ & & \ \ \Leftarrow \text{ by part 1), as } M' \text{ is passive} \\ & \Leftrightarrow & \exists B, C.\ A = BC \ \wedge\ B \trianglelefteq M \ \wedge\ C \trianglelefteq N. \end{array}$$

From the direction $\Rightarrow$ follows: $\overline{\mathcal{S}}[\![MN]\!]\varepsilon \subseteq |\overline{\mathcal{S}}[\![M]\!]\varepsilon|\ (\overline{\mathcal{S}}[\![N]\!]\varepsilon)$.
The direction $\Leftarrow$ gives:

$$\begin{array}{rcl} |\overline{\mathcal{S}}[\![M]\!]\varepsilon|\ (\overline{\mathcal{S}}[\![N]\!]\varepsilon) & = & |\bigcup_{B \trianglelefteq M} \mathcal{S}[\![B]\!]\varepsilon|\ (\bigcup_{C \trianglelefteq N} \mathcal{S}[\![C]\!]\varepsilon) \\ & = & \bigcup_{B \trianglelefteq M} \bigcup_{C \trianglelefteq N} \mathcal{S}[\![BC]\!]\varepsilon, \text{ by continuity} \\ & \subseteq & \overline{\mathcal{S}}[\![MN]\!]\varepsilon, \text{ from } \Leftarrow . \end{array}$$

∎

Lemma 5.13 *If $M \in \mathcal{T}_\sigma$ is passive and $a \in \overline{\mathcal{S}}[\![M]\!]\varepsilon$, then $a <^\sigma_\varepsilon M$.*

Proof: by structural induction on a.

• $\sigma = \tau + \varrho$:
The lemma is clear for $a = 0$ and $a = 1$.

Now let $a = (0, a')$. As M is passive, M will not reduce to the form $\mathsf{0}M'$ or $\mathsf{1}M'$. Therefore $\mathit{Out0}(M) = \mathsf{case}\, M(\lambda y.y)\, \Omega$ is passive, too.

$$\begin{aligned} a' &\in \mathsf{case}\,(\overline{\mathcal{S}}[\![M]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![\lambda y.y]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![\Omega]\!]\varepsilon), \text{ as } a \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \\ &= \overline{\mathcal{S}}[\![\mathsf{case}\, M(\lambda y.y)\,\Omega]\!]\varepsilon, \text{ by Lemma 5.6} \\ &= \overline{\mathcal{S}}[\![\mathit{Out0}(M)]\!]\varepsilon. \end{aligned}$$

By the induction hypothesis we get $a' <^{\tau}_{\varepsilon} \mathit{Out0}(M)$.
The case $a = (1, a')$ is analogous.

• $\sigma = \tau \times \varrho$ is like $\sigma = \tau + \varrho$.

• $\sigma = \tau \to \varrho$:
Let $a = (X, a')$.
Let $N \in \mathcal{T}_\tau$ and $X <^{\tau}_{\varepsilon} N$. Then MN is passive by Lemma 5.12, 2).
$(X, a') \in \overline{\mathcal{S}}[\![M]\!]\varepsilon$ and $X \subseteq \overline{\mathcal{S}}[\![N]\!]\varepsilon$ imply

$$a' \in |\overline{\mathcal{S}}[\![M]\!]\varepsilon|\,(\overline{\mathcal{S}}[\![N]\!]\varepsilon) = \overline{\mathcal{S}}[\![MN]\!]\varepsilon, \text{ by Lemma 5.12, 3).}$$

By induction hypothesis we get $a' <^{\varrho}_{\varepsilon} MN$.
Thus we have shown $a <^{\sigma}_{\varepsilon} M$. ■

We need a special lemma for pcase giving its properties with respect to the inclusive predicates. It must be proved by induction on primes. Note that such a lemma is not necessary for the other constants.

Lemma 5.14

1) If $0 \in \overline{\mathcal{S}}[\![M_0]\!]\varepsilon$ *and* $a <^{\sigma}_{\varepsilon} M_1$*, then* $a <^{\sigma}_{\varepsilon} \mathsf{pcase}\, M_0 M_1 M_2$*.*

2) If $1 \in \overline{\mathcal{S}}[\![M_0]\!]\varepsilon$ *and* $a <^{\sigma}_{\varepsilon} M_2$*, then* $a <^{\sigma}_{\varepsilon} \mathsf{pcase}\, M_0 M_1 M_2$*.*

3) If $a <^{\sigma}_{\varepsilon} M_1$ *and* $a <^{\sigma}_{\varepsilon} M_2$*, then* $a <^{\sigma}_{\varepsilon} \mathsf{pcase}\, M_0 M_1 M_2$*.*

Proof: We abbreviate $M = \mathsf{pcase}\, M_0 M_1 M_2$.

1) The proof is by structural induction on a.
If $M_0 \to^* \mathsf{0}M_0'$ for some M_0', then $M \to^* M_1$, and $a <^{\sigma}_{\varepsilon} M$ follows from Lemma 5.10.
We assume in the following that not $M_0 \to^* \mathsf{0}M_0'$. (Also $M_0 \to^* \mathsf{1}M_0'$ is not possible because of $0 \in \overline{\mathcal{S}}[\![M_0]\!]\varepsilon$.)
We give a case analysis on a:

• $\sigma = \tau + \varrho$:
Let $a = (0, a')$:

a) We assume $M_1 \to^* \mathsf{0}M_1'$ and $M_2 \to^* \mathsf{0}M_2'$ for some M_1', M_2'.
Then $M \to^* \mathsf{0}\,(\mathsf{pcase}\, M_0 M_1' M_2')$.
$(0, a') <^{\sigma}_{\varepsilon} M_1$ implies $a' <^{\tau}_{\varepsilon} \mathit{Out0}(M_1)$.

From Lemma 5.10 and $\mathit{Out0}(M_1) \to^* M_1'$ follows $a' <^\tau_\varepsilon M_1'$.
The induction hypothesis gives $a' <^\tau_\varepsilon \mathsf{pcase}\, M_0 M_1' M_2'$.
Therefore $a' \in \overline{\mathcal{S}}[\![\mathsf{pcase}\, M_0 M_1' M_2']\!]\varepsilon$ and

$$\begin{array}{rcl} (0, a') & \in & 0\,(\overline{\mathcal{S}}[\![\mathsf{pcase}\, M_0 M_1' M_2']\!]\varepsilon) \\ & = & \overline{\mathcal{S}}[\![\mathsf{0}\,(\mathsf{pcase}\, M_0 M_1' M_2')]\!]\varepsilon, \text{ by Lemma 5.6} \\ & = & \overline{\mathcal{S}}[\![M]\!]\varepsilon, \text{ by Lemma 5.5.} \end{array}$$

Furthermore $a' <^\tau_\varepsilon \mathit{Out0}(M)$, as $\mathit{Out0}(M) \to^* \mathsf{pcase}\, M_0 M_1' M_2'$, by Lemma 5.10.

b) We assume that *not* ($M_1 \to^* \mathsf{0} M_1'$ and $M_2 \to^* \mathsf{0} M_2'$) for any M_1', M_2'.
Together with the assumption (not $M_0 \to^* \mathsf{0} M_0'$) there is no reduct of M that is a redex. Then

$$a \in \mathit{pcase}\,(\overline{\mathcal{S}}[\![M_0]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![M_1]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![M_2]\!]\varepsilon) = \overline{\mathcal{S}}[\![M]\!]\varepsilon, \text{ by Lemma 5.6.}$$

M is passive (note that M_1, M_2 are not of functional type). By Lemma 5.13 we get $a <^\sigma_\varepsilon M$.

The case $a = 0$ is contained in the proof for $a = (0, a')$, and the cases $a = 1$, $a = (1, a')$ are analogous.

• $\sigma = \tau \times \varrho$ is like $\sigma = \tau + \varrho$.

• $\sigma = \tau \to \varrho$: Let $a = (X, a')$.
With the assumption (not $M_0 \to^* \mathsf{0} M_0'$) there is no reduct of M that is a redex. Then

$$a \in \mathit{pcase}\,(\overline{\mathcal{S}}[\![M_0]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![M_1]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![M_2]\!]\varepsilon) = \overline{\mathcal{S}}[\![M]\!]\varepsilon, \text{ by Lemma 5.6.}$$

It remains to show: $\forall N \in \mathcal{T}_\tau.\ (X <^\tau_\varepsilon N \ \Rightarrow\ a' <^\varrho_\varepsilon MN)$.
It is $MN = \mathsf{pcase}\, M_0 M_1 M_2 N \to \mathsf{pcase}\, M_0 (M_1 N)(M_2 N)$. We get:

$$\begin{array}{rcl} X <^\tau_\varepsilon N & \Rightarrow & a' <^\varrho_\varepsilon M_1 N, \text{ as } (X, a') <^\sigma_\varepsilon M_1 \\ & \Rightarrow & a' <^\varrho_\varepsilon \mathsf{pcase}\, M_0 (M_1 N)(M_2 N), \text{ by induction hypothesis} \\ & \Rightarrow & a' <^\varrho_\varepsilon MN, \text{ by Lemma 5.10.} \end{array}$$

This concludes part 1) of the lemma.

2) Part 2) is analogous to part 1).

3) The proof is by structural induction on a.
If $M_0 \to^* \mathsf{0} M_0'$ for some M_0', then $M \to^* M_1$, and $a <^\sigma_\varepsilon M$ follows from Lemma 5.10.
If $M_0 \to^* \mathsf{1} M_0'$ for some M_0', then $M \to^* M_2$, and again $a <^\sigma_\varepsilon M$.
We assume in the following that neither $M_0 \to^* \mathsf{0} M_0'$ nor $M_0 \to^* \mathsf{1} M_0'$. We give a case analysis on a:

• $\sigma = \tau + \varrho$:
Let $a = (0, a')$.

a) We assume $M_1 \to^* \mathsf{0}M_1'$ and $M_2 \to^* \mathsf{0}M_2'$ for some M_1', M_2'.
Then $M \to^* \mathsf{0}\,(\mathsf{pcase}\, M_0 M_1' M_2')$.
From $a <_\varepsilon^\sigma M_1$, $a <_\varepsilon^\sigma M_2$ we conclude by Lemma 5.10 that $a' <_\varepsilon^\tau M_1'$ and $a' <_\varepsilon^\tau M_2'$.
By induction hypothesis $a' <_\varepsilon^\tau \mathsf{pcase}\, M_0 M_1' M_2'$. As in part 1) we conclude $a <_\varepsilon^\sigma M$.

b) We assume that not ($M_1 \to^* \mathsf{0}M_1'$ and $M_2 \to^* \mathsf{0}M_2'$) for any M_1', M_2'.
As in part 1) we conclude $a <_\varepsilon^\sigma M$.

The case $a = 0$ is contained in the proof for $a = (0, a')$, and the cases $a = 1$, $a = (1, a')$ are analogous.

- $\sigma = \tau \times \varrho$ is like $\sigma = \tau + \varrho$.
- $\sigma = \tau \to \varrho$:

The argumentation is just the same as in part 1), except that we conclude:
$X <_\varepsilon^\tau N \;\Rightarrow\; a' <_\varepsilon^\varrho M_1 N$ and $a' <_\varepsilon^\varrho M_2 N$. ■

In the following lemma we collect all the properties of the relations $<_\varepsilon^\sigma$ on elements of D_σ that we need in the proof of the Approximation Theorem.

Lemma 5.15 (Inclusive Predicate Lemma) *In the following d is an element of D_σ, D_τ, or D_ϱ, and $M, N \in \mathcal{T}_\sigma$.*

1) $\perp <_\varepsilon^\sigma M$.

2) $\sigma = \tau + \varrho$:

$$0d <_\varepsilon^{\tau+\varrho} M \;\Leftrightarrow\; 0d \subseteq \overline{\mathcal{S}}[\![M]\!]\varepsilon \textit{ and } d <_\varepsilon^\tau \textit{Out0}(M)$$
$$1d <_\varepsilon^{\tau+\varrho} M \;\Leftrightarrow\; 1d \subseteq \overline{\mathcal{S}}[\![M]\!]\varepsilon \textit{ and } d <_\varepsilon^\varrho \textit{Out1}(M)$$

3) $\sigma = \tau \times \varrho$:

$$d <_\varepsilon^{\tau\times\varrho} M \;\Leftrightarrow\; d \subseteq \overline{\mathcal{S}}[\![M]\!]\varepsilon \textit{ and } \textit{fst}\, d <_\varepsilon^\tau \mathsf{fst}\, M \textit{ and } \textit{snd}\, d <_\varepsilon^\varrho \mathsf{snd}\, M$$

4) $\sigma = \tau \to \varrho$:

$$d <_\varepsilon^{\tau\to\varrho} M \;\Leftrightarrow\; d \subseteq \overline{\mathcal{S}}[\![M]\!]\varepsilon \textit{ and } \forall e \in D_\tau, N \in \mathcal{T}_\tau.\, (e <_\varepsilon^\tau N \Rightarrow |d|e <_\varepsilon^\varrho MN)$$

5) Let $n \geq 0$ and c be a constant of type $\sigma = \tau_1 \to \ldots \to \tau_n \to \varrho$, such that there is no reduction rule for c with less than n arguments. Then $\mathcal{S}[\![c]\!]\perp <_\varepsilon^\sigma c$ iff

$$d_i <_\varepsilon^{\tau_i} M_i \textit{ for } 1 \leq i \leq n \;\Rightarrow\; (\mathcal{S}[\![c]\!]\perp) d_1 \ldots d_n <_\varepsilon^\varrho c M_1 \ldots M_n.$$

6) If $d <_\varepsilon^\sigma M$ and $M \downarrow N$, then also $d <_\varepsilon^\sigma N$.

7) If $M \in \mathcal{T}_\sigma$ is passive and $d \subseteq \overline{\mathcal{S}}[\![M]\!]\varepsilon$, then $d <_\varepsilon^\sigma M$.

8) If $0 \in \overline{\mathcal{S}}[\![M_0]\!]\varepsilon$ and $d <_\varepsilon^\sigma M_1$, then $d <_\varepsilon^\sigma \mathsf{pcase}\, M_0 M_1 M_2$.

9) If $1 \in \overline{\mathcal{S}}[\![M_0]\!]\varepsilon$ and $d <_\varepsilon^\sigma M_2$, then $d <_\varepsilon^\sigma \mathsf{pcase}\, M_0 M_1 M_2$.

10) If $d_1 <_\varepsilon^\sigma M_1$ and $d_2 <_\varepsilon^\sigma M_2$, then $d_1 \cap d_2 <_\varepsilon^\sigma \mathsf{pcase}\, M_0 M_1 M_2$.

Note: The parts 6) and 7) of this lemma replace the Lemma 5 of the proof of the Approximation Theorem for the untyped λ-calculus in [MP87]. A condition for the recursively typed λ-calculus corresponding to that of Lemma 5 would be too complicated.

Proof: 1), 2), and 3) are simple consequences of the definition of $<_\varepsilon^\sigma$.

4) $\Rightarrow$: $d \subseteq \overline{\mathcal{S}}[\![M]\!]\varepsilon$ is clear.
Now let $e \in D_\tau$, $N \in \mathcal{T}_\tau$ and $e <_\varepsilon^\tau N$.
Let $a \in |d|e$. Then there is $X \subseteq e$ with $(X, a) \in d$.
From $(X, a) <_\varepsilon^{\tau \to \varrho} M$ and $X <_\varepsilon^\tau N$ follows $a <_\varepsilon^\varrho MN$.
$\Leftarrow$: Let $(X, a) \in d$. We show: $\forall N.\ X <_\varepsilon^\tau N \ \Rightarrow\ a <_\varepsilon^\varrho MN$.
Let $e = X{\downarrow}$. By Lemma 5.9 we get $e <_\varepsilon^\tau N$. Then $a \in |d|e <_\varepsilon^\varrho MN$.

5) The proof is by induction on n. Note that ϱ may be a functional type that varies with n. $n = 0$ is clear.
Now assume the proposition for c is true for some $n \geq 0$; we prove it for $n + 1$:

$$\begin{array}{lll} \mathcal{S}[\![c]\!]\bot <_\varepsilon^\sigma c & & \\ \text{iff } d_i <_\varepsilon^{\tau_i} M_i \text{ for } 1 \le i \le n & \Rightarrow & (\mathcal{S}[\![c]\!]\bot)d_1 \ldots d_n <_\varepsilon^{\tau_{n+1} \to \varrho} cM_1 \ldots M_n, \\ & & \text{by induction hypothesis} \\ \text{iff } d_i <_\varepsilon^{\tau_i} M_i \text{ for } 1 \le i \le n & \Rightarrow & (\mathcal{S}[\![c]\!]\bot)d_1 \ldots d_n \subseteq \overline{\mathcal{S}}[\![cM_1 \ldots M_n]\!]\varepsilon \text{ and} \\ & & (d_{n+1} <_\varepsilon^{\tau_{n+1}} M_{n+1} \Rightarrow \\ & & (\mathcal{S}[\![c]\!]\bot)d_1 \ldots d_{n+1} <_\varepsilon^\varrho cM_1 \ldots M_{n+1}), \\ & & \text{by part 4).} \end{array}$$

Lemma 5.6 says $\overline{\mathcal{S}}[\![cM_1 \ldots M_n]\!]\varepsilon = (\mathcal{S}[\![c]\!]\bot)(\overline{\mathcal{S}}[\![M_1]\!]\varepsilon) \ldots (\overline{\mathcal{S}}[\![M_n]\!]\varepsilon)$, therefore $(\mathcal{S}[\![c]\!]\bot)d_1 \ldots d_n \subseteq \overline{\mathcal{S}}[\![cM_1 \ldots M_n]\!]\varepsilon$ is fulfilled.

6) Follows from Lemma 5.10.

7) Follows from Lemma 5.13.

8), 9) and 10) follow from Lemma 5.14. ■

The Approximation Theorem would be proved if we could show that $\mathcal{S}[\![M]\!]\varepsilon <_\varepsilon^\sigma M$ for all $M \in \mathcal{T}_\sigma$. We will now prove, by structural induction on M, a stronger statement in order to handle free variables in the case of abstraction.

Lemma 5.16 (Approximation Lemma) *Let $M \in \mathcal{T}_\sigma$, $\varepsilon \in Env$, $x_i^{\sigma_i}$ $(1 \le i \le n, n \ge 0)$ be a sequence of distinct variables, $d_i \in D_{\sigma_i}$ and $N_i \in \mathcal{T}_{\sigma_i}$ for all i.
If $d_i <_\varepsilon^{\sigma_i} N_i$ for all i, then*

$$\mathcal{S}[\![M]\!](\varepsilon[x_1 \mapsto d_1, \ldots, x_n \mapsto d_n]) <_\varepsilon^\sigma M[x_1{:=}N_1, \ldots, x_n{:=}N_n].$$

Here $\varepsilon[x_1 \mapsto d_1, \ldots, x_n \mapsto d_n]$ is the environment that maps x to $\varepsilon(x)$ if $x \neq x_i$ for all i, and x_i to d_i. $M[x_1{:=}N_1, \ldots, x_n{:=}N_n]$ is the result of the simultaneous substitution of *the N_i for the free occurrences of x_i in M, with appropriate renaming of bound variables of M.*

Proof: by structural induction on M.
For any $\varepsilon' \in Env$ we abbreviate $\overline{\varepsilon'} = \varepsilon'[x_1 \mapsto d_1, \ldots, x_n \mapsto d_n]$, and for any term L we write $\overline{L} = L[x_1{:=}N_1, \ldots, x_n{:=}N_n]$.
We cite the parts of the Inclusive Predicate Lemma simply by part i). The use of parts 1) – 5) should be obvious and is often not mentioned.

- $M = \Omega$: $\mathcal{S}[\![\Omega]\!]\overline{\varepsilon} = \bot <^{\sigma}_{\varepsilon} \Omega$.
- $M = \mathsf{0},\ \sigma = \tau \to (\tau + \varrho)$:

To show $\mathcal{S}[\![\mathsf{0}]\!]\overline{\varepsilon} <^{\sigma}_{\varepsilon} \mathsf{0}$, we prove $d <^{\tau}_{\varepsilon} N \ \Rightarrow\ 0d <^{\tau+\varrho}_{\varepsilon} \mathsf{0}N$.
We have $0d \subseteq 0(\overline{\mathcal{S}}[\![N]\!]\varepsilon) = \overline{\mathcal{S}}[\![\mathsf{0}N]\!]\varepsilon$. Furthermore $d <^{\tau}_{\varepsilon} \mathit{Out0}(\mathsf{0}N)$ by part 6), as $\mathit{Out0}(\mathsf{0}N) \to^{*} N$.

- $M = \mathsf{1}$ is analogous.
- $M = \mathsf{case},\ \sigma = (\tau + \varrho) \to (\tau \to \psi) \to (\varrho \to \psi) \to \psi$:

To show $\mathcal{S}[\![\mathsf{case}]\!]\overline{\varepsilon} <^{\sigma}_{\varepsilon} \mathsf{case}$, we have to prove:

$$d_0 <^{\tau+\varrho}_{\varepsilon} M_0 \ \wedge\ d_1 <^{\tau\to\psi}_{\varepsilon} M_1 \ \wedge\ d_2 <^{\varrho\to\psi}_{\varepsilon} M_2 \ \Rightarrow\ \mathit{case}\, d_0 d_1 d_2 <^{\psi}_{\varepsilon} \mathsf{case}\, M_0 M_1 M_2.$$

This is clear for $d_0 = \bot$.
Now let $d_0 = 0d_0'$.

a) We assume $M_0 \to^{*} \mathsf{0}M_0'$ for some M_0'.
   As $d_1 <^{\tau\to\psi}_{\varepsilon} M_1$ and $d_0' <^{\tau}_{\varepsilon} \mathit{Out0}(M_0)$, we get
   $\mathit{case}\, d_0 d_1 d_2 = |d_1| d_0' <^{\psi}_{\varepsilon} M_1(\mathit{Out0}(M_0))$.
   We have $\mathsf{case}\, M_0 M_1 M_2 \to^{*} M_1 M_0'$ and $M_1(\mathit{Out0}(M_0)) \to^{*} M_1 M_0'$,
   so $\mathit{case}\, d_0 d_1 d_2 <^{\psi}_{\varepsilon} \mathsf{case}\, M_0 M_1 M_2$ by part 6).

b) We assume that not $M_0 \to^{*} \mathsf{0}M_0'$ for any M_0'.
   $M_0 \to^{*} \mathsf{1}M_0'$ is also impossible. So there is no reduct of $\mathsf{case}\, M_0 M_1 M_2$ that is a redex.
   From Lemma 5.6 we conclude:

$$\begin{aligned} \mathit{case}\, d_0 d_1 d_2 &\subseteq \mathit{case}\,(\overline{\mathcal{S}}[\![M_0]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![M_1]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![M_2]\!]\varepsilon) \\ &= \overline{\mathcal{S}}[\![\mathsf{case}\, M_0 M_1 M_2]\!]\varepsilon. \end{aligned}$$

   Furthermore $\mathsf{case}\, M_0 M_1 M_2$ is passive, and $\mathit{case}\, d_0 d_1 d_2 <^{\psi}_{\varepsilon} \mathsf{case}\, M_0 M_1 M_2$ follows from part 7).

The case $d_0 = 1d_0'$ is analogous.

- $M = \mathsf{pcase},\ \sigma = (\tau + \varrho) \to \psi \to \psi \to \psi$:

We have to prove:
$d_0 <^{\tau+\varrho}_{\varepsilon} M_0 \ \wedge\ d_1 <^{\psi}_{\varepsilon} M_1 \ \wedge\ d_2 <^{\psi}_{\varepsilon} M_2 \ \Rightarrow\ \mathit{pcase}\, d_0 d_1 d_2 <^{\psi}_{\varepsilon} \mathsf{pcase}\, M_0 M_1 M_2$.
For $d_0 = \bot$ we have $\mathit{pcase}\, d_0 d_1 d_2 = d_1 \cap d_2$. The result follows from part 10).
For $d_0 = 0d_0'$ we use part 8), for $d_0 = 1d_0'$ part 9).

• $M = \mathsf{pair}$ is like $M = \mathsf{0}$.

• $M = \mathsf{fst}$, $\sigma = (\tau \times \varrho) \to \tau$:
$d <_\varepsilon^{\tau\times\varrho} N \;\Rightarrow\; \mathit{fst}\, d <_\varepsilon^\tau \mathsf{fst}\, N$ follows directly from part 3).

• $M = \mathsf{snd}$ is analogous.

• $M = x$:
If $x = x_i$ for some i, then $\mathcal{S}[\![x]\!]\overline{\varepsilon} = d_i <_\varepsilon^\sigma N_i = \overline{x}$.
Now let $x \neq x_i$ for all i. Then $\mathcal{S}[\![x]\!]\overline{\varepsilon} = \varepsilon(x) \subseteq \overline{\mathcal{S}}[\![x]\!]\varepsilon$. x is passive. From part 7) follows $\mathcal{S}[\![x]\!]\overline{\varepsilon} <_\varepsilon^\sigma x$.

• $M = NP$, where $N : \tau \to \sigma$ and $P : \tau$:
By induction hypothesis we have $\mathcal{S}[\![N]\!]\overline{\varepsilon} <_\varepsilon^{\tau\to\sigma} \overline{N}$ and $\mathcal{S}[\![P]\!]\overline{\varepsilon} <_\varepsilon^\tau \overline{P}$.
Therefore $|\mathcal{S}[\![N]\!]\overline{\varepsilon}|\,(\mathcal{S}[\![P]\!]\overline{\varepsilon}) <_\varepsilon^\sigma \overline{N}\,\overline{P}$, by part 4).
Thus we get $\mathcal{S}[\![NP]\!]\overline{\varepsilon} <_\varepsilon^\sigma \overline{NP}$.

• $M = \lambda x^\tau.M'$, $\sigma = \tau \to \varrho$:
We may assume that x is no x_i and x does not occur free in any N_i. (x can be renamed by α-conversion.)
First we prove that $\mathcal{S}[\![\lambda x.M']\!]\overline{\varepsilon} \subseteq \overline{\mathcal{S}}[\![\overline{\lambda x.M'}]\!]\varepsilon$.

$$\begin{array}{rcl}
\mathcal{S}[\![\lambda x.M']\!]\overline{\varepsilon} & = & \mathit{Pr}(d \in D_\tau \mapsto \mathcal{S}[\![M']\!](\overline{\varepsilon}[x \mapsto d])) \\
& = & \mathit{Pr}(d \in D_\tau \mapsto \mathcal{S}[\![M']\!](\overline{\varepsilon[x \mapsto d]})),\ \text{as } x \text{ is no } x_i \\
& \subseteq & \mathit{Pr}(d \in D_\tau \mapsto \overline{\mathcal{S}}[\![\overline{M'}]\!](\varepsilon[x \mapsto d])), \\
& & \text{as } \mathcal{S}[\![M']\!](\overline{\varepsilon[x \mapsto d]}) <_{\varepsilon[x\mapsto d]}^\varrho \overline{M'} \text{ by induction hypothesis} \\
& = & \mathit{Pr}(d \in D_\tau \mapsto \bigcup\limits_{A \lhd \overline{M'}} \mathcal{S}[\![A]\!](\varepsilon[x \mapsto d])) \\
& = & \bigcup\limits_{A \lhd \overline{M'}} \mathit{Pr}(d \in D_\tau \mapsto \mathcal{S}[\![A]\!](\varepsilon[x \mapsto d])) \\
& = & \bigcup\limits_{A \lhd \overline{M'}} \mathcal{S}[\![\lambda x.A]\!]\varepsilon \\
& = & \bigcup\limits_{B \lhd \overline{\lambda x.M'}} \mathcal{S}[\![B]\!]\varepsilon, \\
& & \text{as } A \lhd \overline{M'} \;\Leftrightarrow\; \lambda x.A \lhd \lambda x.\overline{M'} = \overline{\lambda x.M'},\ \text{since } x \text{ is no } x_i \\
& = & \overline{\mathcal{S}}[\![\overline{\lambda x.M'}]\!]\varepsilon
\end{array}$$

Now we prove that: $d <_\varepsilon^\tau N \;\Rightarrow\; |\mathcal{S}[\![M]\!]\overline{\varepsilon}|\, d <_\varepsilon^\varrho \overline{M}N$.

$$\begin{array}{rcl}
|\mathcal{S}[\![M]\!]\overline{\varepsilon}|\, d & = & \mathcal{S}[\![M']\!](\overline{\varepsilon}[x \mapsto d]) \\
& = & \mathcal{S}[\![M']\!](\varepsilon[x_1 \mapsto d_1, \ldots, x_n \mapsto d_n, x \mapsto d]),\ \text{as } x \text{ is no } x_i \\
& <_\varepsilon^\varrho & M'[x_1{:=}N_1, \ldots, x_n{:=}N_n, x{:=}N],\ \text{by induction hypothesis}
\end{array}$$

Furthermore we have:

$$\begin{array}{rcl}
\overline{M}N & = & (\lambda x.\overline{M'})N,\ \text{as } x \text{ is no } x_i \\
& \to & (M'[x_1{:=}N_1, \ldots, x_n{:=}N_n])[x{:=}N] \\
& = & M'[x_1{:=}N_1, \ldots, x_n{:=}N_n, x{:=}N],\ \text{as } x \text{ is not free in any } N_i
\end{array}$$

From part 6) follows $|\mathcal{S}[\![M]\!]\overline{\varepsilon}|\ d <^{\varrho}_{\varepsilon} \overline{M}N$. ∎

Proof of the Approximation Theorem:
$\mathcal{S}[\![M]\!]\varepsilon \subseteq \overline{\mathcal{S}}[\![M]\!]\varepsilon$ follows from $\mathcal{S}[\![M]\!]\varepsilon <^{\sigma}_{\varepsilon} M$, which holds by the preceding lemma. ∎

Corollary 5.17 *For all terms M and environments ε:*

$$\mathcal{S}[\![M]\!]\varepsilon = \bigcup\{\mathcal{S}[\![A]\!]\varepsilon \mid A \textit{ is a normal form and } \exists N.\ M \to^* N \ \wedge\ A \prec N\}$$

Proof: $\overline{\mathcal{S}}[\![M]\!]\varepsilon \subseteq$ the right-hand side, and the right-hand side $\subseteq \mathcal{S}[\![M]\!]\varepsilon$. ∎

Note: The original paper [Wad78] gives a definition of approximations in the form of this corollary, for the untyped λ-calculus.

Corollary 5.18 *The semantics of the fixed point combinator* $Y_\sigma = \lambda y^{\sigma\to\sigma}.(\lambda x.y(xx))(\lambda x.y(xx))$ *is*

$$\mathcal{S}[\![Y_\sigma]\!]\varepsilon = \mathit{Pr}(f \in D_{\sigma\to\sigma} \mapsto \bigcup_{n\geq 0} f^n(\bot)),$$

so $|\mathcal{S}[\![Y_\sigma]\!]\varepsilon| f$ *is the least fixed point of* $|f|$.

Proof: The approximations of Y_σ are just the terms $\lambda y.y^n\Omega$, with $y^0\Omega = \Omega$ and $y^{n+1}\Omega = y(y^n\Omega)$.

$$\begin{aligned}
\mathcal{S}[\![Y_\sigma]\!]\varepsilon &= \overline{\mathcal{S}}[\![Y_\sigma]\!]\varepsilon \\
&= \bigcup_{n\geq 0} \mathcal{S}[\![\lambda y.y^n\Omega]\!]\varepsilon \\
&= \bigcup_{n\geq 0} \mathit{Pr}(f \in D_{\sigma\to\sigma} \mapsto f^n(\bot)) \\
&= \mathit{Pr}(f \in D_{\sigma\to\sigma} \mapsto \bigcup_{n\geq 0} f^n(\bot))
\end{aligned}$$

∎

Let us continue our discussion of the definition of approximations. In the case of the untyped λ-calculus [Bar84] it is possible to define least approximations via head normal forms. Let us look at this approach more abstractly: We are given a set H of normal forms with the property: If $A \in H$ and $A \prec M$, then $A \lhd M$. This means that an H-prefix of a term M does not change by reductions of M. In the case of the untyped λ-calculus H is the set consisting just of Ω and all terms of the form $\lambda x_1 \dots x_n.yA_1 \dots A_m$ with $A_i \in H$. We define

$$\mathcal{S}^H[\![M]\!]\varepsilon = \bigcup\{\mathcal{S}[\![A]\!]\varepsilon \mid A \in H \text{ and } \exists N.\ M \to^* N \ \wedge\ A \prec N\}.$$

H should fulfill: $\mathcal{S}^H[\![M]\!]\varepsilon = \mathcal{S}[\![M]\!]\varepsilon$ for all M, ε. We show that a set H with this property and the property above does not exist for our calculus with `pcase`:

Let $M = \mathsf{pcase}\, x\, \mathsf{0}\, \Omega$. It is $M \notin H$, because of the first property of H and as not $M \lhd \mathsf{pcase}\, x\, \mathsf{0}\, \mathsf{0}$. For all $A \prec M$ with $A \neq M$ we have $\mathcal{S}[\![A]\!](\bot[x \mapsto 0]) = \bot$. Therefore $\mathcal{S}^H[\![M]\!](\bot[x \mapsto 0]) = \bot \neq 0 = \mathcal{S}[\![M]\!](\bot[x \mapsto 0])$.

Let us now consider the sequential calculus *without* pcase. In this case we can define two sets H with the desired properties.

Definition 5.19 A normal form A is a *minimum normal form* (mnf) iff for all $B \prec A$: $\mathcal{S}[\![B]\!] = \mathcal{S}[\![A]\!] \;\Rightarrow\; B = A$.
A normal form A is a *constant normal form* (cnf) iff

$$A = \Omega \text{ or } A = \lambda x_1 \ldots x_n.yA_1 \ldots A_m,$$

where $n \geq 0$, $m \geq 0$, y is a variable or a constant $\notin \{\Omega, \mathsf{pcase}\}$, the A_i are cnfs and for $y \in \{\mathsf{fst}, \mathsf{snd}, \mathsf{case}\}$ and $m \geq 1$ it is $A_1 \neq \Omega$.

Constant normal forms resemble the normal forms of H defined by head normal forms above, for the untyped λ-calculus.

Lemma 5.20 *Every minimum normal form without* pcase *is a constant normal form.*

Proof: Suppose A is a normal form without pcase that is no cnf. We show by structural induction on A that A is no mnf.
We have $A = \lambda x_1 \ldots x_n.yA_1 \ldots A_m$, $n \geq 0$, $m \geq 0$, y a variable or a constant, and one of the following three cases:

1) $y = \Omega$ and ($n > 0$ or $m > 0$).
   Then $\Omega \prec A$, $\Omega \neq A$ and $\mathcal{S}[\![\Omega]\!] = \mathcal{S}[\![A]\!]$, so A is no mnf.
2) Some A_i is no cnf.
   By induction hypothesis A_i is no mnf. Then also A is no mnf.
3) y is fst, snd or case and $A_1 = \Omega$.
   Then $\mathcal{S}[\![A]\!] = \mathcal{S}[\![\Omega]\!]$, A is no mnf. ■

Lemma 5.21 *If A is a constant normal form and $A \prec M$, then $A \lhd M$.*

Proof: We prove: If A is a cnf, $A \prec M$ and $M \to N$, then $A \prec N$, by structural induction on A. (The lemma follows by simple induction on reductions $M \to^* N$.)
The case $A = \Omega$ is clear.
Now let $A = \lambda x_1 \ldots x_n.yA_1 \ldots A_m$. Then $M = \lambda x_1 \ldots x_n.yM_1 \ldots M_m$ with $A_i \prec M_i$ for all i.
The term $yM_1 \ldots M_m$ is no redex:
This is clear if y is a variable or $\mathsf{0}$, $\mathsf{1}$, or pair.
If $y = \mathsf{fst}$ or $y = \mathsf{snd}$, and $m \geq 1$, then $A_1 \neq \Omega$ and A_1 is not of the form $\mathsf{pair}\, A'A''$. So M_1 is not of this form either.

If $y = \mathsf{case}$ and $m \geq 1$, then $A_1 \neq \Omega$ and A_1 and M_1 are not of the form $\mathsf{0}A'$ or $\mathsf{1}A'$. Thus there is some j with $M_j \to N_j$ and $N = \lambda x_1 \ldots x_n.yM_1 \ldots M_{j-1}N_jM_{j+1} \ldots M_m$. By the induction hypothesis we get $A_j \prec N_j$, therefore $A \prec N$. ■

By this lemma the set of cnfs (and the set of mnfs) has the first of the two properties of H. We define two new approximation sets for terms:

$$\begin{aligned}\mathcal{B}(M) &= \{A \mid A \text{ is a mnf and } \exists N.\ M \to^* N \ \wedge\ A \prec N\}\\ \mathcal{C}(M) &= \{A \mid A \text{ is a cnf and } \exists N.\ M \to^* N \ \wedge\ A \prec N\}\end{aligned}$$

For the sequential calculus without pcase we have:

$$\mathcal{B}(M) \subseteq \mathcal{C}(M) \subseteq \mathcal{A}(M).$$

The first inclusion follows from Lemma 5.20, the second from Lemma 5.21.
$\mathcal{B}(M) \subseteq \mathcal{A}(M)$ is not valid for $M = \mathsf{pcase}\, x\, \mathsf{0}\, \mathsf{0}$: We have $\mathsf{pcase}\, x\, \mathsf{0}\, \Omega \in \mathcal{B}(M)$, but $\mathsf{pcase}\, x\, \mathsf{0}\, \Omega \notin \mathcal{A}(M)$.
In every case, also for pcase:

$$\overline{\mathcal{S}}[\![M]\!]\varepsilon = \bigcup_{A \in \mathcal{A}(M)} \mathcal{S}[\![A]\!]\varepsilon \subseteq \bigcup_{A \in \mathcal{B}(M)} \mathcal{S}[\![A]\!]\varepsilon \text{ for all } \varepsilon \in \mathit{Env}.$$

This is because for every normal form A there is a mnf $B \prec A$ with $\mathcal{S}[\![A]\!] = \mathcal{S}[\![B]\!]$.
We combine these results with the Approximation Theorem:

Theorem 5.22 *In the sequential calculus without* pcase*: For all terms M and environments ε,*

$$\bigcup_{A \in \mathcal{B}(M)} \mathcal{S}[\![A]\!]\varepsilon = \bigcup_{A \in \mathcal{C}(M)} \mathcal{S}[\![A]\!]\varepsilon = \overline{\mathcal{S}}[\![M]\!]\varepsilon = \mathcal{S}[\![M]\!]\varepsilon.$$

With this theorem the set of mnfs and the set of cnfs both have the second property of H.

[My conjecture is that in the sequential calculus $\mathcal{B}(M)$ is the least approximation of M with the same semantics as M. More precisely the conjecture is: Let I be an ideal of normal forms such that for all $A \in I$ there is N with $M \to^* N$ and $A \prec N$, and $\mathcal{S}[\![M]\!] = \bigcup_{A \in I} \mathcal{S}[\![A]\!]$. Then $\mathcal{B}(M) \subseteq I$.]

6 Adequacy and full abstraction

The classical semantical analysis of the programming language PCF [Plo77] proceeds as follows: The closed terms of the ground type integer are singled out as *programs*. Programs are regarded as the only terms whose syntactical values (integers) can be observed directly. All other terms must be observed through program contexts. If the semantics of a programm M is an integer value i, then M can be reduced to i. This result is called the *adequacy* of the semantics. Then an operational preorder is defined on terms: $M \sqsubseteq N$ iff for all contexts $C[\,]$ such that $C[M]$ and $C[N]$ are programs, if $C[M] \to^* i$, then also $C[N] \to^* i$. If $\mathcal{S}[\![M]\!] \subseteq \mathcal{S}[\![N]\!]$, then $M \sqsubseteq N$; this follows from soundness and adequacy. The converse, *full abstraction*, is not true for sequential PCF, but holds for PCF with a parallel conditional.

We follow the same programme for our recursively typed λ-calculus. We choose the closed terms of type $\mathsf{bool} = \mathsf{void} + \mathsf{void}$ as our programs. Thus the observable non-bottom syntactical values are the terms of the form $\mathsf{0}M$ or $\mathsf{1}M$. We have chosen the smallest type with more than one element. (Any non-functional, non-trivial type, built from $+$ and $\times$ only, would do as well.)

Definition 6.1 The set of *programs* is $\mathit{Prog} = \mathcal{T}^c_{\mathsf{bool}}$.
We define the *operational evaluation function* $\mathcal{O} : \mathit{Prog} \to D_{\mathsf{bool}}$ by $\mathcal{O}[\![M]\!] = 0$ if $M \to^* \mathsf{0}M'$, $\mathcal{O}[\![M]\!] = 1$ if $M \to^* \mathsf{1}M'$, for some M', and $\mathcal{O}[\![M]\!] = \bot$ otherwise.

We want to prove adequacy (that the reduction of a program reaches its semantic value) from the Approximation Theorem of the preceding chapter. We need the following lemma:

Lemma 6.2 *Let $\sigma \in T_\infty$ and $A \in \mathcal{N}_\sigma$ be a normal form with $\mathcal{S}[\![A]\!]\bot \neq \bot$.*
If $\sigma = \tau + \varrho$, then $A = \mathsf{0}A'$ or $A = \mathsf{1}A'$ for some A'.
If $\sigma = \tau \times \varrho$, then $A = \mathsf{pair}\, A'A''$ for some A', A''.

Proof: by structural induction on A.
We suppose A is of type $\tau + \varrho$ or $\tau \times \varrho$. Then $A = cA_1 \ldots A_n$, $n \geq 0$, with c a constant and the A_i normal forms. We give a case analysis on c:

$c = \mathsf{0}, \mathsf{1}$ or pair: The lemma is fulfilled.

$c = \mathsf{fst}$ or snd:
Then $n \geq 1$. $\mathcal{S}[\![A]\!]\bot \neq \bot$ implies $\mathcal{S}[\![A_1]\!]\bot \neq \bot$ implies $A_1 = \mathsf{pair}\, A'A''$ by induction hypothesis. Then A is no normal form.

$c = \mathsf{case}$:
Then $n \geq 3$. $\mathcal{S}[\![A]\!]\bot \neq \bot$ implies $\mathcal{S}[\![A_1]\!]\bot \neq \bot$ implies $A_1 = \mathsf{0}A_1'$ or $A_1 = \mathsf{1}A_1'$ by induction hypothesis. Then A is no normal form.

$c = \mathsf{pcase}$:
Then $n = 3$. If $\mathcal{S}[\![A_1]\!]\bot \neq \bot$, then $A_1 = \mathsf{0}A_1'$ or $A_1 = \mathsf{1}A_1'$ by induction hypothesis and A is no normal form.
If $\mathcal{S}[\![A_1]\!]\bot = \bot$, then $\mathcal{S}[\![A]\!]\bot = \mathcal{S}[\![A_2]\!]\bot \cap \mathcal{S}[\![A_3]\!]\bot \neq \bot$.
If $\sigma = \tau + \varrho$, then by induction hypothesis either ($A_2 = \mathsf{0}A_2'$, $A_3 = \mathsf{0}A_3'$) or ($A_2 = \mathsf{1}A_2'$, $A_3 = \mathsf{1}A_3'$). In both cases A is no normal form.
If $\sigma = \tau \times \varrho$, then by induction hypothesis $A_2 = \mathsf{pair}\, A_2'A_2''$ and $A_3 = \mathsf{pair}\, A_3'A_3''$ and A is no normal form. ■

Theorem 6.3 (Adequacy) *For all $M \in \mathit{Prog}$: $\mathcal{O}[\![M]\!] = \mathcal{S}[\![M]\!]\bot$.*

Proof: $\mathcal{O}[\![M]\!] \subseteq \mathcal{S}[\![M]\!]\bot$ follows from soundness: If $M \to^* \mathsf{0}M'$, then $\mathcal{S}[\![M]\!]\bot = \mathcal{S}[\![\mathsf{0}M']\!]\bot = 0$; and if $M \to^* \mathsf{1}M'$, then $\mathcal{S}[\![M]\!]\bot = \mathcal{S}[\![\mathsf{1}M']\!]\bot = 1$.
It remains to show the adequacy: $\mathcal{S}[\![M]\!]\bot \subseteq \mathcal{O}[\![M]\!]$.
Suppose $\mathcal{S}[\![M]\!]\bot = 0$. By the Approximation Theorem there is an approximation $A \trianglelefteq M$ with $\mathcal{S}[\![A]\!]\bot = 0$. From the preceding lemma follows $A = \mathsf{0}A'$ for some A', therefore $\mathcal{O}[\![M]\!] = 0$. Analogously $\mathcal{S}[\![M]\!]\bot = 1$ implies $\mathcal{O}[\![M]\!] = 1$. ■

Note that this theorem is also valid for the sequential calculus without pcase. It can also be proved directly using the inclusive predicate technique, with a proof a bit easier than the proof of the Approximation Theorem, e.g. the passive terms are not needed.

Now we define the operational preorder on terms, based on the observation of terms through program contexts.

Definition 6.4 Let $M, N \in \mathcal{T}_\sigma$. $M \sqsubseteq N$ iff for all contexts $C[\,]$, such that $C[M]$ and $C[N]$ are programs, $\mathcal{O}[\![C[M]]\!] \subseteq \mathcal{O}[\![C[N]]\!]$ holds.

Theorem 6.5 (Full abstraction) *For all $M, N \in \mathcal{T}_\sigma$: $M \sqsubseteq N$ iff $\mathcal{S}[\![M]\!] \subseteq \mathcal{S}[\![N]\!]$.*

The direction "If $\mathcal{S}[\![M]\!] \subseteq \mathcal{S}[\![N]\!]$ then $M \sqsubseteq N$" follows easily from soundness and adequacy: $\mathcal{O}[\![C[M]]\!] = \mathcal{S}[\![C[M]]\!]\bot \subseteq \mathcal{S}[\![C[N]]\!]\bot = \mathcal{O}[\![C[N]]\!]$. This holds also for the sequential calculus without pcase. In this case the contexts are restricted. Therefore the opposite direction is not valid for the sequential calculus, as can be shown by the same example as in [Plo77].

For the proof of the opposite direction (for the parallel calculus) we prove a lemma that states the definability of all finite elements of the semantics.

Lemma 6.6 (Definability) *For all finite $d \in D_\sigma$ there is a closed term $M \in \mathcal{T}^c_\sigma$ with $\mathcal{S}[\![M]\!]\bot = d$.*

We recall that finite elements are the elements that are downward closures of finite sets of primes. In our term construction we use the following parallel function and instead of pcase:

$$\begin{aligned} \mathsf{and} &: \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}, \text{ defined as} \\ \mathsf{and} &= \lambda xy.\mathsf{pcase}\, x\, y\, \mathsf{1}. \end{aligned}$$

Here and in the following we interpret the Boolean value 0 as true and 1 as false, and chose the names of our functions accordingly. (We made this choice in order to interpret case like if-then-else, with the second argument as true-part and the third argument as false-part.) The semantics of and fulfills: $(\mathcal{S}[\![\mathsf{and}]\!]\bot)00 = 0$, $(\mathcal{S}[\![\mathsf{and}]\!]\bot)1\bot = 1$, $(\mathcal{S}[\![\mathsf{and}]\!]\bot)\bot 1 = 1$. Here we show that all finite elements are definable from and and the sequential constants. In the next chapter we will show that also pcase (which is not finite) is definable from and.

Proof: We have to introduce some notions first. A term $C : \mathsf{bool}$ is called a *condition* iff for every environment ε:

$$(\forall \varepsilon' \supseteq \varepsilon.\ \mathcal{S}[\![C]\!]\varepsilon' \neq 0) \ \Rightarrow\ \mathcal{S}[\![C]\!]\varepsilon = 1.$$

The semantics of a condition is so "dense" that it gives the value 1 for every environment that cannot be enlarged to give the value 0.
A *conditioned prime* is a pair $C{\rightarrow}a$ of a condition C and a prime a. In the course of our construction the condition of $C{\rightarrow}a$ will be used to accumulate a term that checks function arguments. The intuitive semantics of the "mixed term" $C{\rightarrow}a$ is the prime a for every environment ε with $\mathcal{S}[\![C]\!]\varepsilon = 0$. For a set P of primes, $Cond(P)$ is the set of all conditioned primes $C{\rightarrow}a$ with $a \in P$.
A set X of conditioned primes is called *consistent* iff for all $C{\rightarrow}a, C'{\rightarrow}a' \in X$ holds: $(\exists \varepsilon.\ \mathcal{S}[\![C]\!]\varepsilon = \mathcal{S}[\![C']\!]\varepsilon = 0) \ \Rightarrow\ a \uparrow a'$.
For $M \in \mathcal{T}_\sigma$, $X \subseteq Cond(P(\sigma))$ finite and consistent, we define a predicate *term*:

$$M\ term\ X \text{ iff } \mathcal{S}[\![M]\!]\varepsilon = \{a \mid \exists C.\ (C{\rightarrow}a) \in X \ \wedge\ \mathcal{S}[\![C]\!]\varepsilon = 0\}{\downarrow} \ \text{ for all } \varepsilon.$$

For $M \in \mathcal{T}^c_{\sigma\rightarrow\mathsf{bool}}$, $X \subseteq P(\sigma)$ finite and consistent, we define a predicate *eq*:

$$M\ eq\ X \text{ iff } |\mathcal{S}[\![M]\!]\bot|\ d = \begin{cases} 0, & \text{if } X \subseteq d \\ 1, & \text{if } d \not\uparrow X \\ \bot & \text{otherwise} \end{cases}$$

where $d \not\uparrow X$ means: $\exists a \in d, b \in X.$ not $a \uparrow b$.

We prove for every $n \geq 0$ and every $\sigma \in T_\infty$:

1) For every finite and consistent $X \subseteq Cond(P_n(\sigma))$ there is $M \in \mathcal{T}_\sigma$ with $M\ term\ X$.

2) For every finite and consistent $X \subseteq P_n(\sigma)$ there is $M \in \mathcal{T}^c_{\sigma\rightarrow\mathsf{bool}}$ with $M\ eq\ X$.

We use abbreviations for the following function terms:

$$\begin{array}{lcll} \mathsf{if} & = & \lambda xyz.\mathsf{case}\ x\,(\lambda w.y)\,(\lambda w.z): & \mathsf{bool} \rightarrow \sigma \rightarrow \sigma \rightarrow \sigma \\ \mathsf{not} & = & \lambda x.\mathsf{if}\, x\, 1\, 0: & \mathsf{bool} \rightarrow \mathsf{bool} \\ \mathsf{or} & = & \lambda xy.\mathsf{not}\,(\mathsf{and}\,(\mathsf{not}\, x)\,(\mathsf{not}\, y)): & \mathsf{bool} \rightarrow \mathsf{bool} \rightarrow \mathsf{bool} \end{array}$$

The semantics of or is: $(\mathcal{S}[\![\mathsf{or}]\!]\bot)11 = 1$, $(\mathcal{S}[\![\mathsf{or}]\!]\bot)0\bot = 0$, $(\mathcal{S}[\![\mathsf{or}]\!]\bot)\bot 0 = 0$.

The proof of statements 1) and 2) is by simultaneous induction on n:

$n = 0$: 1) $X = \emptyset$. $\Omega\ term\ X$.
2) $X = \emptyset$. $(\lambda x.0)\ eq\ X$.

Induction step:

1) Let $X \subseteq \mathit{Cond}(P_{n+1}(\sigma))$ be finite and consistent. We construct M *term* X by case analysis over σ.

• $\sigma = \mathsf{void}$: $X = \emptyset$, Ω *term* X.

• $\sigma = \tau + \varrho$:
Define the condition sets $C^0 = \{C \mid \exists a \geq 0.\ (C \rightarrow a) \in X\}$ and $C^1 = \{C \mid \exists a \geq 1.\ (C \rightarrow a) \in X\}$.
Define the term $M_0 : \mathsf{bool}$ as $M_0 = \mathsf{1}$ for $C^0 = \emptyset$, otherwise as $M_0 = \mathsf{or}\, C_1^0\, (\mathsf{or}\, C_2^0 \ldots C_j^0)$ for some enumeration $\{C_1^0, C_2^0, \ldots, C_j^0\} = C^0$. Analogously, M_1 is defined as an or-term of the elements of C^1.
Let $X^0 = \{C \rightarrow a \mid (C \rightarrow (0, a)) \in X\}$ and $X^1 = \{C \rightarrow a \mid (C \rightarrow (1, a)) \in X\}$. It is $X^0 \subseteq \mathit{Cond}(P_n(\tau))$ and $X^1 \subseteq \mathit{Cond}(P_n(\varrho))$, both are finite and consistent. By the induction hypothesis there are terms $N_0 \in \mathcal{T}_\tau$, $N_1 \in \mathcal{T}_\varrho$ with N_0 *term* X^0 and N_1 *term* X^1.
We build the term

$$M = \mathsf{if}\, M_0\, (\mathsf{0} N_0)\, (\mathsf{if}\, M_1\, (\mathsf{1} N_1)\, \Omega)$$

and show that M *term* X,
i.e. for all ε, $\mathcal{S}[\![M]\!]\varepsilon = Y{\downarrow}$ with $Y = \{a \mid \exists C.\ (C \rightarrow a) \in X \ \wedge\ \mathcal{S}[\![C]\!]\varepsilon = 0\}$:

$\star$ $\mathcal{S}[\![M]\!]\varepsilon \subseteq Y{\downarrow}$:
Let $a \in \mathcal{S}[\![M]\!]\varepsilon$. We show $a \in Y{\downarrow}$ in each of the two cases:

a) $\mathcal{S}[\![M_0]\!]\varepsilon = 0$: Then $a \in 0\,(\mathcal{S}[\![N_0]\!]\varepsilon)$.
First let $a = 0$. There is some $C \in C^0$ with $\mathcal{S}[\![C]\!]\varepsilon = 0$. $(C \rightarrow a') \in X$ for some $a' \geq 0$, therefore $0 \in Y{\downarrow}$.
Now let $a = (0, a')$. Then $a' \in \mathcal{S}[\![N_0]\!]\varepsilon$. Since N_0 *term* X^0, there is $(C \rightarrow a'') \in X^0$ with $\mathcal{S}[\![C]\!]\varepsilon = 0$ and $a' \leq a''$. $(C \rightarrow (0, a'')) \in X$, therefore $(0, a') \in Y{\downarrow}$.

b) $\mathcal{S}[\![M_0]\!]\varepsilon = 1$ and $\mathcal{S}[\![M_1]\!]\varepsilon = 0$: Then $a \in 1\,(\mathcal{S}[\![N_1]\!]\varepsilon)$.
Analogously to case a) we show that $a \in Y{\downarrow}$.

$\star$ $\mathcal{S}[\![M]\!]\varepsilon \supseteq Y{\downarrow}$:
Let $a \in Y$, i.e. $(C \rightarrow a) \in X$ and $\mathcal{S}[\![C]\!]\varepsilon = 0$ for some C. We show $a \in \mathcal{S}[\![M]\!]\varepsilon$ in each of the four cases:

a) $a = 0$:
$C \in C^0$, therefore $\mathcal{S}[\![M_0]\!]\varepsilon = 0$ and $0 \in \mathcal{S}[\![M]\!]\varepsilon$.

b) $a = (0, a')$:
Again $C \in C^0$, therefore $\mathcal{S}[\![M_0]\!]\varepsilon = 0$ and $\mathcal{S}[\![M]\!]\varepsilon = 0\,(\mathcal{S}[\![N_0]\!]\varepsilon)$. $(C \rightarrow a') \in X^0$, therefore $a' \in \mathcal{S}[\![N_0]\!]\varepsilon$, as N_0 *term* X^0. It follows $(0, a') \in \mathcal{S}[\![M]\!]\varepsilon$.

c) $a = 1$:
Then $C \in C^1$, therefore $\mathcal{S}[\![M_1]\!]\varepsilon = 0$.
We show that $\mathcal{S}[\![M_0]\!]\varepsilon = 1$, i.e. for all $C' \in C^0$: $\mathcal{S}[\![C']\!]\varepsilon = 1$. Here we use the fact that C' is a condition:

Let $\varepsilon' \supseteq \varepsilon$. Then $\mathcal{S}[\![C]\!]\varepsilon' = 0$. $\mathcal{S}[\![C']\!]\varepsilon' = 0$ would contradict the consistency of X, as $C \in C^1$ and $C' \in C^0$. Therefore $\mathcal{S}[\![C']\!]\varepsilon' \neq 0$. We conclude $\mathcal{S}[\![C']\!]\varepsilon = 1$.
So we have $\mathcal{S}[\![M_0]\!]\varepsilon = 1$, $\mathcal{S}[\![M_1]\!]\varepsilon = 0$ and $1 \in \mathcal{S}[\![M]\!]\varepsilon$.

d) $a = (1, a')$:
As in case c) we have $\mathcal{S}[\![M_0]\!]\varepsilon = 1$, $\mathcal{S}[\![M_1]\!]\varepsilon = 0$ and $\mathcal{S}[\![M]\!]\varepsilon = 1\,(\mathcal{S}[\![N_1]\!]\varepsilon)$.
$(C {\rightarrow} a') \in X^1$, therefore $a' \in \mathcal{S}[\![N_1]\!]\varepsilon$, as N_1 *term* X^1. It follows $(1, a') \in \mathcal{S}[\![M]\!]\varepsilon$.

• $\sigma = \tau \times \varrho$:
Let $X^0 = \{C {\rightarrow} a \mid (C {\rightarrow} (0, a)) \in X\} \subseteq \mathit{Cond}(P_n(\tau))$, and $X^1 = \{C {\rightarrow} a \mid (C {\rightarrow} (1, a)) \in X\} \subseteq \mathit{Cond}(P_n(\varrho))$. Both sets are finite and compatible.
By the induction hypothesis there are terms N_0, N_1 with N_0 *term* X^0 and N_1 *term* X^1.
Let $M = (N_0, N_1)$.

$$\begin{aligned}
\mathcal{S}[\![M]\!]\varepsilon &= \mathit{pair}\,(\mathcal{S}[\![N_0]\!]\varepsilon)\,(\mathcal{S}[\![N_1]\!]\varepsilon) \\
&= \{0\} \times \{a \mid \exists C.\ (C {\rightarrow} a) \in X^0 \ \wedge\ \mathcal{S}[\![C]\!]\varepsilon = 0\}{\downarrow} \cup \\
&\ \ \{1\} \times \{a \mid \exists C.\ (C {\rightarrow} a) \in X^1 \ \wedge\ \mathcal{S}[\![C]\!]\varepsilon = 0\}{\downarrow} \\
&= \{a \mid \exists C.\ (C {\rightarrow} a) \in X \ \wedge\ \mathcal{S}[\![C]\!]\varepsilon = 0\}{\downarrow}
\end{aligned}$$

• $\sigma = \tau \rightarrow \varrho$:
Let $X = \{C_i {\rightarrow} (Y_i, a_i) \mid 1 \leq i \leq k\}$ be an enumeration of the elements of X.
For all i, $Y_i \subseteq P_n(\tau)$ is finite and consistent. By the induction hypothesis there is N_i *eq* Y_i for all i.
Let x be a variable of type τ that does not occur free in any C_i. Let $D_i = \mathsf{and}\, C_i\,(N_i x)$.
We define $Z = \{D_i {\rightarrow} a_i \mid 1 \leq i \leq k\}$ and first prove that $Z \subseteq \mathit{Cond}(P_n(\varrho))$ and Z is consistent:

$\star$ $D_i = \mathsf{and}\, C_i\,(N_i x)$ is a condition:
Let ε be an environment such that for all $\varepsilon' \supseteq \varepsilon$, $\mathcal{S}[\![D_i]\!]\varepsilon' \neq 0$. We have to show that $\mathcal{S}[\![D_i]\!]\varepsilon = 1$.
Assume $\mathcal{S}[\![C_i]\!]\varepsilon \neq 1$. As C_i is a condition, there is $\varepsilon'' \supseteq \varepsilon$ with $\mathcal{S}[\![C_i]\!]\varepsilon'' = 0$. Let $\varepsilon' = \varepsilon''[x \mapsto Y_i{\downarrow}]$.
Then $\mathcal{S}[\![C_i]\!]\varepsilon' = 0$, as x does not occur free in C_i. Furthermore $\mathcal{S}[\![N_i x]\!]\varepsilon' = 0$, as N_i *eq* Y_i. Together we get $\mathcal{S}[\![D_i]\!]\varepsilon' = 0$.
Then ε and ε' cannot have an upper bound. (For such an upper bound δ would be: $\delta \supseteq \varepsilon$ and $\mathcal{S}[\![D_i]\!]\delta = 0$.) As $\varepsilon'' \supseteq \varepsilon$, it must be $\varepsilon(x) \not\uparrow \varepsilon'(x) = Y_i{\downarrow}$. Hence $\mathcal{S}[\![N_i x]\!]\varepsilon = 1$, and we conclude $\mathcal{S}[\![D_i]\!]\varepsilon = 1$.

$\star$ Z is consistent:
Let $\mathcal{S}[\![D_i]\!]\varepsilon = \mathcal{S}[\![D_j]\!]\varepsilon = 0$ for some i, j, ε. Then $\mathcal{S}[\![C_i]\!]\varepsilon = \mathcal{S}[\![C_j]\!]\varepsilon = 0$, hence $(Y_i, a_i) \uparrow (Y_j, a_j)$. Also $\mathcal{S}[\![N_i x]\!]\varepsilon = \mathcal{S}[\![N_j x]\!]\varepsilon = 0$, therefore $Y_i \subseteq \varepsilon(x)$ and $Y_j \subseteq \varepsilon(x)$. So $Y_i \uparrow Y_j$ and we conclude $a_i \uparrow a_j$.

We have proved that $Z \subseteq Cond(P_n(\varrho))$ is a finite, consistent, conditioned prime set. By induction hypothesis there is N *term* Z. Let $M = \lambda x.N$. We prove M *term* X, i.e.

$$\mathcal{S}[\![M]\!]\varepsilon = Pr(d \in D_\tau \mapsto \mathcal{S}[\![N]\!](\varepsilon[x \mapsto d])) = \{(Y_i, a_i) \mid 1 \le i \le k \;\wedge\; \mathcal{S}[\![C_i]\!]\varepsilon = 0\}\!\downarrow .$$

$\subseteq$: Let $(Y, a) \in \mathcal{S}[\![M]\!]\varepsilon$. Then

$$\begin{aligned} a &\in \mathcal{S}[\![N]\!](\varepsilon[x \mapsto Y\!\downarrow]) \\ &= \{a_i \mid \mathcal{S}[\![D_i]\!](\varepsilon[x \mapsto Y\!\downarrow]) = 0\}\!\downarrow, \text{ as } N \textit{ term } Z. \end{aligned}$$

Let $a \le a_i$ and $\mathcal{S}[\![D_i]\!](\varepsilon[x \mapsto Y\!\downarrow]) = 0$. Then $|\mathcal{S}[\![N_i]\!]\bot|\,(Y\!\downarrow) = 0$. Hence $Y_i \subseteq Y\!\downarrow$, as N_i *eq* Y_i. So we get $(Y, a) \le (Y_i, a_i)$.
Furthermore $\mathcal{S}[\![C_i]\!]\varepsilon = \mathcal{S}[\![C_i]\!](\varepsilon[x \mapsto Y\!\downarrow]) = 0$.

$\supseteq$: Let $\mathcal{S}[\![C_i]\!]\varepsilon = 0$.
We have $|\mathcal{S}[\![N_i]\!]\bot|\,(Y_i\!\downarrow) = 0$, as N_i *eq* Y_i. Therefore $\mathcal{S}[\![D_i]\!](\varepsilon[x \mapsto Y_i\!\downarrow]) = 0$. As N *term* Z, it is $a_i \in \mathcal{S}[\![N]\!](\varepsilon[x \mapsto Y_i\!\downarrow])$. Hence $(Y_i, a_i) \in \mathcal{S}[\![M]\!]\varepsilon$.

2) Let $X \subseteq P_{n+1}(\sigma)$ be finite and consistent. We construct M *eq* X by case analysis over σ.

• $\sigma = \mathsf{void}$: $X = \emptyset$, $(\lambda x.\mathsf{0})$ *eq* X.

• $\sigma = \tau + \varrho$:
If $X = \emptyset$, then $(\lambda x.\mathsf{0})$ *eq* X.
Now let $a \in X$ for some $a \ge 0$. Let $Y = \{a \mid (0, a) \in X\} \subseteq P_n(\tau)$. By induction hypothesis there is some N with N *eq* Y. Take $M = \lambda x.\mathsf{case}\; x\, N\, \mathsf{1}$. It can be easily checked that M *eq* X.
The case $a \in X$ for some $a \ge 1$ is similar.

• $\sigma = \tau \times \varrho$:
Let $X_0 = \{a \mid (0, a) \in X\} \subseteq P_n(\tau)$ and $X_1 = \{a \mid (1, a) \in X\} \subseteq P_n(\varrho)$.
There are N_0 *eq* X_0 and N_1 *eq* X_1 by induction hypothesis.
Let $M = \lambda x.\mathsf{and}\; (N_0(\mathsf{fst}\; x))\; (N_1(\mathsf{snd}\; x))$. We check easily that M *eq* X.

• $\sigma = \tau \rightarrow \varrho$:
If $X = \emptyset$, then $(\lambda x.\mathsf{0})$ *eq* X.
Otherwise, let $X = \{(Y_i, a_i) \mid 1 \le i \le k\}$ be an enumeration of X.
Let $Y_i' = \{\mathsf{0}{\rightarrow}b | b \in Y_i\} \subseteq Cond(P_n(\tau))$ for all i, it is finite and consistent. By induction hypothesis there is N_i *term* Y_i' for all i. Furthermore, by induction hypothesis there is Q_i *eq* a_i for all i. We define

$$M = \lambda x.\mathsf{and}\; (Q_1(xN_1))(\mathsf{and}\; (Q_2(xN_2)) \ldots (Q_k(xN_k))).$$

We check that M *eq* X: Let $d \in D_\sigma$.
If $X \subseteq d$, then for all i:
$|\mathcal{S}[\![Q_i]\!]\bot|\,(|d|\,(\mathcal{S}[\![N_i]\!]\bot)) = |\mathcal{S}[\![Q_i]\!]\bot|\,(|d|\,(Y_i\!\downarrow)) = 0$, as $a_i \in |d|\,(Y_i\!\downarrow)$.

Therefore $|\mathcal{S}[\![M]\!]\bot|\ d = 0$.
If $d \not\uparrow X$, then there is some j with $d \not\uparrow (Y_j, a_j)$, i.e. $|d|\ (Y_j\!\downarrow) \not\uparrow \{a_j\}$. Therefore $|\mathcal{S}[\![Q_j]\!]\bot|\ (|d|\ (\mathcal{S}[\![N_j]\!]\bot)) = 1$, and $|\mathcal{S}[\![M]\!]\bot|\ d = 1$.
Otherwise, $d \uparrow (Y_i, a_i)$ for all i and $(Y_j, a_j) \notin d$ for some j. Then $|\mathcal{S}[\![M]\!]\bot|\ d = \bot$.

We have now proved statements 1) and 2) for all n and σ. The lemma follows easily from 1): If $d \in D_\sigma$ is finite, it has the form $d = X\!\downarrow$ with $X \subseteq P_n(\sigma)$ for some n, X finite and consistent. There is a term M with M *term* $\{\mathsf{0}{\to}a \mid a \in X\}$, i.e. $\mathcal{S}[\![M]\!]\bot = X\!\downarrow$. ∎

Proof of the Full Abstraction Theorem:
It remains to show for all $M, N \in \mathcal{T}_\sigma$: If $M \sqsubseteq N$, then $\mathcal{S}[\![M]\!]\varepsilon \subseteq \mathcal{S}[\![N]\!]\varepsilon$ for all ε.

First suppose that M and N are closed terms.
Let $a \in \mathcal{S}[\![M]\!]\varepsilon$. Define $f = (\{a\}, 0)\!\downarrow \in D_{\sigma\to\mathsf{bool}}$. By the Definability Lemma, there is $P \in \mathcal{T}^c_{\sigma\to\mathsf{bool}}$ with $\mathcal{S}[\![P]\!]\bot = f$. $P[\,]$ serves as a context such that PM and PN are programs.
$0 = \mathcal{S}[\![PM]\!]\bot = \mathcal{O}[\![PM]\!] \subseteq \mathcal{O}[\![PN]\!] = \mathcal{S}[\![PN]\!]\bot$, therefore $a \in \mathcal{S}[\![N]\!]\varepsilon$.

Now let M and N be terms with their free variables in $\{x_1, \ldots, x_n\}$. We get $\lambda x_1 \ldots x_n.M \sqsubseteq \lambda x_1 \ldots x_n.N$: For all contexts $C[\,]$ apply the context $C[\lambda x_1 \ldots x_n.[\,]]$ to M and N. For the closed terms follows: $\mathcal{S}[\![\lambda x_1 \ldots x_n.M]\!]\varepsilon \subseteq \mathcal{S}[\![\lambda x_1 \ldots x_n.N]\!]\varepsilon$ for all ε. Hence $\mathcal{S}[\![M]\!]\varepsilon \subseteq \mathcal{S}[\![N]\!]\varepsilon$ for all ε. ∎

7 Interdefinability of constants

Our first observation is that case can be defined from pcase and $\mathsf{out0}$, $\mathsf{out1}$(see page 9 for the def. of $\mathsf{out0}$, $\mathsf{out1}$). We have

$$\mathcal{S}[\![\mathsf{case}]\!] = \mathcal{S}[\![\lambda xyz.\mathsf{pcase}\, x\, (\mathsf{pcase}\, x\, (y\, (\mathsf{out0}\, x))\, \Omega)(\mathsf{pcase}\, x\, \Omega\, (z\, (\mathsf{out1}\, x)))]\!].$$

In the preceding chapter we used the function $\mathsf{and} : \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}$, defined as $\mathsf{and} = \lambda xy.\mathsf{pcase}\, x\, y\, \mathsf{1}$, to build defining terms for all finite elements of the semantic model. Now we will show that also pcase (whose semantics is not finite) is definable from and and the sequential constants. Compare the definition of PCF's parallel conditional in terms of the parallel or in [Sto91]. We assume a constant $\mathsf{and} : \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}$ with the semantics:

$$(\mathcal{S}[\![\mathsf{and}]\!]\bot)00 = 0,\ (\mathcal{S}[\![\mathsf{and}]\!]\bot)1\bot = 1,\ (\mathcal{S}[\![\mathsf{and}]\!]\bot)\bot 1 = 1.$$

Without loss of generality, we will define only $\mathsf{pcase}_{\mathsf{void},\mathsf{void},\sigma} : \mathsf{bool} \to \sigma \to \sigma \to \sigma$ for all types σ, and write simply pcase_σ. The general pcase can be easily defined from this.

In order to cope with recursive types, we have to extend the inductive definition of pcase_σ to general type expressions σ (with free type variables). Then we have to associate with each type variable t of σ some type τ and a term variable $p : \mathsf{bool} \to \tau \to \tau \to \tau$, that stands for the pcase_τ-function in its recursive definition.

So we are lead to define an operation $Pcase(\theta, \sigma)$ that produces terms for pcase-functions. Its second argument is a type expression $\sigma \in T_\mu$. The first argument is a partial map $\theta : V_T \to V$ from type variables to term variables, with $\theta(t) \neq \theta(s)$ for $t \neq s$. θ is defined on a finite set of type variables that contains all free variables of σ. $\theta(t)$ must be of the type $\mathsf{bool} \to \tau \to \tau \to \tau$ for some type τ. We associate with θ the partial map $\overline{\theta} : V_T \to T^c_\mu$ defined by $\overline{\theta}(t) = \tau$ for $\theta(t) : \mathsf{bool} \to \tau \to \tau \to \tau$. $Pcase(\theta, \sigma)$ will be a term of type $\mathsf{bool} \to \overline{\theta}(\sigma) \to \overline{\theta}(\sigma) \to \overline{\theta}(\sigma)$, where $\overline{\theta}$ is naturally extended to the substitution of free type variables of type expressions. $[\,]$ is the totally undefined map. The notation $\theta[t \mapsto p]$ will be used as for environments.

In the definition of $Pcase$ we use abbreviations for the following function terms:

$\mathsf{if} \quad : \mathsf{bool} \to \sigma \to \sigma \to \sigma$

$\mathsf{if} \quad = \lambda xyz.\mathsf{case}\, x\, (\lambda w.y)\, (\lambda w.z)$

$\mathsf{not} : \mathsf{bool} \to \mathsf{bool}$

$\mathsf{not} = \lambda x.\mathsf{if}\, x\, 1\, 0$

$\mathsf{or} \quad : \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}$

$\mathsf{or} \quad = \lambda xy.\mathsf{not}\, (\mathsf{and}\, (\mathsf{not}\, x)\, (\mathsf{not}\, y))$

$\mathsf{pc} \quad : \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}$

$\mathsf{pc} \quad = \lambda xyz.\mathsf{or}\, (\mathsf{or}\, (\mathsf{and}\, x\, y)\, (\mathsf{and}\, (\mathsf{not}\, x)\, z))\, (\mathsf{and}\, y\, z)$

It is $\mathcal{S}[\![\mathsf{pc}]\!]\bot = \mathcal{S}[\![\mathsf{pcase}_{\mathsf{bool}}]\!]\bot$.

$\mathsf{sb} \quad : \tau + \varrho \to \mathsf{bool}$

$\mathsf{sb} \quad = \lambda x.\mathsf{case}\, x\, (\lambda y.0)\, (\lambda y.1)$

$Pcase(\theta, \sigma)$ is defined by structural induction on the type expression σ:

$$
\begin{aligned}
Pcase(\theta, t) &= \theta(t) \\
Pcase(\theta, \tau + \varrho) &= \lambda x^{\mathsf{bool}} y^{\overline{\theta}(\tau+\varrho)} z^{\overline{\theta}(\tau+\varrho)}.\mathsf{if}\ (\mathsf{pc}\, x\, (\mathsf{sb}\, y)\, (\mathsf{sb}\, z)) \\
&\qquad\qquad (0\, (Pcase(\theta, \tau)\, x\, (\mathsf{out0}\, y)\, (\mathsf{out0}\, z))) \\
&\qquad\qquad (1\, (Pcase(\theta, \varrho)\, x\, (\mathsf{out1}\, y)\, (\mathsf{out1}\, z))) \\
Pcase(\theta, \tau \times \varrho) &= \lambda x^{\mathsf{bool}} y^{\overline{\theta}(\tau\times\varrho)} z^{\overline{\theta}(\tau\times\varrho)}.(\ Pcase(\theta, \tau)\, x\, (\mathsf{fst}\, y)\, (\mathsf{fst}\, z), \\
&\qquad\qquad Pcase(\theta, \varrho)\, x\, (\mathsf{snd}\, y)\, (\mathsf{snd}\, z)) \\
Pcase(\theta, \tau \to \varrho) &= \lambda x^{\mathsf{bool}} y^{\overline{\theta}(\tau\to\varrho)} z^{\overline{\theta}(\tau\to\varrho)} w^{\overline{\theta}(\tau)}.Pcase(\theta, \varrho)\, x\, (y\, w)(z\, w) \\
Pcase(\theta, \mu t.\tau) &= Y_\pi(\lambda p^\pi.Pcase(\theta[t \mapsto p^\pi], \tau)), \\
&\quad \text{where } \pi = \mathsf{bool} \to \overline{\theta}(\mu t.\tau) \to \overline{\theta}(\mu t.\tau) \to \overline{\theta}(\mu t.\tau), \\
&\quad \text{and } p^\pi \text{ denotes the first variable in } V^\pi \text{ that is not in the image of } \theta \\
Pcase(\theta, \mathsf{void}) &= \Omega
\end{aligned}
$$

It is easy to show by induction that $Pcase(\theta, \sigma) : \mathsf{bool} \to \overline{\theta}(\sigma) \to \overline{\theta}(\sigma) \to \overline{\theta}(\sigma)$.
In the case of the recursive type expression we have

$$\begin{array}{rcl} Pcase(\theta[t \mapsto p^\pi], \tau) & : & \mathsf{bool} \to \varrho \to \varrho \to \varrho \\ \text{with } \varrho & = & \overline{\theta[t \mapsto p^\pi]}(\tau) \\ & = & (\overline{\theta}[t \mapsto \overline{\theta}(\mu t.\tau)])(\tau) \\ & = & \overline{\theta}([\,][t \mapsto \mu t.\tau](\tau)) \\ & \approx & \overline{\theta}(\mu t.\tau), \end{array}$$

so $Pcase(\theta[t \mapsto p^\pi], \tau) : \pi$, therefore $Pcase(\theta, \mu t.\tau) : \pi$.
$Pcase(\theta, \sigma)$ has the free variables $\theta(t)$ for all t free in σ.

Definition 7.1 Let $f \in D_{\mathsf{bool}\to\tau\to\tau\to\tau}$ for some type τ.
We say that f *approximates* the function *pcase to level* n, $app_n(f)$, iff $f\,c\,a\,b \supseteq (pcase\,c\,a\,b)|_n$ for all $c \in D_{\mathsf{bool}}$ and $a, b \in D_\tau$.

Lemma 7.2 *Let θ, σ be admissible arguments in $Pcase(\theta, \sigma)$, as described above. Let $n \geq 0$ and ε be an environment with $app_n(\varepsilon(\theta(t)))$ for all t free in σ.*
Then for $f = \mathcal{S}[\![Pcase(\theta, \sigma)]\!]\varepsilon$ we have $app_n(f)$.
If σ is not of the form $\mu t_1 \ldots \mu t_m.t$, with $m \geq 0$, t a type variable and $t \neq t_i$ for all i, then $app_{n+1}(f)$.

Proof: by structural induction on σ.
• $\sigma = t$: $f = \varepsilon(\theta(t))$, hence $app_n(f)$.
• $\sigma = \tau + \varrho$:
We show $app_{n+1}(f)$, i.e. $f\,c\,a\,b \supseteq (pcase\,c\,a\,b)|_{n+1}$ for all $c \in D_{\mathsf{bool}}$, $a, b \in D_{\overline{\theta}(\sigma)}$.

1) $c = \perp$:
The case $a \cap b = \perp$ is clear.
Now let $a = 0a'$, $b = 0b'$.

$$\begin{array}{rcl} f \perp (0a')\,(0b') & = & 0\,((\mathcal{S}[\![\mathsf{pcase}(\theta, \tau)]\!]\varepsilon) \perp a'\,b') \\ & \supseteq & 0\,((a' \cap b')|_n), \text{ by induction hypothesis} \\ & = & (a \cap b)|_{n+1} \\ & = & (pcase\,c\,a\,b)|_{n+1} \end{array}$$

The case $a = 1a'$, $b = 1b'$ is analogous.

2) $c = 0$:
The case $a = \perp$ is clear.
Now let $a = 0a'$.

$$\begin{array}{rcl} f\,0\,(0a')\,b & = & 0\,((\mathcal{S}[\![Pcase(\theta, \tau)]\!]\varepsilon)\,0\,a'\,((\mathcal{S}[\![\mathsf{out0}]\!]\perp)\,b)) \\ & \supseteq & 0\,(a'|_n), \text{ by induction hypothesis} \\ & = & a|_{n+1} \\ & = & (pcase\,c\,a\,b)|_{n+1} \end{array}$$

The case $a = 1a'$ is analogous.

3) $c = 1$ is analogous to $c = 0$.

• $\sigma = \tau \times \varrho$:
We show $app_{n+1}(f)$. For all $c \in D_{\mathsf{bool}}$, $a_1, b_1 \in D_{\overline{\theta}(\tau)}$ and $a_2, b_2 \in D_{\overline{\theta}(\varrho)}$ we have:

$$\begin{array}{rcl} f\, c\, (pair\, a_1 a_2)\, (pair\, b_1 b_2) & = & pair\, ((\mathcal{S}[\![Pcase(\theta,\tau)]\!]\varepsilon)\, c\, a_1\, b_1)((\mathcal{S}[\![Pcase(\theta,\varrho)]\!]\varepsilon)\, c\, a_2\, b_2) \\ & \supseteq & pair\, (pcase\, c\, a_1\, b_1)|_n\, (pcase\, c\, a_2\, b_2)|_n, \text{ by induction hyp.} \\ & = & (pair\, (pcase\, c\, a_1\, b_1)\, (pcase\, c\, a_2\, b_2))|_{n+1} \\ & = & (pcase\, c\, (pair\, a_1\, a_2)\, (pair\, b_1\, b_2))|_{n+1} \end{array}$$

• $\sigma = \tau \to \varrho$:
We prove $app_{n+1}(f)$. Let $c \in D_{\mathsf{bool}}$ and $a, b \in D_{\overline{\theta}(\sigma)}$.
$f\, c\, a\, b = Pr(d \in D_{\overline{\theta}(\tau)} \mapsto (\mathcal{S}[\![Pcase(\theta,\varrho)]\!]\varepsilon)\, c\, (a\, d)\, (b\, d))$.
Let $(X, r) \in (pcase\, c\, a\, b)|_{n+1}$. Then

$$\begin{array}{rcl} r & \in & ((pcase\, c\, a\, b)\, (X{\downarrow}))|_n \\ & = & (pcase\, c\, (a\, (X{\downarrow}))\, (b\, (X{\downarrow})))|_n \\ & \subseteq & (\mathcal{S}[\![Pcase(\theta,\varrho)]\!]\varepsilon)\, c\, (a\, (X{\downarrow}))\, (b\, (X{\downarrow})), \text{ by induction hypothesis} \end{array}$$

Hence $(X, r) \in f\, c\, a\, b$.

• $\sigma = \mu t.\tau$:

1) We assume that τ is *not* of the form $\mu t_1 \ldots \mu t_m.s$ with $m \geq 0$, s a type variable, $s \neq t$, and $s \neq t_i$ for all i. We have to show $app_{n+1}(f)$.

1.1) We assume $\tau = \mu t_1 \ldots \mu t_m.t$.
Then $\overline{\theta}(\sigma) = \sigma \approx \mathsf{void}$, hence $f \in D_{\mathsf{bool}\to\mathsf{void}\to\mathsf{void}\to\mathsf{void}}$ and $app_{n+1}(f)$.

1.2) Otherwise, τ is not of the form $\mu t_1 \ldots \mu t_m.s$ with $m \geq 0$, s a type variable and $s \neq t_i$ for all i.
We have $f = \bigcup_{i \geq 0} g^i \bot$ with $g = |\mathcal{S}[\![\lambda p.Pcase(\theta[t \mapsto p], \tau)]\!]\varepsilon|$.
We show by induction on i that $app_i(g^i \bot)$ for $0 \leq i \leq n+1$:
$app_0(g^0 \bot)$ is trivial.
Induction step: We assume $app_i(g^i \bot)$ for some $i \leq n$.
$g^{i+1}\bot = g(g^i \bot) = \mathcal{S}[\![Pcase(\theta[t \mapsto p], \tau)]\!](\varepsilon[p \mapsto g^i \bot])$.
By the general induction hypothesis (for the type expression τ) we get $app_{i+1}(g^{i+1}\bot)$.
Especially we have $app_{n+1}(g^{n+1}\bot)$, hence $app_{n+1}(f)$.

2) We assume $\tau = \mu t_1 \ldots \mu t_m.s$ with $m \geq 0$, s a type variable, $s \neq t$, and $s \neq t_i$ for all i.
Then $f = \mathcal{S}[\![Pcase(\theta, s)]\!]\varepsilon = \varepsilon(\theta(s))$, so $app_n(f)$.

• $\sigma = \mathsf{void}$: Trivial. ■

Theorem 7.3 *Let θ, σ be admissible arguments in $Pcase(\theta, \sigma)$, as described above. Let ε be an environment with $\varepsilon(\theta(t)) = \mathcal{S}[\![\mathsf{pcase}_{\overline{\theta}(t)}]\!]\perp$ for all t free in σ. Then $\mathcal{S}[\![Pcase(\theta, \sigma)]\!]\varepsilon = \mathcal{S}[\![\mathsf{pcase}_{\overline{\theta}(\sigma)}]\!]\perp$. Especially for all types σ we have: $\mathcal{S}[\![Pcase([], \sigma)]\!]\perp = \mathcal{S}[\![\mathsf{pcase}_{\sigma}]\!]\perp$.*

Proof: $\mathcal{S}[\![Pcase(\theta, \sigma)]\!]\varepsilon \supseteq \mathcal{S}[\![\mathsf{pcase}_{\overline{\theta}(\sigma)}]\!]\perp$ follows from the preceding lemma.
Now let $f = \mathcal{S}[\![Pcase(\theta, \sigma)]\!]\varepsilon$. We show $f \subseteq \mathcal{S}[\![\mathsf{pcase}_{\overline{\theta}(\sigma)}]\!]\perp$ by structural induction on σ:

• $\sigma = t : f = \varepsilon(\theta(t)) = \mathcal{S}[\![\mathsf{pcase}_{\overline{\theta}(t)}]\!]\perp$.

• $\sigma = \tau + \varrho :$
We show $f\,c\,a\,b \subseteq pcase\,c\,a\,b$ for all $c \in D_{\mathsf{bool}}$ and $a, b \in D_{\overline{\theta}(\sigma)}$.

1) $c = \perp :$
For $a \cap b = \perp$ it is $f \perp a\,b = \perp$.
Now let $a = 0a'$, $b = 0b'$.

$$\begin{array}{rcl} f \perp (0a')\,(0b') & = & 0\,((\mathcal{S}[\![Pcase(\theta, \tau)]\!]\varepsilon) \perp a'\,b') \\ & \subseteq & 0\,(pcase \perp a'\,b'), \text{ by induction hypothesis} \\ & = & pcase \perp a\,b \end{array}$$

The case $a = 1a'$, $b = 1b'$ is analogous.

2) $c = 0 :$
For $a = \perp$ it is $f\,0 \perp b = \perp$.
Now let $a = 0a'$.

$$\begin{array}{rcl} f\,0\,(0a')\,b & = & 0\,((\mathcal{S}[\![Pcase(\theta, \tau)]\!]\varepsilon)\,0\,a'\,((\mathcal{S}[\![\mathsf{out0}]\!]\perp)b)) \\ & \subseteq & 0\,(pcase\,0\,a'\,((\mathcal{S}[\![\mathsf{out0}]\!]\perp)b)), \text{ by induction hypothesis} \\ & = & pcase\,0\,a\,b \end{array}$$

The case $a = 1a'$ is analogous.

3) $c = 1$ is analogous to $c = 0$.

• $\sigma = \tau \times \varrho :$
For all $c \in D_{\mathsf{bool}}$, $a_1, b_1 \in D_{\overline{\theta}(\tau)}$ and $a_2, b_2 \in D_{\overline{\theta}(\varrho)}$:

$$\begin{array}{rcl} f\,c\,(pair\,a_1 a_2)\,(pair\,b_1 b_2) & = & pair\,((\mathcal{S}[\![Pcase(\theta, \tau)]\!]\varepsilon)\,c\,a_1\,b_1)((\mathcal{S}[\![Pcase(\theta, \varrho)]\!]\varepsilon)\,c\,a_2\,b_2) \\ & \subseteq & pair\,(pcase\,c\,a_1\,b_1)\,(pcase\,c\,a_2\,b_2), \text{ by induction hyp.} \\ & = & pcase\,c\,(pair\,a_1\,a_2)\,(pair\,b_1\,b_2) \end{array}$$

• $\sigma = \tau \to \varrho :$
For all $c \in D_{\mathsf{bool}}$, $a, b \in D_{\overline{\theta}(\sigma)}$ and $d \in D_{\overline{\theta}(\tau)}$:

$$\begin{array}{rcl} f\,c\,a\,b\,d & = & (\mathcal{S}[\![Pcase(\theta, \varrho)]\!]\varepsilon)\,c\,(a\,d)\,(b\,d) \\ & \subseteq & pcase\,c\,(a\,d)\,(b\,d), \text{ by induction hypothesis} \\ & = & pcase\,c\,a\,b\,d \end{array}$$

• $\sigma = \mu t.\tau$:
f is the least fixed point of $g = |\mathcal{S}[\![\lambda p.Pcase(\theta[t \mapsto p], \tau)]\!]\varepsilon|$.
Let $d = \mathcal{S}[\![\mathsf{pcase}_{\overline{\theta}(\sigma)}]\!]\perp$. Then

$$\begin{array}{rcl} g\,d &=& \mathcal{S}[\![Pcase(\theta[t \mapsto p], \tau)]\!](\varepsilon[p \mapsto d]) \\ &\subseteq& \mathcal{S}[\![\mathsf{pcase}_{\varrho}]\!]\perp \text{ with } \varrho = \overline{\theta[t \mapsto p]}(\tau) \approx \overline{\theta}(\sigma), \text{ by induction hypothesis} \\ &=& d. \end{array}$$

Therefore $f \subseteq d$.
• $\sigma = \mathsf{void}$: Trivial. ∎

8 Conclusion

We have given the syntax and reduction relation of a recursively typed λ-calculus with a parallel conditional pcase on all types. The calculus was proved to be confluent, with the aid of a general result on the confluence of the λ-calculus with algebraic term rewriting rules. Our reduction relation simply defines the reduction of a redex in any context. It remains to define a reduction strategy that effectively finds the normal form approximations of a term. Such a strategy cannot prescribe deterministically which redex to reduce, as we have the parallel pcase. Instead, it should give for every term a set of its outermost redexes to be reduced in the next reduction steps. Such a strategy could be given for general algebraic term rewriting rules.

We unfolded the recursive types to (possibly infinite) type trees and interpreted these type trees as prime systems. With this interpretation of types, we gave a denotational semantics of terms. The Approximation Theorem was the key result on the strength of reduction with respect to the denotational semantics: The semantics of a term equals the limit of the semantics of its normal form approximations. From this followed the adequacy of the semantics with respect to the observation of Boolean values: If the semantics of a program is 0 or 1, then the program reduces to this value. Furthermore, we showed full abstraction of the semantics. To achieve this, the syntax must contain a parallel function like pcase or and. These functions are definable from each other, so a calculus with the same expressive power could be given with reduction rules for and instead of pcase. The same expressive power means that the same elements of the semantic model are definable in both calculi. The semantic model corresponds to the observation of Boolean values, as we have seen. There are other operational, intensional properties of the original pcase that are not valid for the pcase-function defined from and, e.g. the reduction $\mathsf{pcase}\,\mathsf{0}\,M N \to^* M$. The proofs of confluence and of the Approximation Theorem would be (slightly) easier for a calculus with and. Nevertheless, we preferred to make these investigations with a pcase-calculus.

Acknowledgements: I thank Reinhold Heckmann for carefully reading a draft of this paper.

References

[AC90] R. M. Amadio and L. Cardelli. Subtyping recursive types. Report 62, Digital Systems Research Center, 1990.

[Bar84] H. P. Barendregt. *The Lambda Calculus. Its Syntax and Semantics*. North-Holland, revised edition, 1984.

[BC94] B. Blaaberg and C. Clausen. Adequacy for a lazy functional language with recursive and polymorphic types. *Theoretical Computer Science*, 136:243–275, 1994.

[Ber79] G. Berry. *Modèles complètement adéquats et stables des lambda-calculs typés.* PhD thesis, Université Paris VII, 1979.

[CC90] F. Cardone and M. Coppo. Two extensions of Curry's type inference system. In P. Odifreddi, editor, *Logic and computer science*, pages 19–75. Academic Press, 1990.

[CC91] F. Cardone and M. Coppo. Type inference with recursive types: Syntax and semantics. *Information and Computation*, 92:48–80, 1991.

[Cos89] S. Cosmadakis. Computing with recursive types. In *Proc. Logic in Computer Science*, pages 24–38. IEEE, 1989.

[Gun92] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. MIT Press, 1992.

[Hue80] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *J. of the ACM*, 27(4):797–821, 1980.

[LW91] K. G. Larsen and G. Winskel. Using information systems to solve recursive domain equations. *Information and Computation*, 91:232–258, 1991.

[MP87] P. D. Mosses and G. D. Plotkin. On proving limiting completeness. *SIAM J. Comput.*, 16:179–194, 1987.

[Mül92] F. Müller. Confluence of the lambda calculus with left-linear algebraic rewriting. *Information Processing Letters*, 41:293–299, 1992.

[NPW81] M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, part I. *Theoretical Computer Science*, 13:85–108, 1981.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–256, 1977.

[Sto91] A. Stoughton. Interdefinability of parallel operations in PCF. *Theoretical Computer Science*, 79:357–358, 1991.

[Toy88] Y. Toyama. Commutativity of term rewriting systems. In K. Fuchi and L. Kott, editors, *Programming of Future Generation Computers II*, pages 393–407. North-Holland, 1988.

[Wad78] C. P. Wadsworth. Approximate reduction and lambda calculus models. *SIAM J. Comput.*, 7:337–356, 1978.

[Win93] G. Winskel. *The Formal Semantics of Programming Languages*. MIT Press, 1993.