On Confluence and Semantic Full Abstraction of Lambda Calculus Languages

Dissertation
zur Erlangung des Grades
des Doktors der Naturwissenschaften
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

von

Fritz Müller

Saarbrücken
Mai 2016

Abstract

This thesis joins my four articles:

(1) Confluence of the lambda calculus with left-linear algebraic rewriting,
where we show that a lambda calculus with additional algebraic term rewrite rules that are confluent, left-linear and not-variable-applying, is again confluent,

(2) Full abstraction for a recursively typed lambda calculus with parallel conditional,
where we apply the theorem (1) to show that a recursively typed lambda calculus with a parallel conditional operator is confluent and has a fully abstract Scott-like (denotational) semantics,

(3) On Berry's conjectures about the stable order in PCF,
where we refute Berry's conjectures (a) that the fully abstract model of PCF together with the stable order forms bidomains, and (b) that the stable order has the syntactic order as its image,

(4) From Sazonov's non-dcpo natural domains to closed directed-lub partial orders,
where we explore Sazonov's notion of natural domains and derive the canonical subclass of closed directed-lub partial orders, which can be realized by dcpos with a restricted subset of their elements.

Kurzfassung

Diese Arbeit vereint meine vier Artikel (die englischen Titel sind übersetzt):

(1) Konfluenz des Lambda-Kalküls mit links-linearer algebraischer Termersetzung,
in dem wir zeigen, dass ein Lambda-Kalkül mit zusätzlichen algebraischen Termersetzungsregeln, die konfluent, links-linear und nicht-Variable-anwendend sind, wieder konfluent ist,

(2) Vollständige Abstraktion für einen rekursiv getypten Lambda-Kalkül mit parallelem Konditional,
in dem wir das Theorem (1) anwenden, um einen rekursiv getypten Lambda Kalkül mit einem parallelen Konditional als konfluent zu beweisen, und zu zeigen, dass er eine vollständig abstrakte Scott-artige (denotationale) Semantik hat,

(3) Über Berrys Vermutungen über die stabile Ordnung in PCF,
in dem wir Berrys Vermutungen widerlegen, (a) dass das vollständig abstrakte Modell von PCF mit der stabilen Ordnung Bidomains bildet, und (b) dass die stabile Ordnung die syntaktische Ordnung als Bild hat,

(4) Von Sazonovs nicht-dcpo natürlichen Bereichen zu geschlossenen gerichtet-sup partiellen Ordnungen,
in dem wir Sazonovs Begriff der natürlichen Bereiche erforschen und die kanonische Teilklasse von geschlossenen gerichtet-sup partiellen Ordnungen ableiten, die durch Dcpos mit einer eingeschränkten Teilmenge ihrer Elemente realisiert werden können.

Contents

Chapter 0

Introduction and extended abstract

This thesis joins my four articles:

(1) Confluence of the lambda calculus with left-linear algebraic rewriting,
Information Processing Letters, 41:293-299, 1992,
`www.rw.cdl.uni-saarland.de/~mueller/lconfluence.ps.gz`

(2) Full abstraction for a recursively typed lambda calculus with parallel conditional,
revised Report 12/1993 of SFB 124, Informatik, Universität des Saarlandes,
arxiv:0806.1827,

(3) On Berry's conjectures about the stable order in PCF,
Logical Methods in Computer Science, 8(4:7):1-39, 2012, arxiv:1108.0556,

(4) From Sazonov's non-dcpo natural domains to closed directed-lub partial orders,
first version May 2016, arxiv:1605.01886.

(1) is purely syntactic work that is applied in (2), (2) and (3) are works in denotational semantics, and (4) is purely domain-theoretic. The common background of (2-4) is the semantic full abstraction problem.

This introduction first gives a conceptual and historical account of denotational semantics and the full abstraction problem, which is not in any sense complete, but only serves as a preparation for our themes, and then introduces the four chapters separately. Their whole content is covered.

0.1 PCF: its syntax and semantics (Scott, Plotkin)

Several fundamental concepts and inventions of computer science have their origin, their idea, in the work of logicians. Think of the concept of the program stored in memory, which was invented by Alan Turing for his universal machine, in clarifying the notion of computable function. Think of the notion of type, which was invented by Bertrand Russell for his logical foundation of mathematics, and later developed by Alonzo Church (lambda calculus, simple theory of types), Jean-Yves Girard (polymorphically typed lambda calculus) and Per Martin-Löf (dependent types), among others. The development in computer science started as a slow re-invention, though with different aims, of the concept of type, until the two communities finally noticed each other.

In the year 1969 (and before) the logician Dana Scott seeked for a mathematical theory of computation. At that time there were several suggestions of functional programming languages based on the untyped λ-calculus, and suggestions to translate other programming languages into the λ-calculus. Scott insisted on giving mathematical meaning to the expressions of λ-calculus, to overcome pure formalism. But he did not succeed at that time in giving a model of the untyped λ-calculus (he only succeeded later). So in 1969 he developed a type-theoretical alternative, a logic of computable functions (later abbreviated as LCF) based on type theory; and his paper "A type theoretical alternative to ISWIM, CUCH, OWHY" [Sco93] became the founding document of typed functional programming languages and denotational semantics. We cite from the introduction of this paper: "No matter how much wishful thinking we do, the theory of types is here to stay. There is *no other way* to make sense of the foundation of mathematics. ... My point is that formalism *without eventual interpretation* is in the end useless." This paper was first circulated privately, and only published in 1993, but it had a profound influence, particularly on the theoretical (and practical) work of Robin Milner (who implemented LCF and the functional programming language ML) and Gordon Plotkin.

LCF is a system to reason about computable functions, whose syntax comprises types, expressions and logical formulas. The part of types and expressions has been extracted as the language PCF (programming language for computable functions) by Plotkin [Plo77], who also replaced Scott's combinator expressions by lambda expressions. It is only this part that will interest us. Scott did not intend to define a programming language, only a logical calculus, but Plotkin has added the reduction rules to PCF. Scott and Plotkin have two ground types o (booleans) and ι (the type of individuals, specialized to integers $0, 1, 2, \ldots$). Here we present the syntax of our simplified PCF (without booleans, which can be coded by integers), as we use it in our third paper:

The **types** are formed by ι (integer) and function types $\sigma \to \tau$ for types σ and τ.

The typed **constants** are:
$\mathsf{0}, \mathsf{1}, \mathsf{2}, \ldots : \iota$, the integers;
$\mathsf{suc}, \mathsf{pre} \colon \iota \to \iota$, successor and predecessor function;
$\mathsf{if}\ \_\ \mathsf{then}\ \_\ \mathsf{else}\ \_ \colon \iota \to \iota \to \iota \to \iota$, this conditional tests if the first argument is $\mathsf{0}$.

The PCF **terms** comprise the constants and the typed constructs by the following rules:
$\perp^\sigma \colon \sigma$ for any type σ, the undefined term.
$x^\sigma \colon \sigma$ for any variable x^σ.
If $M \colon \tau$, then $\lambda x^\sigma.M \colon \sigma \to \tau$, lambda abstraction.
If $M \colon \sigma \to \tau$ and $N \colon \sigma$, then $MN \colon \tau$, function application.
If $M \colon \sigma \to \sigma$, then $\mathsf{Y}M \colon \sigma$, Y is the fixpoint operator.

The **reduction rules** are (where n is a variable for integer constants):
$(\lambda x.M)N \to M[x := N]$, the usual β-reduction;
$\mathsf{Y}M \to M(\mathsf{Y}M)$;
$\mathsf{suc}\, n \to (n + 1)$;

$\mathsf{pre}\, n \to (n-1)$, for $n \geq \mathsf{1}$;
$\mathsf{if}\, \mathsf{0}\, \mathsf{then}\, M\, \mathsf{else}\, N \to M$;
$\mathsf{if}\, n\, \mathsf{then}\, M\, \mathsf{else}\, N \to N$, for $n \geq \mathsf{1}$.
The *reduction relation* $\to$ on terms is one step of reduction by these rules in any term context. It is confluent. $\to^*$ is the reflexive, transitive closure of $\to$.

Please note here that the conditional treats $\mathsf{0}$ as the value true, and all other integers as false, as it is natural that the $\mathsf{0}$-branch comes before the n-branch. Think of the conditional as a kind of case-operator.

Scott's main contribution was the semantic interpretation of his calculus. For every type τ there is a domain D^τ of elements of this type. We retell the conceptual development from his paper: The first insight was that we have to deal with partial functions because of possible non-termination, and this can be done in a frame of total functions where we adjoin the undefined bottom element $\perp$ to the ground domain D^ι, and an order $\sqsubseteq$ on D^ι with $\perp \sqsubseteq n$ for every integer n. Every domain carries such a partial order with the meaning "less defined or equal". Every function $f\colon D^\sigma \to D^\tau$ in the domain $D^{\sigma\to\tau}$ must be monotonic with respect to the orders $\sqsubseteq$ on D^σ resp. D^τ.

When we come to the domain $D^{(\iota\to\iota)\to\iota}$ of functionals, we see that the restriction to monotonic functions is not enough: A functional h of this type must be continuous in the sense that the computation of one result hf for an argument function f depends only on finitely many values fx. This can be ensured with the chosen definition of continuity: For *every* ascending chain $(f_n)_{n\geq 0}$, $f_n \sqsubseteq f_{n+1}$, in $D^{\iota\to\iota}$ there exists a least upper bound (lub) $f = \bigsqcup_{n\geq 0} f_n$, and continuity of h should mean $hf = \bigsqcup_{n\geq 0} hf_n$. ($(f_n)$ may be a chain of finite pieces of f, the result hf must be already reached for one argument f_n.) We point to this definition as the important idealization (for *every* ascending chain). It turns out that with this definition of continuity, and with the definition of $D^{\sigma\to\tau}$ as the domain of all monotonic and continuous functions $h\colon D^\sigma \to D^\tau$, the domain $D^{\sigma\to\tau}$ has a lub for every ascending chain again, it is an ω-chain-complete partial order (ω-cpo). We call this the birth of the "tacit agreement" (or perhaps "dogma") of the completeness of semantic domains, as from this time on it was generally agreed that models of PCF have to consist of cpos, until a model emerged that is not complete, which we will see below.

In this framework we can define a function $[\![\ ]\!]$ that gives the semantics of terms of PCF [Sco93, Plo77]. For every PCF-term $M\colon \sigma$ and environment ρ (a map from every variable x^τ to some $\rho(x^\tau) \in D^\tau$) it is $[\![M]\!]\rho \in D^\sigma$. It is defined in the obvious way on constants, and by recursion on the language constructs by the equations:

$$\begin{aligned}
[\![\lambda x.M]\!]\rho\, d &= [\![M]\!]\rho[x := d], \text{ where } \rho[x := d] \text{ is like } \rho \text{, but maps } x \text{ to } d\\
[\![MN]\!]\rho &= ([\![M]\!]\rho)([\![N]\!]\rho)\\
[\![\mathsf{Y}M]\!]\rho &= \bigsqcup_{n\geq 0}(([\![M]\!]\rho)^n\perp)
\end{aligned}$$

Here we see another reason for the idealization of complete domains: It is guaranteed in cpos that the lub of the ascending chain $([\![M]\!]\rho)^n\perp$ in the interpretation of the fixpoint combinator Y exists. The reason for the tacit agreement of completeness was that it works and makes life easy.

In domain theory we generalize ω-chains to *directed sets*, which are non-empty subsets S such that for $a, b \in S$ there is $c \in S$ with $a, b \sqsubseteq c$. A *directed complete partial order (dcpo)* is a partial order with a lub for every directed subset. A *complete partial order (cpo)* is a dcpo with a least element $\bot$.

0.2 Semantic adequacy and full abstraction (Milner, Plotkin)

The question arises: what is the relation between the syntactic reduction relation and the model, the semantic function? Is there some coincidence? Is the denotational semantics "correct" w.r.t. the "operational semantics", the reduction? (The syntax is taken here as the measure of correctness.) The two notions in this context, adequacy and full abstraction, first appeared in the paper "Processes: a mathematical model of computing agents" of Robin Milner [Mil75] and were transferred to PCF by Milner [Mil77] and Gordon Plotkin [Plo77].

First we must decide what kind of syntactic terms we want to *observe*. We take those terms that are so simple that their syntactic form coincides with their semantic form. These are the integer constants n. We define *programs* to be terms of integer type that are closed, i.e. have no free variables. *Adequacy* is this property:

$$\text{For any program } M \text{ and integer constant } n\text{: } M \to^* n \iff [\![M]\!]\bot = n$$

(Here $\bot$ is the environment that maps all variables to $\bot$.)
This was proved by Plotkin [Plo77]. The direction $\Longrightarrow$ is easy to prove, as reduction preserves the meaning. The direction $\Longleftarrow$ is difficult to prove, it requires computability predicates on higher types like those employed in normalization proofs.

So far syntax and semantics coincide. What about the observation of higher type terms? The trick employed here is: let the language do the observation, we observe the terms through syntactic contexts. A context $C[\ ]$ is a term with a hole in it, in this hole a term M (of appropriate type) can be inserted to get a term $C[M]$. (Note that $C[\ ]$ can capture free variables of M.) We observe terms through contexts that yield (integer) programs. We define the *operational preorder* $\sqsubseteq_{op}$ on terms of the same type as:

$M \sqsubseteq_{op} N$ (M is *operationally less defined than* N) iff
$P[M] \to^* n$ implies $P[N] \to^* n$ for all contexts $P[\ ]$ such that $P[M]$ and $P[N]$ are both programs.
The *operational equivalence* is defined as: $M \cong N$ iff $M \sqsubseteq_{op} N$ and $N \sqsubseteq_{op} M$.

$M \cong N$ means that M can be replaced by N in any (program) context to get the same (integer) result. As adequacy is valid, we can express $\sqsubseteq_{op}$ semantically:
$M \sqsubseteq_{op} N$ iff $[\![P[M]]\!]\bot \sqsubseteq [\![P[N]]\!]\bot$ for all contexts $P[\ ]$ such that $P[M]$ and $P[N]$ are both programs.

Full abstraction is this property:

$$\text{For all terms } M, N \text{ of the same type: } M \sqsubseteq_{op} N \iff \text{for all environments } \rho\text{: } [\![M]\!]\rho \sqsubseteq [\![N]\!]\rho.$$

This means the full identity of the operational and semantic preorder on terms. The direction $\Longleftarrow$ follows directly from adequacy, so it is fulfilled for the Scott semantics.

The direction $\Longrightarrow$ is generally problematic and not fulfilled by the Scott semantics. This was discovered by Plotkin [Plo77]. We explain here the reasons informally, they are

proved in [Plo77]. In the semantic domains $D^{\iota\to\iota\to\iota}$ there are many functions that do not appear in the syntax of PCF, they cannot be defined in PCF.
One of them is the "parallel or" por with
$\mathrm{por}\,0\,n = 0$, $\mathrm{por}\,n\,0 = 0$ for all n,
$\mathrm{por}\,n\,m = 1$ for $n, m \neq \bot$, $n, m \neq 0$,
and $\mathrm{por}\,nm = \bot$ otherwise.
This function cannot be defined in PCF, as por has to evaluate its two arguments fairly parallel, and this cannot be done in a sequential language like PCF.

There are two terms $M, N\colon (\iota \to \iota \to \iota) \to \iota$ that can be distinguished semantically by providing them with the argument por.

$$M = \lambda f.\, \mathsf{if}\ f0\bot\ \mathsf{then}\ (\mathsf{if}\ f\bot 0\ \mathsf{then}\ (\mathsf{if}\ f11\ \mathsf{then}\ \bot\ \mathsf{else}\ 0)\ \mathsf{else}\ \bot)\ \mathsf{else}\ \bot,$$
$$N = \lambda f.\, \mathsf{if}\ f0\bot\ \mathsf{then}\ (\mathsf{if}\ f\bot 0\ \mathsf{then}\ (\mathsf{if}\ f11\ \mathsf{then}\ \bot\ \mathsf{else}\ 1)\ \mathsf{else}\ \bot)\ \mathsf{else}\ \bot.$$

It is $([\![M]\!]\bot)\,\mathrm{por} = 0$ and $([\![N]\!]\bot)\,\mathrm{por} = 1$, so $[\![M]\!]\bot \neq [\![N]\!]\bot$ in Scott's semantics.
But M and N cannot be distinguished operationally by $\sqsubseteq_{op}$, it is $M \cong N$, because to distinguish M and N we must provide an argument f with $f0\bot = f\bot 0 = 0$ and $f11 = 1$, which needs parallel evaluation of the arguments of f.

Plotkin has defined reduction rules for a new parallel conditional $\mathsf{pif}\colon \iota \to \iota \to \iota \to \iota$:
$\mathsf{pif}\,0\,MN \to M$, $\mathsf{pif}\,n\,MN \to N$, for integer constant $n \geq 1$, $\mathsf{pif}\,Mn\,n \to n$, for integer constant n.
pif has to evaluate its arguments fairly parallel and reduces when its first argument is $\mathsf{0}$ or $n \geq \mathsf{1}$, or when its second and third arguments are reduced to the same integer constant n.
Plotkin showed that Scott's semantics is fully abstract w.r.t. PCF extended with pif.

This is one way to "solve" the full abstraction problem: to extend the language by new operators. The other way is to restrict the elements of the model. This way was elaborated by Milner in 1977 [Mil77]. According to the established "dogma" of the completeness of semantic domains, the model had to be a cpo. He constructed the unique order-extensional fully abstract cpo-model of PCF out of the terms of an SKI-combinator calculus for PCF (also Scott had used combinators). (Order-extensional means $(\forall x.\ fx \sqsubseteq gx) \implies f \sqsubseteq g$.) Later Gérard Berry constructed this model out of proper λ-terms [Ber79]. In our third paper we show this construction: One defines the (semantically) finite terms of PCF, ordered by $\sqsubseteq_{op}$, forms their equivalence classes under $\cong$ to get the finite elements of the model, the model is the ideal completion of the partial order of the finite elements.

This is a model of syntactic entities. The problem to construct a mathematical fully abstract model of PCF that does not use the syntax of terms (the "full abstraction problem") was the driving force of the subsequent developments.

0.3 Stable functions and stable order (Berry)

In 1979 Gérard Berry published his PhD thesis [Ber79] with the translated title "Fully abstract and stable models of typed lambda calculi", where he tried the second way, to restrict the Scott model to come closer to the fully abstract model. He introduced the notions of stable function and stable order, resp. of conditionally multiplicative (cm) function and cm-order, where cm is more general than stable. We do not here give Berry's official definitions, but introduce stable functions in the context of PCF-functions and the stable order by a shortcut to tokens and traces.

The parallel or function should be ruled out, so what is semantically "wrong" with por? It is $\mathrm{por}\,0\,0 = 0$, and $\mathrm{por}\,0\bot = 0$, $\mathrm{por}\,\bot 0 = 0$. This means: To compute $\mathrm{por}\,0\,0$, there are two *different minimal parts* of the argument vector, namely $0\bot$ and $\bot 0$ that would be sufficient to be computed. ("Parts" are meant in the sense of the semantic order.) A *stable function* is a function f where this is not possible; i.e. for every argument vector $\vec{x}$ with $f\vec{x} = j$ for some integer j, there is one *least* argument vector $\vec{y} \leq \vec{x}$ with $f\vec{y} = j$. (Here $\leq$ is the stable order, for the domain D^ι it is identical to the extensional order $\sqsubseteq$. Here $\leq$ is taken component-wise on vectors.) The tuple $\vec{y}{\mapsto}j = y_1{\mapsto}\dots{\mapsto}y_n{\mapsto}j$ is said to be a *token* of the stable function f, it is some kind of atomic part of f.

The *trace* of f, $\mathcal{T}(f)$, is the set of all tokens of f. The trace is a succinct, irredundant way to notate a value table of a stable function. (There are many other notions called "trace" in mathematics and computer science, our trace has nothing to do with them. Girard calls our trace "skeleton".) A stable function is finite (or compact, in domain-theoretical sense) iff its trace is finite.

The stable order on stable functions can be defined as $f \leq g$ iff $\mathcal{T}(f) \subseteq \mathcal{T}(g)$, i.e. the "value table" of f is contained in that of g. If $f \leq g$, then $f \sqsubseteq g$, but the reverse is not true, the stable order takes into account the way the functions are computed. The y_i in the token $y_1{\mapsto}\dots{\mapsto}y_n{\mapsto}j$ will be recursively notated by traces.

Here are examples of PCF-functions with their traces (for closed terms M we abbreviate $[\![M]\!]\bot$ as $[\![M]\!]$):

$$\begin{array}{ll}
F = \lambda x.\,\mathsf{if}\ x\ \mathsf{then}\ 0\ \mathsf{else}\ \bot & \mathcal{T}[\![F]\!] = \{\{0\}{\mapsto}0\} \\
G = \lambda x.\,\mathsf{if}\ x\ \mathsf{then}\ 0\ \mathsf{else}\ (\mathsf{if}\ \mathsf{pre}\,x\ \mathsf{then}\ 0\ \mathsf{else}\ \bot) & \mathcal{T}[\![G]\!] = \{\{0\}{\mapsto}0, \{1\}{\mapsto}0\} \\
H = \lambda x.0 & \mathcal{T}[\![H]\!] = \{\emptyset{\mapsto}0\},\ \text{also written } \{\bot{\mapsto}0\}.
\end{array}$$

It is $[\![H]\!]n = 0$ for all integers n, but this is redundant and not notated in the trace, because of the minimality of the argument. $[\![H]\!]$ is a finite element.
It is $[\![F]\!] \sqsubseteq [\![G]\!]$ and $[\![F]\!] \leq [\![G]\!]$.
It is $[\![F]\!] \sqsubseteq [\![H]\!]$, but not $[\![F]\!] \leq [\![H]\!]$.
It is $[\![G]\!] \sqsubseteq [\![H]\!]$, but not $[\![G]\!] \leq [\![H]\!]$.
And here a function $K \colon (\iota \to \iota) \to \iota$ of higher type:
$K = \lambda f.\,\mathsf{if}\ f0\ \mathsf{then}\ 0\ \mathsf{else}\ \bot$, $\mathcal{T}[\![K]\!] = \{\{\{0\}{\mapsto}0\}{\mapsto}0\}$.

All functions of PCF are stable, but there are stable functions that cannot be defined in PCF, so that not all unwanted functions can be excluded by stability.

Berry has constructed the bidomain model of PCF whose domains carry two cpo orders $\sqsubseteq$ (the normal extensional order) and $\leq$ (the stable order), and whose functions are continuous w.r.t. both orders and stable w.r.t. the stable order. By our shortcut to traces we can give a simple construction of the bidomain model, followed by a simple construction of the fully abstract cpo model with the stable order. Note that these constructions are not the standard ones given by Berry, they appear only in this introduction. They are formalizations of our explanations above, you can jump to the mark $\rightarrow$ below if you do not want to see them.

For every type σ there is a bidomain with orders $\sqsubseteq$ and $\leq$, a set T^σ of tokens, and a map $\mathcal{T}\colon D^\sigma \to \text{powerset}(T^\sigma)$, such that $a \leq b$ iff $\mathcal{T}(a) \subseteq \mathcal{T}(b)$. The construction is inductive on the type σ:
$D^\iota = \{\bot, 0, 1, 2, \ldots\}$, $\bot \sqsubseteq n$, $\bot \leq n$ for all integers n, take the reflexive closure.
$T^\iota = \{0, 1, 2, \ldots\}$, $\mathcal{T}(\bot) = \emptyset$, $\mathcal{T}(n) = \{n\}$ for all integers n.
$D^{\sigma_1 \to \ldots \to \sigma_n \to \iota}$ is the set of all functions $f\colon D^{\sigma_1} \to \ldots \to D^{\sigma_n} \to D^\iota$ that are monotonic w.r.t. $\sqsubseteq$ and such that for every argument vector $\vec{x} = (x_1, \ldots, x_n)$ with $x_i \in D^{\sigma_i}$ and $f\vec{x} = j$ for some integer j, there is a least vector $\vec{y} \leq \vec{x}$ with $f\vec{y} = j$, and the trace (of each component) of $\vec{y}$ is finite. (The last ensures continuity).
The order $\sqsubseteq$ on $D^{\sigma_1 \to \ldots \to \sigma_n \to \iota}$ is: $f \sqsubseteq g$ iff $f\vec{x} \sqsubseteq g\vec{x}$ for all $\vec{x}$.
$T^{\sigma_1 \to \ldots \to \sigma_n \to \iota}$ is the set of all $t_1 \mapsto \ldots \mapsto t_n \mapsto j$ with $t_i = \mathcal{T}(x_i)$ for some $x_i \in D^{\sigma_i}$ and j integer.
For $f \in D^{\sigma_1 \to \ldots \to \sigma_n \to \iota}$, $\mathcal{T}(f)$ is the set of all $\mathcal{T}(x_1) \mapsto \ldots \mapsto \mathcal{T}(x_n) \mapsto j$ with $f\vec{x} = j$ and $\vec{x}$ is $\leq$-minimal with this property.

Berry has also augmented the fully abstract cpo-model of PCF with the stable order $\leq$. We can give a construction of the finite elements of this model with traces when we change the last construction so that only functions f are taken that are denotations of terms and have *finite* traces. (Their monotonicity and stability is guaranteed by being denotations.) The construction is the same, but we add a map "term" that maps a finite $d \in D^\sigma$ to a closed term term$(d)\colon \sigma$ with the same semantics, and change the construction of the D^σ:

$D^{\sigma_1 \to \ldots \to \sigma_n \to \iota}$ is the set of all functions $f\colon D^{\sigma_1} \to \ldots \to D^{\sigma_n} \to D^\iota$ with a closed term $M\colon \sigma_1 \to \ldots \to \sigma_n \to \iota$ such that (1) f and M have the same semantics, i.e. for all (finite) argument vectors $\vec{x} = (x_1, \ldots, x_n)$ with $x_i \in D^{\sigma_i}$ it is $f\vec{x} = j$ iff $M\,\text{term}(x_1) \ldots \text{term}(x_n) \to^* j$, and such that (2) the set of (finite) argument vectors $\vec{x}$ with $f\vec{x} = j$ for some integer j and with $\vec{x}$ $\leq$-minimal with this property, is finite. It is chosen term$(f) = M$ for some such M.

Note that we have only constructed the *finite* elements of the fully abstract model, the last condition in the specification of f has insured this by demanding a finite trace of f. The whole model is constructed by ideal completion. Note that the construction given is a shortcut and not the standard one given by Berry. We will also see Berry's construction in Section 0.8 below.

$\to$ Berry defined the structure of bicpos $(D, \sqsubseteq, \leq, \bot)$ of domains with two orders, that fulfill some axioms that connect the orders, see Section 3.6. He proved that the fully abstract cpo-model consists of bicpos. He also defined the stronger bidomains. Both structures form cartesian closed categories. The bidomain model above consists, as is intended, of bidomains. Berry conjectured that the fully abstract model (with stable order) consists also of bidomains (Berry's first conjecture). The critical point was to prove that the stable order is bounded complete, i.e. that for a, b with $a, b \leq c$ for some c there is a lub $a \vee b$. We refute this conjecture in Chapter 3.

We now come to Berry's second conjecture about the stable order in PCF. For this we need the *syntactic order* $\prec$ on PCF-terms: For terms M, N of the same type it is $M \prec N$ if N results from the substitution of occurrences of $\bot$ in M by terms of appropriate type. E.g. in the examples above it is $F \prec G$.
The conjecture is this: The stable order of the fully abstract model has the syntactic order as its image: If $a \leq b$ in the model, for finite a and b, then there are closed normal form

terms A, B with $a = [\![A]\!]$, $b = [\![B]\!]$ and $A \prec B$.

Berry proved that this conjecture is true for first-order terms. And he proved the reverse for all types: If $A \prec B$, then $[\![A]\!] \leq [\![B]\!]$. E.g. in the examples above it is $F \prec G$ and $[\![F]\!] \leq [\![G]\!]$. But it is not $[\![F]\!] \leq [\![H]\!]$, so there are no terms F', H' with $[\![F']\!] = [\![F]\!]$, $[\![H']\!] = [\![H]\!]$ and $F' \prec H'$.

We refute also the second conjecture in Chapter 3.

0.4 Game model, sequential functionals and natural domains (Abramsky et al., Hyland/Ong, Normann, Sazonov)

Berry tried to restrict the Scott model to come closer to the fully abstract model and so discovered the concept of stability, but he could not reach the fully abstract model. The widely accepted solution of the full abstraction problem was the game semantics after 1990 [AJM00, HO00, Nic94]. In game semantics a term of PCF is modeled by a strategy of a game, i.e. by a process that performs a dialogue of questions and answers with the environment, the opponent. These strategies are still intensional; the fully abstract model is formed by identifying extensionally equivalent strategies.

The game models are mathematical and independent of the λ-calculus syntax, but their strategies can also be represented by some canonical λ-terms; in [AJM00, section 3.2] they were called (finite and infinite) "evaluation trees", in [HO00, section 7.3] "finite canonical forms" that correspond to compact innocent strategies, and in [AC98, section 6.6] "PCF Böhm trees". In Chapter 3 we call them *game terms*. They are the well-typed terms produced by the grammar:

$$\begin{aligned} M, N ::= {} & \bot^\sigma, \ \sigma \text{ any type} \\ & \lambda x_1 \dots x_n.m, \ m \text{ integer constant}, \ n \geq 0 \\ & \lambda x_1 \dots x_n.\,\mathsf{case}_i(yM_1 \dots M_m)N_0 \dots N_i, \ y \text{ variable}, \ n, m, i \geq 0 \end{aligned}$$

Here we have a new language construct: if $M, N_0, \dots, N_i \colon \iota$, then $\mathsf{case}_i\, MN_0 \dots N_i \colon \iota$ for $i \geq 0$, with the reduction rule:

$$\mathsf{case}_i\, nN_0 \dots N_i \to N_n, \ \text{for } 0 \leq n \leq i$$

Game terms are semantically finite, they denote all finite elements of the fully abstract cpo-model. Infinite game terms are limits of ascending chains $M_1 \prec M_2 \prec \dots$ of (finite) game terms. They denote certain ideals of finite elements of the fully abstract cpo-model. These are called the *sequential functionals*.

It was an open problem whether the sequential functionals cover the whole fully abstract cpo-model, or whether their domains are incomplete. This problem was solved by Dag Normann [Nor06]: Their domains are not cpos, i.e. there are directed sets that have no lub. The example given by Normann is in type 3 and rather sophisticated. Then Vladimir Sazonov made a first attempt to build a general theory for these non-cpo domains [Saz07, Saz09]. His main insight was that functions are continuous only with respect to certain lubs of directed sets that he calls "natural lubs"; these are the hereditarily pointwise lubs in PCF. He defines an abstract structure of "natural domains" [Saz09] as a partial order with an operator that designates certain lubs of (general, not only directed) subsets as "natural lubs" fulfilling some axioms. He shows that the category of natural domains and functions that are continuous w.r.t. the natural directed lubs is a cartesian closed category (ccc). So the tacit agreement of the completeness of semantic domains is finally falling, because there emerged an accepted model (the game model) that is not directed complete.

0.5 Ch. 2: Syntax of a recursively typed lambda calculus with parallel conditional

The preceding sections were the historical account, now we introduce the chapters of our thesis. First we come to the syntax of Chapter 2.

We are interested in full abstraction for a programming language more elaborated than PCF, namely a recursively typed λ-calculus. We want to achieve full abstraction of the normal Scott-like semantics by adding a parallel conditional, like Plotkin did for PCF. Here we introduce the syntax of the language. The type expressions are formed from type variables t and the following constructs:

$\sigma + \tau$	is the separated sum of σ and τ,
$\sigma \times \tau$	is the cartesian separated product of σ and τ,
$\sigma \to \tau$	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 for the undefined type; it has the same meaning as $\mu t.t$. The corresponding domain has just one element $\bot$.

We regard type expressions as equivalent ($\approx$) when they have the same unfolding as regular trees. The term formation is by the usual rules of typed λ-calculus (see above for PCF) and the rule: If $M\colon \sigma$ and $\sigma \approx \tau$, then $M\colon \tau$.

For every type σ we can define a fixpoint combinator:

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

The typed constants are given by:

$\mathsf{0}_{\sigma,\tau}$:	$\sigma \to (\sigma+\tau)$, also called "inleft" in the literature
$\mathsf{1}_{\sigma,\tau}$:	$\tau \to (\sigma+\tau)$, also called "inright"
$\mathsf{case}_{\sigma,\tau,\rho}$:	$(\sigma+\tau)\to(\sigma\to\rho)\to(\tau\to\rho)\to\rho$, sequential conditional
$\mathsf{pcase}_{\sigma,\tau,\rho}$:	$(\sigma+\tau)\to\rho\to\rho\to\rho$, parallel conditional. Note the type different from case's type.
$\mathsf{pair}_{\sigma,\tau}$:	$\sigma\to\tau\to(\sigma\times\tau)$, $\mathsf{pair}\,x\,y$ is also written (x,y)
$\mathsf{fst}_{\sigma,\tau}$:	$(\sigma\times\tau)\to\sigma$
$\mathsf{snd}_{\sigma,\tau}$:	$(\sigma\times\tau)\to\tau$
Ω_σ :	σ, the canonical undefined term of type σ.

The reduction rules for the reduction relation $\rightarrow$ are the usual β-reduction, rules for the reduction in any context and the rules for the constants:

(case0)	$\mathsf{case}\ (\mathsf{0}x)\ y\ z$	$\rightarrow$	$y\ x$
(case1)	$\mathsf{case}\ (\mathsf{1}x)\ y\ z$	$\rightarrow$	$z\ x$
(pair1)	$\mathsf{fst}\ (\mathsf{pair}\ x\ y)$	$\rightarrow$	x
(pair2)	$\mathsf{snd}\ (\mathsf{pair}\ x\ y)$	$\rightarrow$	y
(pcase0)	$\mathsf{pcase}\ (\mathsf{0}x)\ y\ z$	$\rightarrow$	y
(pcase1)	$\mathsf{pcase}\ (\mathsf{1}x)\ y\ z$	$\rightarrow$	z
(pcase00)	$\mathsf{pcase}_{\sigma,\tau,\rho_0+\rho_1}\ x\ (\mathsf{0}y)\ (\mathsf{0}z)$	$\rightarrow$	$\mathsf{0}\ (\mathsf{pcase}_{\sigma,\tau,\rho_0}\ x\ y\ z)$
(pcase11)	$\mathsf{pcase}_{\sigma,\tau,\rho_0+\rho_1}\ x\ (\mathsf{1}y)\ (\mathsf{1}z)$	$\rightarrow$	$\mathsf{1}\ (\mathsf{pcase}_{\sigma,\tau,\rho_1}\ x\ y\ z)$
(pcase$\times\times$)	$\mathsf{pcase}_{\sigma,\tau,\rho_1\times\rho_2}\ x\ (y_1, y_2)\ (z_1, z_2)$	$\rightarrow$	$(\mathsf{pcase}_{\sigma,\tau,\rho_1}\ x\ y_1\ z_1, \mathsf{pcase}_{\sigma,\tau,\rho_2}\ x\ y_2\ z_2)$
(pcase $\rightarrow$)	$(\mathsf{pcase}_{\sigma,\tau,\rho_1\rightarrow\rho_2}\ x\ y\ z)\ w$	$\rightarrow$	$\mathsf{pcase}_{\sigma,\tau,\rho_2}\ x\ (y\ w)\ (z\ w)$

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

The parallel conditional pcase works this way: If $\mathsf{0}$ or $\mathsf{1}$ appears on top of its first argument, then the second resp. third argument can be chosen. Parallel to the first argument, the second and third are evaluated and any constant information, $\mathsf{0}$ or $\mathsf{1}$ or $(\_,\_)$, that appears on top of both of them, can be passed outside.

The problem is to prove confluence of the reduction $\rightarrow$:
For any typed term M with $N \leftarrow^* M \rightarrow^* P$ there is a term Q with $N \rightarrow^* Q \leftarrow^* P$. Confluence ensures determinacy, the uniqueness of normal forms.

We define the *applicative terms* to be the terms without any λ-abstraction, and we call the rules for the constants an *applicative term rewriting system (ATRS)*, as its terms are all applicative. The confluence of our ATRS on applicative terms can be proved from its local confluence (if $N \leftarrow M \rightarrow P$, then there is Q with $N \rightarrow^* Q \leftarrow^* P$) and termination. The local confluence can be proved from the convergence of critical pairs. There are 8 critical pairs, i.e. overlaps of left sides, e.g. $\mathsf{0}y \leftarrow \mathsf{pcase}(\mathsf{0}x)(\mathsf{0}y)(\mathsf{0}z) \rightarrow \mathsf{0}(\mathsf{pcase}(\mathsf{0}x)yz)$. It remains to show the confluence of the combination of the ATRS with β-reduction. This takes us to Chapter 1.

0.6 Ch. 1: Confluence of the lambda calculus with left-linear algebraic rewriting

At the time there was no result in the literature that would provide an easy check of the confluence of β-reduction with our applicative term rewriting system (ATRS) above. There were only results for systems without overlapping rules, or results for systems that are normalizing (terminating, also for β-reduction). Moreover, the prevailing concept of algebraic rewriting excluded variable-applying (i.e. subterms xM) entirely, also in right sides of rules. We had to find a new theorem.

We restrict our ATRS to be any set of pairs $\langle L \rightarrow R\rangle$ (rules) of applicative terms, where L is no variable and all variables of R appear in L.
We call the ATRS *left-linear* if in each rule a variable occurs at most once in the left side.

We call the ATRS *not variable-applying* if there is no *left* side with a subterm of the form xM.

We introduce the notion of $\to$-*closed* subset T of λ-terms which has to fulfill for every $M \in T$:
(1) If $M \to M'$, then $M' \in T$,
(2) Every subterm of M is in T,
(3) Every occurrence of an abstraction in M can be replaced by some variable not in M, so that the new term is in T.
Typically, T is some subset of typed terms.

Our **theorem** is this:
For every left-linear, not variable-applying ATRS with reduction relation $\to$ and every $\to$-closed set T of terms:
$\to$ is confluent on the applicative terms of $T \iff \to$ is confluent on T.

The proof is like this: The reduction $\to$ is split into two parts, β-reduction $\to_\beta$ and the reduction by the ATRS-rules $\to_A$. $\to_\beta$ is known to be confluent on T. We must also prove the confluence of $\to_A$ on all terms of T (not only the applicative terms), which requires some bureaucracy. Then we prove that $\to_\beta$ and $\to_A$ commute, i.e. if $N \leftarrow_\beta^* M \to_A^\star P$, then there is Q with $N \to_A^\star Q \leftarrow_\beta^* P$. For this we need a parallel reduction $\to_{\beta p}$ on a disjoint set of β-redexes, and we have to prove that if $N \leftarrow_{\beta p} M \to_A P$, then there is Q with $N \to_A^\star Q \leftarrow_{\beta p} P$.

Our result is also in the handbook "Term rewriting systems" (ed. Terese) [Bet03, Theorem 10.4.15, page 576].

0.7 Ch. 2: Full abstraction for a recursively typed lambda calculus with parallel conditional

We have described the syntax of our recursively typed lambda calculus in section 0.5 and have proved its confluence. We now want to give a Scott-like semantics of the calculus, which should be fully abstract because of the parallel conditional, like it was for PCF extended by a parallel conditional.

But we have to give domains for recursive types: a recursive type is unfolded to a regular (possibly infinite) type tree, for each finite approximation of the type tree (up to some level) there is a (finite) domain, and we get a chain of domains where each domain is embedded in its successor. The limit of this chain is the desired domain of the recursive type. To make this construction simple (the embedding will be a containment) we use a concrete representation of domains by "prime systems". (The concrete representation will also be necessary to prove full abstraction.) The idea is from [LW91], where the concrete representation of domains by "information systems" is used to solve recursive domain equations.

Prime systems are a specialization of information systems that is sufficient for our needs (they were first introduced in [NPW81] under the name "event structures"):
A *prime system* $\mathcal{A} = (A, \uparrow, \leq)$ consists of a set A (the *primes*), 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 $a \uparrow c$.

The primes are similar to the tokens we have seen in the trace of stable functions. They are atomic, indivisible pieces 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 may be valid of the same element, they are consistent. There is a domain $|\mathcal{A}|$ for a prime system whose elements are the subsets $d \subseteq A$ that are downward closed: if $a \leq b$ and $b \in d$, then $a \in d$, and consistent: $a \uparrow b$ for all $a, b \in d$.

Example: The cartesian product $\mathcal{A}_0 \times \mathcal{A}_1$ of two prime systems has as primes: all $(0, a_0)$ with $a_0 \in A_0$ for the left component of pairs, and all $(1, a_1)$ with $a_1 \in A_1$ for the right component. The elements of the domain $|\mathcal{A}_0 \times \mathcal{A}_1|$ are all sets d of primes with $d_0 = \{\, a_0 \mid (0, a_0) \in d \,\} \in |\mathcal{A}_0|$ and $d_1 = \{\, a_1 \mid (1, a_1) \in d \,\} \in |\mathcal{A}_1|$. d represents the pair (d_0, d_1).

The class of prime systems has the structure of a cpo under the substructure containment $\trianglelefteq$, and our type constructors $+, \times, \rightarrow$ are continuous w.r.t. this cpo. This means that we can give a semantics of the recursive types as described above, as the union of a $\trianglelefteq$-chain of finite domains; a prime at one level of the domain is contained in all following levels. And we give a Scott-like semantics $\mathcal{S}[\![M]\!]\varepsilon$ of terms M with equations like those given for PCF above, where we have to switch between concrete representations in prime systems and the more abstract ones in the semantic domains.

We first prove an approximation theorem for the semantics, which is stronger than adequacy. A normal form A is an *approximation* of a term M if there is a reduct N of M such that $A \prec N'$ for all reducts N' of N. (Here $\prec$ is the syntactic order that we have seen for PCF.) The limit of all approximations of M can be seen as the Böhm tree of M. The approximation theorem says that the semantics of a term is the limit of the semantics of its approximations. We prove it with the method of inclusive predicates (like computability predicates) of [MP87], which we adapt here for the representation by prime systems.

The more interesting part of the paper is the proof of full abstraction. Again we first have to define our notion of observation and program. We choose to observe the values 0 and 1 of type $\mathsf{bool} = \mathsf{void} + \mathsf{void}$, so our programs are the closed terms of type bool. Then full abstraction means: For all terms M, N of the same type:
$\mathcal{S}[\![M]\!]\varepsilon \subseteq \mathcal{S}[\![N]\!]\varepsilon$ for all environments $\varepsilon \iff$ for all contexts $C[\;]$ with $C[M], C[N]$ programs it is $C[M] \rightarrow^* b \Longrightarrow C[N] \rightarrow^* b$ (for $b \in \{0, 1\}$).

The direction $\Longrightarrow$ is again proved from adequacy, which follows from the approximation theorem. The direction $\Longleftarrow$ is the difficult part, it is based (as for PCF) on a definability lemma: For all finite elements d of a semantic domain there is a closed term M with $\mathcal{S}[\![M]\!]\bot = d$.

The proof is much more difficult than for PCF, because of the recursive type structure. It uses expressions that are a mix of syntactic and semantic parts, namely *conditioned primes* $C \rightarrow a$, where C is an (open) term of type bool, the condition, and a is a prime. In the course of the construction the condition C is used to accumulate a term that checks function arguments. The intuitive meaning of $C \rightarrow a$ is: output the prime a for every environment ε with $\mathcal{S}[\![C]\!]\varepsilon = 0$.

We also prove that the parallel conditional and a parallel and-function can be defined from each other.

0.8 Ch. 3: On Berry's conjectures about the stable order in PCF

In this paper we solve the two problems of Berry (see Section 0.3) that were open since 1979. It has three sections of preliminaries: 3.2 Syntax of PCF, 3.3 Semantics of PCF: non-complete partial order f-models, 3.4 Game terms.

In Section 3.3 we generalize Milner's [Mil77] and Berry's [Ber79] construction of the fully abstract cpo-model of PCF to a whole spectrum of not necessarily complete fully abstract models that encompass the standard cpo-model as well as the new model of sequential functionals (the game model, see section 0.4) and the model that consists just of equivalence classes of PCF-terms themselves. We call these models f-models, "f" means: based on finite elements.

The construction of this base of finite elements is the same as in Berry [Ber79], and most phenomena of the Berry conjectures happen only in this base. We have the terms of *finite projections* $\Psi_i^\sigma \colon \sigma \to \sigma$ on type σ of grade i. When applied to a closed term $M \colon \sigma$, the function term Ψ_i^σ serves as a "filter" that lets only pass integer values $\leq i$ as input to M or output from M. $\Psi_i^\sigma M$ is a (semantically) *finite term*, and all the equivalence classes $[\Psi_i^\sigma M]_{op}$ of the finite terms under the operational equivalence $\cong$ are the *finite elements* of our models.

We obtain infinite elements by forming ideals under the order $\sqsubseteq_{op}$: an ideal is here a set S of finite elements of some type σ that is non-empty such that for two elements of S there is an upper bound in S and such that for every element of S all lower ones are in S.

An f-model is given by a set of ideals D^σ for every type σ, ordered by inclusion $\subseteq$ written $\sqsubseteq$, such that application is defined: for $f \in D^{\sigma\to\tau}$, $d \in D^\sigma$ it is $fd \in D^\tau$, and such that every closed term $M \colon \sigma$ has its denotation in D^σ. To these domains Sazonov's theory (Section 0.4) applies: functions must only be continuous w.r.t. special lubs of directed sets that we designate as "f-lubs". A directed set $S \subseteq D^\sigma$ has the *f-lub* $s \in D^\sigma$, $S \to s$, if s is the set-theoretical union of S. We can define a semantic map $[\![\ \ ]\!]$ in f-models and prove that it fulfills the equations of section 0.1. Every f-model is fully abstract for PCF.

Berry had defined the stable order $\leq$ in the fully abstract cpo-model of PCF. We can do the same in every f-model. The stable order on the domain D^ι is defined as the extensional order. For $f, g \in D^{\sigma\to\tau}$ it is $f \leq g$ if for all $x, y \in D^\sigma$: $x \leq y \Longrightarrow fx = fy \sqcap gx$. This is the "official" definition of the stable order. The traces can be defined from this order. We have already seen a shortcut construction (by traces) of the base of finite elements with the stable order in Section 0.3, which deviates from Berry's construction, but gives the same result.

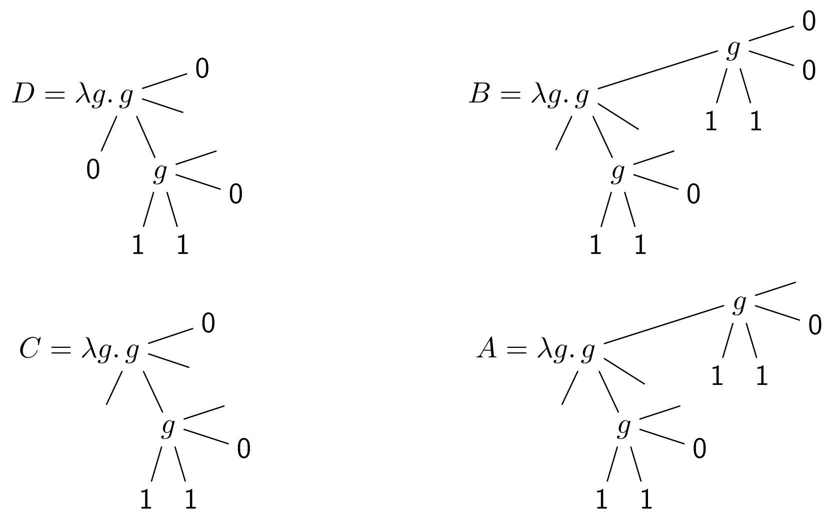
Section 3.4 is about game terms (section 0.4) which are the right medium to investigate Berry's problems. We introduce a graphical representation of game terms that makes their behaviour much better visible. A subterm $\lambda x_1 \dots x_n.\, \mathsf{case}_i(yM_1 \dots M_m)N_0 \dots N_i$ is represented in the graph by a node of the form:

$$\begin{tikzcd}
{} \arrow[d, dash] & & & N_0 \\
\lambda x_1 \dots x_n.\, y \arrow[urrr, dash] \arrow[drrr, dash] \arrow[dd, dash] \arrow[ddrr, dash] & & & \vdots \\
 & & & N_i \\
M_1 & \cdots & M_m &
\end{tikzcd}$$

Subterms $\bot$ of this graph are represented by empty space. We will see examples below.

We prove the existence of an equivalent (finite or infinite) game term for every PCF-term by purely syntactic means. And more: If $M \prec N$ are closed PCF-terms, then there are equivalent game terms $M' \prec N'$. This means that Berry's second conjecture can be restricted to game terms.

We first refute Berry's second conjecture: The syntactic order is not the image of the stable order. Our simplest counter-example is in finitary PCF of second-order type $(\iota \to \iota \to \iota) \to \iota$. We consider the following game terms A, B, C, D:



$$D = \lambda g.\ \mathsf{case}_1(g\,0\,(\mathsf{case}_1(g\,1\ 1\,)\bot\,0\,))\,0\,\bot$$

$$C = \lambda g.\ \mathsf{case}_1(g\bot(\mathsf{case}_1(g\,1\ 1\,)\bot\,0\,))\,0\,\bot$$

$$B = \lambda g.\ \mathsf{case}_1(g\bot(\mathsf{case}_1(g\,1\ 1\,)\bot\,0\,))(\mathsf{case}_1(g\,1\ 1\,)\,0\ 0\,)\bot$$

$$A = \lambda g.\ \mathsf{case}_1(g\bot(\mathsf{case}_1(g\,1\ 1\,)\bot\,0\,))(\mathsf{case}_1(g\,1\ 1\,)\bot\,0\,)\bot$$

We give the trace semantics of these terms:

$$\left.\begin{array}{l}\left.\begin{array}{l}A\left\{\begin{array}{l}\{1\,1\mapsto 1,\ \ \bot\,0\mapsto 0\}\mapsto 0\\ \{\bot\,1\mapsto 1,\ \ \bot\,0\mapsto 0\}\mapsto 0\end{array}\right.\\ \quad\ \{\qquad\qquad\ \bot\bot\mapsto 0\}\mapsto 0\end{array}\right\}B\cong C\\ \quad\ \{\qquad\qquad\ 0\,\bot\mapsto 0\}\mapsto 0\\ \quad\ \{1\,1\mapsto 1,\ \ 0\,0\mapsto 0\}\mapsto 0\\ \quad\ \{1\,\bot\mapsto 1,\ \ 0\,0\mapsto 0\}\mapsto 0\\ \quad\ \{\bot\,1\mapsto 1,\ \ 0\,0\mapsto 0\}\mapsto 0\end{array}\right\}D$$

We have $A \prec B \cong C \prec D$, therefore $[\![A]\!] \leq [\![D]\!]$. This chain of two steps of $\prec$ cannot be replaced by one single step. The intuitive reason for this is: To get down from D to A we must eliminate the two tokens $t = \{\bot\bot\mapsto 0\}\mapsto 0$ and $s = \{0\bot\mapsto 0\}\mapsto 0$. To eliminate t, we must (before) lift the subterm $(g11)$ of D to the top level, which is done in B. But this is only possible, if the token s is eliminated before, namely by deleting the left argument 0 of the top g in D, which is done in C. So there is a precedence: token s before token t.

The possibility of such an example depends on several language features: at least two different ground values 0 and 1, at least a second-order type, and for second-order type some functional parameter of arity at least 2, and the need for nested function calls.

If we restrict the calculus to a single ground value $\mathsf{0}$, we get unary PCF, and in this case both of Berry's conjectures are true: The fully abstract model is a bidomain, in fact it is the standard semantical bidomain construction, proved by Jim Laird in [Lai05]. And we prove that the syntactic order is the image of the stable order, using Laird's proof that every type in unary PCF is a definable retract of some first-order type.

We also refute Berry's first conjecture (Section 0.3): the bicpos of the fully abstract model of PCF are not bidomains. We show that the stable order is not bounded complete (and not distributive). The idea is that the stable lub of two stably bounded elements a and b may entail a new token that was not present in a or b. This new token must be used in the syntax to separate a subterm denoting a from a subterm denoting b that cannot be unified in a common term. Therefore distributivity is not fulfilled, stable lubs are not taken pointwise. And worse: There may be a choice between different new tokens to be entailed, then there is a choice between different minimal stable upper bounds of a and b, but there is no stable lub. The examples use the same essential language features as the example above.

Back to Berry's second conjecture: The counter-example above is a necessary chain of two $\prec$-steps. We can generalize this: For every $n \geq 1$ we give an example of finite elements $a \leq b$ such that there is a chain $C_1 \prec D_1 \cong C_2 \prec D_2 \dots C_n \prec D_n$ of terms with $a = [\![C_1]\!]$, $b = [\![D_n]\!]$, and there is no shorter chain with this property. We call it a chain of least length n. The examples are sequential compositions of n copies of our first example, each copy for a different argument g_i.

This result suggests an improvement of Berry's second conjecture, the chain conjecture: If $a \leq b$ are finite elements, then there is a chain between a and b. We also refute this conjecture by giving terms $A, B \colon (\iota \to \iota \to \iota) \to (\iota \to \iota \to \iota) \to \iota$ with $[\![A]\!] \leq [\![B]\!]$ and no chain between them.

This shows that there seems to be no simple syntactic characterization of the stable order. But the problem remains to find a more complicated one. If we look closer at the terms $A \leq B$, we see that A is produced from B by "forcing" an argument g in B to be strict in one of its two arguments. This leads us to define a new syntactic relation of "strictification" $A \prec^s B$. For this we need a language extension by a new strictness operator str with the trace $\{\{0{\mapsto}0\}{\mapsto}0\}$. $\mathsf{str}\, M$ tests if $M\mathsf{0}$ evaluates to $\mathsf{0}$ and checks if M demands its argument $\mathsf{0}$, only then the output is $\mathsf{0}$. We suggest the *improved chain conjecture*:

In PCF we have: For all finite elements $a \leq b$ there is a sequence (M_i) of terms with $1 \leq i \leq n$, $[\![M_1]\!] = a$, $[\![M_n]\!] = b$, and for every $i < n$ it is $M_i \prec M_{i+1}$ or $M_i \prec^s M_{i+1}$.

This points to an interesting connection of Berry's second conjecture with language extensions of PCF. Our (PCF+str) is the "weakest" sequential extension of PCF with a control operator. It is properly included in (PCF+$\mathsf{strict?}$) of Luca Paolini [Pao06], this in turn is included in (PCF+H), the sequentially realizable functionals of John Longley [Lon02]. (PCF+H) is included in SPCF [Lai07], which is no more extensional. For all these extensions of PCF it would be interesting to give syntactic characterizations of the stable order. First it should be clarified if all types are definable retracts of some lower order types, as is the case for (PCF+H) and SPCF. This could make the proofs easier. For SPCF I conjecture that Berry's second conjecture is valid, as every type of SPCF is a definable retract of some first-order type [Lai07].

0.9 Ch. 4: From Sazonov's non-dcpo natural domains to closed directed-lub partial orders

In this paper we further develop Sazonov's theory of natural domains, see Section 0.4. We first revisit Sazonov's definition of natural domain and give a simpler equivalent axiom system, for which we derive several new axioms.

But our development starts with the most general conceivable structure of incomplete domains, the directed-lub partial orders (dlubpo) $D = (|D|, \sqsubseteq_D, \to_D)$. These are partial orders $(|D|, \sqsubseteq_D)$ with designated directed subsets with their lubs, in the form of a relation $A \to_D a$ meaning that the directed subset A has the natural lub a. The only axiom they have to obey is the singleton axiom: $\{a\} \to_D a$. With continuous functions (defined to be monotonic and respecting natural lubs) they form a category **Dlubpo**.

We go down from these general structures step by step to find cartesian closed categories (ccc). First we find out that the dlubpos do not form a ccc, but that the dlubpos with Sazonov's axiom S5 already form a ccc, though one with a function space that is defined differently from the pointwise function space of natural domains.

In order to classify the common kind of axiom systems of incomplete domains, we introduce lub-rules and lub-rule classes. Let us take an example axiom, axiom S6 (cofinality): If $X, Y \subseteq |D|$, $X \to x$ and $X \sqsubseteq Y \sqsubseteq x$, then $Y \to x$.
(Here $X \sqsubseteq Y$ means: for all $a \in X$ there is $b \in Y$ with $a \sqsubseteq b$. $Y \sqsubseteq x$ means: $Y \sqsubseteq \{x\}$.)
Here we first have as hypothesis of the axiom a subset with a natural lub, $X \to x$, and a further subset Y with an order-theoretic configuration, $X \sqsubseteq Y \sqsubseteq x$. From this configuration follows $\bigsqcup Y = x$, and $Y \to x$ is the conclusion of the axiom.

The important property of the configuration, $X \sqsubseteq Y \sqsubseteq x$, is that it is invariant against monotonic functions f into another partial order E: it is $f^+X \sqsubseteq f^+Y \sqsubseteq fx$. (It is defined $f^+X = \{\, fx \mid x \in X \,\}$.) If f also respects the lub $\bigsqcup X = x$, i.e. $\bigsqcup f^+X = fx$, then also in E we can draw the conclusion $\bigsqcup f^+Y = fx$. We can say this property of invariance means that the conclusion $\bigsqcup Y = x$, and $Y \to x$ in the axiom, is "necessary".

We formalize this by the notion of lub-rule on a partial order $D = (|D|, \sqsubseteq_D)$: this is a triple $(D, P \rightsquigarrow A)$ with P a set of subsets of $|D|$ that each have a lub, and A a subset of $|D|$ that has a lub. P is the pattern of the lub-rule, A the result. Our example axiom is translated to the class of all lub-rules $(D, \{X\} \rightsquigarrow Y)$ such that D is a partial order, $X \subseteq |D|$ with $\bigsqcup X = x$ for some x, and $X \sqsubseteq Y \sqsubseteq x$. A lub-rule is called *valid* if it is "invariant" under monotonic functions, i.e. every monotonic function $f\colon D \to E$ for some partial order E that respects the lubs of the elements of P respects also the lub of A.

All axioms encountered so far (including those of natural domains) describe such valid lub-rules, the axiom systems describe lub-rule classes of valid lub-rules. We explain the philosophical significance of our translation from an axiom system S to a lub-rule class as a "partial extensionalization" of (the intension of) S, i.e. an extension that is between the pure syntax of S and the full extension, i.e. the class of dlubpos that fulfill S.

A lub-rule class is *invariant* if every (valid) lub-rule of the class is transformed by monotonic functions that respect the pattern, to a lub-rule of the same class. The dlubpos generated by an invariant lub-rule class form a full reflective subcategory of **Dlubpo**.

We introduce a new axiom S10 for dlubpos. The category of these dlubpos is the largest full sub-ccc of **Dlubpo** that is generated by an invariant lub-rule class and has the pointwise function spaces. All natural domains are in this category.

The validity of a lub-rule can be characterized by a closure operator cl_D on subsets of dlubpos D that infers from one element all elements below it, and from a natural subset the natural lub of it. This closure operator already appeared in the work of Bruno Courcelle and Jean-Claude Raoult on completions of ordered magmas [CR80].

A lub-rule class is complete if it encompasses all valid lub-rules. The dlubpos generated by a complete lub-rule class are called closed dlubpos (cdlubpo). They can be characterized by the closure operator cl_D, they fulfill the axiom S9 (closure):
If $A \subseteq |D|$ is directed with lub a and $a \in \mathrm{cl}_D\, A$, then $A \to_D a$.
They form a ccc. There are natural domains that are no cdlubpo.

In contrast to natural domains, cdlubpos have several characterizations as canonical structures. Cdlubpos are the dlubpos that are "realized" by "restricted dcpos" (rdcpo). So complete domains are coming in again, and the "dogma" of directed completeness could be "saved". The idea is to complete every cdlubpo with new improper elements (the "blind realizers") to a dcpo that retains the data of the incomplete cdlubpo as a subset, as an order embedding. This idea of realization of a partial order by a dcpo goes back to Alex Simpson [Sim95]: In Simpson's approach every element of the partial order is realized by one or several realizers of the dcpo, while every realizer of the dcpo realizes exactly one element of the partial order. In our approach every element of the partial order is realized by exactly one realizer of the dcpo, while every realizer of the dcpo realizes at most one element of the partial order.

0.10 Outlook

We indicate further directions of research for our four papers.

(1) Confluence of the lambda calculus with left-linear algebraic rewriting:
The theorem has found many applications. It has also been generalized to combinatory reduction systems in [vOvR94]. It is conceivable that weakenings or modifications of the conditions of the theorem are possible. But for this it would be better to wait for applications that show the need for it.

(2) Full abstraction for a recursively typed lambda calculus with parallel conditional:
The reduction relation of the calculus 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 next, as we have the parallel pcase. Instead, it should give for every term a set of its outermost redexes to be reduced parallelly in the next reduction steps. Such a strategy could be given for general algebraic term rewriting rules combined with lambda calculus.

My original aim was to design a functional programming language to compute with sets of elements of any type in a lazy way, which would be based on the language here presented.

Such a language would permit the search and filtering over sets of any type, as far as it is possible, and would need parallel evaluation.

(3) On Berry's conjectures about the stable order in PCF:
Berry's conjectures are refuted, but the problem is not yet solved, we get new conjectures which open new directions in semantics research. This is mainly the "improved chain conjecture", see the end of Section 0.8, for the syntactic characterization of the stable order in PCF. This conjecture suggests that the problem of Berry's second conjecture must be solved with reference to language extensions of PCF, here a strictness operator. Independently, the syntactic characterization of the stable order could be explored in all known extensions of PCF.

(4) From Sazonov's non-dcpo natural domains to closed directed-lub partial orders:
There will soon be a sequel paper to this, where we explore the connection between dlubpos and rdcpos. It will turn out that this is an adjunction between the categories and that it permits the transfer of much of the theory of cccs of algebraic dcpos to closed rdcpos resp. closed dlubpos.

0.11 Acknowledgement

First, I want to thank Reinhold Heckmann for very carefully reading the drafts of all my papers and many fruitful discussions and hints. Proposition 4.8.5 is due to him.

I want to thank my advisor Reinhard Wilhelm for his patience with me over so many years. He gave me the opportunity to work at his Compiler Design Lab from 1981 to 1997 and to be a guest afterwards. He let me the freedom to follow my own interests and pursue my theoretical research in an engineering chair. He always supported me and encouraged me in difficult situations. He also helped to improve this introduction.

I also thank Sebastian Hack and the members of his chair for their support.

There are many colleagues that helped me, esp. with the computer, recently Peter Backes, Roland Leißa and Klaas Boesche. I thank them all.

The work on my first and second paper was supported by Deutsche Forschungsgemeinschaft, SFB 124.

I thank the referees of my first (Jan Willem Klop, Harald Ganzinger) and my third paper.

Chapter 1

Confluence of the lambda calculus with left-linear algebraic rewriting

Information Processing Letters 41(1992)293–299
doi: 10.1016/0020-0190(92)90155-O

Abstract: We consider the untyped λ-calculus with β-reduction and algebraic rewrite rules for some constants. The rules must be left-linear and must not contain applications of variables in their left sides. We prove that the combined reduction relation is confluent (Church-Rosser) if the algebraic rewrite system alone is confluent. This result also holds when reduction is restricted to a subset of terms with suitable closure conditions. (This subset may be a set of well-typed terms.)

1.1 Motivating example

We introduce an example to motivate the general result of section 1.2: Consider a recursively typed λ-calculus [CC91, Cos89] with the type forming constructors + (separated sum) and $\to$ (function space), among others. The terms of the programming language are typed λ-expressions with higher order constants for constructing and destructing terms of non-functional type. In our case we consider the constructors $\mathsf{0}_{\sigma,\tau} : \sigma \to (\sigma + \tau)$ and $\mathsf{1}_{\sigma,\tau} : \tau \to (\sigma + \tau)$ for elements of sum type ("inleft", "inright") and a *parallel* conditional $\mathsf{pcond}_{\sigma,\tau,\rho} : (\sigma + \tau) \to \rho \to \rho \to \rho$. (We will omit the type subscripts and regard the operators as single and untyped in our rewrite system.)

We give a reduction relation, $\longrightarrow$, on terms by the usual β-reduction and the following algebraic rewrite rules:

(pcond 0)	$\mathsf{pcond}\ (\mathsf{0}\ x)\ y\ z$	$\longrightarrow$	y
(pcond 1)	$\mathsf{pcond}\ (\mathsf{1}\ x)\ y\ z$	$\longrightarrow$	z
(pcond+0)	$\mathsf{pcond}\ x\ (\mathsf{0}\ y)\ (\mathsf{0}\ z)$	$\longrightarrow$	$\mathsf{0}\ (\mathsf{pcond}\ x\ y\ z)$
(pcond+1)	$\mathsf{pcond}\ x\ (\mathsf{1}\ y)\ (\mathsf{1}\ z)$	$\longrightarrow$	$\mathsf{1}\ (\mathsf{pcond}\ x\ y\ z)$
(pcond $\to$)	$\mathsf{pcond}\ x\ y\ z\ w$	$\longrightarrow$	$\mathsf{pcond}\ x\ (y\ w)\ (z\ w)$

There may be also other rules, e.g. for the product type or for the usual sequential conditional.

How can we prove the confluence (Church-Rosser property) of this system? I do not know of any theorem in the literature that provides an easy check. Regard the peculiarities here:

- The left sides of the rewrite rules overlap, they form critical pairs. The rules are left-linear, but not "non-ambiguous" in the sense of [Klo80, p. 130], so Klop's results on the confluence of "regular combinatory reduction systems" cannot be applied.

  Overlapping reduction rules are needed for the syntactical definition of several "fairly parallel" functions, see also the parallel or and the classical parallel if_then_else of [Plo77]. The definability of these functions is in turn necessary for the full abstractness of the usual denotational semantics.

- Although we have a typed system, it is not normalizing. (There are non-terminating programs, e.g. the fixed point combinator is definable for every type.) Therefore all the results on the confluence of λ-calculus + algebraic rewriting that rely on strong normalization [BT88, BTG89, Bar90, Dou91, JO91] do not apply here.

  But well-typing is also essential for the confluence of our system:
  The term pcond x (0 y) (0 z) w by (pcond $\rightarrow$) reduces to pcond x (0 y w) (0 z w) and by (pcond+0) to 0 (pcond x y z) w, and these two terms have no common reduct. But pcond x (0 y) (0 z) w is not typable, as (0 y) is not of function type.

- The terms of our algebraic rewrite system are constructed of application as the only non-0-ary symbol (it is invisible) and of 0-ary constants and variables. (These arities are meant in the (low level) sense of algebraic terms, not in the type sense!) We will exclude terms with an application of a variable (xM) only on the left side of rules, but not on the right side. But the prevailing concept of algebraic rewriting + λ-calculus defines algebraic terms as applications of constants to (sequences of) algebraic terms, so it excludes (xM) entirely, e.g. [Dou91]. This would exclude e.g. our rule (pcond $\rightarrow$). In order to make the distinction clear, we use the usual notion "algebraic" only in the title and replace it by our "applicative" (terms, term rewriting systems) in the following section 1.2.

In section 1.2 we prove the confluence of the λ-calculus with left-linear, not variable-applying, confluent rules. For our example it remains to show the confluence of the rewrite rules alone. This can be done by proving the termination, e.g. by mapping each term to a number that decreases in every reduction step, and the convergence of all critical pairs.
Our notation $x \equiv y$ means: x is defined to be y.

1.2 Results

1.2.1 Terms and occurrences

We fix a finite or denumerable set $\mathcal{C}$ of higher-order *constants* and a denumerable set $\mathcal{V}$ of *variables*, with $\mathcal{C} \cap \mathcal{V} = \emptyset$. Variables are denoted by x, y. (λ-)*terms* are built from constants and variables by *application* MN and (λ-)*abstraction* $\lambda x.M$. They are considered equal modulo renaming of bound variables (α-conversion). Λ is the set of all terms. The set $\mathcal{A}$

of *applicative terms* is the set of all terms without any abstraction. The function *Var* maps an applicative term to the set of its variables.

Occurrences of subterms are (as usual) sequences of $0, 1, \lambda$, denoted by u, v, w, z. $v \leq w$ iff v is a prefix of w. v, w are *disjoint* ($v \mid w$) iff neither $v \leq w$ nor $w \leq v$. We say u *is in* term M if M has a subterm at u:
$M/\varepsilon \equiv M$, $(MN)/0w \equiv M/w$, $(MN)/1w \equiv N/w$, $(\lambda x.M)/\lambda w \equiv M/w$.

Replacement at an occurrence: $M[\varepsilon{\leftarrow}N] \equiv N$, $(MN)[0w{\leftarrow}P] \equiv M[w{\leftarrow}P]N$, $(MN)[1w{\leftarrow}P] \equiv MN[w{\leftarrow}P]$, $(\lambda x.M)[\lambda w{\leftarrow}P] \equiv \lambda x.M[w{\leftarrow}P]$.

If (w_i), $1 \leq i \leq n$, is a sequence of pairwise disjoint occurrences and (N_i) a corresponding sequence of terms, then $M[\vec{w_i}{\leftarrow}\vec{N_i}] \equiv M[w_1{\leftarrow}N_1]\ldots[w_n{\leftarrow}N_n]$.

1.2.2 Applicative term rewriting system

An *applicative term rewriting system (ATRS)* is any set of pairs $\langle L{\to}R\rangle$ of applicative terms, where L is no variable and $\mathit{Var}\, R \subseteq \mathit{Var}\, L$. Each pair is called a rule of the ATRS. (Note that applicative terms are built only from constants, variables, and application.)

A *substitution* is a map $\sigma : \mathcal{V} \to \Lambda$. It naturally extends to applicative terms.
The ATRS together with β-reduction defines a one step reduction relation, $\longrightarrow$, on Λ; it is the least relation satisfying:

(β) $(\lambda x.M)N \longrightarrow M[x{:=}N]$ 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,

($\langle L{\to}R\rangle$) for all rules $\langle L{\to}R\rangle$: $\sigma L \longrightarrow \sigma R$ for any substitution σ,

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

(λ) $M \longrightarrow M' \Longrightarrow \lambda x.M \longrightarrow \lambda x.M'$.

$\longrightarrow$ reduces applicative terms to applicative terms. The restriction of $\longrightarrow$ to $\mathcal{A}$ is the least relation on $\mathcal{A}$ satisfying ($\langle L{\to}R\rangle$) for all rules and the two (app) conditions.

Let $M \longrightarrow N$ hold. Then there is an occurrence u in M where either M/u is of the form $(\lambda x.M_1)M_2$ and $N = M[u{\leftarrow}M_1[x{:=}M_2]]$ or M/u is of the form σL and $N = M[u{\leftarrow}\sigma R]$ for the rule $\langle L{\to}R\rangle$. $M \longrightarrow N$ together with such u and the used rule is said to be a *reduction step*. M/u is called the *redex* of the reduction step, u is called the occurrence of the redex resp. of the reduction step, and the reduction *is at* u *with rule* (β) resp. $\langle L{\to}R\rangle$.

We define descendants of occurrences for the second case $\langle L{\to}R\rangle$: Let w be an occurrence in M beneath a variable of L, i.e. $w = uvz$ with $L/v = y \in \mathcal{V}$. Then the *descendants* of w after the reduction step $M \longrightarrow N$ at u with rule $\langle L{\to}R\rangle$ are all $uv'z$ with $R/v' = y$. If $w|u$ then w is the only *descendant* of w. (The other cases for w do not occur in the following.)

An ATRS is called *left-linear* iff the left side of each rule is linear, i.e. every variable has at most one occurrence in the left side.

An ATRS is called *variable-applying* iff there is a left side with a subterm of the form (xM), where $x \in \mathcal{V}$.

We will impose the restriction that our ATRS is left-linear and *not* variable-applying. Left-linearity is motivated by non-confluent examples of the λ-calculus with non-left-linear rewrite rules, e.g. surjective pairing [Klo80, III.1]. This restriction seems to be reasonable for our intended use of ATRSs as definitions of deterministic programming languages. The detection of a non-linear left side cannot be seen as an atomic operational step.

The not-variable-applying condition is motivated e.g. by the ATRS with the single rule $\mathsf{x}\ \mathsf{y} \longrightarrow \mathsf{x}$. We get the following non-converging reductions: $\lambda x.x \longleftarrow (\lambda x.x)y \longrightarrow y$. "Not variable-applying" ensures that there is no overlap between the left side of some rule of the ATRS and the left side of the β-reduction rule. Then the ATRS can be said to be "orthogonal" to β-reduction, there are no "critical pairs" between the two reduction systems. We will prove that in this case, together with left-linearity, the two reduction relations commute.

Remarks: [RV80, Proposition 10] and [Toy88, Corollary 3.1] prove a similar result: Two left-linear term rewriting systems commute if they do not overlap with each other.
The not-variable-applying condition appears also in [Nip91, Lemma 5.1]: The β-reduction rule and the applicative rules are both formalized as rules of a single higher-order rewrite system, so higher-order critical pairs between them can be defined. Due to a general critical pair lemma, the *local* confluence of the combined reduction system follows from the local confluence of the applicative rules and the not-variable-applying condition. Left-linearity is not needed there. Note that we prove full confluence.

1.2.3 $\rightarrow$-closed sets of terms

To obtain an abstract notion of "well-typing" we must restrict the reduction to certain subsets of Λ. These can be rather arbitrary, but must fulfill three closure conditions with respect to our reduction $\longrightarrow$:

A set $T \subseteq \Lambda$ is called *$\rightarrow$-closed* iff for every $M \in T$ the following holds:

1) $M \longrightarrow M' \Longrightarrow M' \in T$.

2) Every subterm of M is in 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 $M[u{\leftarrow}x] \in T$.

The set of applicative terms of T is also $\rightarrow$-closed. In our example T would be the set of typable terms; its closedness is a consequence of the "subject reduction property".

Remark: [Dou91] uses two more closure conditions concerning strong normalization, as he does not assume left linearity. Condition 3 is missing in that work. Our reason for it is the weaker premise of our confluence theorem: While [Dou91] assumes confluence on all algebraic terms, we assume confluence on algebraic, resp. applicative terms of $\underline{T}$ only. Condition 3 ensures that there are enough applicative terms in T.

1.2.4 Confluence Theorem

$\xrightarrow{*}$ means the reflexive, transitive closure of the relation $\longrightarrow$.
$\longrightarrow$ is called *confluent* (Church-Rosser) on the $\rightarrow$-closed set T iff for all terms $M \in T$ with $N \xleftarrow{*} M \xrightarrow{*} P$ there is Q with $N \xrightarrow{*} Q \xleftarrow{*} P$. (These terms are all in T because of $\rightarrow$-closedness of T.)

Main theorem: *For every left-linear, not variable-applying ATRS with reduction relation $\longrightarrow$ and every $\rightarrow$-closed set T:*
$\longrightarrow$ is confluent on the applicative terms of $T \iff \longrightarrow$ is confluent on T.

Proof: The direction $\Longleftarrow$ is trivial.
For showing $\Longrightarrow$, we split $\longrightarrow$ into two parts: $\xrightarrow[\beta]{}$ is the least relation on Λ satisfying (β), (app) and (λ). $\xrightarrow[A]{}$ is the least relation on Λ satisfying ($\langle L \rightarrow R\rangle$) for all rules, (app) and (λ). We have $\longrightarrow = \xrightarrow[\beta]{} \cup \xrightarrow[A]{}$. $\longrightarrow$ is confluent on the applicative terms of T iff $\xrightarrow[A]{}$ is confluent on the applicative terms of T.
$\xrightarrow[\beta]{}$ is known to be confluent on all terms [Bar84, p. 59]. After two simple lemmas we will prove the confluence of $\xrightarrow[A]{}$ on T (Lemma 1.2.3), and then the commutativity of $\xrightarrow[\beta]{}$ and $\xrightarrow[A]{}$ (Lemma 1.2.4). The assumptions "left-linear" and "not variable-applying" for the ATRS will be used only in the proof of the last Lemma 1.2.4.

Lemma 1.2.1. *If $M \xrightarrow[A]{} M'$ then $M[x{:=}N] \xrightarrow[A]{} M'[x{:=}N]$.*

Proof: The reduction step of the consequence is at the same occurrence, with the same rule $\langle L \rightarrow R\rangle$. We have to show $(\sigma L)[x{:=}N] \xrightarrow[A]{} (\sigma R)[x{:=}N]$ for any σ.
Let σ' be the substitution $y \mapsto (\sigma y)[x{:=}N]$. Then:

$$(\sigma L)[x{:=}N] = \sigma' L \xrightarrow[A]{} \sigma' R = (\sigma R)[x{:=}N]$$

Lemma 1.2.2. *If $N \xrightarrow[A]{} N'$ then $M[x{:=}N] \xrightarrow[A]{*} M[x{:=}N']$.*

Proof: If the reduction step $N \xrightarrow[A]{} N'$ is at u with rule $\langle L \rightarrow R\rangle$, we make a reduction with rule $\langle L \rightarrow R\rangle$ at all occurrences vu in $M[x{:=}N]$, where v is any free occurrence of x in M.

Lemma 1.2.3. *$\xrightarrow[A]{}$ is confluent on T.*
(This lemma does not use the assumptions "left-linear" and "not variable-applying" for the ATRS, but all the conditions of $\rightarrow$-closedness of T.)

Proof: We prove for any $M \in T$:
If $N \xleftarrow[A]{*} M \xrightarrow[A]{*} P$ then there is Q with $N \xrightarrow[A]{*} Q \xleftarrow[A]{*} P$, under the assumption of this property for all proper subterms of M (induction on M). (The subterms of M are all in T.)
Case 1: M is of the form $\lambda x.M'$. Then $N = \lambda x.N'$, $P = \lambda x.P'$ with $N' \xleftarrow[A]{*} M' \xrightarrow[A]{*} P'$. By the induction hypothesis there is Q' with $N' \xrightarrow[A]{*} Q' \xleftarrow[A]{*} P'$. We choose $Q \equiv \lambda x.Q'$.
Case 2: M is no abstraction. Let (u_i), $1 \leq i \leq m$ be a sequence of the outermost, i.e. minimal, occurrences of abstractions in M. The u_i's are pairwise disjoint. Let (x_i), $1 \leq i \leq m$, be a sequence of pairwise different variables, each one different from any variable in M, that can be used to replace the abstractions at u_i according to condition 3 of $\rightarrow$-closed

sets. $M' \equiv M[\vec{u_i} \leftarrow \vec{x_i}]$ is an applicative term of T with $M = M'[x_1{:=}M/u_1]\ldots[x_m{:=}M/u_m]$. We proceed with the following property of the reduction $M \xrightarrow[A]{*} N$:

Proposition: For every reduction $M \xrightarrow[A]{*} N$ there is an (applicative) term N' with

$$M' \xrightarrow[A]{*} N', \quad N = N'[\vec{v_j^1} \leftarrow \vec{N_j^1}]\ldots[\vec{v_j^m} \leftarrow \vec{N_j^m}],$$

where (v_j^i) is a sequence of the occurrences of variable x_i in N' for every i, and (N_j^i) is a corresponding sequence of terms with $M/u_i \xrightarrow[A]{*} N_j^i$ for any i,j.
This means: The variables (x_i) keep track of the corresponding subterms of M during the reduction $M' \xrightarrow[A]{*} N'$. The reductions in M' are independent of the reductions made in subterms M/u_i.

Proof: The proposition can be proved by induction on the length of the reduction $M \xrightarrow[A]{*} N$: For length 0 we have $N = M$ of the required form, $N' = M'$. Now assume we have reached a term N of the required form and make the next reduction step $N \xrightarrow[A]{} L$ at w with rule $\langle S {\to} T\rangle$. There are two cases:
Case 1: w is an occurrence in N'. Then the left side of the rule lies entirely in N', because all the N_j^i are abstractions. So there is a reduction step $N' \xrightarrow[A]{} L'$ at w with rule $\langle S {\to} T\rangle$. Every v_j^i has a set D of descendants after the reduction step $N \xrightarrow[A]{} L$. v_j^i is replaced in the sequence for i by a sequence of the elements of D, and N_j^i is replaced by a corresponding multiple sequence of the same N_j^i. Then L is L' with the new replacements of variable occurrences.
Case 2: Some $v_j^i \leq w$. Then L is N' with the old replacement, except that N_j^i is exchanged with its reduct.
End of Proof Proposition

Now we have $N \xleftarrow[A]{*} M \xrightarrow[A]{*} P$ with N in the form of the proposition, and P in analogous form, namely there is P' with

$$M' \xrightarrow[A]{*} P', \quad P = P'[\vec{w_j^1} \leftarrow \vec{P_j^1}]\ldots[\vec{w_j^m} \leftarrow \vec{P_j^m}],$$

(w_j^i) a sequence of occurrences of variable x_i in P', $M/u_i \xrightarrow[A]{*} P_j^i$.

We have $N' \xleftarrow[A]{*} M' \xrightarrow[A]{*} P'$. As $\xrightarrow[A]{}$ is confluent on applicative terms of T (and M' is one), there is Q' with $N' \xrightarrow[A]{*} Q' \xleftarrow[A]{*} P'$.
For $1 \leq i \leq m$: $M/u_i \xrightarrow[A]{*} N_j^i$ and $M/u_i \xrightarrow[A]{*} P_j^i$ for all j. M/u_i is a proper subterm of M (since M is no abstraction), so we can use the induction hypothesis to find a common reduct Q^i of all the N_j^i, P_j^i (for fixed i), i.e. $N_j^i \xrightarrow[A]{*} Q^i$, $P_j^i \xrightarrow[A]{*} Q^i$.

Then we choose $Q \equiv Q'[x_1{:=}Q^1]\ldots[x_m{:=}Q^m]$. We can reduce

$$N = N'[\vec{v_j^1} \leftarrow \vec{N_j^1}]\ldots[\vec{v_j^m} \leftarrow \vec{N_j^m}] \xrightarrow[A]{*} Q :$$

First we apply all the reductions $N_j^i \xrightarrow[A]{*} Q^i$ under every subterm occurrence v_j^i. As the (v_j^i) are *all* the occurrences of x_i in N', we get the term $N'[x_1{:=}Q^1]\ldots[x_m{:=}Q^m]$. This term reduces by $\xrightarrow[A]{*}$ and repeated application of Lemma 1.2.1 to Q.
Analogously, we can reduce $P \xrightarrow[A]{*} Q$.
End of Proof 1.2.3

Now $\xrightarrow[\beta]{}$ and $\xrightarrow[A]{}$ are confluent on T. We will show that $\xrightarrow[\beta]{}$ and $\xrightarrow[A]{}$ commute (on Λ), i.e. if $N \xleftarrow[\beta]{*} M \xrightarrow[A]{*} P$ then there is Q with $N \xrightarrow[A]{*} Q \xleftarrow[\beta]{*} P$. Then the confluence of $\longrightarrow$ on T follows from the Commutative Union Theorem of Hindley-Rosen [Ros73, 3.5] and from condition 1 of the $\rightarrow$-closedness of T.

For the proof of commutativity, we define a relation $\xrightarrow[\beta p]{}$ on Λ that makes multiple, parallel β-reductions in one step, by the following induction rules:

1) $M \xrightarrow[\beta p]{} M$

2) $(\lambda x.M)N \xrightarrow[\beta p]{} M[x{:=}N]$

3) $M \xrightarrow[\beta p]{} M'$, $N \xrightarrow[\beta p]{} N' \Longrightarrow MN \xrightarrow[\beta p]{} M'N'$

4) $M \xrightarrow[\beta p]{} M' \Longrightarrow \lambda x.M \xrightarrow[\beta p]{} \lambda x.M'$

$\xrightarrow[\beta p]{}$ makes a reduction on a disjoint set of redexes only. Contrary to the similar relation used in the Tait/Martin-Löf proof of the confluence of λ-calculus [Bar84, p. 60] there are no reductions inside M, N in rule 2. This is not necessary here, because the other, commuting relation $\xrightarrow[A]{}$ is "too weak" to cause nested β-reductions.

Lemma 1.2.4. *If $N \xleftarrow[\beta p]{} M \xrightarrow[A]{} P$ then there is Q with $N \xrightarrow[A]{*} Q \xleftarrow[\beta p]{} P$.*
(All terms are in Λ. This lemma uses the assumptions "left-linear" and "not variable-applying" for the ATRS.)

Proof: By induction on the derivation of $M \xrightarrow[\beta p]{} N$.
Case 1: $N = M$. We can choose $Q \equiv P$.
Case 2: $M \xrightarrow[\beta p]{} N$ is derived as $(\lambda x.M_1)M_2 \xrightarrow[\beta p]{} M_1[x{:=}M_2]$.
There is no rule $\langle L \rightarrow R\rangle$ applicable at occurrence ε in M, because the ATRS is not variable-applying and no left side is a variable. Therefore, $M \xrightarrow[A]{} P$ must be either of the form

$$(\lambda x.M_1)M_2 \xrightarrow[A]{} (\lambda x.M_1')M_2 \quad \text{with} \quad M_1 \xrightarrow[A]{} M_1'$$

or of the form

$$(\lambda x.M_1)M_2 \xrightarrow[A]{} (\lambda x.M_1)M_2' \quad \text{with} \quad M_2 \xrightarrow[A]{} M_2'.$$

By Lemma 1.2.1, we get in the first case:

$$M_1[x{:=}M_2] \xrightarrow[A]{} M_1'[x{:=}M_2] \xleftarrow[\beta p]{} (\lambda x.M_1')M_2,$$

by Lemma 1.2.2 in the second case:

$$M_1[x{:=}M_2] \xrightarrow[A]{*} M_1[x{:=}M_2'] \xleftarrow[\beta p]{} (\lambda x.M_1)M_2'.$$

Case 3: $M \xrightarrow[\beta p]{} N$ is derived as $M_1M_2 \xrightarrow[\beta p]{} N_1N_2$, as consequence of $M_1 \xrightarrow[\beta p]{} N_1$, $M_2 \xrightarrow[\beta p]{} N_2$.
Subcase 3.1: $M \xrightarrow[A]{} P$ is of the form $M_1M_2 \xrightarrow[A]{} P_1M_2$ with $M_1 \xrightarrow[A]{} P_1$.
We have $N_1 \xleftarrow[\beta p]{} M_1 \xrightarrow[A]{} P_1$, by induction hypothesis there is Q_1 with
$N_1 \xrightarrow[A]{*} Q_1 \xleftarrow[\beta p]{} P_1$. So $N_1N_2 \xrightarrow[A]{*} Q_1N_2 \xleftarrow[\beta p]{} P_1M_2$.

Subcase 3.2: $M \xrightarrow[A]{} P$ is of the form $M_1M_2 \xrightarrow[A]{} M_1P_2$ with $M_2 \xrightarrow[A]{} P_2$.
Analogous to subcase 3.1.

Subcase 3.3: $M \xrightarrow[A]{} P$ is of the form $\sigma L \xrightarrow[A]{} \sigma R$ for rule $\langle L \to R \rangle$, where (x_i), $1 \leq i \leq n$, are the variables of L and σ is the substitution that maps x_i to S_i.
Let us look at a single β-reduction in $M \xrightarrow[\beta p]{} N$ at occurrence u, thus M/u is of the form $(\lambda y.V)W$. If u is an occurrence in L it must be a variable occurrence, $L/u = x_j$ for some j. (Reason: Any other occurrence would imply L/u of the form (x_kW'), this is not possible since the ATRS is not variable-applying.) Hence in any case there is a variable occurrence v in L with $v \leq u$. The x_i identify the variable occurrences uniquely, since the ATRS is left-linear.
Now there is a substitution σ' mapping x_i to S'_i with $S_i \xrightarrow[\beta p]{} S'_i$ and $N = \sigma' L$. We choose $Q \equiv \sigma' R$ and obtain

$$N = \sigma' L \xrightarrow[A]{} \sigma' R \xleftarrow[\beta p]{} \sigma R = P$$

.

Case 4: $M \xrightarrow[\beta p]{} N$ is derived as $\lambda x.M_1 \xrightarrow[\beta p]{} \lambda x.N_1$, as consequence of $M_1 \xrightarrow[\beta p]{} N_1$.
$M \xrightarrow[A]{} P$ must be of the form $\lambda x.M_1 \xrightarrow[A]{} \lambda x.P_1$ with $M_1 \xrightarrow[A]{} P_1$. By induction hypothesis there is Q_1 with $N_1 \xrightarrow[A]{*} Q_1 \xleftarrow[\beta p]{} P_1$. We choose $Q \equiv \lambda x.Q_1$.
End of Proof 1.2.4

From this lemma and the Commutativity Lemma of [Ros73, 3.6] it follows that $\xrightarrow[\beta]{}$ and $\xrightarrow[A]{}$ commute, because $\xrightarrow[\beta]{*} = \xrightarrow[\beta p]{*}$. As stated above, from this we can deduce the confluence of $\longrightarrow$ on T.
End of Proof Main Theorem

Acknowledgements: I thank Harald Ganzinger, Reinhold Heckmann, Helmut Seidl, Kurt Sieber and Reinhard Wilhelm for their interest in my work, fruitful discussions and careful reading of drafts of this paper. I especially thank Jan Willem Klop for his suggestions, the interpretation of the not-variable-applying condition in terms of orthogonality is due to him.

Chapter 2

Full abstraction for a recursively typed lambda calculus with parallel conditional

revised Report 12/1993 of SFB 124, Universität des Saarlandes,
arxiv:0806.1827

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.

2.1 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 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 $+$, $\times$, $\rightarrow$ 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 $\mathcal{S}[\![\lambda x.\Omega]\!]\varepsilon = \perp$.

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].

2.2 Syntax and reduction

2.2.1 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 $\bot$.

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 2.2.1. The void-prefix order $\prec \subseteq T \times T^c_\mu$ is the least partial order satisfying:

1) $\mathsf{void} \prec \tau$ for all $\tau \in T^c_\mu$,
2) $\sigma \prec \sigma', \tau \prec \tau' \ \Rightarrow\ \sigma \mathbin{@} \tau \prec \sigma' \mathbin{@} \tau'$
   for $@ \in \{+, \times, \rightarrow\}$, $\sigma, \tau \in T$, and $\sigma', \tau' \in T^c_\mu$.

$\prec$ 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 2.2.2. The *unfolding* $\rightsquigarrow \subseteq T^c_\mu \times T^c_\mu$ 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) $\tau \rightsquigarrow \tau' \ \Rightarrow\ (\tau \mathbin{@} \sigma) \rightsquigarrow (\tau' \mathbin{@} \sigma)$ and $(\sigma \mathbin{@} \tau) \rightsquigarrow (\sigma \mathbin{@} \tau')$ for $@ \in \{+, \times, \rightarrow\}$, $\tau, \tau', \sigma \in T^c_\mu$.

$\rightsquigarrow$ 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 $\tau, \tau' \in T^c_\mu$.
For every $\tau \in T^c_\mu$ we define the unfolding

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

Proposition 2.2.3. $\tau^* \in T_\infty$.

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

$\approx$ is decidable [AC90].

2.2.2 Terms

For every type $\tau \in T^c_\mu$ 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 $ctype : \mathcal{C} \to T^c_\mu$.

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, $\sigma \in T^c_\mu$:

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

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

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

($\to$ E) $M : \sigma \to \tau, N : \sigma \;\Rightarrow\; MN : \tau$

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

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 $\sigma \in T^c_\mu$ or with type tree $\sigma \in T_\infty$. $\mathcal{T}^c_\sigma$ 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:

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

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 $M : \mu t.\sigma \to t$ and $N : \mu t.\sigma \to \sigma \to t$ 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:

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

$\mathsf{1}_{\sigma,\tau}$: $\tau \to (\sigma + \tau)$, also called "inright"

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

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

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

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

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

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.

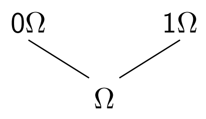
We will frequently omit the type subscripts of the constants. The term rewriting system will treat them as single symbols. Notice that we do not introduce these operators by special term formation rules for the types $\sigma + \tau$ and $\sigma \times \tau$, as it is often done, but as constants of higher 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 $+$.

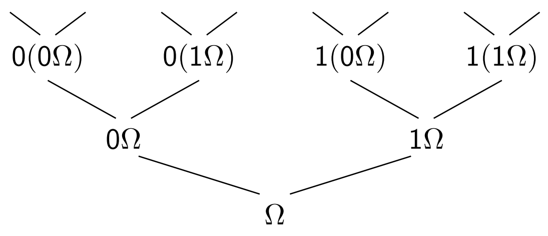
Examples of common types and their canonical terms:

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

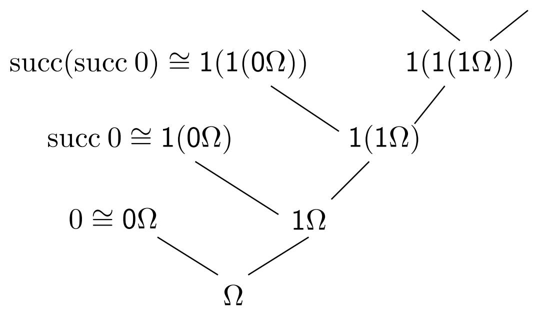
$\mathsf{bool} =_{\text{def}} \mathsf{void} + \mathsf{void}$



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


$\mathsf{0}(\mathsf{0}\Omega)$
$\mathsf{0}(\mathsf{1}\Omega)$
$\mathsf{1}(\mathsf{0}\Omega)$
$\mathsf{1}(\mathsf{1}\Omega)$
$\mathsf{0}\Omega$
$\mathsf{1}\Omega$
Ω

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


succ(succ 0) $\cong \mathsf{1}(\mathsf{1}(\mathsf{0}\Omega))$
$\mathsf{1}(\mathsf{1}(\mathsf{1}\Omega))$
succ 0 $\cong \mathsf{1}(\mathsf{0}\Omega)$
$\mathsf{1}(\mathsf{1}\Omega)$
$0 \cong \mathsf{0}\Omega$
$\mathsf{1}\Omega$
Ω

$\mathsf{boollist} =_{\text{def}} \mu t.\mathsf{void} + (\mathsf{bool} \times t)$
$\mathsf{boollist}$ 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.

2.2.3 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)	$\mathsf{pcase}_{\sigma,\tau,\varrho_0+\varrho_1}\ x\ (0y)\ (0z)$	$\to$	$\mathsf{0}\ (\mathsf{pcase}_{\sigma,\tau,\varrho_0}\ x\ y\ z)$
(pcase11)	$\mathsf{pcase}_{\sigma,\tau,\varrho_0+\varrho_1}\ x\ (1y)\ (1z)$	$\to$	$\mathsf{1}\ (\mathsf{pcase}_{\sigma,\tau,\varrho_1}\ x\ y\ z)$
(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:

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

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 (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 (pcase00)–(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 out0, out1 in the right sides.

Proposition 2.2.5. *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 2.2.6 (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 pcase x $(0y)$ $(0z)$ w. This term is not typable, as $(0y)$ is not of function type. It reduces to pcase x $(0y\,w)$ $(0z\,w)$ by rule (pcase$\rightarrow$), and to 0(pcase $x\,y\,z$) w by rule (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 (case0) ... (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, $\to$ is the reduction relation on applicative terms of Λ. We will first prove that $\to$ is locally confluent on $\mathcal{A}$ via convergence of critical pairs, then prove that $\to$ is noetherian (terminating, strongly normalizing) and conclude the confluence of $\to$ on $\mathcal{A}$ by Newman's Lemma (Lemma 2.4 of [Hue80]). *Local (or weak) confluence* of $\to$ on a set T of terms means: For any $M \in T$ with $N \leftarrow M \to P$ there is a term Q with $N \to^* Q \leftarrow^* P$.

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:
$y\,w \leftarrow (\mathsf{pcase}\,(0x)\,y\,z)\,w \to \mathsf{pcase}\,(0x)\,(y\,w)\,(z\,w)$ that there should be a parallel reduction step: $y\,w \to \mathsf{pcase}\,(0x)\,(y\,w)\,(z\,w)$. 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 $\to$ on $\mathcal{A}$ we will apply a generalized version of Lemma 3.1 of [Hue80]: "For any term rewriting system $\mathcal{R}$: The relation $\to_{\mathcal{R}}$ 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 $\to$-closed sets of terms above. This leads us to:

Definition 2.2.7. A subset T of terms is called $\to_{\mathcal{R}}$*-complete* for a term rewriting system with reduction relation $\to_{\mathcal{R}}$ if for every $M \in T$ the following hold:

1) $M \to_{\mathcal{R}} M' \Rightarrow M' \in T$,
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 2.2.8. Let $\langle S{\to}T\rangle$,$\langle L{\to}R\rangle$ 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 $\langle S{\to}T\rangle$ on $\langle L{\to}R\rangle$ in u determines the *critical pair* (P, Q) defined by $P = (\mu L)[u \leftarrow \mu T]$, $Q = \mu R$. It is $P \leftarrow \mu L \to Q$. We call μL an *overlap* of the critical pair (P, Q).

Our generalization of Huet's Lemma 3.1 is now:

Lemma 2.2.9. *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 $\rightarrow_{\mathcal{R}}$-completeness. The replacement of subterms by variables is possible according to condition 3 of $\rightarrow_{\mathcal{R}}$-completeness, so that the overlap is in T. Thus $P \downarrow Q$ by hypothesis, and the proof proceeds as in [Hue80]. $\square$

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

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

We prove now that $\rightarrow$ 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:

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

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

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

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

We have now proved that $M \to N \Rightarrow \varphi M > \varphi N$. Thus there are no infinite reduction chains. From this and the local confluence of $\to$ on $\mathcal{A}$ follows by Newman's Lemma the confluence of $\to$ on $\mathcal{A}$. As explained above, the confluence of $\to$ on all typed terms follows from the theorem of [Mül92].
(End of proof Theorem 2.2.6.) □

2.3 Prime systems

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

Definition 2.3.1. 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 2.3.2. 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 $X{\downarrow} = \{a \in A \,|\, \exists b \in X.\ a \leq b\}$, 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 2.3.3. 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 2.3.4. *[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 2.3.5. 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 2.3.6. *$\trianglelefteq$ 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 $a \leq b \;\Leftrightarrow\; a \leq_i b \;\Leftrightarrow\; a \leq_B b$. $\square$

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

Definition 2.3.7. 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 2.3.8. *$\trianglelefteq$ is a partial order on $\mathbf{PSys}^n$ with $(\underline{\perp}, \ldots, \underline{\perp})$ as least element. All increasing ω-chains in $(\mathbf{PSys}^n, \trianglelefteq)$ have a least upper bound taken coordinate-wise.*

Definition 2.3.9. 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 2.3.10. *$F : \mathbf{PSys}^n \to \mathbf{PSys}$ 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 2.3.11. $F : \mathbf{PSys} \to \mathbf{PSys}$ 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 2.3.12. *$F : \mathbf{PSys} \to \mathbf{PSys}$ 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 2.3.13. 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

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

Proposition 2.3.14. *$\mathcal{A}_0 + \mathcal{A}_1$ 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 2.3.15. *$+$ is continuous on $(\mathbf{PSys}, \trianglelefteq)$.*

Proof. It is easy to show that $+$ is continuous in its first and second argument, using Lemma 2.3.12. □

Product $\times$

Definition 2.3.16. 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 2.3.17. *$\mathcal{A}_0 \times \mathcal{A}_1$ 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 2.3.18. *$\times$ is continuous on $(\mathbf{PSys}, \trianglelefteq)$.*

Proof. It is easy to show that $\times$ is continuous in its first and second argument, using Lemma 2.3.12. □

Function space $\to$

Definition 2.3.19. 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 $\mathcal{A} \to \mathcal{B} = (C, \uparrow, \leq)$:
$C = \overline{A} \times B$, 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 2.3.20. *$\mathcal{A} \to \mathcal{B}$ 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 2.3.21. *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 2.3.22. *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 $(X,a)\le(Y,b)$. From $Y\le X$ follows $Y{\downarrow}\subseteq X{\downarrow}$. Then $b\in f(X{\downarrow})$, as $b\in f(Y{\downarrow})$ and f is monotonic. As $a\le b$, also $a\in f(X{\downarrow})$ and $(X,a)\in Pr(f)$. □

Theorem 2.3.23. *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 $r\in|\mathcal{A}\to\mathcal{B}|$, $Pr(|r|) = r$:

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

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 $D = \{Y{\downarrow} \mid Y$ 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 $(X, a) \in s$. $\square$

Theorem 2.3.24. *$\to$ 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 $(X, a), (Y, b) \in B$.

$$\begin{array}{rcl}
(X, a) \uparrow (Y, b) & \Leftrightarrow & (X \uparrow_0 Y \;\Rightarrow\; a \uparrow_1 b) \\
& \Leftrightarrow & (X \uparrow'_0 Y \;\Rightarrow\; a \uparrow_1 b) \\
& \Leftrightarrow & (X, a) \uparrow' (Y, b) \\
(X, a) \leq (Y, b) & \Leftrightarrow & Y \leq_0 X \text{ and } a \leq_1 b \\
& \Leftrightarrow & Y \leq'_0 X \text{ and } a \leq_1 b \\
& \Leftrightarrow & (X, a) \leq' (Y, b)
\end{array}$$

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 $\mathcal{A}_0 \trianglelefteq \mathcal{A}_1 \trianglelefteq \ldots$ 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 $\bigcup_i(\mathcal{B} \rightarrow \mathcal{A}_i)$. □

2.4 Denotational semantics

2.4.1 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 $\tau \in T^c_\mu$ by the semantics of its unfolding τ^*.

Definition 2.4.1. The sequence of maps $\mathcal{P}_n : T_\infty \rightarrow \mathbf{PSys}$, $n \geq 0$, is defined inductively by:

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

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

Proposition 2.4.2. *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:
$\mathcal{P}_{n+1}(\mathsf{void}) = \perp \trianglelefteq \mathcal{P}_{n+2}(\mathsf{void})$.
$\mathcal{P}_{n+1}(\sigma \mathbin{@} \tau) = \mathcal{P}_n(\sigma) \mathbin{@} \mathcal{P}_n(\tau) \trianglelefteq \mathcal{P}_{n+1}(\sigma) \mathbin{@} \mathcal{P}_{n+1}(\tau) = \mathcal{P}_{n+2}(\sigma \mathbin{@} \tau)$ for $@ \in \{+, \times, \rightarrow\}$. □

This permits to give the semantics of type trees:

Definition 2.4.3. Define the map $\mathcal{P} : T_\infty \rightarrow \mathbf{PSys}$ by $\mathcal{P}(\sigma) = \bigcup_i \mathcal{P}_i(\sigma)$.
$P(\sigma)$ is the set of primes of $\mathcal{P}(\sigma)$.

Proposition 2.4.4.

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

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

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

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

□

Definition 2.4.5. 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 *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 *level*$(a) <$ *level*$(0, a)$.
If $(1, a) \in P(\sigma + \tau)$, then $a \in P(\tau)$ and *level*$(a) <$ *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 *level*$(x) <$ *level*(X, a), and $a \in P(\tau)$ and *level*$(a) <$ *level*(X, a).
Therefore definitions and proofs for primes may be given by induction on their parts with smaller level.

2.4.2 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 $\mathit{Pr}_3(f) \in |\mathcal{A} \to (\mathcal{B} \to (\mathcal{C} \to \mathcal{D}))|$ by $\mathit{Pr}_3(f) = \mathit{Pr}(\mathit{Pr}_2 \circ f)$. Note that $\mathit{Pr}_2 \circ f$ is continuous as Pr_2 is continuous. It is $(\mathit{Pr}_3(f))\, a\, b\, c = |(|(|\mathit{Pr}_3(f)|\, a)|\, b)|\, c = f\, a\, b\, c$.

Definition 2.4.6. We define the semantics function $\mathcal{S} : \mathcal{T} \to (\mathit{Env} \to \bigcup_{\sigma \in T^c_\mu} D_\sigma)$ by structural induction on the term argument. We write $\mathcal{S}[\![M]\!]$ and $\mathcal{S}[\![M]\!]\varepsilon$, for $M \in \mathcal{T}$, $\varepsilon \in \mathit{Env}$. It is $\mathcal{S}[\![M]\!] \in [\mathit{Env} \to D_\sigma]$ for $M : \sigma$, see the following proposition.

$$\begin{array}{llll}
\mathcal{S}[\![0_{\sigma,\tau}]\!]\varepsilon & = \mathit{Pr}(0), & \text{with} & 0 : D_\sigma \to D_{\sigma+\tau} \\
& & & 0d = \{0\} \cup (\{0\} \times d) \\
\mathcal{S}[\![1_{\sigma,\tau}]\!]\varepsilon & = \mathit{Pr}(1), & \text{with} & 1 : D_\tau \to D_{\sigma+\tau} \\
& & & 1d = \{1\} \cup (\{1\} \times d) \\
\mathcal{S}[\![\mathsf{case}_{\sigma,\tau\varrho}]\!]\varepsilon & = \mathit{Pr}_3(\mathit{case}), & \text{with} & \mathit{case} : D_{\sigma+\tau} \to D_{\sigma\to\varrho} \to D_{\tau\to\varrho} \to D_\varrho \\
& & & \mathit{case}\, d\, f\, g = \begin{cases} \bot, & \text{if } d = \bot \\ |f|e, & \text{if } d = 0e \\ |g|e, & \text{if } d = 1e \end{cases} \\
\mathcal{S}[\![\mathsf{pcase}_{\sigma,\tau,\varrho}]\!]\varepsilon & = \mathit{Pr}_3(\mathit{pcase}), & \text{with} & \mathit{pcase} : D_{\sigma+\tau} \to D_\varrho \to D_\varrho \to D_\varrho \\
& & & \mathit{pcase}\, a\, b\, c = \begin{cases} b \cap c, & \text{if } a = \bot \\ b, & \text{if } a = 0a' \\ c, & \text{if } a = 1a' \end{cases} \\
\mathcal{S}[\![\mathsf{pair}_{\sigma,\tau}]\!]\varepsilon & = \mathit{Pr}_2(\mathit{pair}), & \text{with} & \mathit{pair} : D_\sigma \to D_\tau \to D_{\sigma\times\tau} \\
& & & \mathit{pair}\, d\, e = (\{0\} \times d) \cup (\{1\} \times e) \\
\mathcal{S}[\![\mathsf{fst}_{\sigma,\tau}]\!]\varepsilon & = \mathit{Pr}(\mathit{fst}), & \text{with} & \mathit{fst} : D_{\sigma\times\tau} \to D_\sigma \\
& & & \mathit{fst}\,(\mathit{pair}\, d\, e) = d \\
\mathcal{S}[\![\mathsf{snd}_{\sigma,\tau}]\!]\varepsilon & = \mathit{Pr}(\mathit{snd}), & \text{with} & \mathit{snd} : D_{\sigma\times\tau} \to D_\tau \\
& & & \mathit{snd}\,(\mathit{pair}\, d\, e) = e
\end{array}$$

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

Proposition 2.4.7. *For all terms $M : \psi$, $\mathcal{S}[\![M]\!] \in [\mathit{Env} \to D_\psi]$.*

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 $\psi \approx (\sigma+\tau) \to \varrho \to \varrho \to \varrho$, 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.

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

• 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 $\mathcal{S}[\![NP]\!] \in [Env \to D_\tau]$. □

2.4.3 Soundness of the semantics

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

Lemma 2.4.8. *(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 2.4.9 (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:

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

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. □

2.5 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 2.5.1. 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 2.5.2. 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, $A \lhd\!\!\!\lhd M$, iff $\exists N.\ M \to^* N$ and $A \lhd N$.
$\mathcal{A}(M)$ denotes the set of approximations of M.
We abbreviate $\overline{\mathcal{S}}[\![M]\!]\varepsilon = \bigcup_{A \lhd\!\!\!\lhd M} \mathcal{S}[\![A]\!]\varepsilon$.

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 2.5.3. *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 ($\mathsf{case0}$) – ($\mathsf{pcase}{\to}$). Let $L = M/u$. Let u_i, $1 \le i \le n$, be a sequence of all the outermost occurrences of λ-abstractions in L. Let x_i, $1 \le i \le n$, 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 $\to$ is strongly normalizing (noetherian) on applicative terms, there is a normal form K' of K, $K \to^* K'$. 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 $L \to^* L'$. As K' is a normal form and the L/u_i are λ-abstractions, L' is not a redex of a constant.
It is $M \to^* M[u \leftarrow L']$, as $L \to^* L'$. 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 2.5.4. *$\mathcal{A}(M)$ 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 $A \lhd\!\!\!\lhd M$ and $B \prec A$, then $B \lhd\!\!\!\lhd M$.
$\mathcal{A}(M)$ is directed: Let $A \lhd\!\!\!\lhd M$ and $A' \lhd\!\!\!\lhd M$. There is N with $M \to^* N \ \wedge\ A \lhd N$, and N' with $M \to^* N' \ \wedge\ A' \lhd N'$. By confluence there is a term P with $N \to^* P$ and $N' \to^* P$. 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 $A \sqcup A' \lhd\!\!\!\lhd M$. □

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 2.5.5. *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 $A \mathrel{\underline{\lhd}} M$. 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 $A \mathrel{\underline{\lhd}} N$. This shows $\mathcal{A}(M) \subseteq \mathcal{A}(N)$. Symmetrically $\mathcal{A}(M) \supseteq \mathcal{A}(N)$. □

Lemma 2.5.6. *Let $cM_1 \dots M_n$ be a term where c is a constant and there are no reducts $M_i \to^* M_i'$, $1 \le i \le m \le n$, with $cM_1' \dots M_m'$ a redex. Then*

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

Proof.

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

We have used the fact that $A \mathrel{\underline{\lhd}} cM_1 \dots M_n$ iff $A = cA_1 \dots A_n$ with some $A_i \mathrel{\underline{\lhd}} M_i$; as no $cM_1 \dots M_m$, $m \le n$, can be reduced to a redex. □

Theorem 2.5.7 (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 2.5.8. 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:

$$\begin{array}{lll}
\sigma = \tau + \varrho : & 0 <^{\tau+\varrho}_\varepsilon M & \Leftrightarrow\ 0 \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \\
& (0,a) <^{\tau+\varrho}_\varepsilon M & \Leftrightarrow\ (0,a) \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \text{ and } a <^\tau_\varepsilon \mathit{Out0}(M) \\
& 1 <^{\tau+\varrho}_\varepsilon M & \Leftrightarrow\ 1 \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \\
& (1,a) <^{\tau+\varrho}_\varepsilon M & \Leftrightarrow\ (1,a) \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \text{ and } a <^\varrho_\varepsilon \mathit{Out1}(M)
\end{array}$$

where $\mathit{Out0}(M)$ abbreviates the term $\mathsf{case}\, M\, (\lambda y.y)\, \Omega$,
and $\mathit{Out1}(M)$ the term $\mathsf{case}\, M\, \Omega\, (\lambda y.y)$.

$$\begin{array}{lll}
\sigma = \tau \times \varrho : & (0,a) <^{\tau\times\varrho}_\varepsilon M & \Leftrightarrow\ (0,a) \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \text{ and } a <^\tau_\varepsilon \mathsf{fst}\, M \\
& (1,a) <^{\tau\times\varrho}_\varepsilon M & \Leftrightarrow\ (1,a) \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \text{ and } a <^\varrho_\varepsilon \mathsf{snd}\, M \\
\sigma = \tau \to \varrho : & (X,a) <^{\tau\to\varrho}_\varepsilon M & \Leftrightarrow\ (X,a) \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \text{ and} \\
& & \quad \forall N \in \mathcal{T}_\tau.\ (X <^\tau_\varepsilon N \ \Rightarrow\ a <^\varrho_\varepsilon MN)
\end{array}$$

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 2.5.9. *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:

$$\begin{array}{rcl} X <_\varepsilon^\tau N & \Rightarrow & Y <_\varepsilon^\tau N, \text{ by induction hypothesis} \\ & \Rightarrow & b' <_\varepsilon^\varrho MN, \text{ as } (Y, b') <_\varepsilon^\sigma M \\ & \Rightarrow & a' <_\varepsilon^\varrho MN, \text{ by induction hypothesis} \end{array}$$

Therefore $a = (X, a') <_\varepsilon^\sigma M$. □

Lemma 2.5.10. *If $a <_\varepsilon^\sigma M$ and $M \downarrow N$, then also $a <_\varepsilon^\sigma N$.*

Proof. by structural induction on a.
We have $\overline{\mathcal{S}}[\![M]\!]\varepsilon = \overline{\mathcal{S}}[\![N]\!]\varepsilon$ by Lemma 2.5.5, therefore $a \in \overline{\mathcal{S}}[\![N]\!]\varepsilon$.

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

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

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

$$\begin{array}{rcl} X <_\varepsilon^\tau P & \Rightarrow & a' <_\varepsilon^\varrho MP, \text{ as } a <_\varepsilon^\sigma M \\ & \Rightarrow & a' <_\varepsilon^\varrho NP, \text{ by induction hyp., as } MP \downarrow NP \end{array}$$

Therefore $a <_\varepsilon^\sigma N$. □

We also need the new notion of passive term:

Definition 2.5.11. 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 (case0)...(pcase→) 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:

$\lambda x.N$, $\mathsf{0}$, $\mathsf{0}N$, $\mathsf{1}$, $\mathsf{1}N$,
pair, $\mathsf{pair}\,N_1$, $\mathsf{pair}\,N_1\,N_2$, fst, snd,
case, $\mathsf{case}\,(\mathsf{0}N)$, $\mathsf{case}\,(\mathsf{0}N_1)\,N_2$, $\mathsf{case}\,(\mathsf{1}N)$, $\mathsf{case}\,(\mathsf{1}N_1)\,N_2$,
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 (pcase→).)
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 2.5.12.

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

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

2) Let $MN \to^* P$. By part 1) we have $P = M'N'$ with $M \to^* M'$. 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 $xM_1 \ldots M_n$.)

3) For all A we have:

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

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:

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

□

Lemma 2.5.13. *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.

$$\begin{aligned}
a' \quad &\in \quad \mathsf{case}\,(\overline{\mathcal{S}}[\![M]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![\lambda y.y]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![\Omega]\!]\varepsilon), \text{ as } a \in \overline{\mathcal{S}}[\![M]\!]\varepsilon \\
&= \quad \overline{\mathcal{S}}[\![\mathsf{case}\, M(\lambda y.y)\,\Omega]\!]\varepsilon, \text{ by Lemma 2.5.6} \\
&= \quad \overline{\mathcal{S}}[\![\mathit{Out0}(M)]\!]\varepsilon.
\end{aligned}$$

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 2.5.12, 2).
$(X, a') \in \overline{\mathcal{S}}[\![M]\!]\varepsilon$ and $X \subseteq \overline{\mathcal{S}}[\![N]\!]\varepsilon$ imply

$$a' \in |\overline{\mathcal{S}}[\![M]\!]\varepsilon|\,(\overline{\mathcal{S}}[\![N]\!]\varepsilon) = \overline{\mathcal{S}}[\![MN]\!]\varepsilon, \text{ by Lemma 2.5.12, 3).}$$

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 2.5.14.
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 2.5.10.
We assume in the following that not $M_0 \to^* \mathsf{0}M_0'$. (Also $M_0 \to^* 1M_0'$ 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 2.5.10 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

$$\begin{array}{rcl}(0, a') & \in & 0\,(\overline{\mathcal{S}}[\![\mathsf{pcase}\, M_0 M_1' M_2']\!]\varepsilon) \\ & = & \overline{\mathcal{S}}[\![\mathsf{0}\,(\mathsf{pcase}\, M_0 M_1' M_2')]\!]\varepsilon, \text{ by Lemma 2.5.6} \\ & = & \overline{\mathcal{S}}[\![M]\!]\varepsilon, \text{ by Lemma 2.5.5.}\end{array}$$

Furthermore $a' <^{\tau}_{\varepsilon} \mathit{Out0}(M)$, as $\mathit{Out0}(M) \to^* \mathsf{pcase}\, M_0 M_1' M_2'$, by Lemma 2.5.10.

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

$$a \in \mathit{pcase}\,(\overline{\mathcal{S}}[\![M_0]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![M_1]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![M_2]\!]\varepsilon) = \overline{\mathcal{S}}[\![M]\!]\varepsilon, \text{ by Lemma 2.5.6.}$$

M is passive (note that M_1, M_2 are not of functional type). By Lemma 2.5.13 we get $a <_\varepsilon^\sigma M$.

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 $\sigma = \tau + \varrho$.

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

$$a \in \mathit{pcase}\,(\overline{\mathcal{S}}[\![M_0]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![M_1]\!]\varepsilon)\,(\overline{\mathcal{S}}[\![M_2]\!]\varepsilon) = \overline{\mathcal{S}}[\![M]\!]\varepsilon, \text{ by Lemma 2.5.6.}$$

It remains to show: $\forall N \in \mathcal{T}_\tau.\ (X <_\varepsilon^\tau N \ \Rightarrow\ a' <_\varepsilon^\varrho MN)$.
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:

$$\begin{aligned} X <_\varepsilon^\tau N \quad &\Rightarrow \quad a' <_\varepsilon^\varrho M_1 N, \text{ as } (X, a') <_\varepsilon^\sigma M_1 \\ &\Rightarrow \quad a' <_\varepsilon^\varrho \mathsf{pcase}\, M_0(M_1N)(M_2N), \text{ by induction hypothesis} \\ &\Rightarrow \quad a' <_\varepsilon^\varrho MN, \text{ by Lemma 2.5.10.} \end{aligned}$$

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 $M_0 \to^* 0M_0'$ for some M_0', then $M \to^* M_1$, and $a <_\varepsilon^\sigma M$ follows from Lemma 2.5.10.
If $M_0 \to^* 1M_0'$ for some M_0', then $M \to^* M_2$, and again $a <_\varepsilon^\sigma M$.
We assume in the following that neither $M_0 \to^* 0M_0'$ nor $M_0 \to^* 1M_0'$. We give a case analysis on a:

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

a) We assume $M_1 \to^* 0M_1'$ and $M_2 \to^* 0M_2'$ for some M_1', M_2'.
Then $M \to^* 0\,(\mathsf{pcase}\, M_0 M_1' M_2')$.
From $a <_\varepsilon^\sigma M_1$, $a <_\varepsilon^\sigma M_2$ we conclude by Lemma 2.5.10 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 ($M_1 \to^* 0M_1'$ and $M_2 \to^* 0M_2'$) 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 $<^\sigma_\varepsilon$ on elements of D_σ that we need in the proof of the Approximation Theorem.

Lemma 2.5.15 (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) $\perp <^\sigma_\varepsilon M$.

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

$$\begin{array}{lcl} 0d <^{\tau+\varrho}_\varepsilon M & \Leftrightarrow & 0d \subseteq \overline{\mathcal{S}}[\![M]\!]\varepsilon \textit{ and } d <^\tau_\varepsilon \textit{Out0}(M) \\ 1d <^{\tau+\varrho}_\varepsilon M & \Leftrightarrow & 1d \subseteq \overline{\mathcal{S}}[\![M]\!]\varepsilon \textit{ and } d <^\varrho_\varepsilon \textit{Out1}(M) \end{array}$$

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

$$\begin{array}{lcl} d <^{\tau\times\varrho}_\varepsilon M & \Leftrightarrow & d \subseteq \overline{\mathcal{S}}[\![M]\!]\varepsilon \textit{ and} \\ & & \textit{fst}\, d <^\tau_\varepsilon \mathsf{fst}\, M \textit{ and } \textit{snd}\, d <^\varrho_\varepsilon \mathsf{snd}\, M \end{array}$$

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

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

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 $\mathcal{S}[\![c]\!]\perp <^\sigma_\varepsilon c$ iff

$$d_i <^{\tau_i}_\varepsilon M_i \textit{ for } 1 \leq i \leq n \Rightarrow (\mathcal{S}[\![c]\!]\perp)d_1 \ldots d_n <^\varrho_\varepsilon cM_1 \ldots M_n.$$

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

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

8) If $0 \in \overline{\mathcal{S}}[\![M_0]\!]\varepsilon$ and $d <^\sigma_\varepsilon M_1$, then $d <^\sigma_\varepsilon \mathsf{pcase}\, M_0M_1M_2$.

9) If $1 \in \overline{\mathcal{S}}[\![M_0]\!]\varepsilon$ and $d <^\sigma_\varepsilon M_2$, then $d <^\sigma_\varepsilon \mathsf{pcase}\, M_0M_1M_2$.

10) If $d_1 <^\sigma_\varepsilon M_1$ and $d_2 <^\sigma_\varepsilon M_2$, then $d_1 \cap d_2 <^\sigma_\varepsilon \mathsf{pcase}\, M_0M_1M_2$.

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 $<^\sigma_\varepsilon$.

4) $\Rightarrow$: $d \subseteq \overline{\mathcal{S}}[\![M]\!]\varepsilon$ is clear.
Now let $e \in D_\tau$, $N \in \mathcal{T}_\tau$ and $e <^\tau_\varepsilon N$.
Let $a \in |d|e$. Then there is $X \subseteq e$ with $(X, a) \in d$.
From $(X, a) <^{\tau\to\varrho}_\varepsilon M$ and $X <^\tau_\varepsilon N$ follows $a <^\varrho_\varepsilon MN$.

$\Leftarrow$: Let $(X, a) \in d$. We show: $\forall N.\ X <^\tau_\varepsilon N \Rightarrow a <^\varrho_\varepsilon MN$.
Let $e = X{\downarrow}$. By Lemma 2.5.9 we get $e <^\tau_\varepsilon N$. Then $a \in |d|e <^\varrho_\varepsilon MN$.

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 $n+1$:

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

Lemma 2.5.6 says $\overline{\mathcal{S}}[\![cM_1 \dots M_n]\!]\varepsilon = (\mathcal{S}[\![c]\!]\bot)\,(\overline{\mathcal{S}}[\![M_1]\!]\varepsilon) \dots (\overline{\mathcal{S}}[\![M_n]\!]\varepsilon)$,
therefore $(\mathcal{S}[\![c]\!]\bot)d_1 \dots d_n \subseteq \overline{\mathcal{S}}[\![cM_1 \dots M_n]\!]\varepsilon$ is fulfilled.

6) Follows from Lemma 2.5.10.

7) Follows from Lemma 2.5.13.

8), 9) and 10) follow from Lemma 2.5.14. □

The Approximation Theorem would be proved if we could show that $\mathcal{S}[\![M]\!]\varepsilon <^{\sigma}_{\varepsilon} M$ 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 2.5.16 (Approximation Lemma). *Let $M \in \mathcal{T}_\sigma$, $\varepsilon \in Env$, $x_i^{\sigma_i}$ $(1 \leq i \leq n, n \geq 0)$ be a sequence of distinct variables, $d_i \in D_{\sigma_i}$ and $N_i \in \mathcal{T}_{\sigma_i}$ for all i.*
If $d_i <^{\sigma_i}_{\varepsilon} N_i$ for all i, then

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

Here $\varepsilon[x_1 \mapsto d_1, \dots, x_n \mapsto d_n]$ is the environment that maps x to $\varepsilon(x)$ if $x \neq x_i$ for all i, and x_i to d_i. $M[x_1{:=}N_1, \dots, x_n{:=}N_n]$ 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 $\overline{\varepsilon'} = \varepsilon'[x_1 \mapsto d_1, \dots, x_n \mapsto d_n]$, and for any term L we write $\overline{L} = L[x_1{:=}N_1, \dots, x_n{:=}N_n]$.
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 2.5.6 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 $M = \mathsf{0}$.

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

• $M = \mathsf{snd}$ is analogous.

• $M = x$:
If $x = x_i$ for some i, then $\mathcal{S}[\![x]\!]\overline{\varepsilon} = d_i <^{\sigma}_{\varepsilon} N_i = \overline{x}$.
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 $\mathcal{S}[\![x]\!]\overline{\varepsilon} <^{\sigma}_{\varepsilon} x$.

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

• $M = \lambda x^{\tau}.M'$, $\sigma = \tau \to \varrho$:
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 $\mathcal{S}[\![\lambda x.M']\!]\overline{\varepsilon} \subseteq \overline{\mathcal{S}}[\![\overline{\lambda x.M'}]\!]\varepsilon$.

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

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

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

Furthermore we have:

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

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

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. $\square$

Corollary 2.5.17. *For all terms M and environments ε:*

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

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

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

Corollary 2.5.18. *The semantics of the fixed point combinator $Y_\sigma = \lambda y^{\sigma\to\sigma}.(\lambda x.y(xx))(\lambda x.y(xx))$ is*

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

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 $\lambda x_1 \ldots x_n.yA_1 \ldots A_m$ with $A_i \in H$. We define

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

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 $M = \mathsf{pcase}\, x\, 0\, \Omega$. It is $M \notin H$, because of the first property of H and as not $M \lhd \mathsf{pcase}\, x\, 0\, 0$. 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 2.5.19. 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 2.5.20. *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 2.5.21. *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 $A = \lambda x_1 \dots x_n . y A_1 \dots A_m$. Then $M = \lambda x_1 \dots x_n . y M_1 \dots M_m$ with $A_i \prec M_i$ for all i.
The term $y M_1 \dots M_m$ 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 $N = \lambda x_1 \dots x_n . y M_1 \dots M_{j-1} N_j M_{j+1} \dots M_m$. 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 2.5.20, the second from Lemma 2.5.21.
$\mathcal{B}(M) \subseteq \mathcal{A}(M)$ 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 2.5.22. *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 $\mathcal{B}(M) \subseteq I$.]

2.6 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 program 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 2.6.1. The set of *programs* is $Prog = \mathcal{T}^c_{\mathsf{bool}}$.
We define the *operational evaluation function* $\mathcal{O} : Prog \to D_{\mathsf{bool}}$ 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 2.6.2. *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 $A = cA_1 \dots A_n$, $n \geq 0$, 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 $\mathcal{S}[\![A_1]\!]\perp \neq \perp$, then $A_1 = \mathsf{0}A_1'$ or $A_1 = \mathsf{1}A_1'$ by induction hypothesis and A is no normal form.
If $\mathcal{S}[\![A_1]\!]\perp = \perp$, then $\mathcal{S}[\![A]\!]\perp = \mathcal{S}[\![A_2]\!]\perp \cap \mathcal{S}[\![A_3]\!]\perp \neq \perp$.
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 2.6.3 (Adequacy). *For all $M \in Prog$: $\mathcal{O}[\![M]\!] = \mathcal{S}[\![M]\!]\perp$.*

Proof. $\mathcal{O}[\![M]\!] \subseteq \mathcal{S}[\![M]\!]\perp$ follows from soundness: If $M \to^* \mathsf{0}M'$, then $\mathcal{S}[\![M]\!]\perp = \mathcal{S}[\![\mathsf{0}M']\!]\perp = 0$; and if $M \to^* \mathsf{1}M'$, then $\mathcal{S}[\![M]\!]\perp = \mathcal{S}[\![\mathsf{1}M']\!]\perp = 1$.
It remains to show the adequacy: $\mathcal{S}[\![M]\!]\perp \subseteq \mathcal{O}[\![M]\!]$.
Suppose $\mathcal{S}[\![M]\!]\perp = 0$. By the Approximation Theorem there is an approximation $A \lhd M$ with $\mathcal{S}[\![A]\!]\perp = 0$. From the preceding lemma follows $A = \mathsf{0}A'$ for some A', therefore $\mathcal{O}[\![M]\!] = 0$. Analogously $\mathcal{S}[\![M]\!]\perp = 1$ 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 2.6.4. 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 2.6.5 (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: $\mathcal{O}[\![C[M]]\!] = \mathcal{S}[\![C[M]]\!]\perp \subseteq \mathcal{S}[\![C[N]]\!]\perp = \mathcal{O}[\![C[N]]\!]$. 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 2.6.6 (Definability). *For all finite $d \in D_\sigma$ there is a closed term $M \in \mathcal{T}_\sigma^c$ with $\mathcal{S}[\![M]\!]\perp = d$.*

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:

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

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

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

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

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

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 $X \subseteq \mathit{Cond}(P_n(\sigma))$ there is $M \in \mathcal{T}_\sigma$ with $M\ \mathit{term}\ X$.

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

We use abbreviations for the following function terms:

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

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) $X = \emptyset$. $\Omega\ \mathit{term}\ X$.

2) $X = \emptyset$. $(\lambda x.0)\ \mathit{eq}\ X$.

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 $C^0 = \{C \mid \exists a \geq 0.\ (C{\to}a) \in X\}$ and
$C^1 = \{C \mid \exists a \geq 1.\ (C{\to}a) \in X\}$.
Define the term $M_0 : \mathsf{bool}$ as $M_0 = 1$ for $C^0 = \emptyset$, otherwise as $M_0 = \mathsf{or}\, C^0_1\, (\mathsf{or}\, C^0_2 \ldots C^0_j)$ for some enumeration $\{C^0_1, C^0_2, \ldots, C^0_j\} = C^0$. Analogously, M_1 is defined as an or-term of the elements of C^1.
Let $X^0 = \{C{\to}a \mid (C{\to}(0,a)) \in X\}$ and $X^1 = \{C{\to}a \mid (C{\to}(1,a)) \in X\}$. 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

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

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

$\star$ $\mathcal{S}[\![M]\!]\varepsilon \subseteq Y{\downarrow}$:
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 $\mathcal{S}[\![C]\!]\varepsilon = 0$. $(C{\to}a') \in X$ 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 $(C{\to}a'') \in X^0$ with $\mathcal{S}[\![C]\!]\varepsilon = 0$ and $a' \leq a''$. $(C{\to}(0, a'')) \in X$, 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. $(C{\to}a) \in X$ 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 $\mathcal{S}[\![M]\!]\varepsilon = 0\,(\mathcal{S}[\![N_0]\!]\varepsilon)$. $(C{\to}a') \in X^0$, 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 $\mathcal{S}[\![M]\!]\varepsilon = 1\,(\mathcal{S}[\![N_1]\!]\varepsilon)$.
$(C{\to}a') \in X^1$, 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 $X^0 = \{C{\to}a \mid (C{\to}(0,a)) \in X\} \subseteq \mathit{Cond}(P_n(\tau))$, and $X^1 = \{C{\to}a \mid (C{\to}(1,a)) \in X\} \subseteq \mathit{Cond}(P_n(\varrho))$. 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 $M = (N_0, N_1)$.

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

• $\sigma = \tau \to \varrho$:
Let $X = \{C_i{\to}(Y_i, a_i) \mid 1 \le i \le k\}$ 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 $Z = \{D_i{\to}a_i \mid 1 \le i \le k\}$ 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 $Z \subseteq \mathit{Cond}(P_n(\varrho))$ 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.

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

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

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

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

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

• $\sigma = \tau + \varrho$:
If $X = \emptyset$, then $(\lambda x.\mathsf{0})\ \mathit{eq}\, X$.
Now let $a \in X$ for some $a \geq 0$. Let $Y = \{a \mid (0, a) \in X\} \subseteq P_n(\tau)$. By induction hypothesis there is some N with $N\ \mathit{eq}\ Y$. Take $M = \lambda x.\mathsf{case}\ x\, N\, \mathsf{1}$. It can be easily checked that $M\ \mathit{eq}\, X$.
The case $a \in X$ for some $a \geq 1$ 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 $N_0\ \mathit{eq}\, X_0$ and $N_1\ \mathit{eq}\, X_1$ by induction hypothesis.
Let $M = \lambda x.\mathsf{and}\ (N_0(\mathsf{fst}\ x))\ (N_1(\mathsf{snd}\ x))$. We check easily that $M\ \mathit{eq}\, X$.

• $\sigma = \tau \rightarrow \varrho$:
If $X = \emptyset$, then $(\lambda x.\mathsf{0})\ \mathit{eq}\, X$.
Otherwise, let $X = \{(Y_i, a_i) \mid 1 \leq i \leq k\}$ be an enumeration of X.
Let $Y_i' = \{\mathsf{0}{\rightarrow}b \mid b \in Y_i\} \subseteq \mathit{Cond}(P_n(\tau))$ for all i, it is finite and consistent. By induction hypothesis there is $N_i\ \mathit{term}\ Y_i'$ for all i. Furthermore, by induction hypothesis there is $Q_i\ \mathit{eq}\, a_i$ 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 $M\ \mathit{eq}\, X$: Let $d \in D_\sigma$.
If $X \subseteq d$, then for all i:
$|\mathcal{S}\llbracket Q_i \rrbracket \bot|\,(|d|\,(\mathcal{S}\llbracket N_i \rrbracket \bot)) = |\mathcal{S}\llbracket Q_i \rrbracket \bot|\,(|d|\,(Y_i{\downarrow})) = 0$, as $a_i \in |d|\,(Y_i{\downarrow})$.
Therefore $|\mathcal{S}\llbracket M \rrbracket \bot|\, d = 0$.
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 $|\mathcal{S}\llbracket Q_j \rrbracket \bot|\,(|d|\,(\mathcal{S}\llbracket N_j \rrbracket \bot)) = 1$, and $|\mathcal{S}\llbracket M \rrbracket \bot|\, d = 1$.
Otherwise, $d \uparrow (Y_i, a_i)$ for all i and $(Y_j, a_j) \notin d$ for some j. Then $|\mathcal{S}\llbracket M \rrbracket \bot|\, d = \bot$.

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 $M\ \mathit{term}\ \{\mathsf{0}{\rightarrow}a \mid a \in X\}$, i.e. $\mathcal{S}\llbracket M \rrbracket \bot = X{\downarrow}$. □

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

First suppose that M and N are closed terms.
Let $a \in \mathcal{S}\llbracket M \rrbracket\varepsilon$. Define $f = (\{a\}, 0){\downarrow} \in D_{\sigma \rightarrow \mathsf{bool}}$. 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 $\{x_1,\dots,x_n\}$. We get $\lambda x_1\dots x_n.M \sqsubseteq \lambda x_1\dots x_n.N$: For all contexts $C[\,]$ apply the context $C[\lambda x_1\dots x_n.[\,]]$ to M and N. For the closed terms follows: $\mathcal{S}[\![\lambda x_1\dots x_n.M]\!]\varepsilon \subseteq \mathcal{S}[\![\lambda x_1\dots x_n.N]\!]\varepsilon$ for all ε. Hence $\mathcal{S}[\![M]\!]\varepsilon \subseteq \mathcal{S}[\![N]\!]\varepsilon$ for all ε. □

2.7 Interdefinability of constants

Our first observation is that case can be defined from pcase and out0, out1 (see page 42 for the def. of out0, 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 $\mathsf{and} = \lambda xy.\mathsf{pcase}\,x\,y\,1$, 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 led 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:

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

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

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

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

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

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

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

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

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

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

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

$Pcase(\theta, \sigma)$ 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

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

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 2.7.1. 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 2.7.2. *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) $c = \bot$:
The case $a \cap b = \bot$ 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 $a = \bot$ is clear.
Now let $a = 0a'$.

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

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 $\mu t_1 \dots \mu t_m.s$ 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 $\tau = \mu t_1 \dots \mu t_m.t$.
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 $\mu t_1 \dots \mu t_m.s$ 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 $\tau = \mu t_1 \dots \mu t_m.s$ 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 2.7.3. *Let θ, σ be admissible arguments in $Pcase(\theta, \sigma)$, as described above. Let ε be an environment with $\varepsilon(\theta(t)) = \mathcal{S}[\![\mathsf{pcase}_{\overline{\theta}(t)}]\!]\bot$ for all t free in σ. Then $\mathcal{S}[\![Pcase(\theta, \sigma)]\!]\varepsilon = \mathcal{S}[\![\mathsf{pcase}_{\overline{\theta}(\sigma)}]\!]\bot$. Especially for all types σ we have: $\mathcal{S}[\![Pcase([\,], \sigma)]\!]\bot = \mathcal{S}[\![\mathsf{pcase}_\sigma]\!]\bot$.*

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

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

• $\sigma = \tau + \varrho$:
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) $c = \bot$:
For $a \cap b = \bot$ it is $f \bot a\,b = \bot$.
Now let $a = 0a'$, $b = 0b'$.

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

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

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

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

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 $a_2, b_2 \in D_{\overline{\theta}(\varrho)}$:

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

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

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

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

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

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

2.8 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 $\mathsf{pcase}\,\mathsf{0}\,MN \to^* M$. 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.

Chapter 3

On Berry's conjectures about the stable order in PCF

Logical Methods in Computer Science, 8(4:7):1-39, 2012, arxiv:1108.0556

Abstract: PCF is a sequential simply typed lambda calculus language. There is a unique order-extensional fully abstract cpo-model of PCF, built up from equivalence classes of terms. In 1979, Gérard Berry defined the stable order in this model and proved that the extensional and the stable order together form a bicpo. He made the following two conjectures:
1) "Extensional and stable order form not only a bicpo, but a bidomain."
We refute this conjecture by showing that the stable order is not bounded complete, already for finitary PCF of second-order types.
2) "The stable order of the model has the syntactic order as its image: If a is less than b in the stable order of the model, for finite a and b, then there are normal form terms A and B with the semantics a, resp. b, such that A is less than B in the syntactic order."
We give counter-examples to this conjecture, again in finitary PCF of second-order types, and also refute an improved conjecture: There seems to be no simple syntactic characterization of the stable order. But we show that Berry's conjecture is true for unary PCF.

For the preliminaries, we explain the basic fully abstract semantics of PCF in the general setting of (not-necessarily complete) partial order models (f-models). And we restrict the syntax to "game terms", with a graphical representation.

3.1 Introduction

PCF is a simple functional programming language, a call-by-name typed lambda calculus with integers and booleans as ground types, some simple sequential operations on the ground types, and a fixpoint combinator. The concept of PCF was formed by Dana Scott in 1969, see the historical document [Sco93]. It is used as a prototypical programming language to explore the relationship between operational and denotational semantics, see the seminal paper of Gordon Plotkin [Plo77].

The (operational) *observational preorder* $M \sqsubseteq_{op} N$ of two terms (of equal type) is defined as: For all contexts $C[\ ]$ of integer type, if $C[M]$ reduces to the integer n, then

$C[N]$ also reduces to the same n. The denotational semantics (the model) assigns to every term M an element $[\![M]\!]$ of a partial order $(D, \sqsubseteq)$ (usually a complete partial order, cpo) as meaning. The model is said to be *(order) fully abstract* if the two orders coincide: $M \sqsubseteq_{op} N \iff [\![M]\!] \sqsubseteq [\![N]\!]$. The standard model of Scott domains and continuous functions is adequate (i.e. the direction $\Longleftarrow$ of the coincidence), but not fully abstract, because the semantic domains contain finite elements that are not expressible as terms, like the parallel or function. First Robin Milner [Mil77] constructed in 1977 a unique fully abstract order-extensional cpo-model of PCF that can be built up from equivalence classes of terms by some ideal completion. The problem to construct a fully abstract model of PCF that does not use the syntax of terms (the "full abstraction problem") was the driving force of the subsequent developments, see also the handbook article [Ong95].

In 1979 Gérard Berry published his PhD thesis [Ber79] with the translated title "Fully abstract and stable models of typed lambda-calculi", which is the main basis of our work. In order to sort out functions like the parallel or from the semantic domains, to get "closer" to the fully abstract model, he gave the definition of stable function: A function f is *stable* if for the computation of some finite part of the output a deterministic minimal part of the input is needed. In the case that there are only finitely many elements smaller than a finite element, this definition is equivalent to the definition of a *conditionally multiplicative function* f: If a and b are compatible, then $f(a \sqcap b) = fa \sqcap fb$. To make the operation of functional application of stable functions itself stable, Berry had to replace the pointwise order of functions, the extensional order, by the new stable order: Two functions are in the *stable order*, $f \leq g$, if for all $x \leq y$: $fx = fy \sqcap gx$. This entails the pointwise order, but it demands in addition that g must not output some result for input x that f outputs only for greater y.

Side remark: Stability is a universal concept that was independently (re)discovered in many mathematical contexts. So Jean-Yves Girard found it in the logical theory of dilators and then transferred it to domain theory (qualitative domain, coherence space) to give a model of polymorphism (system F) [Gir86], thereby independently reinventing Berry's stable functions and stable order, see also the textbook [GTL89], chapter 8 and appendix A. For a general theory of stability and an extensive bibliography see [Tay90].

Now Berry had a model (of PCF) of stable functions with the stable order. But this model did not respect the old (pointwise) extensional order of the standard model and so had new unwanted elements not contained in the standard model. To get a proper subset of the standard model, he introduced bicpo models. A *bicpo* is a set with two orders, an extensional and a stable one, both forming cpos and being connected in some way. He augmented Milner's fully abstract cpo model by the stable order and proved that it consists of bicpos and its functions are conditionally multiplicative. In section 3.3 we show in addition that its stable order forms stable bifinite domains and therefore its functions are also stable and can be represented by *traces*, i.e. sets of tokens (or events) like in [CPW00]. E.g. the function $[\![\lambda f.\,\mathsf{if}(\mathsf{zero}(f0))\ \mathsf{then}\ 0\ \mathsf{else}\ \bot]\!]$ can be represented by the trace consisting of the tokens $\{0 \mapsto 0\} \mapsto 0$ and $\{\bot \mapsto 0\} \mapsto 0$. Functions are in the stable order, $f \leq g$, iff the trace of f is a subset of the trace of g.

In his thesis Berry made the following two conjectures that we refute:

1) "Extensional and stable order in the fully abstract cpo-model of PCF form not only a bicpo, but a bidomain."

This would mean (among other things) that the stable order is bounded complete and distributive. We give counter-examples in finitary PCF of second-order types to this conjecture. The idea is that the stable lub of two stably bounded elements a and b may entail a new token that was not present in a or b. This new token must be used in the syntax to separate a subterm denoting a from a subterm denoting b that cannot be unified in a common term. Therefore distributivity is not fulfilled, stable lubs are not taken pointwise. And worse: There may be a choice between different new tokens to be entailed, then there is a choice between different minimal stable upper bounds of a and b, but there is no stable lub. The minimal stable upper bounds are pairwise stably incompatible, and the extensional lub $a \sqcup b$ is one of them.

2) The extensional order of the fully abstract model coincides with the (syntactic) observational preorder. This leads to the question: Is there a syntactic characterization also for the stable order? Berry made the conjecture:
"The stable order of the model has the syntactic order as its image:
If $a \leq b$ in the stable order, for finite a and b, then there are normal form terms A and B with $[\![A]\!] = a$ and $[\![B]\!] = b$, such that $A \prec B$ in the syntactic order."

Berry proved the converse direction: If $A \prec B$, then $[\![A]\!] \leq [\![B]\!]$, and proved the conjecture for first-order types.

Our simplest counter-example to this conjecture is a situation of four terms $A \prec B \cong C \prec D$, where $\cong$ is observational equivalence, so that $[\![A]\!] \leq [\![D]\!]$, but there is no way to find terms $A' \cong A$, $D' \cong D$ with $A' \prec D'$. The elimination of some token of D depends on the prior elimination of some other token, so that two $\prec$-steps are necessary to get from D down to A.

We further give examples where such a chain of $\prec$-steps (with intermediate $\cong$-steps) of any length is necessary. This proposes an improved conjecture, the "chain conjecture": Instead of $A \prec B$ we demand the existence of a chain between A and B. But we also refute this conjecture. Although stable order and syntactic order are connected, there seems to be no simple syntactic characterization of the stable order in PCF.

All our counter-examples for both conjectures are in finitary PCF of second-order types. They all share a common basic idea: We have a term $M\colon (\iota \to \iota \to \iota) \to \iota$ with two tokens (among others) which are in the simplest form like the tokens $\{\bot\bot{\mapsto}\mathtt{0}\}{\mapsto}\mathtt{0}$ and $\{\bot\mathtt{0}{\mapsto}\mathtt{0}, \mathtt{11}{\mapsto}\mathtt{1}\}{\mapsto}\mathtt{0}$. The function call that realizes $\bot\bot{\mapsto}\mathtt{0}$ resp. $\bot\mathtt{0}{\mapsto}\mathtt{0}$ is at the top level of M, the function call for $\mathtt{11}{\mapsto}\mathtt{1}$ is nested below. We want to eliminate the token $\{\bot\bot{\mapsto}\mathtt{0}\}{\mapsto}\mathtt{0}$. For this the function call for $\mathtt{11}{\mapsto}\mathtt{1}$ must be "lifted" to the top level, but this is not possible due to other tokens of M that have to stay.

The necessary ingredients for the counter-examples are: at least second-order type with some functional parameter of arity at least 2, at least two different ground values $\mathtt{0}$ and $\mathtt{1}$, and the need for nested function calls.

If we restrict the calculus to a single ground value $\mathtt{0}$, we get unary PCF, and in this case both of Berry's conjectures are true: The fully abstract model is a bidomain, in fact it is the standard semantical bidomain construction, proved by Jim Laird in [Lai05]. And we prove that the syntactic order is the image of the stable order, using Laird's proof that every type in unary PCF is a definable retract of some first-order type.

The need for nested function calls is the result of a "restriction" of PCF: There is no operator to test if a function demands a certain argument, so that this information could be used in an if-then-else. Jim Laird has shown that in a language with such control operators (SPCF) nested function calls can be eliminated, and also every type of SPCF is a definable retract of a first-order type [Lai07]. Therefore I am convinced, though I do not prove it here, that also for SPCF the syntactic order is the image of the stable order.

The above mentioned "restriction" of PCF is generally the reason for many irregularities of the semantics of PCF and the difficulty of the full abstraction problem. An important result is the undecidability of finitary PCF [Loa01]. This means that the observational equivalence of two terms of finitary PCF is undecidable, and also the question whether there is a term for a functional value table. As remarked in the introduction to [CPW00], this result restricts the possible fully abstract models of PCF to be not "finitary" in some sense. There have been several solutions for semantical fully abstract models of PCF: A model of continuous functions restricted by Kripke logical relations [OR95], and game semantics [AJM00, HO00, Nic94]. In game semantics a term of PCF is modeled by a strategy of a game, i.e. by a process that performs a dialogue of questions and answers with the environment, the opponent. These strategies are still intensional; the fully abstract model is formed by a quotient, the extensional collapse. The strategies can be identified with PCF Böhm trees of a certain normal form, see also [AC98, section 6.6]. We call these Böhm trees "game terms" and prove that it is sufficient to formulate all our results in the realm of game terms, esp. that if two terms are syntactically ordered, then there are equivalent game terms so ordered. This simplifies the proofs of the counter-examples. We also introduce a graphical notation for game terms that facilitates the handling of larger examples.

It was an open problem whether the game model is isomorphic to Milner's fully abstract cpo-model, i.e. whether its domains are cpos. This problem was solved by Dag Normann [Nor06]: Its domains are not cpos, i.e. there are directed sets that have no lub. Then Vladimir Sazonov made a first attempt to build a general theory for these non-cpo domains [Saz07, Saz09, NS12]. His main insight was that functions are continuous only with respect to certain lubs of directed sets that he calls "natural lubs"; these are the hereditarily pointwise lubs.

We want to place our results in the context of these new, more general models. For the semantic preliminaries we give a simple definition of a set of well-behaved (not-necessarily complete) partial order fully abstract models of PCF: These *f-models* are sets of ideals of finite elements, such that application is defined and every PCF-term has a denotation. Sazonov's natural lubs correspond to our *f-lubs*, which are defined with respect to the finite elements.

I found the counter-example to Berry's second conjecture around the year 1990, but did not yet publish it. As far as I know, nobody else tackled Berry's problems. The reason for this seems to be that they were simply forgotten. The stable order in the fully abstract model was never explored after Berry; a reason may be that he never prepared a journal version of his thesis, which is not easily accessible. The recommended introduction to our subject is the report “Full abstraction for sequential languages: The state of the art” [BCL85], which contains the thesis in condensed form, but lacks most proofs. There is also an article [Ber78] published by Berry before his thesis, which is not recommended, because section 4.5 (bidomains) is wrong (different definition of bidomain, the first conjecture is stated as theorem). An excellent general introduction to domains, stability and PCF (and many other things) is the textbook [AC98]. But for the stable order in the fully abstract model of PCF the only detailed source remains Berry’s thesis.

Here is the structure of the paper. The counter-examples are given in the order of their discovery, i.e. in the order of increasing complexity.

2. Syntax of PCF.

3. Semantics of PCF: non-complete partial order f-models:
   We introduce f-models as general (not-necessarily complete) partial order fully abstract models of PCF and give the properties of the stable order in this general context. (The order-extensional fully abstract cpo-model of PCF is a special case.)

4. Game terms:
   We describe the construction of game terms by the finite projections and give a graphical notation for game terms.
   The expert who is interested only in the counter-examples may skip the introductory sections 2-4; reading only the definition of game terms and their graphical notation at the beginning of section 4.

5. The syntactic order is not the image of the stable order:
   We prove Berry’s second conjecture for first-order types, give a counter-example in a second-order type (a chain of length 2), and prove the existence of chains of any least length.

6. The stable order is not bounded complete: no bidomain:
   We prove Berry’s first conjecture for first-order types. In a second-order type we give an example of a stable lub that does not fulfill distributivity, and an example of two stably bounded elements without stable lub.

7. Refutation and improvement of the chain-conjecture:
   We refute the improved second conjecture that the stable order entails a chain of terms. We propose in turn an improvement of the chain conjecture, based on the complementary syntactic relation of strictification.

8. Unary PCF:
   We prove Berry’s second conjecture for unary PCF, with the aid of Jim Laird’s definable retractions from any type to some first-order type [Lai05].

9. Outlook.

3.2 Syntax of PCF

In this section we give the syntactic definitions of PCF [Plo77, BCL85, AC98]. The programming language PCF is a simply typed lambda calculus with arithmetic and fixpoint operators. It usually comes with two ground types ι (integers) and o (booleans). We simplify the language and use only the ground type ι (integers); the booleans are superfluous and can be coded as integers, the intensional structure of the terms stays the same.

The **types** are formed by ι and function types $\sigma \to \tau$ for types σ and τ.

The typed **constants** are:
$0, 1, 2, \ldots : \iota$, the integers;
$\mathsf{suc}, \mathsf{pre} \colon \iota \to \iota$, successor and predecessor function;
$\mathsf{if}\,\_\,\mathsf{then}\,\_\,\mathsf{else}\,\_ \colon \iota \to \iota \to \iota \to \iota$, this conditional tests if the first argument is 0.
(We write e.g. $\mathsf{if}\, x\, \mathsf{then}\, y$ for the application of this function to only two arguments.)

The PCF **terms** comprise the constants and the typed constructs by the following rules:
$\perp^\sigma \colon \sigma$ for any type σ, the undefined term.
$x^\sigma \colon \sigma$ for any variable x^σ.
If $M \colon \tau$, then $\lambda x^\sigma.M \colon \sigma \to \tau$, lambda abstraction.
If $M \colon \sigma \to \tau$ and $N \colon \sigma$, then $MN \colon \tau$, function application.
If $M \colon \sigma \to \sigma$, then $\mathsf{Y}M \colon \sigma$, Y is the fixpoint operator.

PCF$^\sigma$ is the set of all PCF terms of type σ, and PCF$_c^\sigma$ is the set of the closed terms of these.
Type annotations of $\perp$ and of variables will often be omitted.
We use the (semantic) symbol $\perp$ also as syntactic term, instead of the usual Ω.
We define the *syntactic order* $\prec$ (also called $\perp$-match order in the literature) on terms of the same type:
$M \prec N$ iff N can be obtained by replacing some occurrences of $\perp$ in M by terms.

The **reduction rules** are (where n is a variable for integer constants):
$(\lambda x.M)N \to M[x := N]$, the usual β-reduction;
$\mathsf{Y}M \to M(\mathsf{Y}M)$;
$\mathsf{suc}\, n \to (n+1)$;
$\mathsf{pre}\, n \to (n-1)$, for $n \geq 1$;
$\mathsf{if}\, 0\, \mathsf{then}\, M\, \mathsf{else}\, N \to M$;
$\mathsf{if}\, n\, \mathsf{then}\, M\, \mathsf{else}\, N \to N$, for $n \geq 1$.

The *reduction relation* $\to$ is one step of reduction by these rules in any term context. It is confluent. $\to^*$ is the reflexive, transitive closure of $\to$.

A *program* is a closed term of type ι.
The *operational (observational) preorder* $\sqsubseteq_{op}$ on terms of the same type is defined as:
$M \sqsubseteq_{op} N$ (M is *operationally less defined than* N) iff
$P[M] \to^* n$ implies $P[N] \to^* n$ for all contexts $P[\ ]$ such that $P[M]$ and $P[N]$ are both programs.
The *operational equivalence* is defined as: $M \cong N$ iff $M \sqsubseteq_{op} N$ and $N \sqsubseteq_{op} M$.

3.3 Semantics of PCF: non-complete partial order f-models

This section gives an exposition of the fully abstract semantics of PCF with the stable order, as far as it is needed to understand the results of this paper. The proofs are omitted, as they are easy and/or already known in some form.

The order-extensional fully abstract cpo-model of PCF was first constructed by Robin Milner [Mil77] based on terms of an SKI-combinator calculus. Later Gérard Berry's thesis [Ber79] constructed this model based on the proper λ-terms. This model is the ideal completion of the finite elements; every directed set has a lub.

Then came the fully abstract game models of PCF [AJM00, HO00, Nic94]. The elements of these models can be represented by the (infinite) Böhm trees of PCF. It was an open problem whether the game model is isomorphic to Milner's model, i.e. whether its domains are cpos.

This problem was solved by Dag Normann [Nor06]: Its domains are not cpos, i.e. there are directed sets that have no lub. Then Vladimir Sazonov made a first attempt to build a general theory for these non-cpo domains [Saz07, Saz09, NS12]. His main insight was that functions are continuous only with respect to certain lubs of directed sets that he calls "natural lubs"; these are the hereditarily pointwise lubs.

We want to place our results in the context of these new, more general models. Therefore we give a simple definition of a set of well-behaved (not-necessarily complete) partial order fully abstract models of PCF: These *f-models* are sets of ideals of finite elements, such that application is defined and every PCF-term has a denotation. Sazonov's natural lubs correspond to our *f-lubs*, which are defined with respect to the finite elements.

We state the usual properties for these f-models; the essence of their proofs is already contained in Berry's construction. Our aim is the definition of the stable order and of conditionally multiplicative (cm) functions. All functions in f-models are cm. We can further show, in addition to Berry, that the domains have property I under the stable order and therefore the functions are stable and we can work with their traces.

We need the following PCF terms, the *finite projections* on type σ of *grade* i, $\Psi_i^\sigma : \sigma \to \sigma$:

$$\begin{aligned}\Psi_i^\iota &= \lambda x^\iota.\,\mathsf{if}\ x\ \mathsf{then}\ 0\ \mathsf{else}\ \mathsf{if}\ \mathsf{pre}^1\,x\ \mathsf{then}\ 1\ \mathsf{else}\ \ldots\ \mathsf{if}\ \mathsf{pre}^i\,x\ \mathsf{then}\ i\ \mathsf{else}\ \bot\\ \Psi_i^{\sigma\to\tau} &= \lambda f^{\sigma\to\tau}.\lambda x^\sigma.\Psi_i^\tau(f(\Psi_i^\sigma x))\end{aligned}$$

We also need the following terms for the glb functions on all types, $\inf^\sigma : \sigma \to \sigma \to \sigma$, here in a liberal syntax:

$$\begin{aligned}\inf{}^\iota &= \lambda x^\iota y^\iota.\,\mathsf{if}\ x = y\ \mathsf{then}\ x\ \mathsf{else}\ \bot\\ &= \lambda x^\iota y^\iota.\,\mathsf{if}\ x\ \mathsf{then}\ \mathsf{if}\ y\ \mathsf{then}\ 0\ \mathsf{else}\ \bot\\ &\qquad\qquad\qquad \mathsf{else}\ \mathsf{suc}(\inf{}^\iota(\mathsf{pre}\,x)(\mathsf{pre}\,y))\\ \inf{}^{\sigma\to\tau} &= \lambda f^{\sigma\to\tau} g^{\sigma\to\tau}.\lambda x^\sigma.\inf{}^\tau(fx)(gx)\end{aligned}$$

When applied to a closed term $M : \sigma$, the function term Ψ_i^σ serves as a "filter" that lets only pass integer values $\leq i$ as input or output to M. This serves to define the finite elements of the intended model.

Definition 3.3.1. A term $M\colon\sigma$ is a *finite term of grade i* if it is closed and $M \cong \Psi_i^\sigma M$.
$\mathcal{F}_i^\sigma = \{\, [\Psi_i^\sigma M]_{op} \mid M \in \mathrm{PCF}_c^\sigma \,\}$ is the set of *finite elements of grade i* of type σ,
where $[X]_{op}$ is the equivalence class of term X under the operational equivalence $\cong$.
$\mathcal{F}^\sigma = \bigcup_i \mathcal{F}_i^\sigma$ is the set of *finite elements* of type σ.

The finite elements are partially ordered by the extension of the operational preorder $\sqsubseteq_{op}$ to equivalence classes.
An *ideal* of finite elements of type σ is a set $S \subseteq \mathcal{F}^\sigma$ such that: $S \neq \emptyset$ and
$a, b \in S \Longrightarrow \exists c \in S.\ a \sqsubseteq_{op} c$ and $b \sqsubseteq_{op} c$,
and $a \in S, b \in \mathcal{F}^\sigma$ and $b \sqsubseteq_{op} a \Longrightarrow b \in S$.
$I(\mathcal{F}^\sigma)$ is the set of ideals of finite elements of type σ.
There is an operation apply on ideals of finite elements. For $f \in I(\mathcal{F}^{\sigma\to\tau})$, $d \in I(\mathcal{F}^\sigma)$:

$$\mathrm{apply}(f,d) = \downarrow\{\, f'd' \mid f' \in f, d' \in d \,\} \in I(\mathcal{F}^\tau),$$

where $f'd' = [MN]_{op}$ for $M \in f'$, $N \in d'$. $\mathrm{apply}(f,d)$ is simply written fd.
From now on $a \in \mathcal{F}^\sigma$ is identified with the ideal $\downarrow\{a\}$, the downward closure w.r.t. $\sqsubseteq_{op}$ of $\{a\}$. So we have the embedding $\mathcal{F}^\sigma \subseteq I(\mathcal{F}^\sigma)$.

Definition 3.3.2. An *f-model* of PCF ("f" means: based on finite elements) is a collection of $D^\sigma \subseteq I(\mathcal{F}^\sigma)$ for every type σ, each D^σ ordered by inclusion $\subseteq$ written $\sqsubseteq$,
such that for $f \in D^{\sigma\to\tau}$, $d \in D^\sigma$: $fd \in D^\tau$,
and such that every closed term $M\colon\sigma$ has its denotation in D^σ: $\downarrow\{\, [\Psi_i^\sigma M]_{op} \mid i \geq 0 \,\} \in D^\sigma$.
The lubs w.r.t. $\sqsubseteq$ will be written $\sqcup$ and $\bigsqcup$, the glbs $\sqcap$.

All f-models coincide on their part of the finite elements w.r.t. both extensional $\sqsubseteq$ and stable $\leq$ order. In the following sections, propositions will mostly deal with finite elements. The propositions are valid for all f-models if not otherwise stated.

To every f-model we can associate the *semantic map* $[\![\ \ ]\!]\colon \mathrm{PCF}^\sigma \to \mathrm{ENV} \to D^\sigma$, where ENV is the set of environments ρ that map every variable x^σ to some $\rho(x^\sigma) \in D^\sigma$. If $M\colon\sigma$ is a term with the free variables $x_1, \ldots, x_n$, then

$$[\![M]\!]\rho = \downarrow\{\, [\Psi_i^\sigma M[x_1 := N_1, \ldots, x_n := N_n]]_{op} \mid i \geq 0, [N_j]_{op} \in \rho(x_j) \,\}.$$

For closed terms M we also write $[\![M]\!]$ for $[\![M]\!]\bot$.

There are three outstanding examples of f-models: There is the least f-model that consists of just the ideals denoting closed PCF-terms. There is the greatest f-model consisting of all ideals; this is Milner's and Berry's cpo-model. And there is the game model consisting of all denotations of (infinite) PCF-Böhm-trees, i.e. the sequential functionals. By Normann's result [Nor06] we know that the game model is properly between the least and the greatest f-models.

Now we will collect the most important properties of f-models. In the following the D^σ are the domains of some f-model.

Lemma 3.3.3. *Every $\mathcal{F}_i^\sigma$ has finitely many elements.*
The semantics of the inf^σ*-terms are the glb-functions with respect to the order $\sqsubseteq$; we write $\sqcap$ for these functions.*
If $d, e \in \mathcal{F}_i^\sigma$, then $d \sqcap e \in \mathcal{F}_i^\sigma$.
If $d, e \in \mathcal{F}_i^\sigma$ are compatible (bounded), i.e. there is some $a \in D^\sigma$ with $d \sqsubseteq a$ and $e \sqsubseteq a$, then there is a lub $d \sqcup e \in \mathcal{F}_i^\sigma$.

With this lemma we can prove:

Proposition 3.3.4. *All $D^{\sigma\to\tau}$ are order-extensional, i.e. :*

$$\begin{aligned} \textit{If } f, g \in D^{\sigma\to\tau}, \ \textit{then } f \sqsubseteq g &\iff \forall d \in D^{\sigma}.\, fd \sqsubseteq gd \\ &\iff \forall d \in \mathcal{F}^{\sigma}.\, fd \sqsubseteq gd \end{aligned}$$

Elements of $D^{\sigma\to\tau}$ will be identified with the corresponding functions. apply and these functions are all monotone. They are continuous with respect to certain directed lubs, the f-lubs.

Definition 3.3.5. The directed set $S \subseteq D^{\sigma}$ *has the f-lub* $s \in D^{\sigma}$, written $S \to s$, iff s is an upper bound of S and for all finite $x \sqsubseteq s$ there is some $y \in S$ with $x \sqsubseteq y$. (This is equivalent to: s is the set-theoretical union of S. s is also the lub of S w.r.t. $\sqsubseteq$.)
A function $f\colon D^{\sigma} \to D^{\tau}$ is *f-continuous*, iff it is monotone and respects f-lubs of directed sets $S \subseteq D^{\sigma}$, i.e. if $S \to s$, then $fS \to fs$. (With $fS = \{\, fx \mid x \in S \,\}$.)

Proposition 3.3.6. *The* apply *operation is f-continuous on the domain $D^{\sigma\to\tau} \times D^{\sigma}$. (With component-wise order and pairs of finite elements as finite elements.) Therefore* apply *is f-continuous in each argument, and the functions of $D^{\sigma\to\tau}$ are f-continuous.*

In [NS12] it is shown that in the game model there are lubs of directed sets that are not f-lubs; and that there are finite elements that are not compact in the usual sense with respect to general directed lubs.

The f-lubs are exactly the directed lubs for which all functions are continuous: If we have a directed lub that is not an f-lub, then this lub contains a finite element that is not contained in the directed set. The PCF-function that "observes" (or "tests") this finite element is a function that is not continuous for the directed set.

In the greatest f-model all lubs of directed sets are f-lubs. If $S \to s$ in the greatest f-model, then the same holds in all f-models that contain s and the elements of S.

In an f-model we can define natural lubs in the sense of Sazonov as hereditarily pointwise lubs. Then a directed set S has the f-lub s iff S has the natural lub s.

Side remark: Here we must also mention the "rational chains" of Escardó and Ho [EH09]. These are ascending sequences of PCF terms that can be defined syntactically by a PCF procedure. The denotations (in any f-model) of the elements of a rational chain always form a directed set with an f-lub (natural lub). The converse does not hold generally.

Proposition 3.3.7. *The semantic map of an f-model fulfills the usual equations, i.e. the constants have their intended meanings, and:*

$$\begin{aligned} [\![\lambda x.M]\!]\rho d &= [\![M]\!]\rho[x := d] \\ [\![MN]\!]\rho &= [\![M]\!]\rho[\![N]\!]\rho \\ [\![\mathsf{Y}M]\!]\rho &= \bigsqcup_{n\geq 1}(([\![M]\!]\rho)^{n}\bot) \end{aligned}$$

Proposition 3.3.8 (Berry, 3.6.11 in [Ber79]). *Define the functions $\psi_i^\sigma = [\![\Psi_i^\sigma]\!]\bot \colon D^\sigma \to D^\sigma$. For all σ, (ψ_i^σ) is an increasing sequence of finite projections with f-lub the identity* id*:*

$$\begin{aligned} \psi_i^\sigma &\sqsubseteq \mathrm{id} \\ \psi_i^\sigma \circ \psi_i^\sigma &= \psi_i^\sigma, \text{ with } \circ \text{ function composition} \\ \psi_i^\sigma &\sqsubseteq \psi_{i+1}^\sigma \\ \{\, \psi_i^\sigma \mid i \geq 0 \,\} &\to \mathrm{id} \\ \psi_i^\sigma(D^\sigma) &= \mathcal{F}_i^\sigma \end{aligned}$$

Proposition 3.3.9. *Every f-model is fully abstract for PCF: For all terms M, N of the same type*

$$(\forall \rho \in \mathrm{ENV}.\ [\![M]\!]\rho \sqsubseteq [\![N]\!]\rho) \iff M \sqsubseteq_{op} N.$$

In the rest of this section we will define the stable order in f-models and collect the corresponding properties that will be needed in this paper.

The definition of the stable order $\leq$ is given by Berry [Ber79, 4.8.6, page 4-93] for the fully abstract cpo-model as follows:

$$\begin{aligned} &\text{For } d, e \in D^\iota : d \leq e \iff d \sqsubseteq e \\ &\text{For } f, g \in D^{\sigma\to\tau} : f \leq g \iff \forall x \in D^\sigma.\ fx \leq gx \text{ and} \\ &\qquad\qquad \forall x, y \in D^\sigma.\ x \uparrow_\leq y \implies fx \sqcap gy = fy \sqcap gx \end{aligned}$$

(Here $\uparrow_\leq$ means compatibility w.r.t. $\leq$.)
This definition serves as well for our f-models, but I prefer the equivalent (w.r.t. the full type hierarchy) form:

Definition 3.3.10 (stable order $\leq$).

$$\begin{aligned} &\text{For } d, e \in D^\iota : d \leq e \iff d \sqsubseteq e \\ &\text{For } f, g \in D^{\sigma\to\tau} : f \leq g \iff \forall x, y \in \mathcal{F}^\sigma.\ x \leq y \implies fx = fy \sqcap gx \end{aligned}$$

The order $\leq$ is extended pointwise to environments from ENV, here used in the definition of $\leq$ on denotations:

$$\text{For } f, g \in \mathrm{ENV} \to D^\sigma : f \leq g \iff \forall \rho, \varepsilon \in \mathrm{ENV}.\ \rho \leq \varepsilon \implies f\rho = f\varepsilon \sqcap g\rho$$

The lubs w.r.t. $\leq$ will be written $\vee$ and $\bigvee$, the glbs $\wedge$.

Note that $\sqcap$ is by definition the glb w.r.t. the *extensional* order $\sqsubseteq$. But we can prove the following:

Proposition 3.3.11. *In any actual f-model the following holds:*
For $f, g \in D^\sigma$: If f, g are $\leq$-compatible in the greatest f-model, then $f \sqcap g$ is also the glb w.r.t. $\leq$. (Note: If f, g are $\leq$-compatible in the actual f-model, then they are also compatible in the greatest f-model.)
If $f \leq g$ then $f \sqsubseteq g$. $\leq$ is a partial order on D^σ.

$$\begin{aligned} &\textit{For } f, g \in D^{\sigma\to\tau} : f \leq g \iff \forall x \in D^\sigma.\ fx \leq gx \textit{ and} \\ &\qquad\qquad \forall x, y \in D^\sigma.\ x \leq y \implies fx = fy \sqcap gx \end{aligned}$$

The definition of $\leq$ can be given in "uncurried" form with vectors of arguments, the order $\leq$ extended componentwise:

$$\begin{aligned} \textit{For } f,g \in D^{\sigma_1\to\ldots\to\sigma_n\to\iota} : f \leq g \iff & \forall x_1, y_1 \in D^{\sigma_1}, \ldots, x_n, y_n \in D^{\sigma_n}. \\ & (x_1,\ldots,x_n) \leq (y_1,\ldots,y_n) \implies \\ & fx_1\ldots x_n = fy_1\ldots y_n \sqcap gx_1\ldots x_n \end{aligned}$$

Proof. The proof that $f \sqcap g$ is the glb w.r.t. $\leq$ (for $\leq$-compatible f, g) is by induction on the type σ. It uses only the definition of $\leq$ and that $\sqcap$ is the glb w.r.t. $\sqsubseteq$, no stability (or conditional multiplicativity) is used. □

Definition 3.3.12. $f \in D^{\sigma\to\tau}$ is *conditionally multiplicative (cm)* if

$$\forall x, y \in \mathcal{F}^\sigma.\ x \uparrow_\leq y \implies f(x \sqcap y) = fx \sqcap fy$$

Analogously for denotations $f \in \mathrm{ENV} \to D^\sigma$.

This definition can also be given in "uncurried" form: $f \in D^{\sigma_1\to\ldots\to\sigma_n\to\iota}$ is cm iff

$$\begin{aligned} \forall x_1, y_1 \in D^{\sigma_1}, \ldots, x_n, y_n \in D^{\sigma_n}.\ (x_1,\ldots,x_n) \uparrow_\leq (y_1,\ldots,y_n) \implies & \\ f(x_1 \sqcap y_1)\ldots(x_n \sqcap y_n) = fx_1\ldots x_n \sqcap fy_1\ldots y_n & \end{aligned}$$

Theorem 3.3.13 (Berry, 4.8.10 in [Ber79])**.** *In an f-model, all functions from domains $D^{\sigma\to\tau}$ are cm. All denotations $[\![M]\!]$ are cm.*

Proof. Berry first proves the property cm for the denotations of normal form terms by induction on the size of the type. Then it is extended to all functions by continuity. □

Proposition 3.3.14 (Berry [Ber79], syntactic monotony w.r.t. $\leq$)**.**
For every context $C[\quad]$ with hole of type σ, and terms $M, N \colon \sigma$:
If $[\![M]\!] \leq [\![N]\!]$ then $[\![C[M]]\!] \leq [\![C[N]]\!]$.
Therefore, for terms $M, N \colon \sigma$: If $M \prec N$ then $[\![M]\!] \leq [\![N]\!]$.

We will also write $M \leq N$ for $[\![M]\!] \leq [\![N]\!]$.

Now we show property I of $(D^\sigma, \leq)$ and the representation of all functions by traces, which is not contained in Berry's thesis.

Proposition 3.3.15. *For the finite projections we have: $\psi_i^\sigma \leq \psi_{i+1}^\sigma$ and $\psi_i^\sigma \leq \mathrm{id}$.*
The $\mathcal{F}_i^\sigma$ are downward closed w.r.t. $\leq$: If $d \in \mathcal{F}_i^\sigma$, $e \in D^\sigma$ and $e \leq d$, then $e \in \mathcal{F}_i^\sigma$.
Therefore the domains $(D^\sigma, \leq)$ have the property I: There are only finitely many elements under each finite element.

Proof. The proof of $\psi_i^\sigma \leq \mathrm{id}$ is by induction on the type σ; the induction step is in the proof of proposition 12.4.4 in the section on stable bifinite domains of [AC98, page 287]. The downward closedness of $\mathcal{F}_i^\sigma$ is an easy consequence and can be found at the same place. □

Because of property I, all our functions of $D^{\sigma\to\tau}$ (which are cm) are also stable, and therefore can be represented by traces. We chose the trace of the uncurried form.

Definition 3.3.16. Let $f \in D^{\sigma_1\to\dots\to\sigma_n\to\iota}$, $n \geq 0$, $x_i \in D^{\sigma_i}$ and $fx_1\dots x_n = j$ for some integer j.
Then there are $y_i \in \mathcal{F}^{\sigma_i}$, $y_i \leq x_i$, with $fy_1\dots y_n = j$ and $(y_1,\dots,y_n)$ is the $\leq$-least vector with this property. (This is the meaning of: f is stable.)
In this case we say that $y_1{\mapsto}\dots{\mapsto}y_n{\mapsto}j$ is a *token* of f.
The set of all tokens of f is called the *trace* of f, written $\mathcal{T}(f)$.
The y_i in the token will be represented by traces again. We will use a liberal syntax for tokens and traces, writing $\perp$ for the trace $\emptyset$, $\mathsf{0}$ for the trace $\{\mathsf{0}\}$ of $\mathsf{0}$, and also $\mathsf{00}{\mapsto}\mathsf{0}$ for the token $\{\mathsf{0}\}{\mapsto}\{\mathsf{0}\}{\mapsto}\mathsf{0}$. If M is a closed term, we write simply $\mathcal{T}[\![M]\!]$ for the trace of its denotation $[\![M]\!]\perp$.

Proposition 3.3.17. *For $f, g \in D^\sigma$: $f \leq g$ iff $\mathcal{T}(f) \subseteq \mathcal{T}(g)$.*
If f, g are $\leq$-compatible in the greatest f-model, then $\mathcal{T}(f \sqcap g) = \mathcal{T}(f) \cap \mathcal{T}(g)$.
$f \in D^\sigma$ is finite of grade i, $f \in \mathcal{F}^\sigma_i$, iff all numbers in the trace of f are $\leq i$.

3.4 Game Terms

Berry's conjectures demand the existence of certain finite PCF-terms. In this section we show that we may restrict these finite terms to terms in a certain standard normal form that we call *game terms*. This will simplify the proofs of the counter-examples, and is also an interesting result itself. Game terms first appeared in the literature on game semantics as terms representing game strategies; in [AJM00, section 3.2] they were called (finite and infinite) "evaluation trees", in [HO00, section 7.3] "finite canonical forms" that correspond to compact innocent strategies, and in [AC98, section 6.6] "PCF Böhm trees". The textbook article on "PCF Böhm trees" comes closest to our approach, as it introduces a semantics in the form of Böhm trees and has to solve similar problems in the needed syntactic transformations. But we do not employ a (game or other) semantics, i.e. we do not interpret the PCF-constants by infinite strategies or Böhm trees; our approach is purely syntactic. We take a finite PCF-term, apply an operator that resembles the finite projection Ψ^σ_i and reduce the resulting term to its game term form. We show that the transforming reductions respect the syntactic order $\prec$ (used in the refutation of Berry's second conjecture), and this will also enable us to proceed to infinite game terms. We also introduce a graphical representation of game terms that makes the behaviour of terms better visible.

First we introduce an additional new construct for the PCF language, for every $i \geq 0$:
If $M, N_0, \dots, N_i \colon \iota$, then $\mathsf{case}_i\, MN_0\dots N_i \colon \iota$.
Please note that case_i is not a constant, but the whole case-expression is a new construct of the language, it is no application. We call the new terms (PCF-)case-terms, and a case-term with all case-expressions as case_i for fixed i we call case_i-term. The reduction rule for case_i is:

$$\mathsf{case}_i\, nN_0\dots N_i \to N_n, \text{ for } 0 \leq n \leq i$$

The case-expression is equivalent to a PCF-term:

$$\mathsf{case}_i\, MN_0\dots N_i \cong \mathsf{if}\ M\ \mathsf{then}\ N_0\ \mathsf{else\ if\ pre}^1\, M\ \mathsf{then}\ N_1\ \mathsf{else}\dots\mathsf{if\ pre}^i\, M\ \mathsf{then}\ N_i\ \mathsf{else}\ \perp$$

This is the "filter" as it appears in the finite projection term Ψ_i^ι. So case_i does not enhance the expressiveness of PCF. It is merely a "macro" that is used as short expression for the filter term above, to keep the unity of the filter term in the transformation to game terms.

The syntactic order $\prec$ is defined on case-terms as follows:

$$\mathsf{case}_i\, MN_0 \dots N_i \prec \mathsf{case}_j\, M'N_0' \dots N_j' \text{ iff } i \leq j,\ M \prec M' \text{ and } N_k \prec N_k' \text{ for } 0 \leq k \leq i.$$

This is equivalent to the syntactic order on the macro expansions of the case-expressions.

Definition 3.4.1. *Game terms* are the well-typed PCF-case-terms that are furthermore produced by the following grammar:

$$\begin{aligned} M, N ::= {} & \bot^\sigma,\ \sigma \text{ any type} \\ & \lambda x_1 \dots x_n.m,\ m \text{ integer constant},\ n \geq 0 \\ & \lambda x_1 \dots x_n.\,\mathsf{case}_i(yM_1 \dots M_m)N_0 \dots N_i,\ y \text{ variable},\ n, m, i \geq 0 \end{aligned}$$

Please note that $\lambda x_1 \dots x_n.$ vanishes for $n = 0$, so needed for the N_k of type ι.

A *game term of grade* i, $i \geq 0$, is a game term that is a case_i-term (every case is case_i) with all integer constants $\leq i$. (This entails that a closed game term of grade i is a finite term of grade i.)

A *game term of pregrade* i, $i \geq 0$, is a game term that is furthermore produced by the following grammar for the non-terminal N:

$$\begin{aligned} N ::= {} & \bot^\sigma,\ \sigma \text{ any type} \\ & \lambda x_1 \dots x_n.m,\ m \text{ integer constant},\ n \geq 0 \\ & \lambda x_1 \dots x_n.\,\mathsf{case}_i(yM_1 \dots M_m)N_0 \dots N_i,\ y \text{ variable, all } M_k \text{ game term of grade } i, \\ & \qquad n, m \geq 0 \end{aligned}$$

(A game term of pregrade i is a case_i-term.)

Informally, we call the positions in a game term of integer constants at the top level, i.e. where this integer serves as output of the term, *output positions.* So a game term of pregrade i is a game term such that for all integer constants m that are *not* in output position it is $m \leq i$. (So the integers at output positions are not restricted.)

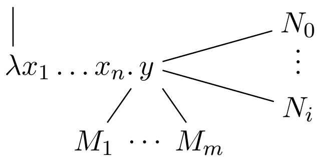
We define a notion for the replacement of integers in output positions of game terms.

Definition 3.4.2. Let P, L be game terms, $L \colon \iota$ and $l \geq 0$. We define $P\lceil l := L\rceil$ by recursion on P:

$$\begin{aligned} \bot\lceil l := L\rceil &= \bot \\ (\lambda x_1 \dots x_n.l)\lceil l := L\rceil &= \lambda x_1 \dots x_n.L \\ (\lambda x_1 \dots x_n.m)\lceil l := L\rceil &= \lambda x_1 \dots x_n.m, \text{ for } m \neq l \\ (\lambda x_1 \dots x_n.\,\mathsf{case}_i(yM_1 \dots M_m)N_0 \dots N_i)\lceil l := L\rceil &= \lambda x_1 \dots x_n.\,\mathsf{case}_i(yM_1 \dots M_m) \\ & \qquad N_0\lceil l := L\rceil \dots N_i\lceil l := L\rceil \end{aligned}$$

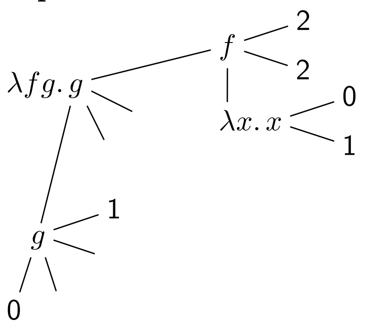
We also write multiple replacements, e.g. $P\lceil l := L_l \text{ for } l \geq 0\rceil$. These multiple replacements are done in parallel, the whole replacement moves down the term.

We will use a graphical representation of game terms in the next sections:
A subterm $\lambda x_1 \dots x_n.\,\mathsf{case}_i(yM_1 \dots M_m)N_0 \dots N_i$ is represented in the graph by a node of the form:



The upper parent of this node is connected to the λ; if the λ is missing, the upper or left parent is connected to the y. The $M_1, \dots, M_m$ are the *legs* of y; the $N_0, \dots, N_i$ are the *arms* of y. A leg or arm that points to a $\perp$ is mostly represented simply by a leg or arm pointing to empty space. This graphical representation makes the behaviour of game terms much better visible.

Example:



This is the representation of the term:

$$\lambda fg.\,\mathsf{case}_1[g(\mathsf{case}_1(g\mathsf{0}\perp)\mathsf{1}\perp)\perp][\mathsf{case}_1(f(\lambda x.\,\mathsf{case}_1\,x\mathsf{0}\mathsf{1}))\mathsf{2}\mathsf{2}]\perp$$

of type $((\iota \to \iota) \to \iota) \to (\iota \to \iota \to \iota) \to \iota$. It is a game term of *pregrade* 1. The output positions are the two positions of the number $\mathsf{2}$. If we replace the number $\mathsf{2}$ at the output positions by $\perp$, $\mathsf{0}$ or $\mathsf{1}$, then we get a game term of *grade* 1.

Game terms are the real "medium" in which to investigate Berry's problems: First, if one seeks terms M which have many semantically different syntactic parts $N \prec M$, according to Berry's second conjecture, then one is naturally led to game terms, because they have a very fine syntactic structure. Second, they simplify the proofs of the counter-examples. The conditional always appears together with a variable, cutting down the cases to be analysed and simplifying the induction hypotheses considerably.

In the next subsection we develop a map gt_i^σ from finite terms to equivalent game terms such that $M \prec N \colon \sigma$ entails $\mathrm{gt}_i^\sigma(M) \prec \mathrm{gt}_j^\sigma(N)$, where M, N are of grade i resp. j, $i \leq j$. This means that the refutation of Berry's conjectures may be restricted to game terms. In the following subsection we extend our result to infinite game terms. They are needed for a full formulation of Berry's conjectures for first-order types (where they are valid).

3.4.1 Finite Game Term Theorem

We are given finite terms $M \prec N$ and want to find equivalent game terms. First we must get rid of the Ys in the terms.

The map $\omega\colon \mathrm{PCF}^\sigma \to \mathrm{PCF}^\sigma$ (for all types σ) is taken from [BCL85, Ber79] and called the *immediate syntactic value*:

$$\omega(M) = \begin{cases} \lambda x_1 \ldots x_n . u\, \omega(M_1) \ldots \omega(M_m), & \text{if } M = \lambda x_1 \ldots x_n . u M_1 \ldots M_m \\ & \text{with } u \text{ a variable or constant,} \\ & \text{i.e. } M \text{ is in head normal form} \\ \bot \text{ else} & \end{cases}$$

Please note here that a constant is $\mathsf{suc}, \mathsf{pre}, \mathsf{if}, \mathsf{0}, \mathsf{1}, \mathsf{2}, \ldots$ A constant is not $\bot$ or Y.
$\to_{\beta Y}$ is the one-step reduction with the β-rule or the rule $\mathsf{Y}M \to M(\mathsf{Y}M)$ in any context. As is known from [BCL85, Ber79], if $M \to^*_{\beta Y} N$, then $\omega(M) \prec \omega(N)$.

Lemma 3.4.3 (Approximation Lemma)**.** *For every finite term M there is a term N' such that $M \to^*_{\beta Y} N$ for some N, $N' \prec \omega(N)$, $M \cong N'$ and N' is the $\prec$-least term with this property. This unique N' is called* $\mathrm{approx}(M)$*.*

Proof. For the fully abstract cpo-model (and therefore for all f-models) the approximation continuity theorem [BCL85, theorem 4.3.1] is valid:

$$\{\, [\![\omega(N)]\!] \mid M \to^*_{\beta Y} N \,\} \to [\![M]\!].$$

The set on the left is directed and M is finite, therefore there is N with $M \to^*_{\beta Y} N$ and $[\![M]\!] = [\![\omega(N)]\!]$.

Now assume the type of M, N is $\sigma_1 \to \ldots \to \sigma_n \to \iota$. Take any vector of closed terms $A_1 \colon \sigma_1, \ldots, A_n \colon \sigma_n$ with $\omega(N) A_1 \ldots A_n \to^* m$ (integer constant).

By syntactic stability [Ber79, theorem 2.8.8] [BCL85, theorem 3.6.7] there is a $\prec$-least term $N^* \prec \omega(N)$ with $N^* A_1 \ldots A_n \to^* m$. Take as N' the $\prec$-lub of all these N^*. □

Lemma 3.4.4. *For all finite terms $M \prec N$ it is* $\mathrm{approx}(M) \prec \mathrm{approx}(N)$*.*

Proof. Let M' be a term with $M \to^*_{\beta Y} M'$ and $M \cong \omega(M')$.
As the β-rule and the Y-rule do not involve $\bot$, all these reductions $M \to^*_{\beta Y} M'$ can also be done in N. (If $A \prec B$ and $A \to_{\beta Y} A'$, then there is B' with $B \to_{\beta Y} B'$ and $A' \prec B'$.)
So there is N' with $N \to^*_{\beta Y} N'$ and $M' \prec N'$, and of course $\omega(M') \prec \omega(N')$.

By confluence of $\to_{\beta Y}$ there is N'' with $N' \to^*_{\beta Y} N''$ and $N \cong \omega(N'')$.
It is $\omega(N') \prec \omega(N'')$, therefore $\omega(M') \prec \omega(N'')$.
$\mathrm{approx}(M)$ is the least term X with $X \prec \omega(N'')$ and $M \sqsubseteq_{op} X$.
$\mathrm{approx}(N)$ fulfills the two conditions for X, therefore $\mathrm{approx}(M) \prec \mathrm{approx}(N)$. □

Now we have finite terms $\mathrm{approx}(M) \prec \mathrm{approx}(N)$ without Y. The next step is to apply a Ψ^σ_i-like operator to the terms and reduce according to some reduction rules to game terms. The proof can be done in different ways:

In my first version I proved the termination of the reductions, formulated an invariant of the (eta-expanded) term structure, proved the invariance under the reductions and that they lead to game terms. This resulted in an induction on the reduction sequence, the induction step done by induction on the term, causing much rewriting bureaucracy. (This ugly proof is available as supplementary material from my home page.)

Here we will see a more elegant half-sized proof based on an induction on the term from the beginning, with the aid of a reducibility predicate (see e.g. [Plo77, theorem 3.1]). (Jim Laird also uses a reducibility predicate to produce eta-expanded normal forms of a simply typed λ-calculus with lifting (without inconsistent values) [Lai05, proposition 4.2].)

To produce the game terms we define for every $i \geq 0$ a big-step reduction relation $M \downarrow_i N$ on case_i-terms. The mere existence of the game terms could be proved without $\downarrow_i$, but we want to give an explicit deterministic algorithm. (Determinism is easily built into big-step reduction.) The values for $\downarrow_i$, i.e. the terms that we consider as the results of reductions, are the game terms of pregrade i.

Here are the rules for $\downarrow_i$. In the hypothesis of a rule the abbreviation $M \downarrow_i N\ gi$ means "$M \downarrow_i N$ and N is a game term of grade i", $M \downarrow_i N\ pi$ means "$M \downarrow_i N$ and N is a game term of pregrade i".

(0) $n \downarrow_i n$ for all integer constants n

(1) $\dfrac{M[y := M_1]M_2 \dots M_m \downarrow_i P}{(\lambda y.M)M_1M_2 \dots M_m \downarrow_i P}$, for $m \geq 1$ (2) $\bot M_1 \dots M_m \downarrow_i \bot$, for $m \geq 0$

(3) $\dfrac{A \downarrow_i A'\ pi}{\mathsf{suc}\, A \downarrow_i A'\lceil m := m+1 \text{ for } m \geq 0\rceil}$ (4) $\dfrac{A \downarrow_i A'\ pi}{\mathsf{pre}\, A \downarrow_i A'\lceil 0 := \bot,\ m := m-1 \text{ for } m \geq 1\rceil}$

(5) $\dfrac{A_k \downarrow_i A'_k\ pi, \text{ for } k = 1,2,3}{\mathsf{if}\ A_1\ \mathsf{then}\ A_2\ \mathsf{else}\ A_3 \downarrow_i A'_1\lceil 0 := A'_2,\ m := A'_3 \text{ for } m \geq 1\rceil}$

(6) $\dfrac{A \downarrow_i A'\ pi, A' \neq \bot}{\lambda x_1 \dots x_n.\, \mathsf{case}_i\, A0 \dots i \downarrow_i \lambda x_1 \dots x_n.A'\lceil k := \bot \text{ for } k > i\rceil}$, for $n \geq 0$

(7) $\dfrac{A \downarrow_i \bot}{\lambda x_1 \dots x_n.\, \mathsf{case}_i\, A0 \dots i \downarrow_i \bot}$, for $n \geq 0$

(8) $\dfrac{A_k \downarrow_i A'_k\ gi, \text{ for } 1 \leq k \leq m}{\mathsf{case}_i(xA_1 \dots A_m)0 \dots i \downarrow_i \mathsf{case}_i(xA'_1 \dots A'_m)0 \dots i}$, for $m \geq 0$

Remarks: Not for all case_i-terms $M\colon \sigma$ there is a value V with $M \downarrow_i V$, but there will be a value V with $\Psi_i^\sigma M \downarrow_i V$ for Ψ_i^σ suitably defined. The reduction relations are complete enough for the purposes of the following proofs. So to understand the reductions at this stage, just check the soundness of each rule separately, according to the following lemma, and do not bother about completeness. When you go through the subsequent proofs, you will see that exactly these rules are needed, no more, no less.

Lemma 3.4.5 (soundness of the reduction relations $\downarrow_i$). *For all case_i-terms M, M': If $M \downarrow_i M'$, then $[\![M]\!] = [\![M']\!]$ and M' is a value (i.e. a game term of pregrade i).*

Proof. Translate each reduction rule into a rule with semantic equivalence instead of the reduction relation: Translate statements $A \downarrow_i A'$ into ($[\![A]\!] = [\![A']\!]$ and A' is a value), and keep the statements gi and pi. Then check each translated rule for validity. □

Now we come to the reducibility predicate. We pack all that we want to prove into its definition: the compatibility of the transformation with the order $\prec$ and even the uniqueness of the reduction $\downarrow_i$.

Definition 3.4.6 (reducibility predicate)**.** Let $i \leq j$, A a case_i-term and B a case_j-term of type $\sigma = \sigma_1 \to \ldots \to \sigma_n \to \iota$, $n \geq 0$.
$A \prec B \colon \sigma$ are (i,j)-*transformable*, written $A \prec B \colon \sigma(i,j)$,
iff for all $A_l \prec B_l \colon \sigma_l(i,j)$, $1 \leq l \leq n$, there are game terms $A', B' \colon \iota$ of pregrade i resp. j with $AA_1 \ldots A_n \downarrow_i A'$ and $BB_1 \ldots B_n \downarrow_j B'$,
A' and B' are unique for these reductions, and furthermore $A' \prec B'$.

Note that this definition does not take care of the free variables of A, B. Note also that it does not demand the *grade* i, j of A', B', but the *pregrade.* So it will be applicable to general terms that do not restrict the integer constants, in lemma 3.4.9.

Lemma 3.4.7. *If $A \prec B \colon \sigma(i,j)$, then $\bot \prec B \colon \sigma(k,j)$ for all $k \leq j$.*

Proof. Easy consequence of the definition of the reducibility predicate and of rule (2) for $\bot$-application. □

For the next lemma we need a notion of simultaneous substitution for PCF-terms that properly renames bound variables. We take Allen Stoughton's definitions [Sto88].

A *substitution* is a function s, t from variables to terms (of the type of the variable). The substitution $s[x := N]$ is defined by $(s[x := N])x = N$ and $(s[x := N])y = sy$ for $y \neq x$. id is the identity substitution.

If x is a variable, M a term, s a substitution, then we define

$$\mathrm{new}\, xMs = \{\, y \mid y \text{ variable and for all } z \in FV(M) - \{x\}.\ y \notin FV(sz) \,\},$$

where $FV(X)$ is the set of free variables of term X.

The *simultaneous substitution* Ms of sx for the free occurrences of x in M, for all x, is defined by structural recursion on M:

$$\begin{aligned}
xs &= sx, \text{ for every variable } x \\
cs &= c, \text{ for every constant } c \\
(MN)s &= (Ms)(Ns) \\
(\lambda x.M)s &= \lambda y.(M(s[x := y])), \text{ with } y = \mathrm{choice}(\mathrm{new}\, xMs),
\end{aligned}$$

where choice is a fixed function that chooses some variable y from the argument set of variables.

We suppose that the normal substitution (in the β-rule) behaves like this:

$$P[y := N] = P(\mathrm{id}[y := N]).$$

Lemma 3.4.8. *For terms M, N, substitution s and variables x, y with $y = \mathrm{choice}(\mathrm{new}\, xMs)$ we have:*

$$(M(s[x := y]))[y := N] = M(s[x := N])$$

Proof. Follows from theorem 3.2 of [Sto88]. □

Lemma 3.4.9. *Let $A \prec B\colon \sigma$ be PCF-terms without* Y.
Let $\{x_1^{\tau_1}, \dots, x_m^{\tau_m}\}$ be a superset of the free variables of B.
For $1 \leq k \leq m$ let $A'_k \prec B'_k\colon \tau_k(i,j)$ be case_i*- resp.* case_j*-terms that are (i,j)-transformable.*
Define the substitutions $s = \mathrm{id}[x_1 := A'_1] \dots [x_m := A'_m]$ and $t = \mathrm{id}[x_1 := B'_1] \dots [x_m := B'_m]$.
Then $As \prec Bt\colon \sigma(i,j)$.

Proof. By induction on the term B. (Note: PCF-terms are without case.)
Case $B = B^* B_0$, $B^*\colon \sigma_0 \to \sigma_1 \to \dots \to \sigma_n \to \iota$, for $n \geq 0$:
First let $A = A^* A_0$.
By the induction hypothesis we get $A^* s \prec B^* t\colon \sigma_0 \to \dots \sigma_n \to \iota(i,j)$ and $A_0 s \prec B_0 t\colon \sigma_0(i,j)$.
Let $A_l \prec B_l\colon \sigma_l(i,j)$ for $1 \leq l \leq n$.
By the reducibility predicate there are game terms $A' \prec B'\colon \iota$ of pregrade i resp. j with

$$\begin{aligned}(A^* s)(A_0 s) A_1 \dots A_n &\downarrow_i A'\\ (B^* t)(B_0 t) B_1 \dots B_n &\downarrow_j B'\end{aligned}$$

So $As \prec Bt\colon \sigma(i,j)$.

Now let $A = \bot$. By the same argument we have $Bt \prec Bt\colon \sigma(j,j)$, therefore by lemma 3.4.7: $\bot \prec Bt\colon \sigma(i,j)$.

Case $B = \lambda x.B^*\colon \sigma_1 \to \dots \to \sigma_n \to \iota$, $n \geq 1$:
First let $A = \lambda x.A^*$.
Let $A_l \prec B_l\colon \sigma_l(i,j)$ for $1 \leq l \leq n$.
By the induction hypothesis for B^* we get

$$A^*(s[x := A_1]) \prec B^*(t[x := B_1])\colon \sigma_2 \to \dots \to \sigma_n \to \iota(i,j).$$

Therefore there are game terms $A', B'\colon \iota$ of pregrade i resp. j with

$$\begin{aligned}(A^*(s[x := A_1])) A_2 \dots A_n &\downarrow_i A'\\ (B^*(t[x := B_1])) B_2 \dots B_n &\downarrow_j B',\end{aligned}$$

with A', B' unique and $A' \prec B'$.
By lemma 3.4.8 and the definition of substitution we get:

$$\begin{aligned}A^*(s[x := A_1]) &= (A^*(s[x := y]))[y := A_1], \text{ for } y = \mathrm{choice}(\mathrm{new}\, x A^* s)\\ B^*(t[x := B_1]) &= (B^*(t[x := z]))[z := B_1], \text{ for } z = \mathrm{choice}(\mathrm{new}\, x B^* t)\\ (\lambda x.A^*)s &= \lambda y.A^*(s[x := y])\\ (\lambda x.B^*)t &= \lambda z.B^*(t[x := z])\end{aligned}$$

Then it reduces

$$\begin{aligned}(A^*(s[x := y]))[y := A_1] A_2 \dots A_n \downarrow_i A', &\text{ and therefore by rule (1):}\\ (\lambda y.A^*(s[x := y])) A_1 A_2 \dots A_n \downarrow_i A', &\text{ therefore}\\ (\lambda x.A^*) s A_1 A_2 \dots A_n \downarrow_i A'. &\end{aligned}$$

Analogously:

$$(\lambda x.B^*) t B_1 B_2 \dots B_n \downarrow_j B'.$$

These reductions are unique, and $A' \prec B'$. So $As \prec Bt\colon \sigma(i,j)$.

Now let $A = \bot$. By the same argument we have $Bt \prec Bt\colon \sigma(j,j)$, therefore by lemma 3.4.7: $\bot \prec Bt\colon \sigma(i,j)$.

Cases $B = x$ (variable), $B = n$ (integer constant), $B = \bot$ are clear.
For $B = n$ rule (0) is used, for $B = \bot$ rule (2).
For the subcases $A = \bot$ lemma 3.4.7 is used.

Case $B = \mathsf{if}$:
First let $A = \mathsf{if}$.
Let $A_l \prec B_l\colon \iota(i,j)$ for $1 \le l \le 3$.
Then there are $A_l \downarrow_i A'_l$ and $B_l \downarrow_j B'_l$ (A'_l, B'_l unique) with $A'_l \prec B'_l$, for $1 \le l \le 3$.
It reduces by rule (5):

$$\mathsf{if}\, A_1 \,\mathsf{then}\, A_2 \,\mathsf{else}\, A_3 \downarrow_i A'_1\lceil 0 := A'_2,\ m := A'_3 \text{ for } m \ge 1\rceil$$
$$\mathsf{if}\, B_1 \,\mathsf{then}\, B_2 \,\mathsf{else}\, B_3 \downarrow_j B'_1\lceil 0 := B'_2,\ m := B'_3 \text{ for } m \ge 1\rceil$$

Both reductions are unique and the results are in relation $\prec$.
Now let $A = \bot$. By lemma 3.4.7 it is $\bot \prec \mathsf{if}\colon \sigma(i,j)$.

Cases $B = \mathsf{suc}$, $B = \mathsf{pre}$: analogous to $B = \mathsf{if}$.
For $B = \mathsf{suc}$ rule (3) is used, for $B = \mathsf{pre}$ rule (4). □

Next we prove a lemma that introduces the terms Ψ_i^σ into the transformation. For the rest of this section we redefine the finite projection terms Ψ_i^σ as equivalent case_i-terms:

$$\Psi_i^{\sigma_1 \to \ldots \to \sigma_n \to \iota} = \lambda f.\lambda x_1 \ldots x_n.\, \mathsf{case}_i[f(\Psi_i^{\sigma_1}x_1)\ldots(\Psi_i^{\sigma_n}x_n)]\mathsf{0}\ldots i, \text{ for } n \ge 0.$$

Lemma 3.4.10. *For all types $\sigma = \sigma_1 \to \ldots \to \sigma_n \to \iota$ the following three propositions are valid:*

(1) For all $A \prec B\colon \sigma(i,j)$ it is

$$A(\Psi_i^{\sigma_1}x_1)\ldots(\Psi_i^{\sigma_n}x_n) \prec B(\Psi_j^{\sigma_1}x_1)\ldots(\Psi_j^{\sigma_n}x_n)\colon \iota(i,j).$$

(2) For all $A \prec B\colon \sigma(i,j)$ there are $A', B'\colon \sigma$ with $\Psi_i^\sigma A \downarrow_i A'$ and $\Psi_j^\sigma B \downarrow_j B'$ such that both are unique for this reduction, and furthermore $A' \prec B'$ and they are game terms of grade *i resp. j.*

(3) For all variables x^σ and $i \le j$: $\Psi_i^\sigma x^\sigma \prec \Psi_j^\sigma x^\sigma\colon \sigma(i,j)$.

Proof. By simultaneous induction on the type σ.
(1) By the induction hypothesis for (3) we get $\Psi_i^{\sigma_k}x_k \prec \Psi_j^{\sigma_k}x_k\colon \sigma_k(i,j)$, for $1 \le k \le n$, and the proposition follows.
(2) The proposition (1) means that there are game terms $A'', B''\colon \iota$ with pregrade i resp. j such that $A(\Psi_i^{\sigma_1}x_1)\ldots(\Psi_i^{\sigma_n}x_n) \downarrow_i A''$ and $B(\Psi_j^{\sigma_1}x_1)\ldots(\Psi_j^{\sigma_n}x_n) \downarrow_j B''$, with A'', B'' unique

for this reduction and $A'' \prec B''$.
If $A'' = \bot$ then it reduces by rule (7):

$$\lambda x_1 \dots x_n.\, \mathsf{case}_i[A(\Psi_i^{\sigma_1} x_1) \dots (\Psi_i^{\sigma_n} x_n)]0 \dots i \downarrow_i \bot$$

and therefore by rule (1): $\Psi_i^\sigma A \downarrow_i \bot$.
If also $B'' = \bot$, then likewise $\Psi_j^\sigma B \downarrow_j \bot$ and the proposition follows.
(We still have $A'' = \bot$.) If $B'' \neq \bot$ then it reduces by rule (6):

$$\lambda x_1 \dots x_n.\, \mathsf{case}_j[B(\Psi_j^{\sigma_1} x_1) \dots (\Psi_j^{\sigma_n} x_n)]0 \dots j \downarrow_j \lambda x_1 \dots x_n.B''\lceil k := \bot \text{ for } k > j\rceil = B'$$

and therefore by rule (1): $\Psi_j^\sigma B \downarrow_j B'$, B' is a game term of grade j, and the proposition follows.
If $A'' \neq \bot$ and $B'' \neq \bot$, then we get like the last reduction by rules (6) and (1):

$$\Psi_i^\sigma A \downarrow_i \lambda x_1 \dots x_n.A''\lceil k := \bot \text{ for } k > i\rceil = A'$$
$$\Psi_j^\sigma B \downarrow_j \lambda x_1 \dots x_n.B''\lceil k := \bot \text{ for } k > j\rceil = B'$$

Both reductions are unique, it is $A' \prec B'$ and they are game terms of grade i resp. j.
(3) We have to prove that for all $A_l \prec B_l \colon \sigma_l(i,j)$, $1 \leq l \leq n$, there are game terms $A' \prec B'$ of pregrade i resp. j with $(\Psi_i^\sigma x)A_1 \dots A_n \downarrow_i A'$ and $(\Psi_j^\sigma x)B_1 \dots B_n \downarrow_j B'$ (with uniqueness of the reductions).

By the induction hypothesis of (2) for all l there are game terms $A'_l \prec B'_l \colon \sigma_l$ of grade i resp. j with $\Psi_i^{\sigma_l} A_l \downarrow_i A'_l$ and $\Psi_j^{\sigma_l} B_l \downarrow_j B'_l$ (with uniqueness of the reductions).
It reduces by rule (8)

$$\mathsf{case}_i[x(\Psi_i^{\sigma_1} A_1) \dots (\Psi_i^{\sigma_n} A_n)]0 \dots i \downarrow_i \mathsf{case}_i[xA'_1 \dots A'_n]0 \dots i = A'$$

and therefore by rule (1):

$$(\Psi_i^\sigma x)A_1 \dots A_n \downarrow_i A'$$

Likewise it reduces by rules (8) and (1):

$$(\Psi_j^\sigma x)B_1 \dots B_n \downarrow_j \mathsf{case}_j[xB'_1 \dots B'_n]0 \dots j = B'$$

A', B' are even game terms of *grade* i resp. j. The reductions are unique. It is $A' \prec B'$. □

Definition 3.4.11. Let A be a case_i-term without Y with $A \prec A \colon \sigma(i,i)$.
The unique game term A' of grade i with $\Psi_i^\sigma A \downarrow_i A'$ is called $\mathrm{proj}_i^\sigma(A)$.
For every finite term $M \colon \sigma$ we get $\mathrm{approx}(M)$ without Y with $\mathrm{approx}(M) \prec \mathrm{approx}(M) \colon \sigma(i,i)$ by lemma 3.4.9. (Note that finite terms are closed.)
We define the map $\mathrm{gt}_i^\sigma(M) = \mathrm{proj}_i^\sigma(\mathrm{approx}(M))$, for $M \colon \sigma$ finite term of grade i.

Theorem 3.4.12 (Game Term Theorem). *If $i \leq j$ and $M \prec N \colon \sigma$ are finite PCF-terms of grade i resp. j, then $\mathrm{gt}_i^\sigma(M) \prec \mathrm{gt}_j^\sigma(N)$ are game terms of grade i resp. j with $M \cong \mathrm{gt}_i^\sigma(M)$ and $N \cong \mathrm{gt}_j^\sigma(N)$.*

Proof. By lemma 3.4.3 and 3.4.4 we get $\mathrm{approx}(M) \prec \mathrm{approx}(N)$ without Y. By lemma 3.4.9 it is $\mathrm{approx}(M) \prec \mathrm{approx}(N) \colon \sigma(i,j)$. By lemma 3.4.10(2) $\mathrm{proj}_i^\sigma(\mathrm{approx}(M)) \prec \mathrm{proj}_j^\sigma(\mathrm{approx}(N))$ are game terms of grade i resp. j. Furthermore $M \cong \Psi_i^\sigma(\mathrm{approx}(M)) \cong \mathrm{proj}_i^\sigma(\mathrm{approx}(M))$ and likewise for N. □

3.4.2 Infinite game terms

Definition 3.4.13. An *infinite game term* of type σ is an ideal of game terms of type σ (of any grade), under the ordering $\prec$. (Infinite game terms can be construed as Böhm trees with infinite case-expressions, which we write as $\mathsf{case}_\infty\, M N_0 N_1 \ldots$.) The order $\prec$ on infinite game terms is the subset order of the ideals. The semantics (in some f-model) of an infinite game term is the lub of the semantics of the members of its ideal, if the lub exists in the f-model.

Definition 3.4.14. Let $M\colon \sigma$ be a closed PCF-term.
$\Psi_0^\sigma M \prec \Psi_1^\sigma M \prec \Psi_2^\sigma M \prec \ldots$ is an ascending chain of finite terms with ascending grade. Define $\mathrm{gt}^\sigma(M)$ as the lub (in the order of infinite game terms) of the ascending chain of game terms $\mathrm{gt}_0^\sigma(\Psi_0^\sigma M) \prec \mathrm{gt}_1^\sigma(\Psi_1^\sigma M) \prec \mathrm{gt}_2^\sigma(\Psi_2^\sigma M) \prec \ldots$.

Theorem 3.4.15 (Infinite Game Term Theorem)**.** *If $M \prec N\colon \sigma$ are closed PCF-terms, then $\mathrm{gt}^\sigma(M) \prec \mathrm{gt}^\sigma(N)$ are infinite game terms with $[\![M]\!] = [\![\mathrm{gt}^\sigma(M)]\!]$ and $[\![N]\!] = [\![\mathrm{gt}^\sigma(N)]\!]$ in any f-model.*

Proof. By proposition 3.3.8 it is $[\![\Psi_i^\sigma M]\!] \to [\![M]\!]$, therefore $[\![M]\!] = [\![\mathrm{gt}^\sigma(M)]\!]$, and likewise $[\![N]\!] = [\![\mathrm{gt}^\sigma(N)]\!]$. As $\mathrm{gt}_i^\sigma(\Psi_i^\sigma M) \prec \mathrm{gt}_i^\sigma(\Psi_i^\sigma N)$ for all i, we get $\mathrm{gt}^\sigma(M) \prec \mathrm{gt}^\sigma(N)$. □

3.5 The syntactic order is not the image of the stable order

Berry's second conjecture in its finite form says that the stable order of the order-extensional fully abstract cpo-model of PCF (our greatest f-model) has the syntactic order as its image: If $a \leq b$ for finite a, b in the model, then there are normal form terms A, B with $[\![A]\!] = a$, $[\![B]\!] = b$ and $A \prec B$.
(The choice of the greatest f-model is not important, as all f-models coincide on their finite parts.)

In this section we will first show that Berry's second conjecture is valid in first-order types. Then we give our simplest counter-example in finitary PCF of second-order type, a chain of length 2. We also give examples of chains of any finite length.

For first-order types Berry's conjecture can be strengthened to the infinite case:

Theorem 3.5.1 (Berry, Theorem 4.1.7 and 4.8.14 in [Ber79])**.** *Let σ be a first-order type, and $b \in D^\sigma$ in the greatest f-model. Then there is an infinite game term B with $b = [\![B]\!]$. Furthermore, for all such infinite game terms B and every subset $t \subseteq \mathcal{T}(b)$ there is an infinite game term $A \prec B$ with $\mathcal{T}[\![A]\!] = t$. (As infinite game term, A has a denotation in the greatest f-model.)*

Proof. Let $\sigma = \iota \to \iota \to \ldots \to \iota$ with $n \geq 1$ arguments. In [Ber79, 4.1.7] Berry shows that $b \in D^\sigma$, as the lub of a growing sequence of finite sequential functions, is itself sequential. Therefore: If b is not some constant function, then b is strict in some j-th argument. So B can be recursively constructed as infinite game term (with case_∞ the infinite case) in the form:

$$B = \lambda x_1 \ldots x_n.\, \mathsf{case}_\infty\, x_j B_1 B_2 \ldots,$$

where B_i is a term with free variables $x_1, \dots x_{j-1}x_{j+1}\dots x_n$ for the residual function b_i given by

$$b_i x_1 \dots x_{j-1}x_{j+1}\dots x_n = b x_1 \dots x_{j-1} i x_{j+1} \dots x_n.$$

In [Ber79, 4.8.14] Berry shows that A can be constructed in the same manner B was constructed, i.e. following the same choice of the variables for which the function is strict. We can describe the construction of A differently by using traces: The tokens of the trace $\mathcal{T}[\![B]\!]$ correspond exactly to the branches of B that output a result, i.e. do not lead to $\perp$. We simply choose $A \prec B$ by setting those branches of B that do not correspond to a token in t to the empty output $\perp$. $\square$

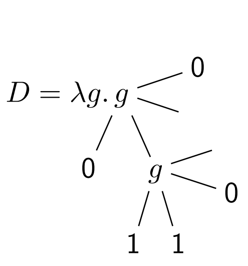
We conjecture that Berry's second conjecture is also true for second-order types with parameters of arity at most one:

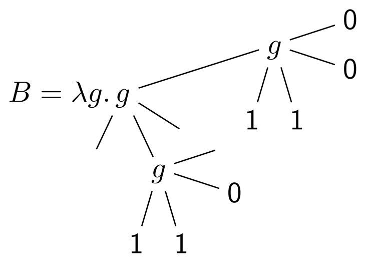
Conjecture 3.5.2. Let $\sigma = \sigma_1 \to \dots \to \sigma_n \to \iota$ with $\sigma_i = \iota$ or $\sigma_i = \iota \to \iota$ for all i. Let $b \in \mathcal{F}_i^\sigma$ be a finite element of grade i.
Then there is a game term B of grade i for b, $b = [\![B]\!]$, such that for every subset $t \subseteq \mathcal{T}(b)$ that is secured in the sense of definition 2 of [CPW00] there is $A \prec B$ with $\mathcal{T}[\![A]\!] = t$. (The trace of every semantic element is secured, so Berry's second conjecture would be fulfilled for these types.)

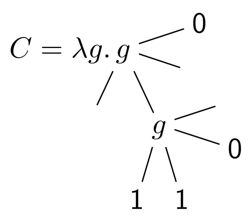
The proof of this conjecture is in preparation. It needs a new theory of (PCF-)terms that would exceed the frame of this paper.

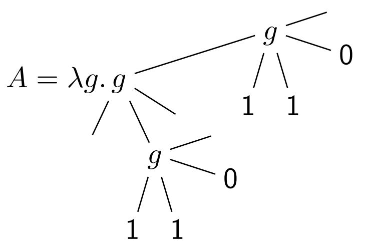
3.5.1 Refutation of Berry's second conjecture: A chain of least length 2

Our simplest counter-example to Berry's second conjecture is in finitary PCF of second-order type $(\iota \to \iota \to \iota) \to \iota$. We consider the following game terms A, B, C, D:









$D = \lambda g.\ \mathsf{case}_1(g\,0\,(\mathsf{case}_1(g\,1\,1)\perp 0))\,0\perp$

$C = \lambda g.\ \mathsf{case}_1(g\perp(\mathsf{case}_1(g\,1\,1)\perp 0))\,0\perp$

$B = \lambda g.\ \mathsf{case}_1(g\perp(\mathsf{case}_1(g\,1\,1)\perp 0))(\mathsf{case}_1(g\,1\,1)\,0\,0)\perp$

$A = \lambda g.\ \mathsf{case}_1(g\perp(\mathsf{case}_1(g\,1\,1)\perp 0))(\mathsf{case}_1(g\,1\,1)\perp 0)\perp$

For illustration (not for the proof) we give the trace semantics of these terms:

$$\left.\begin{array}{l}\left.\begin{array}{l} A\left\{\begin{array}{l} \{1\,1\mapsto 1,\ \ \bot\,0\mapsto 0\}\mapsto 0\\ \{\bot\,1\mapsto 1,\ \ \bot\,0\mapsto 0\}\mapsto 0\end{array}\right. \\ \quad\ \{\qquad\quad\ \ \bot\,\bot\mapsto 0\}\mapsto 0\end{array}\right\} B\cong C \\ \quad\ \{\qquad\quad\ \ 0\,\bot\mapsto 0\}\mapsto 0 \\ \quad\ \{1\,1\mapsto 1,\ \ 0\,0\mapsto 0\}\mapsto 0 \\ \quad\ \{1\,\bot\mapsto 1,\ \ 0\,0\mapsto 0\}\mapsto 0 \\ \quad\ \{\bot\,1\mapsto 1,\ \ 0\,0\mapsto 0\}\mapsto 0 \end{array}\right\} D$$

We have $A \prec B \cong C \prec D$, therefore $[\![A]\!] \leq [\![D]\!]$. We will prove that this chain of two steps of $\prec$ cannot be replaced by one single step.

Proof of the equivalence $B \cong C$: For any argument g, if Cg converges (i.e. reduces to an integer constant), then the subterm $g11$ of C converges also. (There are only two possibilities for g: either $\mathcal{T}(g) = \{\bot\bot\mapsto 0\}$, or g demands its second argument.) Therefore it is possible to safely replace the result 0 in C by the term $\mathsf{case}_1(g11)00$, i.e. to "lift" $g11$ to the top level.

It is important to notice that this transformation cannot be performed with D: Here there are more possibilities for g to make Dg converge. It might be that $\mathcal{T}(g) = \{0\bot\mapsto 0\}$, then the subterm $g11$ does not converge.

The intuition of the example: We start with term D, working downwards step by step to A eliminating tokens of the trace. First the token $\{0\bot\mapsto 0\}\mapsto 0$ is eliminated getting C (and the other tokens with g demanding its first argument 0). Then it becomes possible to lift $g11$, we get $B \cong C$. Next we eliminate the token $\{\bot\bot\mapsto 0\}\mapsto 0$ in B to get A. This is done by "forcing" the evaluation of the second argument of g, by demanding that g delivers different results for different arguments.

Proposition 3.5.3. *Let A, D be the game terms of grade 1 above. There are no game terms A', D' of grade 1 with $A' \prec D'$ and $A' \cong A$, $D' \cong D$. Then by the game term theorem 3.4.12 there are no PCF-terms A', D' with this property. Since we have seen that $[\![A]\!] \leq [\![D]\!]$, the proposition refutes Berry's second conjecture.*

Proof. As game terms of grade 1, A' and D' should be of the form $\lambda g.S$, where $S\colon \iota$ is a game term possibly with the only free variable g. We abbreviate $S[g := M]$ as $S[M]$.

Let $R, P, Q\colon \iota \to \iota \to \iota$ be the following terms:

$$\begin{array}{ll} R = \lambda xy.\,\mathsf{case}_1\, y\, 0(\mathsf{case}_1\, x \bot 1), & \mathcal{T}[\![R]\!] = \{11\mapsto 1,\ \ \bot 0\mapsto 0\} \\ P = \lambda xy.0, & \mathcal{T}[\![P]\!] = \{\bot\bot\mapsto 0\} \\ Q = \lambda xy.\,\mathsf{case}_1\, x\, 0 \bot, & \mathcal{T}[\![Q]\!] = \{0\bot\mapsto 0\} \end{array}$$

We will prove: For any terms S, S' of the form above,

$$\text{if } S' \prec S \text{ and } S[Q] \to^* 0 \text{ and } S'[R] \to^* 0,\ \text{then } S'[P] \to^* 0.$$

The proposition follows from this claim, as $DQ \to^* 0$ and $AR \to^* 0$, but not $AP \to^* 0$.

The proof of the claim is by induction on the term S:
The cases $S = \bot, \mathsf{0}, \mathsf{1}$ are clear.
Let $S = \mathsf{case}_1(gS_1S_2)S_3S_4$ and $S' \prec S$ with $S' = \mathsf{case}_1(gS'_1S'_2)S'_3S'_4$. (The remaining case $S' = \bot$ is clear.)
Suppose $S[Q] \to^* \mathsf{0}$ and $S'[R] \to^* \mathsf{0}$. Then $S_1[Q] \to^* \mathsf{0}$.
R and Q are compatible in the Scott model of all continuous functions, the "parallel or" is an upper bound. Expressed differently, R and Q are compatible in the sense that they produce compatible integer results for the same argument. Therefore the semantics of $S_1[R]$ and $S_1[Q]$ must be compatible, so it is not possible that $S_1[R] \to^* \mathsf{1}$.
As $S'_1 \prec S_1$, it is also not possible that $S'_1[R] \to^* \mathsf{1}$.
Therefore $(gS'_1S'_2)[R] \to^* \mathsf{0}$ (it must converge to get $S'[R] \to^* \mathsf{0}$).
Hence $S'[R] \to^* S'_3[R] \to^* \mathsf{0}$.
On the other side we have $S[Q] \to^* S_3[Q] \to^* \mathsf{0}$.
Together we have $S_3[Q] \to^* \mathsf{0}$ and $S'_3[R] \to^* \mathsf{0}$, and by the induction hypothesis for S_3 follows: $S'_3[P] \to^* \mathsf{0}$.
Therefore $S'[P] \to^* S'_3[P] \to^* \mathsf{0}$. □

Remark 3.5.4. As we base our proof on game terms, we gave a special induction hypothesis for the combination of case_1 and g. The proof for general normal form terms is more complicated as it must work with if and g separately and use a more general induction hypothesis, i.e. one proves by induction on S:

$$\text{If } S' \prec S, \text{ then } [\![S[Q]]\!] = [\![S'[R]]\!] = [\![S'[P]]\!] \text{ or } [\![S[Q]]\!] = \bot \text{ or } [\![S'[R]]\!] = \bot$$

This has on the surface the *form* of the Sieber sequentiality logical relation $S^3_{\{1,2\}\{1,2,3\}}$, see [Sie92]. (It is $(d_1, d_2, d_3) \in S^3_{\{1,2\}\{1,2,3\}}$ iff $d_1 = d_2 = d_3$ or $d_1 = \bot$ or $d_2 = \bot$.) This form on the surface is responsible for the fact that the induction hypothesis goes up through the case $S = \mathsf{if}\, S_1\, \mathsf{then}\, S_2\, \mathsf{else}\, S_3$. But for the proof of the case $S = gS_1S_2$ the specific semantics of R, P, Q and the fact $S' \prec S$ are needed.

So a sequentiality relation alone is not sufficient to prove this counter-example: a logical relation is a *semantic* means to prove the undefinability of a function. But here we must prove the undefinability of $S' \prec S$ for two functions $[\![A]\!] \leq [\![D]\!]$, where both functions separately are definable. At first sight this necessitates a *syntactic* proof. But we could ask the question: Are there *semantic* means to prove this? Are there necessary semantic conditions for the syntactic order that are stronger than the condition of stable order? See also the remark in the last section "Outlook".

3.5.2 Chains of any length

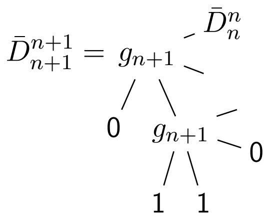
We have seen an example of a chain of two $\prec$-steps. Generally:

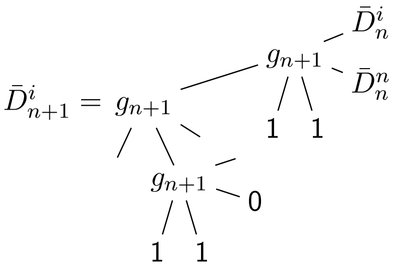
Definition 3.5.5. Let $a \leq b$ be finite elements in an f-model.
A *chain of length $n \geq 1$ between a and b* is a pair of sequences of terms $(C_i), (D_i)$ with $1 \leq i \leq n$ and $a = [\![C_1]\!]$, $b = [\![D_n]\!]$ and $C_i \prec D_i$, $D_i \cong C_{i+1}$.
If $a = b$, then we say there is a chain of length 0 between a and b.
A chain is *of least length n* if there is no shorter chain.

By the game term theorem, if there is a chain of PCF-terms, then there is an equivalent chain of game terms.

Now we construct examples of chains of least length $n+1$ for any finite $n \geq 0$, by a sequential composition of n copies of our first example, each copy for a different argument g_i. For every $n \geq 0$ let σ_n be the type $(\iota \to \iota \to \iota) \to \ldots \to (\iota \to \iota \to \iota) \to \iota$ with n parameters. For every n we define two sequences of game terms $C_n^i, D_n^i \colon \sigma_n$ with $0 \leq i \leq n$.
First we define by induction on n the versions $\bar{C}_n^i, \bar{D}_n^i$ without λ-binder:

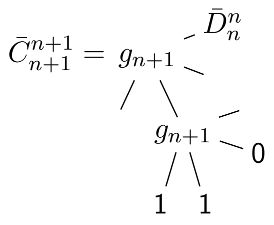
$\bar{D}_0^0 = 0$

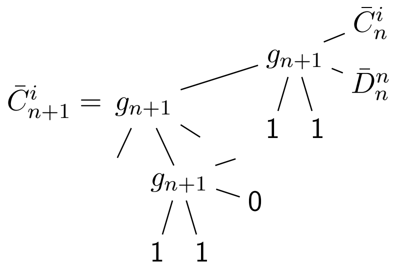




for $i \leq n$

$\bar{C}_0^0 = \bot$





for $i \leq n$

We define $C_n^i = \lambda g_n \ldots g_1.\bar{C}_n^i$ and $D_n^i = \lambda g_n \ldots g_1.\bar{D}_n^i$.
For all $n \geq 0$, $0 \leq i \leq n$: $C_n^i \prec D_n^i$. The proof is an easy induction on n.
For all $n \geq 1$, $i < n$: $D_n^i \cong C_n^{i+1}$. Proof by induction on n:
For $n = 1$, $i = 0$ we have that D_1^0 is the term B, and C_1^1 the term C of our former example, both only with g replaced by g_1.
For $n := n+1$:
For $i = n$ we have $D_{n+1}^n \cong C_{n+1}^{n+1}$ by the same argument as in our former example for $B \cong C$.
For $i < n$ we get $D_{n+1}^i \cong C_{n+1}^{i+1}$ by the induction hypothesis.

All together for any $n \geq 0$ we get a chain of length $n+1$ between $[\![C_n^0]\!]$ and $[\![D_n^n]\!]$:

$$C_n^0 \prec D_n^0 \cong C_n^1 \prec D_n^1 \ldots D_n^{n-1} \cong C_n^n \prec D_n^n.$$

We want to prove that this chain has the least length.

First the intuition of the example: We use the terms R, P, Q of the proof of proposition 3.5.3 and name their traces:

$$r = \mathcal{T}[\![R]\!] = \{11 \mapsto 1, \bot 0 \mapsto 0\}, \quad p = \mathcal{T}[\![P]\!] = \{\bot\bot \mapsto 0\}, \quad q = \mathcal{T}[\![Q]\!] = \{0\bot \mapsto 0\}$$

The trace of D_n^n contains all tokens $p \ldots pq \ldots q \mapsto 0$, with j arguments p, $0 \leq j \leq n$. These tokens are in the upper branch of D_n^n. We work down from D_n^n eliminating all these tokens in $n+1$ steps.

In the j-th step $(0 \leq j \leq n)$ the token $p \ldots pq \ldots q \mapsto 0$, with j arguments p, is eliminated in D_n^{n-j}. (In D_n^{n-j} all the tokens of this form with less arguments p have already been eliminated.) If $j < n$ we proceed as follows: Following the upper branches in D_n^{n-j} we come to an occurrence of the variable g_{n-j}. It is the root of a subterm $\bar{D}^{n-j}_{n-j}$, its upper arm is $\bar{D}^{n-j-1}_{n-j-1}$. The elimination is by setting the first argument of this g_{n-j} to $\perp$, getting C^{n-j}_n. Only then it is possible to lift the lower $g_{n-j}\mathsf{11}$ to the top level, getting D^{n-j-1}_n. There the new $g_{n-j}\mathsf{11}$ at the top level gets two arms which are copies of $\bar{D}^{n-j-1}_{n-j-1}$. The lower arm (of these two) stays the same in the following transformations (it contains the token $p\ldots prq\ldots q\mapsto\mathsf{0}$ with j arguments p). The upper arm undergoes further eliminations of tokens $p\ldots pq\ldots q\mapsto\mathsf{0}$. These further eliminations are only possible after the separation of the two arms.

Finally in the n-th step the $\mathsf{0}$ which stands at the end of the upper branches of D^0_n is set to $\perp$ getting C^0_n, eliminating the token $p\ldots p\mapsto\mathsf{0}$.

Proposition 3.5.6. *Let $n \geq 0$ and C^i_n, D^i_n be the terms defined above. Then the chain*

$$C^0_n \prec D^0_n \cong C^1_n \prec D^1_n \ldots D^{n-1}_n \cong C^n_n \prec D^n_n$$

between $[\![C^0_n]\!]$ and $[\![D^n_n]\!]$ has the least length $n+1$.

Proof. We assume $n \geq 1$ and suppose any chain between C^0_n and D^n_n and look at an intermediate $\prec$-step of this chain, i.e. we have the situation

$$C^0_n \leq M \prec N \leq D^n_n.$$

We assume that some token of the form $p\ldots pq\ldots q\mapsto\mathsf{0}$ is eliminated in this step. Let t be such token with the minimal number j of arguments p, and assume $j<n$.
Then we have

$$NP\ldots PQ\ldots Q \to^* \mathsf{0}, \text{ and } MP\ldots PRQ\ldots Q \to^* \mathsf{0},$$

because $C^0_n \leq M$ (both with j arguments P).
We can abstract the $(j+1)$st argument in these terms and build the terms

$$N' = \lambda g.NP\ldots PgQ\ldots Q \text{ and } M' = \lambda g.MP\ldots PgQ\ldots Q.$$

It is $M' \prec N'$. We can transform M', N' to game terms and apply the argument in the proof of proposition 3.5.3 to deduce: $M'P \to^* \mathsf{0}$.
So $MP\ldots PPQ\ldots Q \to^* \mathsf{0}$ (with $j+1$ arguments P).
As $Q \sqsubseteq_{op} P$, we also have $MP\ldots PQ\ldots Q \to^* \mathsf{0}$ for all $k \geq j+1$ arguments P.
All these arguments of M are minimal w.r.t. the stable order, because they are also minimal for D^n_n and it is $M \leq D^n_n$.
Therefore every token $p\ldots pq\ldots q\mapsto\mathsf{0}$ with $k \geq j+1$ arguments p is in M.
This shows that from the tokens of the form $p\ldots pq\ldots q\mapsto\mathsf{0}$ only the token t is eliminated in the step $M \prec N$. (For $j=n$ this is trivially the case.) As there are $n+1$ of these tokens to be eliminated, the chain must have at least $n+1$ steps. □

Our example of a chain of least length $n+1$ has n functional parameters g_i of arity 2 and is of grade 1. We could transform it into an "equivalent" example with only one functional parameter g of arity 3 and terms of grade n, by coding g_iMN as $giMN$.

Our results suggest an improvement of Berry's second conjecture:

Conjecture 3.5.7 (Chain Conjecture)**.** If $a \leq b$ are finite elements in an f-model, then there is a chain between a and b.
We will refute also this conjecture in section 3.7.

3.6 The stable order is not bounded complete: no bidomain

Gérard Berry showed that the fully abstract order-extensional cpo-model of PCF (our greatest f-model) together with the stable order forms a bicpo, and conjectured that it is also a bidomain (Berry's first conjecture). Here we repeat the definitions of both structures. We prove the conjecture for first-order types. Then we refute the general conjecture. Our first example is the stable lub of two finite elements for which the distributive law is not valid. Our second example consists of two finite elements with stable upper bound but without stable lub. Both examples are in PCF of second-order type of grade 2.

Definition 3.6.1 (Berry: 4.7.2 in [Ber79]). A *bicpo* is a structure $(D, \sqsubseteq, \leq, \bot)$ such that:

(1) The structure $(D, \sqsubseteq, \bot)$ is a cpo with least element $\bot$ and with a continuous glb-function $\sqcap$.

(2) The structure $(D, \leq, \bot)$ is a cpo with least element $\bot$ such that $a \leq b \Longrightarrow a \sqsubseteq b$ and for all $\leq$-directed sets S the two lubs are equal: $\bigvee S = \bigsqcup S$.

(3) The function $\sqcap$ is $\leq$-monotonic. (With (1) and (2) it follows that it is $\leq$-continuous.)

(4) For all $\leq$-directed sets S and S': If for all $a \in S$, $a' \in S'$ there are $b \in S$, $b' \in S'$ with $a \sqsubseteq b$, $a' \sqsubseteq b'$, $b \leq b'$, then $\bigsqcup S \leq \bigsqcup S'$.

In a bicpo: For all $a \uparrow_{\leq} b$, $a \sqcap b$ is also the glb w.r.t. $\leq$.

Theorem 3.6.2 (Berry: 4.8.10 in [Ber79]). *The domains $(D^\sigma, \sqsubseteq, \leq, \bot)$ of the fully abstract order-extensional cpo-model of PCF are bicpos.* $\square$

Definition 3.6.3 (Berry: 4.4.10 in [Ber79]). A cpo $(D, \leq, \bot)$ is *distributive* if

(1) it is bounded complete
(This means that for $a \uparrow_{\leq} b$ there is a lub $a \vee b$. And this entails with completeness that there is also a glb $a \wedge b$ for all a, b, even for $\leq$-incompatible ones.)
and

(2) for all $a, b, c \in D$ with $b \uparrow_{\leq} c$: $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$.

Definition 3.6.4 (Berry: 4.7.9 in [Ber79]). A bicpo $(D, \sqsubseteq, \leq, \bot)$ is *distributive* if $(D, \leq, \bot)$ is distributive and for all $a \uparrow_{\leq} b$: $a \vee b$ is also the lub w.r.t. $\sqsubseteq$.

(Please note that in a distributive bicpo only for $a \uparrow_{\leq} b$ it must be $a \wedge b = a \sqcap b$.)

Definition 3.6.5 (Berry: 4.7.12 in [Ber79]). A distributive bicpo $(D, \sqsubseteq, \leq, \bot)$ is a *bidomain* if there is a $\leq$-growing sequence $(\psi_i)_{i \geq 1}$ of finite projections w.r.t. $\leq$ and with lub $\bigvee \psi_i = \text{id}$. (This means: $\psi_i \colon D \to D$ is continuous w.r.t. $\sqsubseteq$ and $\leq$, $\psi_i \leq \text{id}$, $\psi_i \circ \psi_i = \psi_i$, $\psi_i \leq \psi_{i+1}$, $\psi_i(D)$ finite, $\bigvee \psi_i = \text{id}$.)

In this definition the sequence (ψ_i) is also a $\sqsubseteq$-growing sequence of finite projections w.r.t. $\sqsubseteq$ and with lub id. Together with the the glb-function $\sqcap$ it follows that $(D, \sqsubseteq, \bot)$ is a Scott domain, a bounded complete ω-algebraic cpo.

As we have explained in proposition 3.3.8 and 3.3.15, the conditions for (ψ_i) in the definition of bidomain are fulfilled for the fully abstract order-extensional cpo-model (and furthermore for all f-models) by the projections ψ_i^σ. In fact the $(D^\sigma, \leq)$ are stable ω-bifinite domains for the cpo-model, in the sense of definition 12.4.3 of [AC98].

To be precise, the condition of distributivity of the stable order was not conjectured by Berry in his thesis; there he remained agnostic. But in the state-of-the-art paper [BCL85] we can read: "Unfortunately we are not able to show that the domains of the fully abstract model are bidomains, although we definitely believe it; the problem is to show that the $\leq_{cm}$-lubs are taken pointwise."

First we clarify the situation for first-order types:

Theorem 3.6.6. *Let σ be a first-order type and $(D^\sigma, \sqsubseteq, \leq, \bot)$ be the corresponding domain of any f-model.*
The finite elements of D^σ fulfill distributivity w.r.t. $\leq$ in D^σ in the following sense:
For $a, b \in \mathcal{F}^\sigma$ the glb in D^σ exists and is given by $\mathcal{T}(a \wedge b) = \mathcal{T}(a) \cap \mathcal{T}(b)$.
For $a, b \in \mathcal{F}^\sigma$ with $a \uparrow_\leq b$ the lub in D^σ exists and is given by $\mathcal{T}(a \vee b) = \mathcal{T}(a) \cup \mathcal{T}(b)$. It is taken pointwise and it is also the lub w.r.t. $\sqsubseteq$.
Then the distributive law is fulfilled by set theory on traces.

If D^σ contains a denotation for every infinite game term of type σ (this is the case for the game model and every greater f-model), then D^σ is the domain of the greatest f-model. In this case all elements $a, b \in D^\sigma$ fulfill distributivity in the sense above. Therefore D^σ is a bidomain in this case.

Proof. Let $a, b \in \mathcal{F}^\sigma$.
We can apply theorem 3.5.1 and get a game term A with $a = [\![A]\!]$, and a game term $C \prec A$ with $\mathcal{T}[\![C]\!] = \mathcal{T}(a) \cap \mathcal{T}(b)$. Define $a \wedge b = [\![C]\!]$; it is finite and therefore in D^σ.

Now let $a \uparrow_\leq b$, i.e. there is some d with $a \leq d$ and $b \leq d$. By theorem 3.5.1 there are an infinite game term D with $d = [\![D]\!]$, and finite game terms A, B with $a = [\![A]\!]$, $b = [\![B]\!]$, $A \prec D$, $B \prec D$. Take the syntactical lub E of A and B. It is $\mathcal{T}[\![E]\!] = \mathcal{T}[\![A]\!] \cup \mathcal{T}[\![B]\!]$, because in first-order game terms branches correspond to tokens. Define $a \vee b = [\![E]\!]$; it is finite and therefore in D^σ. This lub is pointwise on the uncurried argument and therefore also the lub w.r.t. $\sqsubseteq$.

If D^σ contains a denotation for every infinite game term of type σ, then by theorem 3.5.1 D^σ is exactly the domain of the greatest f-model. The construction of $a \wedge b$ and $a \vee b$ for any $a, b \in D^\sigma$ is as above, only with infinite game terms. □

Conjecture 3.6.7. For all types of the form $\sigma = \sigma_1 \to \ldots \to \sigma_n \to \iota$, with $\sigma_i = \iota$ or $\sigma_i = \iota \to \iota$, Berry's first conjecture is valid, i.e. D^σ is a bidomain in the greatest f-model.

The proof of this conjecture is in preparation. It relies on the conjecture 3.5.2.

Now we prove some properties of stable upper bounds (sub) in f-models. (These are properties that are also valid in stable bifinite domains, see lemma 12.4.7 in [AC98].)

Theorem 3.6.8. *Let D^σ be a domain of an f-model, $\sigma = \sigma_1 \to \ldots \to \sigma_n \to \iota$, $n \geq 0$. Let X be a finite set of finite elements of D^σ that has a stable upper bound (sub) in D^σ. Let m be the maximal grade of the elements of X. For every sub x of X there is a unique minimal (w.r.t. $\leq$) sub y of X with $y \leq x$. Every minimal sub of X is finite of grade m; they are pairwise $\leq$-incompatible. The extensional lub $\bigsqcup X$ is one of those.*

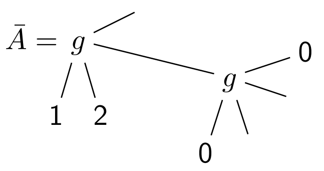
Proof. Let x be a sub of X. Then the projection $\psi^\sigma_m x$ is also a sub of X. Let Z be the set of all subs z of X with $z \leq \psi^\sigma_m x$; it is a non-empty finite set of finite elements. Then $y =$ (the glb of Z) is the desired unique minimal sub of X with $y \leq x$.

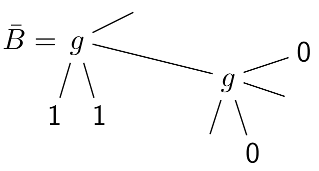
Let a, b be two minimal subs of X that are $\leq$-compatible. Then $a \sqcap b$ is also a sub of X, therefore $a = b$.

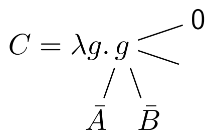
Let $g = \bigsqcup X$ and h some sub of X. We have to show that $f \leq g$ for every $f \in X$.
This is clear for $n = 0$, in the type ι.
Now let $n > 0$ and $\vec{x}$, $\vec{y}$ be two vectors of arguments of type $\sigma_1 \times \ldots \times \sigma_n$ with $\vec{x} \leq \vec{y}$.
We have to show that $f\vec{x} = f\vec{y} \sqcap g\vec{x}$.
It is $f\vec{x} = f\vec{y} \sqcap h\vec{x} \sqsupseteq f\vec{y} \sqcap g\vec{x}$. And $f\vec{x} \sqsubseteq f\vec{y} \sqcap g\vec{x}$ is clear.
This shows that g is a sub of X; of course it is also minimal w.r.t. $\leq$. □

3.6.1 A stable lub without distributivity

Our first counter-example to Berry's first conjecture is of type $(\iota \to \iota \to \iota) \to \iota$ and of grade 2. We consider the following game terms A, B, C, where we use a case_1 for a case_2 with the third arm $\bot$:







$A = \lambda g.\bar{A}$ $\qquad$ $B = \lambda g.\bar{B}$

Here are the traces of these terms:

$$\left.\begin{array}{l} A\left\{\begin{array}{ll}\{0\,\bot\mapsto 0, & 1\,2\mapsto 1\}\mapsto 0\\ \{0\,\bot\mapsto 0, & 1\,\bot\mapsto 1\}\mapsto 0\end{array}\right.\\ B\left\{\begin{array}{ll}\{\bot\,0\mapsto 0, & 1\,1\mapsto 1\}\mapsto 0\\ \{\bot\,0\mapsto 0, & \bot\,1\mapsto 1\}\mapsto 0\end{array}\right.\\ \quad\;\;\{\bot\bot\mapsto 0 \qquad\qquad\;\; \}\mapsto 0\end{array}\right\} C$$

It is $A \leq C$ and $B \leq C$. We will show that C is the stable lub of A and B.

The intuition of the example: A and B do not contain the token $\{\bot\bot\mapsto 0\}\mapsto 0$, because their two occurrencies of g are forced to evaluate their first resp. second argument, to get different results for different arguments. (This is the same trick that was used in the preceding section.) C adds to the tokens of A and B just the token $\{\bot\bot\mapsto 0\}\mapsto 0$, to separate $\bar{A}$ and $\bar{B}$. (Note that a g for which Cg converges cannot demand both its arguments 00.) Therefore this lub does not fulfill distributivity. In C it is not possible to lift a differing term gMN to the top level that would eliminate that token, because the five occurrences of g in C cannot be "unified" to a common term that would always converge.

Proposition 3.6.9. *Let A, B, C be the game terms above. $[\![C]\!]$ is the stable lub of $a = [\![A]\!]$ and $b = [\![B]\!]$. Let d be the finite element with the trace $\{\{\bot\bot\mapsto 0\}\mapsto 0\}$. Then $d \wedge (a \vee b) \neq (d \wedge a) \vee (d \wedge b)$. This refutes Berry's first conjecture.*

Proof. By the game term theorem 3.4.12 and the preceding theorem 3.6.8, every minimal sub of A and B can be represented by a game term of grade 2. Such a game term is of the form $\lambda g.S$, where $S\colon \iota$ is a game term possibly with the only free variable g. We abbreviate $S[g := M]$ as $S[M]$.

We use the following terms as arguments:

$$\begin{array}{ll} Q = \lambda xy.\,\mathsf{case}_1\,x\mathsf{0}(\mathsf{case}_2\,y\bot\bot\mathsf{1}) & \mathcal{T}[\![Q]\!] = \{\mathsf{0}\bot{\mapsto}\mathsf{0},\ \ \mathsf{12}{\mapsto}\mathsf{1}\} \\ R = \lambda xy.\,\mathsf{case}_1\,y\mathsf{0}(\mathsf{case}_1\,x\bot\mathsf{1}) & \mathcal{T}[\![R]\!] = \{\bot\mathsf{0}{\mapsto}\mathsf{0},\ \ \mathsf{11}{\mapsto}\mathsf{1}\} \\ P = \lambda xy.\mathsf{0} & \mathcal{T}[\![P]\!] = \{\bot\bot{\mapsto}\mathsf{0}\} \end{array}$$

Q and R are compatible in the sense that they produce compatible results for the same argument. We will prove that for any term S of the form above:

$$\text{If } S[Q] \to^* \mathsf{0} \text{ and } S[R] \to^* \mathsf{0},\ \text{then } S[P] \to^* \mathsf{0}.$$

The proof is by induction on the term S: The cases $S = \bot, \mathsf{0}, \mathsf{1}, \mathsf{2}$ are clear.
Let $S = \mathsf{case}_2(gS_1S_2)S_3S_4S_5$.
For $S[Q] \to^* \mathsf{0}$ it must be $S_1[Q] \to^* \mathsf{0}$ or $S_2[Q] \to^* \mathsf{2}$.

(1) case $S_1[Q] \to^* \mathsf{0}$:
For $S[R] \to^* \mathsf{0}$ it must be $S_2[R] \to^* \mathsf{0}$ or $S_1[R] \to^* \mathsf{1}$.

(1.1) case $S_2[R] \to^* \mathsf{0}$:
We have $S[Q] \to^* S_3[Q] \to^* \mathsf{0}$ and $S[R] \to^* S_3[R] \to^* \mathsf{0}$.
By the induction hypothesis for S_3 we get $S_3[P] \to^* \mathsf{0}$, therefore $S[P] \to^* \mathsf{0}$.

(1.2) case $S_1[R] \to^* \mathsf{1}$:
This is not possible, as Q and R are compatible in the sense above.

(2) case $S_2[Q] \to^* \mathsf{2}$:
For $S[R] \to^* \mathsf{0}$ it must be $S_2[R] \to^* \mathsf{0}$ or $S_2[R] \to^* \mathsf{1}$.
Both cases are not possible, as Q and R are compatible in the sense above.

So we have shown that for every $\sqsubseteq$-upper bound D of grade 2 of A and B it must be $DP \to^* \mathsf{0}$. For a $\leq$-upper bound it cannot be $D\bot \to^* \mathsf{0}$. Therefore P is a $\leq$-minimal argument to fulfill $DP \to^* \mathsf{0}$. This means: Any minimal stable upper bound of A and B must contain the token $\{\bot\bot{\mapsto}\mathsf{0}\}{\mapsto}\mathsf{0}$. So C is the stable lub of A and B. (It is also the $\sqsubseteq$-lub.) $\square$

Remark 3.6.10 (alternative proof with Sieber sequentiality relation)**.** Because we work in the proof above on game terms, the induction hypothesis is simpler and the proof shorter than a proof by induction on general terms. A short purely semantic proof for general terms is possible with a Sieber sequentiality logical relation [Sie92].

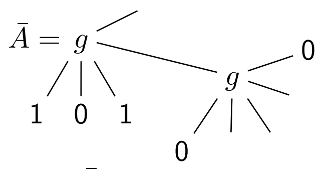
We can show that there is no definable function that fulfills the value table $[\![Q]\!]{\mapsto}\mathsf{0}$, $[\![R]\!]{\mapsto}\mathsf{0}$, $[\![P]\!]{\mapsto}n$ for $n \neq \mathsf{0}$. We use the sequentiality relation $rel = S^3_{\{1,2\}\{1,2,3\}}$.
For $d_1, d_2, d_3\colon \iota$ it is $(d_1, d_2, d_3) \in rel$ iff $d_1 = \bot$ or $d_2 = \bot$ or $d_1 = d_2 = d_3$.
First, the output column $(\mathsf{0}, \mathsf{0}, n)$ of the value table is not in this relation.
Then we have to show that $([\![Q]\!], [\![R]\!], [\![P]\!]) \in rel$ (on the type $\iota \to \iota \to \iota$).
Suppose we have

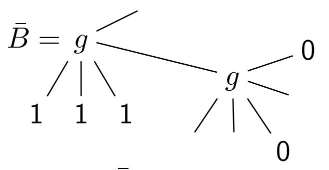
$$[\![Q]\!]a_1b_1 = c_1,$$
$$[\![R]\!]a_2b_2 = c_2,$$
$$[\![P]\!]a_3b_3 = c_3$$

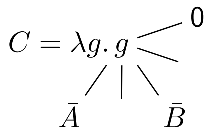
and suppose $(c_1, c_2, c_3) \notin rel$. We have to show that $(a_1, a_2, a_3) \notin rel$ or $(b_1, b_2, b_3) \notin rel$.
It must be $c_3 = 0$.
It cannot be $c_1 = \bot$, so it must be $c_1 = 0$ or $c_1 = 1$:
If $c_1 = 0$, then it cannot be $c_2 = 0$, so it must be $c_2 = 1$, then $a_1 = 0$, $a_2 = 1$, therefore $(a_1, a_2, a_3) \notin rel$, end of proof for $c_1 = 0$.
If $c_1 = 1$, then it is $c_2 = 0$ or $c_2 = 1$:
If $c_2 = 0$, then $b_1 = 2$, $b_2 = 0$, therefore $(b_1, b_2, b_3) \notin rel$.
If $c_2 = 1$, then $b_1 = 2$, $b_2 = 1$, therefore $(b_1, b_2, b_3) \notin rel$.

It is no surprise that we have to perform a case analysis of similar complexity as in the proof above. But it is interesting that the whole proof of this remark can be done mechanically by the computer program written by Allen Stoughton [Sto94]. For a general system of ground constants, this program takes a value table of a second-order function and returns either a term defining such a function or a logical relation proving its undefinability.

Our counter-example is of grade 2 with g of arity 2. There is an "equivalent" example of grade 1 with g of arity 3:





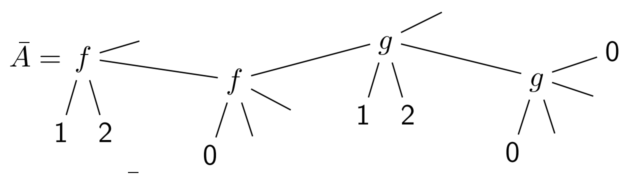


$A = \lambda g.\bar{A}$ $B = \lambda g.\bar{B}$

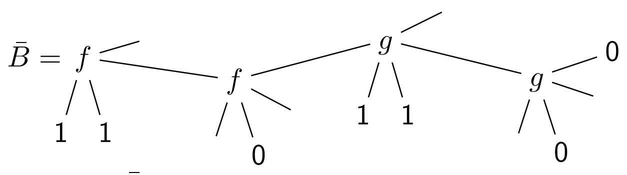
Conjecture 3.6.11. In $\mathcal{F}_1^{(\iota\to\iota\to\iota)\to\iota}$, the finite elements of grade 1 of the type $(\iota\to\iota\to\iota)\to\iota$, Berry's first conjecture is valid; this subdomain is a bidomain. (This is a finite combinatorial problem and could be solved by a computer program.)

3.6.2 Two elements without stable lub

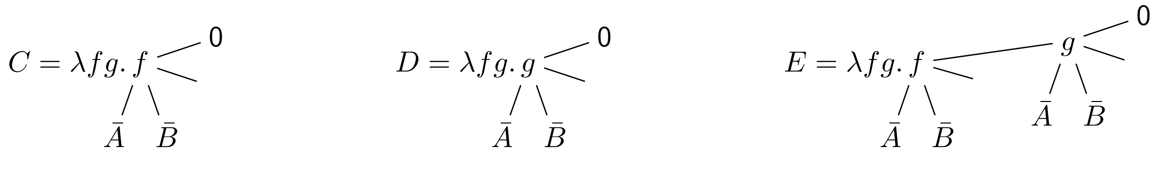
Now to our counter-example to bounded completeness of the stable order. It is of type $(\iota \to \iota \to \iota) \to (\iota \to \iota \to \iota) \to \iota$ and of grade 2. It employs the trick of our last example twice to two functional parameters. Consider the following game terms A, B, C, D, E, where we use a case$_1$ for a case$_2$ with the third arm $\bot$.



$A = \lambda fg.\bar{A}$



$B = \lambda fg.\bar{B}$



The traces of the terms are:

$$\begin{aligned}
\mathcal{T}[\![A]\!] &= \{\,0\,\bot\mapsto 0\,,\ 1\ \underset{\bot}{2}\mapsto 1\,\}\mapsto\{\,0\,\bot\mapsto 0\,,\ 1\ \underset{\bot}{2}\mapsto 1\,\}\mapsto 0\\
\mathcal{T}[\![B]\!] &= \{\bot\,0\mapsto 0\,,\ \underset{\bot}{1}\ 1\mapsto 1\,\}\mapsto\{\bot\,0\mapsto 0\,,\ \underset{\bot}{1}\ 1\mapsto 1\,\}\mapsto 0\\
\mathcal{T}[\![C]\!] &= \mathcal{T}[\![A]\!]\cup\mathcal{T}[\![B]\!]\cup\{\{\bot\bot\mapsto 0\,\}\mapsto\bot \qquad\qquad \mapsto 0\,\}\\
\mathcal{T}[\![D]\!] &= \mathcal{T}[\![A]\!]\cup\mathcal{T}[\![B]\!]\cup\{\bot \qquad\qquad \mapsto\{\bot\bot\mapsto 0\,\}\mapsto 0\,\}\\
\mathcal{T}[\![E]\!] &= \mathcal{T}[\![A]\!]\cup\mathcal{T}[\![B]\!]\cup\{\{\bot\bot\mapsto 0\,\}\mapsto\{\bot\bot\mapsto 0\,\}\mapsto 0\,\}
\end{aligned}$$

The token of $\mathcal{T}[\![A]\!]$ entails three more tokens: (1) with the first indicated 2 replaced by $\bot$, (2) with the second indicated 2 replaced by $\bot$, (3) with both replaced by $\bot$. Likewise for the token of $\mathcal{T}[\![B]\!]$. (These entailments are due to securedness, see the definition 2 of [CPW00].)

C, D, E are three stable upper bounds of A and B; we will show that they are just the minimal stable upper bounds. E is the $\sqsubseteq$-lub of A and B.

The intuition of the example: In an upper bound of A and B, both have to be separated by some function call at the top level; because A and B cannot be "unified". There are three ways to choose the separator: f or g or (both f and g), realized by C, D, E resp.

Proposition 3.6.12. *Let A, B, C, D, E be the game terms above. $[\![C]\!], [\![D]\!], [\![E]\!]$ are the minimal stable upper bounds of $[\![A]\!]$ and $[\![B]\!]$. So $[\![A]\!]$ and $[\![B]\!]$ have no stable lub. (This again refutes Berry's first conjecture.)*

Proof. By theorem 3.6.8, every minimal sub of A and B is of grade 2. By the game term theorem, we restrict to game terms of grade 2. These game terms must have the form $\lambda fg.S$. We use the terms Q, R, P of the proof of proposition 3.6.9. Our claim is: For every term S of the form above,

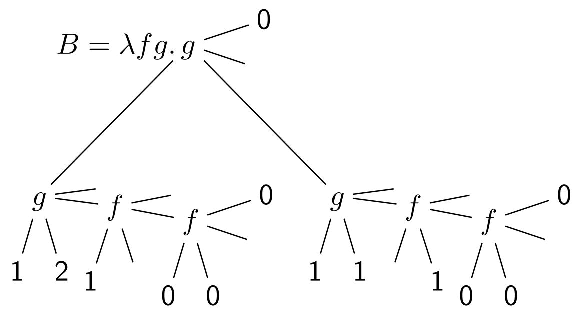
$$\text{if } S[f := Q, g := Q] \rightarrow^* \mathsf{0} \text{ and } S[f := R, g := R] \rightarrow^* \mathsf{0}, \text{ then } S[f := P, g := P] \rightarrow^* \mathsf{0}.$$

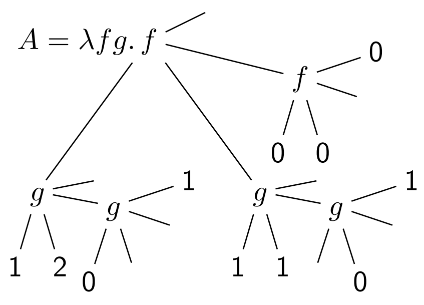
The proof of the claim is by induction on the term S and follows exactly the proof of proposition 3.6.9. There is only one additional case $S = \mathsf{case}_2(fS_1S_2)S_3S_4S_5$ of the same scheme.

So we have shown that for every $\sqsubseteq$-upper bound F of grade 2 of A and B it must be $FPP \rightarrow^* \mathsf{0}$. For a $\leq$-upper bound it cannot be $F\bot\bot \rightarrow^* \mathsf{0}$. Hence the minimal arguments to fulfill $FPP \rightarrow^* \mathsf{0}$ must be (P, P), $(P, \bot)$ or $(\bot, P)$. This is fulfilled by E, C, D respectively. $\square$

3.7 Refutation and improvement of the chain conjecture

The chain conjecture 3.5.7 said that for finite elements $a \leq b$ there is a chain between a and b, see the definition 3.5.5 of chain. We give here a counter-example in the type $(\iota \to \iota \to \iota) \to (\iota \to \iota \to \iota) \to \iota$ of grade 2. Consider the following game terms A, B:





Here are the traces of these terms:

$$A\left\{\begin{array}{l} \{0\,0 \mapsto 0,\ \ 1\,\bot \mapsto 1\} \mapsto \{0\,\bot \mapsto 0,\ \ 1\,2 \mapsto 1\} \mapsto 0 \\ \ \ \bot \qquad\qquad\qquad\qquad\qquad\quad \bot \\ \{0\,0 \mapsto 0,\ \ \bot\,1 \mapsto 1\} \mapsto \{\bot\,0 \mapsto 0,\ \ 1\,1 \mapsto 1\} \mapsto 0 \\ \bot \qquad\qquad\qquad\qquad\qquad\quad \bot \end{array}\right. \hspace{-1em}\left.\begin{array}{l} \\ \\ \\ \\ \bot \qquad\qquad\quad \mapsto \{\bot\bot \mapsto 0 \qquad\quad\ \} \mapsto 0 \end{array}\right\} B$$

The first token entails three more tokens: (1) with the indicated 0 replaced by $\bot$, (2) with the indicated 2 replaced by $\bot$, (3) with both replaced by $\bot$. Likewise for the second token.

It is $A \leq B$. B contains just one more token t than A. Assume that there is a chain between $[\![A]\!]$ and $[\![B]\!]$. Then t is eliminated in a definite step $A' \prec B'$ of the chain, with $A \cong A'$ and $B \cong B'$. We will show that such $A' \prec B'$ do not exist.

The intuition of the example: It is derived from the example of subsection 3.6.1. A and B are like the term C of that example. For B: In the left leg of the upper g the subterm $g0\bot$ (of C) is replaced by the subterm demanding the first argument of f. In the right leg the subterm $g\bot 0$ (of C) is replaced by the subterm demanding the second argument of f. This ensures that not both legs (of the upper g) can be evaluated. There is again no term with g that could be lifted to the top level and that would eliminate the token $\bot \mapsto \{\bot\bot \mapsto 0\} \mapsto 0$. Therefore there is no $\prec$-step leading from A to B. But the subterms with f can be lifted to the top replacing the upper g of B (as "separator" of $g12$ and $g11$), so we get A with that token eliminated. Here the subterms $g0\bot$ and $g\bot 0$ of the former example C appear again; they must appear to ensure that A gets the first eight tokens of B and ensure that not both legs of the upper f can be evaluated.

Proposition 3.7.1. *Let A, B be the game terms of grade 2 above. There are no game terms A', B' of grade 2 with $A' \prec B'$ and $A' \cong A$, $B' \cong B$. Then by the game term theorem there are no PCF-terms A', B' with this property. This refutes the chain conjecture.*

Proof. As game terms of grade 2, A' and B' should be of the form $\lambda fg.S$, where $S\colon \iota$ is a game term possibly with the only free variables f, g. We abbreviate $S[f := M, g := N]$ as $S[M, N]$.

We use the terms of the proof of proposition 3.6.9 as arguments for g:

$$\begin{array}{ll} Q = \lambda xy.\, \mathsf{case}_1\, x 0 (\mathsf{case}_2\, y \bot\bot 1) & \mathcal{T}[\![Q]\!] = \{0\bot \mapsto 0,\ \ 12 \mapsto 1\} \\ R = \lambda xy.\, \mathsf{case}_1\, y 0 (\mathsf{case}_1\, x \bot 1) & \mathcal{T}[\![R]\!] = \{\bot 0 \mapsto 0,\ \ 11 \mapsto 1\} \\ P = \lambda xy. 0 & \mathcal{T}[\![P]\!] = \{\bot\bot \mapsto 0\} \end{array}$$

We use the following terms as arguments for f:

$$Q' = \lambda xy.\,\mathsf{case}_1\, x(\mathsf{case}_1\, y0\bot)1 \qquad \mathcal{T}[\![Q']\!] = \{00{\mapsto}0,\ \ 1\bot{\mapsto}1\}$$
$$R' = \lambda xy.\,\mathsf{case}_1\, y(\mathsf{case}_1\, x0\bot)1 \qquad \mathcal{T}[\![R']\!] = \{00{\mapsto}0,\ \ \bot 1{\mapsto}1\}$$

The pairs (Q', Q) and (R', R) are compatible in the sense that their replacement into the same integer term leads to compatible results.

We will prove that for any terms S, S' of the form above:

$$\text{If } S' \prec S \text{ and } S[\bot, P] \to^* 0, \quad S'[Q', Q] \to^* 0, \quad S'[R', R] \to^* 0, \text{ then } S'[\bot, P] \to^* 0.$$

The proposition follows immediately from this claim.

The proof is by induction on the term S: The cases $S = \bot, 0, 1, 2$ are clear.
Let $S = \mathsf{case}_2(gS_1S_2)S_3S_4S_5$ and $S' \prec S$ with $S' = \mathsf{case}_2(gS_1'S_2')S_3'S_4'S_5'$. (The case $S' = \bot$ is clear.)
Assume the three conditions of the claim.
Let $(gS_1'S_2')[Q', Q] \to^* q$ and $(gS_1'S_2')[R', R] \to^* r$, both terms must converge to integer constants.
From the compatibility of (Q', Q) and (R', R) follows the compatibility of q and r, so either $q = r = 0$ or $q = r = 1$.
As (Q', Q) and (R', R) are compatible, it cannot be $S_2'[Q', Q] \to^* 2$ and $S_2'[R', R] \to^* 1$.
Therefore $q = r = 0$.
Then we get $S_3[\bot, P] \to^* 0$, $S_3'[Q', Q] \to^* 0$, $S_3'[R', R] \to^* 0$.
By the induction hypothesis for S_3 we conclude $S_3'[\bot, P] \to^* 0$, hence $S'[\bot, P] \to^* 0$. This fulfills the claim.

Now let $S = \mathsf{case}_2(fS_1S_2)S_3S_4S_5$.
Then $S[\bot, P] \cong \bot$, so the claim is fulfilled. □

The refutation of the chain conjecture shows that already for second-order types the correspondence of stable and syntactic order is destroyed; there seems to be no simple syntactic characterization of the stable order. But certainly the two orders are related, but in which sense? A weaker conjecture that is now open is the following:

Conjecture 3.7.2 (Maximality Conjecture)**.** Every PCF-term without Y that is syntactically maximal (i.e. contains no $\bot$) is also stably maximal.

The existence of chains of any length suggests a kind of "metric" on finite elements $a \leq b$: If there is a chain between a and b of least length n, then the distance of a and b is n. If there is no chain, then the distance is ∞. But it might be doubted if this is meaningful, or if a transition $A \leq B$ like the example above (without chain) should also be counted as some kind of elementary step of finite distance.

The example $A \leq B$ above shows us that the syntactic order $\prec$ is not enough to give a syntactic description of the stable order; there are more "syntactic" relations needed. We can imagine that A is produced from B by "forcing" the upper g in B to be strict in one of its two arguments, so that the token $\bot{\mapsto}\{\bot\bot{\mapsto}0\}{\mapsto}0$ is eliminated.

We tentatively propose an improved chain conjecture with such a new syntactic relation of "strictification". For this we have to extend PCF with a new operator. The theory of this extension has still to be properly developed; so all propositions in the rest of this section have the status of conjectures.

In [Pao06] Luca Paolini extends PCF with two new operators, one of them called strict? of type $(\iota \to \iota) \to \iota$. Suppose the operational semantics is given by an evaluation procedure eval. Then strict? obeys the rules:

$$\begin{aligned}&\text{If } \operatorname{eval}(M\mathsf{0}){\downarrow} \text{ and } \operatorname{eval}(M\bot){\uparrow} \text{ then } \operatorname{eval}(\mathsf{strict?}\,M) = \mathsf{0}\\ &\text{If } \operatorname{eval}(M\mathsf{0}){\downarrow} \text{ and } \operatorname{eval}(M\bot){\downarrow} \text{ then } \operatorname{eval}(\mathsf{strict?}\,M) = \mathsf{1}\end{aligned}$$

Here $X{\downarrow}$ means that X evaluates to some integer constant, $X{\uparrow}$ is the negation. Paolini also gives an effective evaluation for strict?.

We use instead a new constant $\mathsf{str}\colon (\iota \to \iota) \to \iota$ that is the "strict half" of strict?, i.e. we have the only rule:

$$\text{If } \operatorname{eval}(M\mathsf{0}) = \mathsf{0} \text{ and } \operatorname{eval}(M\bot){\uparrow} \text{ then } \operatorname{eval}(\mathsf{str}\,M) = \mathsf{0}$$

str can be expressed by a term with strict?, but strict? cannot be expressed by str. Note that our str is finite. An effective evaluation could also be given for str. ($\mathsf{str}\,M$ tests if $M\mathsf{0}$ evaluates to 0 and in this process checks if M demands its argument 0.)

On the extended language (PCF+str) the operational equivalence $\cong$ is defined in the usual way by observation through program contexts. It is extensional, i.e. $M \cong N$ iff for all $M' \cong N'$ it is $MM' \cong NN'$. There is a fully abstract semantics $[\![\ \ ]\!]$ given by equivalence classes of terms; these equivalence classes are construed as functions. These functions are stable; we can define a trace semantics $\mathcal{T}[\![\ \ ]\!]$ in the usual way, with the stable order $\leq$ as the inclusion relation on traces. All denotations are monotonic w.r.t. the stable order $\leq$.

str has the trace semantics

$$\mathcal{T}[\![\mathsf{str}]\!] = \{\{\mathsf{0}{\mapsto}\mathsf{0}\}{\mapsto}\mathsf{0}\}$$

Note that the token $\{\mathsf{0}{\mapsto}\mathsf{0}\}{\mapsto}\mathsf{0}$ expresses the fact that the argument function $\{\mathsf{0}{\mapsto}\mathsf{0}\}$ is strict, its argument 0 is needed. Note that str is not monotonic w.r.t. the extensional order of PCF; it is $[\![\mathsf{str}]\!]\{\mathsf{0}{\mapsto}\mathsf{0}\} = \mathsf{0}$, but $[\![\mathsf{str}]\!]\{\bot{\mapsto}\mathsf{0}\} = \bot$. It is

$$\mathcal{T}[\![\mathsf{str}]\!] \subseteq \mathcal{T}[\![\lambda f.\,\mathsf{if}\ f\mathsf{0}\ \mathsf{then}\ \mathsf{0}\ \mathsf{else}\ \bot]\!] = \{\{\mathsf{0}{\mapsto}\mathsf{0}\}{\mapsto}\mathsf{0},\ \{\bot{\mapsto}\mathsf{0}\}{\mapsto}\mathsf{0}\}$$

All semantic elements preserve compatibility in the following sense. Let us define the relation $\uparrow_h$ of *hereditary compatibility* on denotations: for integers it is $m \uparrow_h n$ if $m = \bot$ or $n = \bot$ or $m = n$. For functions it is $f \uparrow_h g$ if for all $x \uparrow_h y$: $fx \uparrow_h gy$. All our functions f of (PCF+str) have the property that $f \uparrow_h f$. Paolini's operator strict? does not have it.

With str we can define functions $\text{strictify}_n\colon \sigma_n \to \sigma_n$, where $\sigma_n = (\iota \to \ldots \to \iota \to \iota)$ with $n \geq 1$ arguments. E.g. $\text{strictify}_2\colon (\iota \to \iota \to \iota) \to (\iota \to \iota \to \iota)$,

$$\text{strictify}_2 = \lambda gxy.\,\mathsf{if}\,(\mathsf{str}[\lambda z.\,\mathsf{if}\ g(\mathsf{if}\ z\,\mathsf{then}\,x\,\mathsf{else}\,\bot)(\mathsf{if}\ z\,\mathsf{then}\,y\,\mathsf{else}\,\bot)\ \mathsf{then}\ \mathsf{0}\ \mathsf{else}\ \mathsf{0}])\ \mathsf{then}\ gxy\ \mathsf{else}\ \bot$$

$\text{strictify}_2\,gxy$ tests if gxy converges and $g\bot\bot$ diverges, and outputs gxy in this case. So $\text{strictify}_2\,gxy$ "forces" g to be strict in one of its two arguments. If it is not, then the output is $\bot$.

Let us replace in the example term B above the upper occurrence of g by (strictify$_2$ g) to get a new term B'. Then $[\![B']\!] = [\![A]\!]$, in the semantics of (PCF+str). A is a "strictification" of B.

If M is a term of (PCF+str), then unstr(M) is defined as the term M with all occurrences of str replaced by $\lambda f.\,\mathsf{if}\, f\mathsf{0}\;\mathsf{then}\;\mathsf{0}\;\mathsf{else}\;\bot$. So unstr($M$) is a PCF-term and $M \leq \text{unstr}(M)$, in the semantics of the extended language.

Now we can define our complementary "syntactic" relation.

Definition 3.7.3. Let M, N be PCF-terms of the same type. M is a *strictification* of N, written $M \prec^s N$, if there is a (PCF+str)-term M' with $[\![M]\!] = [\![M']\!]$ (in the semantics of (PCF+str)) and $[\![\text{unstr}(M')]\!] = [\![N]\!]$ (in the semantics of PCF).

Note that for PCF-terms M, N: $(M \prec^s N \Longrightarrow [\![M]\!] \leq [\![N]\!])$ and $(M \cong N \Longrightarrow M \prec^s N)$.

Conjecture 3.7.4 (improved chain conjecture)**.** In PCF we have: For all finite elements $a \leq b$ there is a sequence (M_i) of terms with $1 \leq i \leq n$, $[\![M_1]\!] = a$, $[\![M_n]\!] = b$, and for every $i < n$ it is $M_i \prec M_{i+1}$ or $M_i \prec^s M_{i+1}$.

A proof of this conjecture would be non-trivial and should first be tried on second-order types. (It might be that types higher than second-order need new higher-type strictness operators that cannot be defined from str.) Perhaps the situation should first be clarified in the realm of (PCF+str) and a conjecture of this kind should be proved there.

Our (PCF+str) is the "weakest" sequential extension of PCF with a control operator. It is properly included in (PCF+$\mathsf{strict?}$), this in turn is included in (PCF+H), the sequentially realizable functionals of John Longley [Lon02]; see section 9 in [Pao06] for an overview of such extensions of PCF. (PCF+H) is included in SPCF (mentioned in the introduction), which is no more extensional. For all these extensions of PCF it would be interesting to give syntactic characterizations of the stable order. First it should be clarified if all types are definable retracts of some lower order types, as is the case for (PCF+H) and SPCF. This could make the proofs easier, as we will see for unary PCF in the following section.

3.8 Unary PCF

Here we will prove Berry's conjectures for unary PCF, with the aid of Jim Laird's results [Lai05]. Unary PCF is the calculus of PCF without Y and with the only constant $\mathsf{0}$ and case_0-expressions. Its semantics is given by the finite elements of $\mathcal{F}_0^\sigma$ for all σ, with the orders $\sqsubseteq$ and $\leq$.

We first repeat the general closure properties of the $\mathcal{F}_i^\sigma$, seen as embedded in the D^σ of an f-model, taken from lemma 3.3.3, proposition 3.3.15 and theorem 3.6.8.

Proposition 3.8.1. *The $\mathcal{F}_i^\sigma$ are finite and downward closed w.r.t. $\leq$.*
For $a, b \in \mathcal{F}_i^\sigma$, $a \sqcap b \in \mathcal{F}_i^\sigma$ is the glb w.r.t. $\sqsubseteq$ in D^σ and $\mathcal{F}_i^\sigma$. For $a \uparrow_\leq b$ it is also the glb w.r.t. $\leq$.
For $a, b \in \mathcal{F}_i^\sigma$ that are $\sqsubseteq$-bounded in D^σ, $a \sqcup b \in \mathcal{F}_i^\sigma$ is the lub w.r.t. $\sqsubseteq$ in D^σ and $\mathcal{F}_i^\sigma$.
For a finite set $X \subseteq \mathcal{F}_i^\sigma$ that has a stable upper bound, all minimal stable upper bounds of X are in $\mathcal{F}_i^\sigma$. The extensional lub $\bigsqcup X$ is one of those. If X has a stable lub, then it is $\bigsqcup X$.

To apply Laird's results on definable retractions, we augment unary PCF with product types $\sigma \times \tau$. The constructs of the whole language are:
$\mathsf{0}\colon \iota$, $\perp^\sigma\colon \sigma$, $x^\sigma\colon \sigma$
If $M\colon \tau$, then $\lambda x^\sigma.M\colon \sigma \to \tau$.
If $M\colon \sigma \to \tau$ and $N\colon \sigma$, then $MN\colon \tau$.
If $M, N\colon \iota$, then $\mathsf{case}_0\, MN\colon \iota$.
If $M\colon \sigma$ and $N\colon \tau$, then $\langle M, N\rangle\colon \sigma \times \tau$.
If $M\colon \sigma \times \tau$, then $\pi_1 M\colon \sigma$ and $\pi_2 M\colon \tau$.

The reduction rules are:
$(\lambda x.M)N \to M[x := N]$
$\mathsf{case}_0\, \mathsf{0} M \to M$
$\pi_1\langle M, N\rangle \to M$
$\pi_2\langle M, N\rangle \to N$

This section needs the products only as auxiliary constructions for the first-order types that are the targets of Laird's retractions. In this section the underlying language is always the augmented unary PCF with products if products are not explicitly excluded.

Laird defines in [Lai05] a categorical notion of *standard model* of unary PCF together with order-extensionality and partial extensional order at each type. He defines *parallel composition* as the function f with $f\langle\perp,\perp\rangle = \perp$, $f\langle\perp,\mathsf{0}\rangle = f\langle\mathsf{0},\perp\rangle = \mathsf{0}$, $f\langle\mathsf{0},\mathsf{0}\rangle = \mathsf{0}$. A model is *universal* at type τ if every element of τ is the denotation of a term.

Definition 3.8.2 (Laird, definition 3.4 in [Lai05]). Given types σ, τ, a *definable retraction from σ to τ* (in a model $\mathcal{M}$) (written $\mathrm{Inj} : \sigma \trianglelefteq \tau : \mathrm{Proj}$ or just $\sigma \trianglelefteq \tau$) is a pair of (closed) terms $\mathrm{Inj}\colon \sigma \to \tau$ and $\mathrm{Proj}\colon \tau \to \sigma$ such that $[\![\lambda x.\, \mathrm{Proj}(\mathrm{Inj}\, x)]\!] = \mathrm{id}$ in $\mathcal{M}$.

Lemma 3.8.3 (Laird, lemma 3.10 in [Lai05]). *For any type τ there is a natural number n such that there is a definable retraction from τ to some binary product form of $(\iota \to \iota)^n$; the same retraction for any standard order-extensional model without parallel composition.*

Theorem 3.8.4 (Laird, theorem 3.11 in [Lai05]). *Any standard model of unary PCF which is order-extensional and excludes parallel composition is universal.*

We can build the stable biorder model of unary PCF as a collection of bicpos $(E^\sigma, \sqsubseteq, \leq)$ for every type σ: We start with $E^\iota = \{\perp, \mathsf{0}\}$ and $\perp \sqsubseteq \mathsf{0}$, $\perp \leq \mathsf{0}$.
$E^{\sigma\times\tau} = E^\sigma \times E^\tau$ with the usual $\sqsubseteq$ and $\leq$.
$E^{\sigma\to\tau}$ is the set of stable and monotone functions $f\colon (E^\sigma, \sqsubseteq, \leq) \to (E^\tau, \sqsubseteq, \leq)$. (If $x \sqsubseteq y$ then $fx \sqsubseteq fy$. If $x \leq y$ then $fx \leq fy$. If $x \uparrow_\leq y$ then $f(x \sqcap y) = fx \sqcap fy$. Continuity conditions are not necessary as the domains are finite.) $E^{\sigma\to\tau}$ is ordered by the usual $\sqsubseteq$ and $\leq$.

$(E^\sigma, \sqsubseteq, \leq)$ is not only a bicpo, but a distributive bicpo where the stable lub of two $\leq$-compatible functions is defined pointwise, by proposition 4.7.10 in Berry's thesis [Ber79]. (If $f \uparrow_\leq f'$, then $(f \vee f')x = fx \vee f'x$.) Therefore the stable lub of two elements is also defined by union on traces.

The stable biorder model fulfills the conditions of theorem 3.8.4, therefore it is universal (and fully abstract). This means that $(E^\sigma, \sqsubseteq, \leq)$ is isomorphic to $(\mathcal{F}_0^\sigma, \sqsubseteq, \leq)$ for types σ without products. In the following the semantics of unary PCF-terms is always taken in the model $(E^\sigma, \sqsubseteq, \leq)$. All this proves Berry's first conjecture for unary PCF:

Theorem 3.8.5 (Laird [Lai05]). *For every type σ without products, the structure $(\mathcal{F}_0^\sigma, \sqsubseteq, \leq)$ is a distributive bicpo (hence also a bidomain as it is finite).*
For $a, b \in \mathcal{F}_0^\sigma$ with $a \uparrow_\leq b$, $a \vee b$ is given by $\mathcal{T}(a \vee b) = \mathcal{T}(a) \cup \mathcal{T}(b)$ and this lub is taken pointwise for functions a, b.

With the aid of Laird's definable retractions we can prove a strong form of Berry's second conjecture for unary PCF, based on the fact that it is valid for first-order types. First we need two lemmas on the reduction.

Lemma 3.8.6. *The reduction $\to$ on unary PCF with products is confluent and strongly normalizing. Therefore it has unique normal forms. The normal form of a term of a type without products does not contain any product subterm.*

Proof. The confluence can be proved with the main theorem of [Mül92], see also [Bet03, theorem 10.4.15, page 576]: The rules of $\to$ without the β-rule are confluent on the applicative terms (i.e. the terms without λ), as they are orthogonal; they are left-linear and not variable-applying. Therefore their combination with the β-rule is confluent.

For the proof of strong normalization there seems to be no theorem in the literature that would provide an easy modular check for the simply typed λ-calculus with algebraic rewrite rules of our form.

Therefore we take the proof of strong normalization of the simply typed λ-calculus with products in the textbook [GTL89, chapter 6] for the only atomic type ι and augment it by the constant $\mathsf{0}$ and case_0-expressions. The proof stays literally the same. The only thing we have to add is a proof that if M, N are strongly normalizable, then $\mathsf{case}_0\, MN$ is so; in the proof that all terms are reducible. $\square$

Lemma 3.8.7. *Let ω be the following map on unary PCF-terms (where $n, m \geq 0$):*

$$\begin{aligned}
\omega(\lambda x_1 \ldots x_n.\mathsf{0}) &= \lambda x_1 \ldots x_n.\mathsf{0} \\
\omega(\lambda x_1 \ldots x_n.y M_1 \ldots M_m) &= \lambda x_1 \ldots x_n.y\, \omega(M_1) \ldots \omega(M_m), \textit{ for } y \textit{ variable} \\
\omega(\lambda x_1 \ldots x_n.\, \mathsf{case}_0\, MN) &= \lambda x_1 \ldots x_n.\, \mathsf{case}_0\, \omega(M)\, \omega(N), \\
&\qquad \textit{if } \omega(M) = \mathsf{case}_0 \ldots \textit{ or } \omega(M) = y \ldots \textit{ with } y \textit{ variable} \\
\omega(\lambda x_1 \ldots x_n.\langle M, N\rangle) &= \lambda x_1 \ldots x_n.\langle \omega(M), \omega(N)\rangle \\
\omega(\lambda x_1 \ldots x_n.\pi_1 M) &= \lambda x_1 \ldots x_n.\pi_1\, \omega(M), \textit{ if } \omega(M) = y \ldots \textit{ with } y \textit{ variable} \\
\omega(\lambda x_1 \ldots x_n.\pi_2 M) &= \lambda x_1 \ldots x_n.\pi_2\, \omega(M), \textit{ if } \omega(M) = y \ldots \textit{ with } y \textit{ variable} \\
\omega(M) &= \bot, \textit{ in all other cases}
\end{aligned}$$

$\omega(M)$ is a normal form prefix of M, it pushes $\bot$s upwards.
If M is a normal form, then $\omega(M) \cong M$.
If $M \to^* N$, then $\omega(M) \prec \omega(N)$.
If $M \prec N$, then $\omega(M) \prec \omega(N)$.
We define $\mathrm{nf}(M) = \omega($*the normal form of* $M)$.
For all $M \prec N$ it is $\mathrm{nf}(M) \prec \mathrm{nf}(N)$.

Proof. The first four propositions are clear, we prove here the last one; the proof is similar to the one of lemma 3.4.4.

Let M', N' be the normal forms of M, N.
As the reduction rules for $\to$ do not involve $\bot$, all the reductions $M \to^* M'$ can also be done in N. (If $A \prec B$ and $A \to A'$, then there is B' with $B \to B'$ and $A' \prec B'$.)
So there is N'' with $N \to^* N''$ and $M' \prec N''$.
By confluence of $\to$ it is $N'' \to^* N'$.
Then we get $\mathrm{nf}(M) = \omega(M') \prec \omega(N'') \prec \omega(N') = \mathrm{nf}(N)$. □

Theorem 3.8.8. *For every type σ without products, for every $a \in \mathcal{F}_0^\sigma$ there is a game term $A\colon \sigma$ with $a = [\![A]\!]$ such that for every $b \leq a$ there is $B \prec A$ with $b = [\![B]\!]$.*

Proof. By Laird's lemma 3.8.3 there is a number n and a definable retraction $\mathrm{Inj}\colon \sigma \trianglelefteq \tau\colon \mathrm{Proj}$, with τ some binary product form of $(\iota \to \iota)^n$.
Let A' be a term for a, $[\![A']\!] = a$.
Let $A'' = \mathrm{nf}(\mathrm{Proj}(\mathrm{Inj}\, A'))$. A'' does not contain any subterm of product type.
By the game term theorem 3.4.12 we get the desired game term $A = \mathrm{gt}_0^\sigma(\Psi_0^\sigma A'')$ with $A \cong A''$, so $[\![A]\!] = a$.

Let $C = \mathrm{nf}(\mathrm{Inj}\, A')$. $C = \langle C_1, \ldots, C_n\rangle$ in some binary pair form, where $C_i \cong \lambda x.\bot$ or $\lambda x.\mathsf{0}$ or $\lambda x.x$.
Let $b \leq a$. Then $[\![\mathrm{Inj}]\!]b \leq [\![\mathrm{Inj}]\!]a = [\![C]\!]$.
For every i, if $x \leq [\![C_i]\!]$ then $x = [\![C_i]\!]$ or $x = \bot$. Therefore there is $B' \prec C$ with $[\![B']\!] = [\![\mathrm{Inj}]\!]b$.
Let $B'' = \mathrm{nf}(\mathrm{Proj}\, B')$. It is $A'' = \mathrm{nf}(\mathrm{Proj}\, C)$. Therefore $B'' \prec A''$.
By the game term theorem 3.4.12 there is a game term $B = \mathrm{gt}_0^\sigma(\Psi_0^\sigma B'')$ with $B \cong B''$ and $B \prec A$.
We have $b = [\![\mathrm{Proj}]\!]([\![\mathrm{Inj}]\!]b) = [\![\mathrm{Proj}]\!][\![B']\!] = [\![B]\!]$. □

Remark: Please note that Laird's retractions are incredibly intelligent, because they must introduce in the term $A'' = \mathrm{nf}(\mathrm{Proj}(\mathrm{Inj}\, A'))$ some nestings of variables that were not present in A', to fulfill the proposition of the theorem.

It is a nice exercise (of three pages) to compute an example: Take $\sigma = (\iota \to \iota \to \iota) \to \iota$ and $A' = \lambda g.g\mathsf{0}\mathsf{0}\colon \sigma$. The trace of A' is

$$\mathcal{T}[\![A']\!] = \{\{\bot\bot\mapsto\mathsf{0}\}\mapsto\mathsf{0}, \{\mathsf{0}\bot\mapsto\mathsf{0}\}\mapsto\mathsf{0}, \{\bot\mathsf{0}\mapsto\mathsf{0}\}\mapsto\mathsf{0}, \{\mathsf{0}\mathsf{0}\mapsto\mathsf{0}\}\mapsto\mathsf{0}\}.$$

Going through Laird's proof of lemma 3.8.3, we get complicated terms $\mathrm{Inj}\colon \sigma \trianglelefteq \tau\colon \mathrm{Proj}$ with $\tau = (((\iota \to \iota) \times (\iota \to \iota)) \times (\iota \to \iota)) \times ((\iota \to \iota) \times \iota)$.
We compute the normal forms:

$$\mathrm{Inj}\, A' \to^* C = \langle\langle\langle\lambda x.x, \lambda x.x\rangle, \lambda x.x\rangle, \langle\lambda x.x, \underline{\mathsf{0}}\rangle\rangle$$
$$\mathrm{Proj}(\mathrm{Inj}\, A') \to^* A'' = \lambda g.\, \mathsf{case}_0[g(g\mathsf{0}(g\underline{\mathsf{0}}\mathsf{0}))\mathsf{0}][g\mathsf{0}(g\underline{\mathsf{0}}\mathsf{0})]$$

This term is much more expanded than needed.

If we replace the underlined $\underline{\mathsf{0}}$ in C by $\bot$, we get a term A'' with both underlined $\underline{\mathsf{0}}$ replaced by $\bot$. The trace of this new term A'' is $\{\{\bot\bot\mapsto\mathsf{0}\}\mapsto\mathsf{0}, \{\mathsf{0}\bot\mapsto\mathsf{0}\}\mapsto\mathsf{0}, \{\bot\mathsf{0}\mapsto\mathsf{0}\}\mapsto\mathsf{0}\}$.
Note that there was no syntactically lesser term than A' with this trace.

Remark: Another recommended exercise for the reader is to encode our first counterexample (to Berry's second conjecture) of subsection 3.5.1 in unary PCF. The booleans are encoded by the type $\beta = \iota \to \iota \to \iota$ as usual. The value $\mathsf{0}$ is represented by $\lambda xy.x$, $\mathsf{1}$ is represented by $\lambda xy.y$. There are three more inhabitants of β: $\perp$, $\lambda xy.\,\mathsf{case}_0\, xy$ and $\lambda xy.\mathsf{0}$. The example is now of type $(\beta \to \beta \to \beta) \to \beta$. The term D can be given an expanded form such that $A \prec B = C \prec D$. In D the top boolean $\lambda xy.\mathsf{0}$ is used (in one position) as the lub of $\lambda xy.x$ and $\lambda xy.y$.

3.9 Outlook

We have seen one trick to produce several examples which show that the stable order in PCF is not so regular as Berry had expected. These counter-examples have as necessary ingredients: at least two incompatible values and at least a second-order type with at least arity two of some functional parameter. To be precise, we still have to show that Berry's conjectures are valid in all second-order types with functional parameters of only arity one, see conjectures 3.6.7 and 3.5.2.

With the refutation of the chain conjecture in section 3.7 we have shown that there is no simple characterization of the stable order in terms of the syntactic order. In fact the counter-example shows that there is not only the syntactic order that causes the stable order, but that there are other syntactic relations needed with this property. Such another relation was identified as the relation of "strictification", and an improved chain conjecture 3.7.4 was tentatively proposed.

There should be some kind of full syntactic account of the stable order, at least for second-order types. For any type there should be syntactic conditions that are necessary for the relation $A \leq B$ of terms. These should at least prove the maximality conjecture 3.7.2: Every PCF-term without Y that is syntactically maximal is also stably maximal.

It would also be interesting to find syntactic characterizations of the stable order in extensions of PCF by sequential control operators, i.e. in (PCF+str), (PCF+$\mathsf{strict?}$), (PCF+H) and SPCF, see the remarks at the end of section 3.7.

In this paper we have treated the problem of the syntactic characterization of the stable order, but Berry originally had in mind the semantic characterization of the syntactic order. In the light of the results of this paper this seems to be a problem of similar difficulty. One should first seek necessary conditions for the syntactic order that are stronger than the stable order.

Chapter 4

From Sazonov's non-dcpo natural domains to closed directed-lub partial orders

First version May 2016, arxiv:1605.01886

Abstract: Normann proved that the domains of the game model of PCF (the domains of sequential functionals) need not be dcpos. Sazonov has defined natural domains for a theory of such incomplete domains.

This paper further develops that theory. It defines lub-rules that infer natural lubs from existing natural lubs, and lub-rule classes that describe axiom systems like that of natural domains. There is a canonical proper subcategory of the natural domains, the closed directed lub partial orders (cdlubpo), that corresponds to the complete lub-rule class of all valid lub-rules. Cdlubpos can be completed to restricted dcpos, which are dcpos that retain the data of the incomplete cdlubpo as a subset.

4.1 Introduction

Is the tacit agreement (perhaps a kind of "dogma") of the directed completeness of semantic domains falling? We might get this impression from the recent results of Dag Normann and Vladimir Sazonov on the game model of PCF. So we begin this introduction with a brief history of the models of PCF.

History

PCF is a simply typed λ-calculus on integers with higher-order recursion. The concept of PCF was formed by Dana Scott in 1969, see the historical document [Sco93]. It is used as a prototypical programming language to explore the relationship between operational and denotational semantics, see the seminal paper of Gordon Plotkin [Plo77]. The first model of Scott was made of directed complete partial orders, beginning with flat domains for integers and booleans and the full domain of continuous functionals for higher-order types. It was natural to demand directed completeness of the domains, so that we get a definition of continuity that leads to closure under function spaces and the existence of all conceivable fixpoints for the semantics of recursive functions. But the model contained ideal elements that were not realized in the language: These were finite elements, like the "parallel or" function, or infinite elements (as lubs of directed sets of finite elements). The presence of the finite unrealized elements (like "parallel or") already causes the model to be not fully abstract (i.e. the denotational semantics does not match the operational one), as was observed by Gordon Plotkin [Plo77].

The question of a fully abstract model arose, where a model was supposed to consist of directed complete partial orders, following the established "dogma". The first fully abstract cpo model was constructed by Robin Milner [Mil77] in 1977 from equivalence classes of finite combinator terms by an inverse limit of domains. Later the same model was constructed by Gérard Berry [Ber79] from equivalence classes of proper PCF-terms by an ideal completion.

So this model was built on syntactic terms of the language, which was not considered satisfactory, and the search for a purely mathematical fully abstract cpo model began. The widely accepted solution was the game semantics after 1990 [AJM00, HO00, Nic94]. In game semantics a term of PCF is modeled by a strategy of a game, i.e. by a process that performs a dialogue of questions and answers with the environment, the opponent. These strategies are still intensional; the fully abstract model is formed by a quotient, the extensional collapse. The strategies can be identified with PCF Böhm trees of a certain normal form, see [AC98, section 6.6].

It was an open problem whether the model of game domains is isomorphic to Milner's fully abstract cpo-model, i.e. whether its domains are cpos and so contain every element of the cpo-model. This problem was solved by Dag Normann [Nor06]: its domains are not cpos, i.e. there are directed sets that have no lub. The example given by Normann is in type 3 and rather sophisticated. Then Vladimir Sazonov made a first attempt to build a general theory for these non-cpo domains [Saz07, Saz09]. His main important insight was that functions are continuous only with respect to certain lubs of directed sets that he calls "natural lubs"; these are the hereditarily pointwise lubs in PCF. He defines an abstract structure of "natural domains" [Saz09] as a partial order with an operator that designates certain lubs of (general, not only directed) subsets as "natural lubs", fulfilling some axioms. He shows that the category of natural domains and functions that are continuous w.r.t. the directed natural lubs is a cartesian closed category (ccc). He defines naturally finite elements and natural algebraicity w.r.t. the natural directed lubs, and also shows that naturally algebraic, bounded complete natural domains form a ccc. In a recent paper Normann and Sazonov [NS12] show that in the game model of PCF, the sequential functionals, there is a Normann-example in a second-order type, there are directed lubs that are not natural, and there are naturally finite elements that are not finite in the classical sense. The main results of the last paper are also covered in the recent textbook of Longley and Normann: "Higher-order computability" [LN15], section 7.6.

Generally speaking, all these problems are due to a fundamental mismatch between the two worlds that semantics is relating: The world of syntax, of mechanism (in the form of programming languages and abstract machines), of intension on one side, and the more abstract world of domains and continuous functions, of extension on the other side. The problems are generally caused by restrictions on the syntactic side. So the restriction to sequentiality caused the full abstraction problem for PCF. Its solution, games, are constructions that stand somewhat in the middle between the two worlds.

Sazonov's natural domain theory accounts for another syntactic restriction: that limits of ascending chains of finite elements can only be formed for those chains that are ascending with the syntactic order, as a Boehm tree. (This is not bound to sequentiality, as Sazonov shows in [Saz07] a corresponding model for PCF with parallel conditional.) Natural domains model this incompleteness, but they miss an important property: the existence of fixpoints of endofunctions (on domains with $\perp$). (The domains of game semantics have this property, a mechanism always has the fixpoints by construction.) So a category of abstract incomplete domains (e.g. natural domains) must be understood as a (cartesian closed) "house" in which several more syntactic programming language models (with the fixpoint property) live together. We pose as an open problem to find categories of abstract incomplete domains with the existence of fixpoints.

We have seen that the game domain model, the model of sequential functionals, results from a syntactic restriction to Boehm trees. There are other syntactic restrictions conceivable, the most extreme being a restriction just to the terms of PCF itself. So in my paper [Mül12, section 3] on Berry's conjectures I have given the definition of a whole spectrum of fully abstract models of PCF ("f-models") as sets of ideals of equivalence classes of finite terms, such that application is defined and every PCF-term has a denotation. In this spectrum Milner's cpo model is the largest model, the pure term model is the least, and the game model is properly between the two.

Ideas of this paper

In this paper we further explore the abstract domain theory of incomplete domains. Our main objective is to find cartesian closed categories. We begin high above the natural domains with the most general conceivable structure, the directed-lub partial orders (dlubpo), partial orders with designated directed lubs, in the form of a relation $A \rightarrow_D a$, meaning that the directed subset A has the natural lub a. The only axiom they must obey is the singleton axiom $\{a\} \rightarrow_D a$ (Sazonov's axiom 3).

We define lub-rule classes to classify axiom systems with a form like that of natural domains. A lub-rule on a partial order D is a triple $(D, P \rightsquigarrow A)$ with P a set of subsets of $|D|$ that each have a lub and A a subset of $|D|$ that has a lub. This expresses the fact that in D we can infer from the existence of lubs of the elements of P the existence of the lub of A. This inference, this lub-rule, is "valid" if it is invariant under monotonic functions, i.e. every monotonic function $f\colon D \rightarrow E$ for some partial order E that respects the lubs of the elements of P respects also the lub of A. We explore axiom systems (of dlubpos) that can be described by classes of valid lub-rules.

We explain the philosophical significance of our translation from an axiom system S to a lub-rule class as a "partial extensionalization" of (the intension of) S, i.e. an extension that is between the pure syntax of S and the full extension, the class of dlubpos that fulfill S.

The validity of a lub-rule can be characterized by a closure operator cl_D on subsets of dlubpos D that infers from one element all elements below it, and from a natural subset the natural lub of it. This closure operator already appeared in the work of Bruno Courcelle and Jean-Claude Raoult on completions of ordered magmas [CR80].

A lub-rule class is complete if it encompasses all valid lub-rules. The dlubpos generated by a complete lub-rule class are called closed dlubpos (cdlubpo). They can be characterized by the closure operator cl_D, i.e. they fulfill the axiom S9 (closure):
If $A \subseteq |D|$ is directed with lub a and $a \in \mathrm{cl}_D A$, then $A \to_D a$.
They form a ccc. There are natural domains that are no cdlubpo.

In contrast to natural domains, cdlubpos have several characterizations as canonical structures. Cdlubpos are the dlubpos that are "realized" by "restricted dcpos" (rdcpo). So complete domains are coming in again, and the "dogma" of completeness could be "saved". The idea is to complete every cdlubpo with new improper elements (the "blind realizers") to a dcpo that retains the data of the incomplete cdlubpo as a subset, by an order embedding.

This idea of realization of a partial order by a dcpo goes back to Alex Simpson [Sim95]: In Simpson's approach every element of the partial order is realized by one or several realizers of the dcpo, while every realizer of the dcpo realizes exactly one element of the partial order. In our approach every element of the partial order is realized by exactly one realizer of the dcpo, while every realizer of the dcpo realizes at most one element of the partial order. (The two approaches could be combined, see the last section 4.10 Outlook.)

In a sequel paper we will work out the connection between the categories **Dlubpo** and **Rdcpo**. There is an adjunction between them, which establishes an adjoint equivalence between the sub-cccs **Cdlubpo** and **Crdcpo** (closed rdcpos). This connection also makes it possible to transfer the theory of cccs of algebraic dcpos to the realm of closed rdcpos resp. cdlubpos.

Outline of the paper

2. Preliminaries and notation:
   We repeat a concrete definition of cartesian closed categories and basic definitions of dcpo theory. Throughout the paper we will encounter closure and completion procedures that all follow the same abstract scheme. Here we extract this scheme as a fixpoint lemma on powersets and the definition of "rule systems" with their deductions.

3. Sazonov's definition of natural domains revisited:
   We repeat Sazonov's definition of natural domains, and give a simpler equivalent axiom system. Although we deviate from natural domains in the following sections, we will refer to the new axioms that are inferred here. We also show an example of a natural domain where natural lubs are needed for general subsets, not only directed ones.

4. Directed-lub partial orders (dlubpo):
   We define directed-lub partial orders as the most general structure we conceive. In this category the exponents, if they exist, have a definition that is generally different from the pointwise exponents of natural domains. We give sufficient conditions on subcategories of **Dlubpo** to have terminal, products and exponents like the normal ones. The main theorems are that **Dlubpo** is no ccc, but that the full subcategory of dlubpos that fulfill Sazonov's axiom S5 is already a ccc (with exponents of the general form).

   This section contains the basic definitions of dlubpos, but the results are mainly unrelated to the rest of the paper, only few minor ones are used in the following sections. For a first reading, I recommend to read only the basic definitions 4.1 to 4.4 and skip the rest of the section.

5. Lub-rule classes and closed dlubpos:
   We introduce the (valid) lub-rules, lub-rule systems and (complete, invariant) lub-rule classes we described above. We define closed dlubpos (cdlubpo) fulfilling the closure axiom S9. These are the dlubpos fulfilling all valid (directed) lub-rules. Every cdlubpo is a natural domain. The dlubpos generated by an invariant lub-rule class form a full reflective subcategory of **Dlubpo**.

6. The ccc of S10-dlubpos:
   We introduce a new axiom S10 for dlubpos. The category of these dlubpos is the largest full sub-ccc of **Dlubpo** that is generated by an invariant lub-rule class and has the pointwise exponents. Every natural domain and every cdlubpo is in this category.

7. Example of a natural domain that is no cdlubpo:
   We first show by a simple finite example that the lub-rule class corresponding to the axioms of natural domains is not complete. Then we give an example of a natural domain that is no cdlubpo.

8. Algebraic dlubpos:
   We show that every algebraic dlubpo that fulfills axiom S6 (cofinality) is a cdlubpo. This means that algebraic natural domains and algebraic cdlubpos are the same.

9. Restricted partial orders and restricted dcpos:
   We give a sufficient condition based on subcategory morphisms for a dlubpo to be a cdlubpo. We define restricted partial orders (rpo) and restricted dcpos (rdcpo) and show that cdlubpos are exactly the dlubpos realized by rpos resp. rdcpos.

10. Outlook

4.2 Preliminaries and notation

Notation: We mostly write function application without brackets, if possible. Function application associates to the left.
If f is a function that is defined on the elements of a set A, then we write $f^+A = \{\, fa \mid a \in A \,\}$. If A is a set of sets of elements a, and f is defined on all a, then we write $f^{++}A = (f^+)^+A = \{\, f^+B \mid B \in A \,\}$.
$\mathcal{P}(S)$ is the powerset of the set S.

We give some basic definitions of category theory and domain theory, and then a fixpoint lemma on powersets with a definition of "rule systems".

4.2.1 Category theory

The main property of our categories of domains that interests us here is cartesian closedness. We adopt a concrete definition from [AC98, def. 4.2.5]:

Definition 4.2.1 (cartesian closed category)**.** Let **K** be a category.

(1) A *terminal object* in $\mathbf{K}$ is a $\top \in \mathbf{K}$ such that:
$\forall C \in \mathbf{K}.\ \exists! h\colon C \to \top$.

(2) A *product* of $D, E \in \mathbf{K}$ is an object $D \times E \in \mathbf{K}$ with *projections* $\pi_1\colon D \times E \to D$ and $\pi_2\colon D \times E \to E$ such that:
$\forall C \in \mathbf{K}.\ \forall f\colon C \to D.\ \forall g\colon C \to E.\ \exists! h\colon C \to D \times E.\ (\pi_1 \circ h = f \text{ and } \pi_2 \circ h = g)$.
The morphism h is denoted by $\langle f, g\rangle$, $\langle \_, \_\rangle$ is called the *pairing operator*.
For $f\colon D \to D'$, $g\colon E \to E'$: $f \times g\colon D \times E \to D' \times E'$ is defined as $f \times g = \langle f \circ \pi_1, g \circ \pi_2\rangle$.

(3) Let $\mathbf{K}$ have all (binary) products.
An *exponent* of $D, E \in \mathbf{K}$ is an object $D \Rightarrow E \in \mathbf{K}$ with an *evaluation morphism* $\mathrm{eval}\colon (D \Rightarrow E) \times D \to E$ such that:
$\forall C \in \mathbf{K}.\ \forall f\colon C \times D \to E.\ \exists! h\colon C \to (D \Rightarrow E).\ (\mathrm{eval} \circ (h \times \mathrm{id}) = f)$.
The morphism h is denoted by $\mathrm{curry}(f)$, curry is called the *currying operator*.

A *cartesian closed category (ccc)* is a category that has a terminal object and all (binary) products and all exponents.

We will also see adjunctions and reflective subcategories, we will mainly use the notations of [Lan98].

4.2.2 Domain theory

We take our definitions from the excellent textbook [AC98].

A structure $D = (|D|, \sqsubseteq_D)$ is a *partial order (po)* if $|D|$ is a set and $\sqsubseteq_D$ is a binary relation on $|D|$ that is reflexive, transitive and antisymmetric.
$\sqsubseteq_D$ is simply written $\sqsubseteq$ if D is clear from the context, and this abbreviation also applies to other structures and indices.
A non-empty subset $A \subseteq |D|$ is *directed* if for all $a, b \in A$ there is some $c \in A$ with $a \sqsubseteq c$ and $b \sqsubseteq c$.
For a partial order D (or some extension of a partial order with more data) $\bar{\mathcal{P}}(D)$ is the set of subsets of $|D|$ that have a least upper bound (lub) in D; and $\mathcal{P}^{\uparrow}(D)$ is the set of directed subsets of $|D|$ that have a lub in D.
For subsets $A, B \subseteq |D|$ it is written $A \sqsubseteq B$ if for all $a \in A$ there is $b \in B$ with $a \sqsubseteq b$; and for $A \subseteq |D|$, $b \in |D|$: $A \sqsubseteq b$ if for all $a \in A$ it is $a \sqsubseteq b$.

D is a *directed complete partial order (dcpo)* if every directed subset of $A \subseteq |D|$ has a least upper bound (lub), denoted $\bigsqcup A$. If furthermore D has a least element (written $\bot$), then it is called a *complete partial order (cpo)*.
For D, E dcpos, a function $f\colon |D| \to |E|$ is *continuous* if it preserves directed lubs: for all directed $A \subseteq |D|$ it is $f(\bigsqcup A) = \bigsqcup (f^{+} A)$. (Then it is also monotonic, i.e. $a \sqsubseteq_D b \implies fa \sqsubseteq_E fb$.)

Dcpo is the category of dcpos and continuous functions. The continuity of f is specified by $f\colon D \to E$, which always means that f is a morphism in the category of D and E.
Dcpo is a ccc: The terminal object is the one point dcpo. For dcpos D, E the product is $D \times E = (|D| \times |E|, \sqsubseteq_{D \times E})$ where $\sqsubseteq_{D\times E}$ is pointwise. The exponent is $D \Rightarrow E = (|D \Rightarrow E|, \sqsubseteq_{D \Rightarrow E})$ where $|D \Rightarrow E|$ is the set of continuous functions, and $\sqsubseteq_{D \Rightarrow E}$ is the pointwise order. Lubs of directed sets of continuous functions are taken pointwise.

4.2.3 Fixpoint lemma and rule systems

We will encounter many closure and completion procedures that all follow a certain abstract scheme that is here extracted. These results are certainly all well known, but I did not find a reference with proofs in the literature that matches. We will use ordinals, a short introduction to them can be found in [Joh87].

Definition 4.2.2. Let S be a set.
A function $f\colon \mathcal{P}(S) \to \mathcal{P}(S)$ is a *preclosure* if it is *increasing*: for all $A \subseteq S$ it is $fA \supseteq A$, and it is *monotonic*: for $A, B \subseteq S$, $A \subseteq B$ implies $fA \subseteq fB$.
An $A \subseteq S$ with $fA = A$ is called *closed under* f.
For $A \subseteq S$ we define $f^0 A = A$, $f^{\alpha+1} A = f(f^\alpha A)$ for all ordinals α, $f^\beta A = \bigcup_{\alpha<\beta} f^\alpha A$ for all limit ordinals β.

Lemma 4.2.3 (fixpoint lemma)**.**
Let S be a set, $f\colon \mathcal{P}(S) \to \mathcal{P}(S)$ a preclosure and $A \subseteq S$.
Let B be the intersection of all $A' \supseteq A$ with $fA' = A'$.
Then B is the least set with $A \subseteq B \subseteq S$ and $fB = B$. B is called the closure *of A under f.*
The map from $A \subseteq S$ to the closure of A under f is called the closure operator *for f.*
It is $B = \bigcup_{\alpha \text{ ordinal}} f^\alpha A$. Furthermore $B = f^\gamma A$ for some ordinal γ.

Proof. It is $fB \supseteq B$ because f is increasing.
It is $fB \subseteq fA' \subseteq A'$ for all $A' \supseteq A$ with $fA' = A'$, therefore $fB \subseteq B$.
Let $C = \bigcup_{\alpha \text{ ordinal}} f^\alpha A$. We show $C \subseteq B$, i.e. $f^\alpha A \subseteq B$ for all ordinals α, by induction on α:
It is $f^0 A = A \subseteq B$. $f^{\alpha+1} A = f(f^\alpha A) \subseteq fB = B$.
$f^\beta A = \bigcup_{\alpha<\beta} f^\alpha A \subseteq B$ for all limit ordinals β.
To prove $B \subseteq C$ we show that $fC = C$:
By Hartogs' lemma [Har15][Joh87, lemma 7.1], there is an ordinal γ that cannot be mapped injectively into S. Assume $f(f^\gamma A) \neq f^\gamma A$.
Then for every element α of γ, i.e. for every ordinal $\alpha < \gamma$, we have $f(f^\alpha A) \neq f^\alpha A$.
If we map each $\alpha < \gamma$ to some new element of $f(f^\alpha A)$ that is not in $f^\alpha A$, we get an injective map from γ into S, contradiction. So it must be $f(f^\gamma A) = f^\gamma A = C$.
As B is the least set with $fB = B$ and $A \subseteq B$, we get $B \subseteq C$. $\square$

In all our applications of the fixpoint lemma, the preclosure is generated by a rule system on the underlying set S:

Definition 4.2.4 (rule system)**.** A *rule system* on a set S is a relation $\rightsquigarrow\ \subseteq \mathcal{P}(S) \times S$.
The elements of $\rightsquigarrow$ are called *rules*. The *preclosure of* this rule system is
$f\colon \mathcal{P}(S) \to \mathcal{P}(S)$ defined by $fA = A \cup \{\, a \in S \mid \exists B \subseteq A.\ B \rightsquigarrow a \,\}$.
The closure operator for f is called the *closure operator* of the rule system, corresponding to *closures under* the rule system.
A *deduction of $a \in S$ from $A \subseteq S$* in this rule system is some $(N, r, \mathit{lab}, \mathit{pre})$ where N is the set of *nodes*, $r \in N$ is the *root*, $\mathit{lab}\colon N \to S$ is the *labelling function*,
$\mathit{pre}\colon \{\, n \in N \mid \mathit{lab}\, n \notin A \,\} \to \mathcal{P}(N)$ is the *predecessor function*,
such that $\mathit{lab}\, r = a$,
for all $n \in N$ with $\mathit{lab}\, n \notin A$: $\mathit{lab}^+(\mathit{pre}\, n) \rightsquigarrow \mathit{lab}\, n$,

(we say that $\mathit{lab}\, n$ is *deduced* from $\mathit{lab}^+(\mathit{pre}\, n)$),
for all $n \in N$ there is a unique finite path of length $i \geq 1$:
$r = n_1,\ n_2 \in \mathit{pre}\, n_1,\ n_3 \in \mathit{pre}\, n_2, \ldots,\ n_i = n$,
and for all $n \in N$ there is no infinite path $n = n_1,\ n_2 \in \mathit{pre}\, n_1,\ n_3 \in \mathit{pre}\, n_2, \ldots$. (So such paths end in some $n_i \in N$ with $\mathit{lab}\, n_i \in A$.)

If there is a deduction of a from A, then the pair (A, a) is called a *derived rule* of the rule system.

So the deduction is a tree with root r labelled a and leaves labelled by elements of A, such that each non-leaf node is deduced by a rule from its predecessors. It has a well-founded structure and we can prove properties of its nodes by induction on this structure.

Lemma 4.2.5. *Let $\rightsquigarrow$ be a rule system on a set S, and f its preclosure. Let $A \subseteq S$ and $a \in S$.*
There is a deduction of a from A in $\rightsquigarrow$ $\iff$ a is in the closure of A under f.

Proof. $\Longrightarrow$: Let $(N, r, \mathit{lab}, \mathit{pre})$ be the deduction of a from A in $\rightsquigarrow$. We prove by induction on the deduction that for every node $n \in N$, $\mathit{lab}\, n$ is in the closure B of A under f. This is clear as $A \subseteq B$ and $fB = B$.

$\Longleftarrow$: We show by induction on α that for every ordinal α and every $a \in f^\alpha A$, there is a deduction of a from A. It is clear for $\alpha = 0$ and α a limit ordinal. For $\alpha = \beta + 1$, a is already in $f^\beta A$ or else there is some $C \subseteq f^\beta A$ with $C \rightsquigarrow a$. In the second case, we build the deduction with root r and $\mathit{lab}\, r = a$ and connect r to all the deductions of the elements of C as predecessors. □

4.3 Sazonov's definition of natural domains revisited

In this section we repeat Sazonov's definition of natural domains, and give a simpler equivalent axiom system. We also give an example that shows that natural lubs are needed for general subsets, not only directed ones.

Sazonov's natural domains [Saz09] are a general extrapolation of the domains of the game model of PCF. They are just partial orders with a designation of some of the (general) subsets with lubs as "natural lubs". The natural lubs of directed sets must be respected by the naturally continuous functions. The natural lubs have to obey certain closure conditions which are taylored to ensure cartesian closedness of the resulting category under a certain function space.

We translate the definition into a new notation: instead of the partial natural-lub-operator $\biguplus$ we use a relation symbol $\to$ for natural convergence.

Definition 4.3.1 (Sazonov, def. 2.1 of [Saz09])**.**
A structure $D = (|D|, \sqsubseteq_D, \to_D)$ is a *natural domain* if
$|D|$ is a set of *elements*,
$\sqsubseteq_D$ is a partial order on $|D|$,
$\to_D \subseteq \mathcal{P}^\uparrow(|D|) \times |D|$ is the *(natural) convergence relation* on directed sets,
($X \to_D x$ means: X has the *natural lub* x, such an X is called *natural*)
such that $\to_D$ can be extended by pairs (X, x) with $X \subseteq |D|$ *not* directed, $x \in |D|$, to a

relation on all subsets, which we also denote $\to_D$, and the following axioms are fulfilled for the extended relation (where axiom Sx corresponds to $\biguplus$x in Sazonov):
(S1) If $X \to x$, then $\bigsqcup X = x$.
(S2) If $X \subseteq Y \subseteq |D|$, $X \to x$ and $Y \sqsubseteq x$, then $Y \to x$.
(S3 singleton) For all $x \in |D|$: $\{x\} \to x$.
(S4(1)) If $\{y_{ij}\}_{i\in I, j\in J}$ is a non-empty two-parametric family of elements in $|D|$,
and for every $i \in I$ it is $\{y_{ij}\}_{j\in J} \to d_i$,
then $\{d_i\}_{i\in I} \to d \iff \{y_{ij}\}_{i\in I, j\in J} \to d$.
(S4(2)) If I is an index set with a directed partial order, and $\{y_{ij}\}_{i,j\in I}$ is a two-parametric family of elements in $|D|$ that is monotonic in each parameter i and j,
then $\{y_{ij}\}_{i,j\in I} \to y \iff \{y_{ii}\}_{i\in I} \to y$.

Note that we deviate from Sazonov's definition in that we take only the part of the convergence relation on *directed* subsets as the data of the structure, and leave the existence of the extended relation as a side condition. This is justified by the morphisms of the category, the (naturally) continuous functions, which must be continuous only w.r.t. the *directed* natural lubs. We get our natural domains by identifying Sazonov's natural domains which have the same order and the same *directed* natural lubs. The result is an equivalent category. We make this change because we define all other related structures of incomplete domains by convergence of directed sets only.

The specification of the lub in the conclusions of the axioms is redundant, it suffices to specify that the set in question is natural, e.g. in axiom (S2): Y is natural instead of $Y \to x$.

We do not count S1 as a proper axiom, instead we treat it as a condition for the relation $\to$ that is always presupposed in the sequel. The other proper axioms have a different character, they deduce the naturalness of lubs from existing natural lubs.

We have translated the axiom system into a form that clearly discerns its different parts. It needs a clean-up and simplification. First, we can further split axiom S4(1) into the two axioms S4(1$\Rightarrow$) and S4(1$\Leftarrow$), where the logical equivalence $\iff$ is replaced by the indicated direction.

Proposition 4.3.2. *In the presence of axiom S3 (singleton): axioms S2 and S4(1$\Leftarrow$) together are equivalent to the axiom:*
(S6 cofinality) *If $X, Y \subseteq |D|$, $X \to x$ and $X \sqsubseteq Y \sqsubseteq x$, then $Y \to x$.*

Proof. S2 and S4(1$\Leftarrow$) $\implies$ S6:
Let $X \to x$ and $X \sqsubseteq Y \sqsubseteq x$.
Let $I = \{\, (a,b) \mid a \in X, b \in Y \text{ and } a \sqsubseteq b \,\}$ and $J = \{1,2\}$.
For $(a,b) \in I$ define $y_{(a,b)1} = a$ and $y_{(a,b)2} = b$ and $d_{(a,b)} = b$.
For every $(a,b) \in I$ it is $\{y_{(a,b)j}\}_{j\in J} \to d_{(a,b)}$, by axiom S3 and S2.
It is $\{y_{ij}\}_{i\in I, j\in J} \to x$ by axiom S2, as $X \subseteq \{y_{ij}\}_{i\in I, j\in J}$ by $X \sqsubseteq Y$.
We apply axiom S4(1$\Leftarrow$) and deduce $\{d_{(a,b)}\}_{(a,b)\in I} \to x$.
This set is a subset of Y, therefore by axiom S2 we get $Y \to x$.
The reverse direction is immediate. □

Proposition 4.3.3. *Axiom S4(2) is redundant, it follows from axiom S6 (cofinality).* □

Sazonov proposes an "optional clause, which might be postulated as well", as a replacement for axiom S4(2):

(S5) If $X \subseteq Y \subseteq |D|$, $Y \to y$ and $Y \sqsubseteq X$, then $X \to y$.

This does not lead to a stronger axiom system:

Proposition 4.3.4. *Axioms S2 and S5 together are equivalent to axiom S6 (cofinality).*

Proof. S2 and S5 $\Longrightarrow$ S6:
It is $X \subseteq X \cup Y \subseteq |D|$, $X \to x$, $X \cup Y \sqsubseteq x$, so $X \cup Y \to x$ by axiom S2.
It is $Y \subseteq X \cup Y \subseteq |D|$, $X \cup Y \to x$, $X \cup Y \sqsubseteq Y$, so $Y \to x$ by axiom S5.
The reverse direction is immediate. □

Next, we want to give axiom S4(1⇒) in a form with sets instead of two-parametric families, to get rid of the index sets of unlimited size:

Proposition 4.3.5. *In the presence of axioms S3 (singleton) and S6 (cofinality), axiom S4(1⇒) is equivalent to the axiom:*
(S7 transitivity) *If* $\overline{X} \subseteq \mathcal{P}(|D|)$ *with* $X \to d_X$ *for all* $X \in \overline{X}$,
then $\{\, d_X \mid X \in \overline{X}\,\} \to d \Longrightarrow \bigcup \overline{X} \to d$.

Proof. S4(1⇒) follows immediately from S7.
For the reverse direction, we prove S7 from S4(1⇒).
First, we have to prove the two special cases (1) $\overline{X} = \emptyset$ and (2) $\overline{X} = \{\emptyset\}$.
(1) The conclusion of S7 reads $\emptyset \to d \Longrightarrow \emptyset \to d$.
(2) It is $\emptyset \to d_\emptyset$ and the conclusion reads $\{d_\emptyset\} \to d \Longrightarrow \emptyset \to d$.
Now assume we do not have case (1) or (2).
Take $I = \overline{X}$ and $J = \bigcup \overline{X}$.
Both I and J are not empty, as is required by S4(1⇒). We define y_{ij}:
If $i \in I = \overline{X}$ is not empty, then for all $j \in J$ take $y_{ij} = j$ if $j \in i$ and else some arbitrary element of i.
If $i \in I$ is empty, then for all $j \in J$ take $y_{ij} = \bigsqcup \emptyset$ (the least element), this exists as $i = \emptyset \to d_\emptyset = \bigsqcup \emptyset$.
Then for all nonempty $i \in I$ it is $i \subseteq \{y_{ij}\}_{j \in J}$ and $\{y_{ij}\}_{j \in J} \sqsubseteq d_i$,
so by axiom S6 it is $\{y_{ij}\}_{j \in J} \to d_i$.
If $i \in I$ is empty, then it is $\{y_{ij}\}_{j \in J} = \{\bigsqcup \emptyset\} \to \bigsqcup \emptyset$ by axiom S3.
So in every case it is $\{y_{ij}\}_{j \in J} \to d_i$, and we can draw the conclusion of S4(1⇒).
Furthermore, if not $\emptyset \in I$, then $\{y_{ij}\}_{i \in I, j \in J} = \bigcup \overline{X}$, and the conclusion of S7 follows.
If $\emptyset \in I$, then $\{y_{ij}\}_{i \in I, j \in J} = \bigcup \overline{X} \cup \{\bigsqcup \emptyset\}$.
From $\{y_{ij}\}_{i \in I, j \in J} \to d$ follows $\bigcup \overline{X} \to d$ by axiom S6. This is the conclusion of S7. □

In summary, we get a nice simpler axiom system as a base for the rest of the paper:

Proposition 4.3.6. *The axioms of the Sazonov natural domain are equivalent to S1, S3 (singleton), S6 (cofinality) and S7 (transitivity).* □

We can further combine axioms S6 and S7, for this we need a new definition:

Definition 4.3.7. Let D be the structure in the beginning of the definition of the Sazonov natural domain, and $A, B \subseteq |D|$.
A is *under* B, $A \underset{\leftarrow}{\sqsubseteq}_D B$, if for all $a \in A$:
there is $b \in B$ with $a \sqsubseteq_D b$ or there is $B' \subseteq B$ with $B' \to_D a$.

Proposition 4.3.8. *In the presence of axiom S3 (singleton): axioms S6 (cofinality) and S7 (transitivity) together are equivalent to the axiom:*
(S8 under) *If $A, B \subseteq |D|$, $A \to a$ and $A \underset{\leftarrow}{\sqsubseteq} B \sqsubseteq a$, then $B \to a$.*

Proof. S6 and S7 $\Longrightarrow$ S8:
Let $\varphi\colon A \to \mathcal{P}(|D|)$ be defined as:
$\varphi x = \{x\}$ if there is $y \in B$ with $x \sqsubseteq y$,
otherwise take a choice of some $B' \subseteq B$ with $B' \to x$ and define $\varphi x = B'$.
Let $C = \bigcup(\varphi^+ A)$. By the singleton axiom S3 and transitivity axiom S7 it is $C \to a$.
It is $C \sqsubseteq B \sqsubseteq a$, therefore $B \to a$ by the cofinality axiom S6.
The reverse direction is immediate. □

So the axioms of the Sazonov natural domain are equivalent to S1, S3 and S8.
The proper axioms share a common form: They are rules (in the sense of rule systems, definition 4.2.4) that deduce from the existence of the lubs of some sets the existence of the lub of another set, all in some configuration of a partial order. One can get the impression that the axioms are "complete" in this respect, that all "necessary" deductions can be made.

In section 4.5 we set up the general frame of lub-rule systems where we can define what this completeness means. It will turn out in section 4.7 that the axioms are in fact incomplete. So we will have to replace them by a stronger axiom in order to get a class of domains that can be realized by dcpos.

We now come to the morphisms of the category of natural domains.

Definition 4.3.9 (Sazonov def. 2.3(a) in [Saz09])**.**
Let D, E be natural domains.
A function $f\colon |D| \to |E|$ is *(naturally) continuous* if it is monotonic w.r.t. $\sqsubseteq_D$ and $\sqsubseteq_E$, and if X is directed and $X \to_D x$, then $f^+X \to_E fx$. (Monotonicity is here redundant.)
If f is naturally continuous, we write $f\colon D \to E$.

In [Saz09] it is shown that the natural domains with naturally continuous functions form a ccc.
The terminal object is the one point domain $\{\{\bot\}, \bot \sqsubseteq \bot, \{\bot\} \to \bot\}$.
The product $D \times E$ has $|D \times E| = |D| \times |E|$, $\sqsubseteq_{D\times E}$ is component-wise, and $A \to_{D\times E} a$ iff $\pi_1^+ A \to_D \pi_1 a$ and $\pi_2^+ A \to_E \pi_2 a$.
The exponent $D \Rightarrow E$ has $|D \Rightarrow E|$ the set of naturally continuous functions, $\sqsubseteq_{D\Rightarrow E}$ is pointwise, and $F \to_{D\Rightarrow E} f$ iff $Fx \to_E fx$ for all $x \in |D|$.

As we remarked above, we take only the part of the convergence relation on directed sets as the data of a natural domain, and leave the existence of the extended relation as a side condition. The following example shows a case where natural lubs of general subsets are needed as intermediary steps in the deduction of the natural lubs of directed subsets, so that the extended relation is really needed.

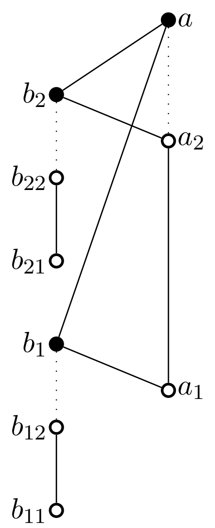


Figure 4.1: Natural domain E (partial)

Figure 4.1 shows only part of the natural domain E. It first has the elements a, a_i and b_i for $i \geq 1$. The order is $a_i \sqsubseteq a_j$ iff $i \leq j$, $a_i \sqsubseteq a$, $a_i \sqsubseteq b_i$ and $b_i \sqsubseteq a$ for $i \geq 1$. And $\{a_i\}_{i\geq 1} \to a$ is natural. It has further the elements of $B = \{b_{ij}\}_{i,j\geq 1}$ with $(b_{ij} \sqsubseteq b_{ik}$ iff $j \leq k)$ and $b_{ij} \sqsubseteq b_i$ for all $i \geq 1$. It is stipulated that the $\{b_{ij}\}_{j\geq 1} \to b_i$ are natural. Moreover, we erect an "artificial" directed set over the elements of B, whose elements are not depicted in the diagram: every finite subset $X \subseteq B$ is an element of E, the set of these elements is called $\overline{B}$. For all i, j it is defined $b_{ij} \sqsubseteq \{b_{ij}\}$. For $X, Y \in \overline{B}$ it is defined $X \sqsubseteq_E a$ and $(X \sqsubseteq_E Y$ iff $X \subseteq Y)$. E is defined to be the natural domain that is completed by all conclusions of the partial order axioms and the axioms of naturality. $\overline{B}$ is a directed subset of $|E|$ with the lub a.

From $\{a_i\}_{i\geq 1} \to a$ we deduce $\{b_i\}_{i\geq 1} \to a$ by axiom S6, then $B \to a$ by axiom S7, then $\overline{B} \to a$ by axiom S6.

Here $\{b_i\}_{i\geq 1} \to a$ and $B \to a$ are intermediary steps that deduce the naturality of non-directed subsets. There is no deduction of the directed natural lub $\overline{B} \to a$ from the given directed natural lubs that does not use a non-directed intermediary step.

4.4 Directed-lub partial orders (dlubpo)

Here we explore categories larger than the category of natural domains. We want to find a "minimal part" of the natural domain axioms that already achieves cartesian closedness. Less axioms means more chaos. We always presuppose axiom S1, which we do not count as proper axiom. When there is no other axiom valid, we get the normal terminal object $\top$, but we do not get all constant element morphisms $\lambda x.d\colon \top \to D$, $d \in D$ some object. So in order to constrain chaos, we start with the "most obvious" axiom, the singleton axiom S3, and get the category **Dlubpo** of directed-lub partial orders. These are partial orders with a designation of some lubs of *directed* subsets as natural. This category has the expected terminal object, products and the constant element morphisms. We prove that the exponents do not always exist; and if they exist, they are generally defined differently from those in the category of natural domains. (But the natural domains, with their *directed* natural subsets as data, are a full sub-ccc of **Dlubpo** such that their exponents coincide with the general exponents of **Dlubpo**.) We also give sufficient conditions on subcategories of **Dlubpo** to have products and exponents like the normal ones, in analogy to lemma 5 of [Smy83]. We show that the dlubpos fulfilling axiom S5 (for directed sets) already form a ccc.

For a first understanding of the following sections only definitions 4.1 to 4.4 are necessary. The rest of this section may be skipped, although definition 4.4.5, propositions 4.4.6 and 4.4.9(1) will be used in the following sections.

Definition 4.4.1. A structure $D = (|D|, \sqsubseteq_D, \rightarrow_D)$ is a *directed-lub partial order (dlubpo)* if
$|D|$ is a set of *elements* of D,
$\sqsubseteq_D$ is a partial order on $|D|$, lubs are denoted by $\bigsqcup_D$,
$\rightarrow_D \subseteq \mathcal{P}^{\uparrow}(|D|, \sqsubseteq_D) \times |D|$ is the *(natural) convergence relation*,
if $A \rightarrow_D a$, then A is a directed subset of $|D|$, and $a \in |D|$ is its lub,
(a is called the *natural lub* of A, A is called a *natural* (directed) subset),
$\rightarrow_D$ fulfills the *singleton axiom* S3: For every $d \in |D|$ it is $\{d\} \rightarrow_D d$.

Definition 4.4.2. Let D, E be dlubpos.
A function $f\colon |D| \rightarrow |E|$ is *continuous* if it is monotonic w.r.t. $\sqsubseteq_D$ and $\sqsubseteq_E$ and if $X \rightarrow_D x$, then $f^+X \rightarrow_E fx$. If f is continuous, we write $f\colon D \rightarrow E$.
The dlubpos with continuous functions form a category **Dlubpo**, with normal function composition and identity functions.

Note that Sazonov calls such a function "naturally continuous", which we abbreviate to "continuous", as it is clear from the context that D, E are dlubpos.

Definition 4.4.3. The dlubpo $\top = (\{\bot\}, \bot \sqsubseteq \bot, \{\bot\} \rightarrow \bot)$ is called the *terminal dlubpo.*
$\top$ is the categorical terminal object in **Dlubpo**. It need not be so in subcategories. If the subcategory has $\top$ as terminal object, we say that it has the normal terminal.

Definition 4.4.4. Let D, E be dlubpos.
We define the *product dlubpo* $D \times E = (|D \times E|, \sqsubseteq_{D\times E}, \rightarrow_{D\times E})$ as follows:
$|D \times E| = |D| \times |E|$,
$(a, b) \sqsubseteq_{D\times E} (a', b')$ iff $a \sqsubseteq_D a'$ and $b \sqsubseteq_E b'$,
if $A \subseteq |D \times E|$ is directed and $(a, b) \in |D \times E|$, then $A \rightarrow_{D\times E} (a, b)$ iff $\pi_1^+ A \rightarrow_D a$ and $\pi_2^+ A \rightarrow_E b$, where we have the normal *projections* $\pi_1\colon |D \times E| \rightarrow |D|$ and $\pi_2\colon |D \times E| \rightarrow |E|$.
(It is clear that (a, b) is the lub of A.)
We also have the normal pairing functions:
For any dlubpo C, $f\colon C \rightarrow D$ and $g\colon C \rightarrow E$ it is $\langle f, g\rangle\colon C \rightarrow D \times E$, $\langle f, g\rangle x = (fx, gx)$.
All these functions are continuous, and $D \times E$ with π_1, π_2 is the categorical product in **Dlubpo**. It need not be so in subcategories. If the subcategory has the $D\times E$ as categorical products, we say it has the normal products.

Definition 4.4.5. Let **K** be a subcategory of **Dlubpo**, and $D, E \in \mathbf{K}$.
We define (polymorphically) the function eval: $\{\, f \mid f\colon D \rightarrow E \text{ in } \mathbf{K}\,\} \times |D| \rightarrow |E|$ by $\mathrm{eval}(f, d) = fd$.
We define the *general function dlubpo* $D \Rightarrow^*_{\mathbf{K}} E$ (relative to **K**) by:
$|D \Rightarrow^*_{\mathbf{K}} E| = \{\, f \mid f\colon D \rightarrow E \text{ in } \mathbf{K}\,\}$,

$f \sqsubseteq_{D \Rightarrow^*_{\mathbf{K}} E} g$ iff for all $x \in |D|$ it is $fx \sqsubseteq_E gx$,
if $F \subseteq |D \Rightarrow^*_{\mathbf{K}} E|$ is directed and $f \in |D \Rightarrow^*_{\mathbf{K}} E|$, then $F \to_{D \Rightarrow^*_{\mathbf{K}} E} f$ iff for all $A \subseteq |D \Rightarrow^*_{\mathbf{K}} E| \times |D|$ that are directed w.r.t. the component-wise order with $\pi_1^+ A = F$ and $\pi_2^+ A \to d$ it is $\mathrm{eval}^+ A \to_E fd$.
For $\mathbf{K} = \mathbf{Dlubpo}$ we define $(D \Rightarrow^* E) = (D \Rightarrow^*_{\mathbf{Dlubpo}} E)$.
We also have the curried functions: For any dlubpo C, $f \colon C \times D \to E$ in **Dlubpo** we define $\mathrm{curry}(f) \colon |C| \to |D \Rightarrow^* E|$ by $\mathrm{curry}(f)c = \lambda d.f(c,d)$.

We have to prove that in the definition $\bigsqcup F = f$: for all $d \in |D|$, take $A = F \times \{d\}$, then $\mathrm{eval}^+ A \to fd$, so f is the pointwise lub of F.
$D \Rightarrow^*_{\mathbf{K}} E$ fulfills the singleton axiom: for $f \in |D \Rightarrow^*_{\mathbf{K}} E|$ it is $\{f\} \to f$ because f is continuous.

The function eval is continuous. The trick of the definition is that not only the continuity in constant second arguments (as in the case of natural domains), but the whole continuity of eval is coded in the definition of $\to_{D \Rightarrow^*_{\mathbf{K}} E}$. We will see that if **Dlubpo** contains an exponent of D and E, then it is isomorphic to $D \Rightarrow^* E$.

Proposition 4.4.6. $\mathrm{eval} \colon (D \Rightarrow^*_{\mathbf{K}} E) \times D \to E$ *is continuous.*
The results of $\mathrm{curry}(f)$, $\mathrm{curry}(f)c$ *for all* $c \in |C|$, *are continuous.*
$\mathrm{curry}(f)$ *is monotonic, but need not be continuous. (We will see a counter-example below.)*

$\square$

We will now give sufficient conditions for subcategories of **Dlubpo** to contain terminal objects, products and exponents that coincide partially with the normal terminals, products and the general function dlubpos. These conditions are extracted from the corresponding results of [Smy83, lemma 5] for full subcategories of the category of ω-algebraic cpos. Those results apply generally to partial order structures, and they achieve full coincidence, i.e. isomorphism, only for the order structure; while in our case the isomorphism does not extend to the additional structure of natural convergence. The natural convergence of a dlubpo can be chosen more arbitrarily. The only way to achieve full isomorphism in our case seems to presuppose the normal structure in the subcategory (or a set of structures that all together force the normal structure on the categorical object in question), which we will do afterwards.

Smyth's results are for full subcategories, while we give them for general subcategories, by extracting the needed morphisms from Smyth's proofs as conditions in the propositions. The proofs are adaptations of Smyth's proofs. We give the proofs mainly for those results that we need in the sequel.

Proposition 4.4.7. *Let* $\mathbf{K}$ *be a subcategory of* **Dlubpo**. *If* $\mathbf{K}$ *contains some object that is not empty, and* $\mathbf{K}$ *contains a terminal object* D *and all constant morphisms* $\lambda x.a \colon D \to D$ *for* $a \in |D|$, *then* D *is isomorphic to the terminal dlubpo* $\top = \{\{\bot\}, \bot \sqsubseteq \bot, \{\bot\} \to \bot\}$.

Proof. D cannot have two different elements, since this would entail two different constant morphisms $D \to D$. D cannot be empty, as there is a non-empty object. Thus D has just one element a, and it must be $\{a\} \to a$ by the singleton axiom. $\square$

Proposition 4.4.8 (Smyth, lemma 5(ii) in [Smy83], also prop. 5.2.17(2) in [AC98])**.**
Let $\mathbf{K}$ *be a subcategory of* **Dlubpo** *with the normal terminal object* $\top$*.*
Let $D, E \in \mathbf{K}$ *and* $D \times_{\mathbf{K}} E$ *be their categorical product in* $\mathbf{K}$*, with the projections* $\pi_{1\mathbf{K}}, \pi_{2\mathbf{K}}$ *and pairing functions* $\langle f, g\rangle_{\mathbf{K}}$*.*
Let $\mathbf{K}$ *have all constant functions* $\lambda x.a\colon \top \to D$*, for* $a \in |D|$*, and* $\lambda x.a\colon \top \to E$*, for* $a \in |E|$*. (As* D, E *fulfill the singleton axiom, all these functions are sure to be continuous, so are morphisms in every* full *subcategory* $\mathbf{K}$*.)*

(1) The function $\varphi\colon D \times_{\mathbf{K}} E \to D \times E$*,* $\varphi = \langle \pi_{1\mathbf{K}}, \pi_{2\mathbf{K}}\rangle$*, is bijective. (And of course it is continuous, so a morphism in* **Dlubpo***.)*
The translation φ *respects projections and pairing functions:*
$\pi_1 \circ \varphi = \pi_{1\mathbf{K}}$*,* $\pi_2 \circ \varphi = \pi_{2\mathbf{K}}$*,* $\varphi \circ \langle f, g\rangle_{\mathbf{K}} = \langle f, g\rangle$*.*

(2) Let, furthermore, some $C \in \mathbf{K}$ *with some* $c, c' \in |C|$*,* $c \sqsubseteq c'$ *and* $c \neq c'$*, and for all* $d \sqsubseteq d'$ *in* D *the function* $f\colon C \to D$ *be in* $\mathbf{K}$ *with* $fx = d$ *for* $x \sqsubseteq c$ *and* $fx = d'$ *else, and likewise for all* $e \sqsubseteq e'$ *in* E *the function* $f\colon C \to E$ *with* $fx = e$ *for* $x \sqsubseteq c$ *and* $fx = e'$ *else.*
(As to the existence of these functions in full *subcategories, the remark above applies again.)*
Then φ^{-1} *is monotonic, so that* φ *is an order-isomorphism.* φ^{-1} *is continuous in each component of its argument separately. (But it need not be continuous.)*

Proof. is an adaptation of the proof of part (ii), lemma 5 of [Smy83]. □

Proposition 4.4.9 (Smyth, lemma 5(iii) in [Smy83], also prop. 5.2.17(3) in [AC98])**.**
Let $\mathbf{K}$ *be a subcategory of* **Dlubpo** *with the normal terminal object* $\top$ *and all normal products* $D \times E$*.*
Let $D, E \in \mathbf{K}$ *and* $D \Rightarrow_{\mathbf{K}} E$ *be their categorical exponent in* $\mathbf{K}$*, with the evaluation* $\mathrm{eval}_{\mathbf{K}}\colon (D \Rightarrow_{\mathbf{K}} E) \times D \to E$ *and curried morphisms* $\mathrm{curry}_{\mathbf{K}}(f)$*.*
Let $\mathbf{K}$ *have all constant functions* $\lambda x.f\colon \top \to (D \Rightarrow_{\mathbf{K}} E)$*, for* $f \in D \Rightarrow_{\mathbf{K}} E$*.*
(As $D \Rightarrow_{\mathbf{K}} E$ *fulfills the singleton axiom, all these functions are sure to be continuous, so are morphisms in every* full *subcategory* $\mathbf{K}$*.)*

(1) The function $\varphi\colon (D \Rightarrow_{\mathbf{K}} E) \to (D \Rightarrow^{*}_{\mathbf{K}} E)$*,* $\varphi = \mathrm{curry}(\mathrm{eval}_{\mathbf{K}})$ *is bijective and monotonic. (But need not be continuous.)*
The translation φ *respects evaluation and curried functions:*
$\mathrm{eval}(\varphi a, d) = \mathrm{eval}_{\mathbf{K}}(a, d)$*,* $\varphi \circ \mathrm{curry}_{\mathbf{K}}(f) = \mathrm{curry}(f)$*.*

(2) Let, furthermore, some $C \in \mathbf{K}$ *with some* $c, c' \in |C|$*,* $c \sqsubseteq c'$ *and* $c \neq c'$*, and for all* $f_1 \sqsubseteq f_2$ *in* $D \Rightarrow^{*}_{\mathbf{K}} E$ *the function* $h\colon C \times D \to E$ *be in* $\mathbf{K}$ *with* $h(u, x) = f_1 x$ *for* $u \sqsubseteq c$ *and* $h(u, x) = f_2 x$ *else. Let* $g_1, g_2\colon \top \to C$ *be in* $\mathbf{K}$ *with* $g_1 \bot = c$ *and* $g_2 \bot = c'$*.*
Then φ^{-1} *is monotonic, so that* φ *is an order isomorphism. (But need not be a* **Dlubpo***-isomorphism.)*

Proof. **(1)** The proof is a detailed version of the first part of the proof of part (iii), lemma 5 of [Smy83].

$$\begin{tikzcd}
\top \arrow[d, "\mathrm{curry}_{\mathbf{K}}(f')"'] & \times & D \arrow[r, "f'"] \arrow[d, "\mathrm{id}"] & E \\
D \Rightarrow_{\mathbf{K}} E & \times_{\mathbf{K}} & D \arrow[ur, "\mathrm{eval}_{\mathbf{K}}"'] &
\end{tikzcd}$$

φ is **surjective**:
Let $f \in D \Rightarrow^{*}_{\mathbf{K}} E$.
Let $f' \colon \top \times D \to E$, with $f'y = f(\pi_2 y)$, $f' = f \circ \pi_2$ is in $\mathbf{K}$.
Let $h = (\mathrm{curry}_{\mathbf{K}}(f'))\bot \in (D \Rightarrow_{\mathbf{K}} E)$. We calculate:

$$\begin{aligned}
(\varphi h)x &= \mathrm{eval}_{\mathbf{K}}(h, x)\\
&= \mathrm{eval}_{\mathbf{K}}((\mathrm{curry}_{\mathbf{K}}(f'))\bot, x)\\
&= f'(\bot, x)\\
&= fx.
\end{aligned}$$

So $\varphi h = f$.

φ is **injective**:
Let $h' \in D \Rightarrow_{\mathbf{K}} E$ with $\varphi h' = f$. We have to show that $h' = h$ above.
Let $g \colon \top \to (D \Rightarrow_{\mathbf{K}} E)$ with $g\bot = h'$, g is in $\mathbf{K}$ by hypothesis.
We show that $\mathrm{eval}_{\mathbf{K}} \circ (g \times \mathrm{id}) = f'$:

$$\begin{aligned}
f'y &= f(\pi_2 y)\\
&= (\varphi h')(\pi_2 y)\\
&= \mathrm{eval}_{\mathbf{K}}(h', \pi_2 y)\\
&= (\mathrm{eval}_{\mathbf{K}} \circ (g \times \mathrm{id}))y.
\end{aligned}$$

By the uniqueness of $\mathrm{curry}_{\mathbf{K}}(f')$ in $\mathbf{K}$, it must be $g = \mathrm{curry}_{\mathbf{K}}(f')$.
Therefore $h' = g\bot = \mathrm{curry}_{\mathbf{K}}(f')\bot = h$, as desired.
(2) The proof is an adaptation of the second part of the proof of part (iii), lemma 5 of [Smy83]. □

We now come to situations, where a subcategory $\mathbf{K}$ of **Dlubpo** has a categorical (terminal, product, exponent) object D and also the normal (terminal, product, general function) dlubpo D' with a certain morphism $\varphi \colon D \to D'$, and where this implies that φ is an isomorphism. For the terminal and product we can give two propositions where we generalize from **Dlubpo** to an arbitrary category $\mathbf{C}$.

Proposition 4.4.10. *Let $\mathbf{C}$ be a category and $\mathbf{K}$ a subcategory of $\mathbf{C}$.*

(1) If $\mathbf{C}$ has a terminal object $\top$, and $\mathbf{K}$ a terminal object $\top_{\mathbf{K}}$, and $\mathbf{K}$ contains $\top$ and a morphism $\varphi \colon \top_{\mathbf{K}} \to \top$, then φ and the unique (in $\mathbf{K}$) $\psi \colon \top \to \top_{\mathbf{K}}$ are an isomorphism in $\mathbf{K}$.

(2) Let $D, E \in \mathbf{K}$. If $\mathbf{C}$ has a product $D \times E$, with $\pi_1, \pi_2, \langle\rangle$, and $\mathbf{K}$ a product $D \times_{\mathbf{K}} E$, with $\pi_{1\mathbf{K}}, \pi_{2\mathbf{K}}, \langle\rangle_{\mathbf{K}}$, and $\mathbf{K}$ contains $D \times E$, π_1, π_2 and the morphism $\varphi \colon D \times_{\mathbf{K}} E \to D \times E$,

$\varphi = \langle \pi_{1\mathbf{K}}, \pi_{2\mathbf{K}} \rangle$,
then φ and $\psi\colon D \times E \to D \times_{\mathbf{K}} E$, $\psi = \langle \pi_1, \pi_2 \rangle_{\mathbf{K}}$, are an isomorphism in $\mathbf{K}$.
The translation φ respects projections and pairing functions.

Proof. The proofs are easy and like the known proofs of the isomorphism of terminal objects resp. products in a single category. □

The situation for exponents is special and more complicated:

Proposition 4.4.11. *Let $\mathbf{K}$ be a subcategory of* **Dlubpo** *with the normal terminal object $\top$ and all normal categorical products $\times$. Let $D, E \in \mathbf{K}$.*
Let $\mathbf{K}$ have a categorical exponent $D \Rightarrow_{\mathbf{K}} E$ of D, E with evaluation $\mathrm{eval}_{\mathbf{K}}$*, curried morphisms* $\mathrm{curry}_{\mathbf{K}}(f)$ *and all constant functions $\lambda x.f\colon \top \to (D \Rightarrow_{\mathbf{K}} E)$, for $f \in |D \Rightarrow_{\mathbf{K}} E|$.*
If $\mathbf{K}$ also contains the general function dlubpo (relative to $\mathbf{K}$) $D \Rightarrow^*_{\mathbf{K}} E$ with the morphism eval*, then*
$\varphi\colon (D \Rightarrow_{\mathbf{K}} E) \to (D \Rightarrow^*_{\mathbf{K}} E)$, $\varphi = \mathrm{curry}(\mathrm{eval}_{\mathbf{K}})$ and
$\psi\colon (D \Rightarrow^*_{\mathbf{K}} E) \to (D \Rightarrow_{\mathbf{K}} E)$, $\psi = \mathrm{curry}_{\mathbf{K}}(\mathrm{eval})$ are an isomorphism in **Dlubpo**.
ψ is, of course, in $\mathbf{K}$. If φ is also in $\mathbf{K}$, then the isomorphism is also in $\mathbf{K}$.
The translation φ respects evaluation and curried functions:
$\mathrm{eval}(\varphi a, d) = \mathrm{eval}_{\mathbf{K}}(a, d)$, $\varphi \circ \mathrm{curry}_{\mathbf{K}}(f) = \mathrm{curry}(f)$.

Proof. By proposition 4.4.9(1), φ is bijective and monotonic. We have to prove that φ is also continuous, and that ψ is the inverse of φ.

$$\begin{tikzcd}
D \Rightarrow_{\mathbf{K}} E \arrow[d, bend right, "\varphi"'] & \times & D \arrow[r, "\mathrm{eval}_{\mathbf{K}}"] \arrow[d, leftrightarrow, "\mathrm{id}"] & E \\
D \Rightarrow^*_{\mathbf{K}} E \arrow[u, bend right, "\psi"'] & \times & D \arrow[ur, "\mathrm{eval}"'] &
\end{tikzcd}$$

For all $f \in D \Rightarrow^*_{\mathbf{K}} E$, $d \in |D|$, it is

$$\begin{aligned}
\mathrm{eval}_{\mathbf{K}}((\psi f), d) &= \mathrm{eval}_{\mathbf{K}}(\mathrm{curry}_{\mathbf{K}}(\mathrm{eval}) f, d) \\
&= \mathrm{eval}(f, d). \\
\text{Then } \varphi(\psi f) &= \lambda d \in D.\, \mathrm{eval}_{\mathbf{K}}(\psi f, d) \\
&= \lambda d \in D.\, \mathrm{eval}(f, d) = f.
\end{aligned}$$

Therefore ψ is the inverse of φ, as φ is bijective.

To show that φ is continuous, let $A \to a$ in $D \Rightarrow_{\mathbf{K}} E$. We have to show that $\varphi^+ A \to \varphi a$. So let $B \subseteq (D \Rightarrow^*_{\mathbf{K}} E) \times |D|$ directed with $\pi_1^+ B = \varphi^+ A$ and $\pi_2^+ B \to d$.
We have to show that $\mathrm{eval}^+ B \to \mathrm{eval}(\varphi a, d)$.
Let $C = \{\, (\psi f, b) \mid (f, b) \in B \,\}$. C is directed, as B is directed and ψ is monotonic.

It is $\pi_1^+ C = A$ and $\pi_2^+ C = \pi_2^+ B \to d$. We calculate:

$$\begin{aligned}\mathrm{eval}^+ B &= \{\, \mathrm{eval}_{\mathbf{K}}(\psi f, b) \mid (f,b) \in B \,\} \\ &= \mathrm{eval}^+_{\mathbf{K}}\, C \\ &\to \mathrm{eval}_{\mathbf{K}}(a,d) \\ &= \mathrm{eval}_{\mathbf{K}}(\psi(\varphi a), d) \\ &= \mathrm{eval}(\varphi a, d).\end{aligned}$$

So φ is an isomorphism in **Dlubpo**. $\square$

Theorem 4.4.12. *In* **Dlubpo** $\top$ *is the terminal object and* $D \times E$ *is the categorical product with* π_1, π_2 *and* $\langle f, g\rangle$.
If there is an exponent of D, E *in* **Dlubpo**, *then it is* $D \Rightarrow^* E$ *with evaluation* eval *and curried functions* curry(f).
Dlubpo *is not a ccc: there are dlubpos* D, E *such that* $D \Rightarrow^* E$ *is not the exponent, i.e. there is some* $f\colon C \times D \to E$ *such that* curry(f) *is not continuous.*

Proof. The first sentence follows from the definitions of $\top$ and $D \times E$. The second follows from proposition 4.4.11.
Here is the counter-example in figure 4.2.

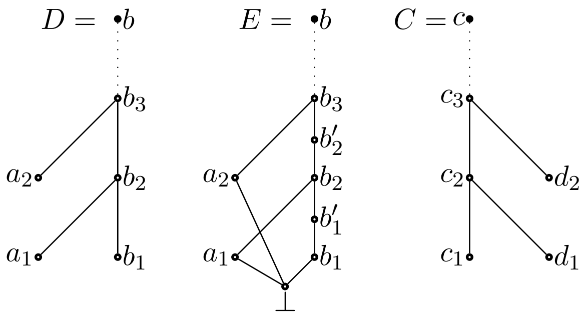


Figure 4.2: Example theorem 4.4.12, dlubpos D, E, C

In the dlubpos D, E, C the order is the reflexive, transitive closure of the relations specified below. They also contain all trivial natural lubs, i.e. all natural lubs $A \to a$ where $a \in A$. So also the subcategory of the dlubpos with all trivial natural lubs is not a ccc.

D is the dlubpo with
$|D| = \{a_i\}_{i\geq 1} \cup \{b_i\}_{i\geq 1} \cup \{b\}$,
for all i: $a_i \sqsubseteq b_{i+1}$, $b_i \sqsubseteq b_{i+1}$, b greatest element,
$D' = \{a_i\}_{i\geq 1} \cup \{b_i\}_{i\geq 1}$, $D' \to b$.

E is the dlubpo with:
$|E| = \{a_i\}_{i\geq 1} \cup \{b_i\}_{i\geq 1} \cup \{b_i'\}_{i\geq 1} \cup \{b\}$,
for all i: $a_i \sqsubseteq b_{i+1}$, $b_i \sqsubseteq b_i' \sqsubseteq b_{i+1}$, b greatest element, $\perp$ least element,
for all directed $S \subseteq |E|$ with the lub b and some $b_i' \in S$ it is $S \to b$.

C is the dlubpo with:
$|C| = \{d_i\}_{i\geq 1} \cup \{c_i\}_{i\geq 1} \cup \{c\}$, it is isomorphic to D,
for all i: $d_i \sqsubseteq c_{i+1}$, $c_i \sqsubseteq c_{i+1}$, c greatest element,
$C' = \{d_i\}_{i\geq 1} \cup \{c_i\}_{i\geq 1}$, $C' \to c$.

We define a continuous function $f\colon C \times D \to E$ by defining curry(f). In the following, we abbreviate curry$(f)x = \overline{x}$ for all $x \in |C|$. For all $n \geq 1$ we define:

$$\begin{aligned}
\overline{c_n} a_1 &= \bot & \overline{d_n} a_1 &= \bot \\
\overline{c_n} a_i &= a_{i-1} \text{ for } 2 \leq i \leq n & \overline{d_n} a_i &= a_{i-1} \text{ for } 2 \leq i < n \\
\overline{c_n} a_i &= \bot \text{ for } i > n & \overline{d_n} a_i &= \bot \text{ for } i \geq n \\
\overline{c_n} b_i &= b'_i \text{ for } i < n & \overline{d_n} b_i &= b'_i \text{ for } i < n \\
\overline{c_n} b_i &= b'_{n-1} \text{ for } i \geq n & \overline{d_n} b_i &= b_n \text{ for } i \geq n \\
\overline{c_n} b &= b'_{n-1} & \overline{d_n} b &= b_n
\end{aligned}$$

$$\begin{aligned}
\overline{c} a_1 &= \bot \\
\overline{c} a_i &= a_{i-1} \text{ for } i \geq 2 \\
\overline{c} b_i &= b'_i \text{ for } i \geq 1 \\
\overline{c} b &= b
\end{aligned}$$

For all $x \in |C|$, the function $\overline{x}$ is monotonic. Furthermore, for all n it is $\overline{c_n} \sqsubseteq \overline{c_{n+1}}$, $\overline{d_n} \sqsubseteq \overline{c_{n+1}}$, $\overline{c_n} \sqsubseteq \overline{c}$, also $\overline{d_n} \sqsubseteq \overline{d_{n+1}}$ and $\overline{c_n} \sqsubseteq \overline{d_{n+1}}$, so that we get the diagram figure 4.3.

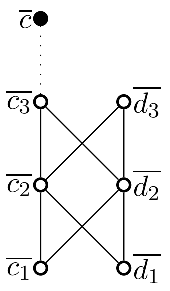


Figure 4.3: Example theorem 4.4.12, functions $\bar{x} = \text{curry}(f)x$

From all this follows that f is monotonic.
f is also continuous: let $A \subseteq |C| \times |D|$ directed with $\pi_1^+ A \to e_1$ and $\pi_2^+ A \to e_2$. We have to show that $f^+ A \to f(e_1, e_2)$. The proof is by going through the cases, we leave out the cases where one (or two) of the natural lubs is a trivial natural lub.
The interesting case is $\pi_1^+ A = C' \to c$ and $\pi_2^+ A = D' \to b$:
For every n there is some $(c_n, y) \in A$ and some $(x, b_n) \in A$.
As A is directed, there must be some $(c_i, b_j) \in A$ with $i \geq n$ and $j \geq n$, then it is $b'_{n-1} \sqsubseteq f(c_i, b_j)$. Therefore b is the lub of $f^+ A$. Also $f(c_i, b_j) = b'_k$ for some k. Therefore $f^+ A \to b = f(c, b)$. So f is continuous.

We now prove that curry(f) is not continuous:
It is $C' \to c$ in C. We define $\overline{C'} = (\text{curry}(f))^+ C'$. It is $\overline{c} = \text{curry}(f)c$. We prove that not $\overline{C'} \to \overline{c}$ in $D \Rightarrow^* E$.
Let $A \subseteq |D \Rightarrow^* E| \times |D|$ be the set $A = \{(\overline{c_n}, a_n)\}_{n \geq 1} \cup \{(\overline{d_n}, b_n)\}_{n \geq 1}$.
A is directed, as for all n it is $(\overline{c_n}, a_n) \sqsubseteq (\overline{d_{n+1}}, b_{n+1})$ and $(\overline{d_n}, b_n) \sqsubseteq (\overline{d_{n+1}}, b_{n+1})$.
It is $\pi_1^+ A = \overline{C'}$ and $\pi_2^+ A = D' \to b$, but eval$^+ A = \{\bot\} \cup \{a_i\}_{i \geq 1} \cup \{b_i\}_{i \geq 1}$ does not have eval$(\overline{c}, b) = b$ as natural lub, as eval$^+ A$ does not contain any b'_i. □

We now come to the problem of finding a large sub-ccc of **Dlubpo** (that uses the general exponents $D \Rightarrow^* E$). We conjecture that it is possible to find such a category that is the largest among those sub-cccs that are generated by some kind of invariant lub-rule class, see the following section 4.5, analogous to the result of theorem 4.6.7 for subcategories that use the (pointwise) exponent $D \Rightarrow E$. But it seems that this comes at a price: the needed axiom will be rather complicated, it encodes the condition that curried functions are continuous. And the definition of the notion of lub-rule class has to be augmented. As we think that such a result will not be worth the effort, we do not follow this, but show here that the simple axiom S5 is sufficient for cartesian closedness, though not in any sense maximal.

Definition 4.4.13. A dlubpo D is an *S5-dlubpo* if it fulfills axiom S5 for directed sets: If $X \subseteq Y \subseteq |D|$, X, Y directed, $Y \to y$ and $Y \sqsubseteq X$, then $X \to y$.

Note that from $X \subseteq Y \subseteq |D|$, Y directed and $Y \sqsubseteq X$ follows that X is also directed. Therefore by axiom S5 we can deduce from natural lubs of directed sets only natural lubs of sets that are directed again. So we get the same class of dlubpos when we demand axiom S5 for general sets in the definition, contrary to the situation for natural domains.

Proposition 4.4.14. *Let D, E, C be dlubpos.*

(1) The terminal dlubpo $\top$ fulfills S5.

(2) If D, E fulfill S5, then $D \times E$ fulfills S5.

(3) If E fulfills S5, then $D \Rightarrow^* E$ fulfills S5.

(4) If E fulfills S5, then for all $f : C \times D \to E$ it is $\mathrm{curry}(f) : C \to (D \Rightarrow^* E)$ *continuous.*

Proof. **(1)** is clear.
(2) Let $X \subseteq Y \subseteq |D \times E|$ directed, $Y \to y$ and $Y \sqsubseteq X$. Then $\pi_1^+ X \subseteq \pi_1^+ Y$ directed, $\pi_1^+ Y \to \pi_1 y$ and $\pi_1^+ Y \sqsubseteq \pi_1^+ X$, so $\pi_1^+ X \to \pi_1 y$. Also $\pi_2^+ X \to \pi_2 y$. Therefore $X \to y$.
(3) Let $F \subseteq G \subseteq |D \Rightarrow^* E|$ directed, $G \to g$ and $G \sqsubseteq F$. We have to show $F \to g$.
Let $A \subseteq |D \Rightarrow^* E| \times |D|$ directed with $\pi_1^+ A = F$ and $\pi_2^+ A \to d$. We have to show $\mathrm{eval}^+ A \to gd$.
Let $B = G \times (\pi_2^+ A)$. B is directed, so $\mathrm{eval}^+ B \to gd$.
For every $(h, a) \in B$ there is some $(h', y) \in A$ with $h \sqsubseteq h'$, and some $(x, a) \in A$.
As A is directed, there is some $(h'', a') \in A$ with $(h, a) \sqsubseteq (h'', a')$.
So we get $B \sqsubseteq A$, therefore $\mathrm{eval}^+ B \sqsubseteq \mathrm{eval}^+ A$.
From $A \subseteq B$ follows $\mathrm{eval}^+ A \subseteq \mathrm{eval}^+ B$.
From all this follows $\mathrm{eval}^+ A \to gd$, as E fulfills S5.
(4) Let $C' \to c$ in C. Let $\overline{C'} = (\mathrm{curry}(f))^+ C'$ and $\overline{c} = \mathrm{curry}(f)c$.
We have to show $\overline{C'} \to \overline{c}$ in $D \Rightarrow^* E$.
Let $A \subseteq |D \Rightarrow^* E| \times |D|$ directed with $\pi_1^+ A = \overline{C'}$ and $\pi_2^+ A \to d$.
We have to show $\mathrm{eval}^+ A \to \overline{c}d$.
Let $B \subseteq |C| \times |D|$ with $B = C' \times (\pi_2^+ A)$. B is directed and $B \to (c, d)$, so $f^+ B \to f(c, d)$.
For every $(b, a) \in B$ there is some $(\mathrm{curry}(f)b, y) \in A$ and some $(x, a) \in A$.
As A is directed, there is some $(g, a') \in A$ with $(\mathrm{curry}(f)b, y) \sqsubseteq (g, a')$ and $(x, a) \sqsubseteq (g, a')$, so $f(b, a) \sqsubseteq \mathrm{eval}(g, a')$.
Therefore we get $f^+ B \sqsubseteq \mathrm{eval}^+ A$.
Together with $\mathrm{eval}^+ A \subseteq f^+ B$ and $f^+ B \to \overline{c}d$ we get $\mathrm{eval}^+ A \to \overline{c}d$, as E fulfills S5. □

It follows this theorem:

Theorem 4.4.15. *The full subcategory* **S5dlubpo** *of* **Dlubpo***, of all S5-dlubpos, is cartesian closed, with the normal terminal object* $\top$*, the normal products* $D \times E$*, the general exponents* $D \Rightarrow^* E$*, and the corresponding morphisms.* $\square$

4.5 Lub-rule classes and closed dlubpos

In this section we set up the general frame of lub-rule systems and classes, in which we can describe axiom systems that are like those of Sazonov natural domains and dlubpos. In this frame we can give a precise definition of the completeness of the axiom system; and in section 4.7 it will turn out that the axioms of Sazonov natural domains are not complete.

Let us have a look at an example axiom from section 4.3:
(S6 cofinality) If $X, Y \subseteq |D|$, $X \to x$ and $X \sqsubseteq Y \sqsubseteq x$, then $Y \to x$.
Here we first have as hypothesis of the axiom a subset with a natural lub, $X \to x$, and a further subset Y with an order-theoretic configuration, $X \sqsubseteq Y \sqsubseteq x$. From this configuration follows $\bigsqcup Y = x$, and $Y \to x$ is the conclusion of the axiom.

The important property of the configuration, $X \sqsubseteq Y \sqsubseteq x$, is that it is invariant against monotonic functions f into another partial order E: it is $f^+X \sqsubseteq f^+Y \sqsubseteq fx$. If f also respects the lub $\bigsqcup X = x$, i.e. $\bigsqcup f^+X = fx$, then also in E we can draw the conclusion $\bigsqcup f^+Y = fx$. We can say this property of invariance means that the conclusion $\bigsqcup Y = x$, and $Y \to x$ in the axiom, is "necessary".

All the axioms we have seen so far are of this form. (Except axiom S1 which we do not count as proper axiom.) They are necessary deductions of a lub from some set of natural lubs in an invariant order-theoretic configuration. We call these axioms "valid".

Here is an axiom that is not valid: If $X \subseteq |D|$ and $\bigsqcup X = x$, then $X \to x$. It is not valid, because there are monotonic functions f from D that do not respect $\bigsqcup X = x$. The monotonic function f needs to respect only the natural lubs of the hypothesis of the axiom, and here there are none.

We now formalize these intuitions. Our results should also be applicable to domains that use natural lubs of *general* subsets, like natural domains. For this we define lub partial orders as general structures, only for use in this section. (Further below we will go over from lubpos to dlubpos.)

Definition 4.5.1. A structure $D = (|D|, \sqsubseteq_D, \to_D)$ is a *lub partial order (lubpo)* if
$|D|$ is a set of *elements*,
$\sqsubseteq_D$ is a partial order on $|D|$,
$\to_D \subseteq \bar{\mathcal{P}}(D) \times |D|$ is the *(natural) convergence relation* with $\bigsqcup A = a$ for $A \to a$. (Remember that $\bar{\mathcal{P}}(D)$ is the set of subsets of $|D|$ that have a lub in $\sqsubseteq_D$.)
If $(|D|, \sqsubseteq_D)$ is a partial order and $P \subseteq \bar{\mathcal{P}}(D)$, then $\to^P \subseteq \bar{\mathcal{P}}(D) \times |D|$ is the relation with $A \to^P (\bigsqcup A)$ for all $A \in P$, so that $(|D|, \sqsubseteq_D, \to^P)$ is the lubpo corresponding to the set P of natural subsets.

A lub-rule models an instance of the application of an axiom:

Definition 4.5.2. A *lub-rule* on a partial order D is a triple $(D, P\rightsquigarrow A)$ with $P \subseteq \bar{\mathcal{P}}(D)$ and $A \in \bar{\mathcal{P}}(D)$. P is called the *pattern* of the lub-rule and A its *result*.

A lub-rule $(D, P\rightsquigarrow A)$ is *valid* if for every partial order E and every monotonic function $f\colon D \to E$ that respects the lubs of the elements of P (i.e. $\forall B \in P.\ f(\bigsqcup B) = \bigsqcup(f^+B)$), f respects also the lub of A (i.e. $f(\bigsqcup A) = \bigsqcup(f^+A)$).

A lubpo E *fulfills* a lub-rule $(D, P\rightsquigarrow A)$ if:
If $(|E|, \sqsubseteq_E) = D$ and all elements of P are natural in E, then A is natural in E.

A *lub-rule system* $\mathcal{R}$ on a partial order D is a set of pairs (rules) $(P\rightsquigarrow A)$ such that $(D, P\rightsquigarrow A)$ is a valid lub-rule on D. This is a rule system in the sense of definition 4.2.4, so we get derived lub-rules from the lub-rule system. (Proposition 4.5.7 below shows that the derived rules are valid.) In this context we implicitely understand every rule $(P\rightsquigarrow A)$ of $\mathcal{R}$ as the lub-rule $(D, P\rightsquigarrow A)$.

This lub-rule system $\mathcal{R}$ is *complete* if every valid lub-rule on D is derivable from it.

A *lub-rule class* is a class of valid lub-rules. (They can be on different partial orders.)

A lub-rule class $\mathcal{R}$ *generates* the class of lubpos that fulfill all lub-rules of $\mathcal{R}$.

A lub-rule class $\mathcal{R}$ *induces* on every partial order D the lub-rule system of all $(P\rightsquigarrow A)$ with $(D, P\rightsquigarrow A) \in \mathcal{R}$. $\mathcal{R}$ is *complete* if all these induced lub-rule systems are complete. A lub-rule is *derived* from $\mathcal{R}$, if it is derived in one of the induced lub-rule systems.

A lub-rule class $\mathcal{R}$ is *invariant* if for every lub-rule $(D, P\rightsquigarrow A) \in \mathcal{R}$, every partial order E and every monotonic function $f\colon D \to E$ that respects the lubs of the elements of P, also the lub-rule $(E, f^{++}P\rightsquigarrow f^+A)$ is derivable in the lub-rule system induced by $\mathcal{R}$ on E. (It is defined $f^{++}P = (f^+)^+P = \{\, f^+B \mid B \in P \,\}$.) (The last is a valid lub-rule as the first is it.)

Proposition 4.5.3. *If we have a lubpo E and a lub-rule system $\mathcal{R}$ on $(|E|, \sqsubseteq_E)$ such that E fulfills every rule of $\mathcal{R}$, then E fulfills also every derived rule of $\mathcal{R}$.*
Every complete lub-rule class is invariant. □

Let us demonstrate the translation from the axiom system to the lub-rule class for the axiom S6 (cofinality). It is translated to the lub-rule class of all lub-rules $(D, \{X\}\rightsquigarrow Y)$ such that D is a partial order, $X \subseteq |D|$ with $\bigsqcup X = x$ for some x, and $X \sqsubseteq Y \sqsubseteq x$. All these are valid lub-rules. The other axioms S3 and S7 of natural domains are likewise translated, so that we get an invariant lub-rule class $\mathcal{S}$ for the axioms of Sazonov natural domains. (The proof is immediate and boring.)

Philosophical significance: the quest for intension by partial extensionalization

Let us reflect on what we have done so far. The following is not mathematics, it is the thoughts that I had when discovering this piece of mathematics.

Since the Logic of Port-Royal (1662), philosophical logic makes the distinction between two kinds of meaning of a concept: the *intension* (or content), i.e. the predicates the concept uses to describe the things which fall under it, and the *extension*, i.e. the class of all these things.

Mathematics, as based on set theory, is always extensional, i.e. we use *intensions* (like expressions, axioms), to describe, define *extensions* (like sets, classes). We always talk in intensions about extensions.

But what can we do when we want to talk *in mathematics* about an intension? We have to *extensionalize* an aspect of the intension, we have to introduce new extensions. This may occur at different levels, the intension covers a whole spectrum of them. We are always sure to have the lowest level, i.e. the purely syntactic text on which the intension is founded, and the highest level, the extension that the intension describes. We call the last one the *full extensionalization*, and a level properly between the lowest and the highest a *partial extensionalization.*

In our concrete case we have as intension an axiom system for lubpos, which defines as extension a class of lubpos. We deliberately leave open the formal logic of the axiom system, so that we describe the following translation informally. The axiom system must be amenable to this translation to a lub-rule class, which is our partial extensionalization. A lub-rule abstracts a certain form of inference from the axiom, in extensional form. It must be defined that the partial extensionalization describes the full one, which is done by our definition of a lub-rule class generating a class of lubpos. We have gained new objects, the lub-rule classes, which can be used to analyse classes of lubpos.

There might be similar examples in the literature, but I know only the following one. Sometimes the partial extensionalization works by forming equivalence classes of the syntactic expressions of the intensions. This is the case for the formalization of the notion of algorithm given by Noson Yanofsky [Yan10]. Here the intensional objects are (primitive recursive) programs, and the full extensions are the functions they describe. The algorithms are defined as certain equivalence classes of programs.

Now we introduce the important cl-rule system on a lubpo and its closure operation, and the lub-completion of a lubpo. These procedures appear in [CR80] and are used there for various completions of partial orders that respect lubs that are already fixed. In this section we use them to give a characterization of valid lub-rules; this will be used in section 4.7 to prove the incompleteness of the Sazonov axioms $\mathcal{S}$.

Definition 4.5.4 (Courcelle/Raoult, proof of theorem 1 in [CR80])**.** Let D be a lubpo. The *cl-rule system* on D is the rule system (in the sense of def. 4.2.4) $\rightsquigarrow$ on $|D|$ given by the rules:
for all $a, b \in |D|$ with $b \sqsubseteq a$: $\{a\} \rightsquigarrow b$,
for all $A \subseteq |D|$, $a \in |D|$ with $A \to a$: $A \rightsquigarrow a$.
The closure operator of the cl-rule system is called cl_D.
An $A \subseteq |D|$ is *closed* if it is closed under cl_D.
(We remark that closed sets are a generalization of subsets of dcpos that are closed w.r.t. the Scott topology.)
If we have a lubpo given this way: $D = (|D|, \sqsubseteq_D, \to^P)$, with its set of natural sets P, then we abuse notation and write cl_P for cl_D, if $|D|$ and $\sqsubseteq_D$ are clear from the context.
$\hat{D}$ is the set of all closed subsets of D.
The *lub-completion* of D is $\text{lub-comp}(D) = (\hat{D}, \sqsubseteq)$, with $\sqsubseteq$ the inclusion $\subseteq$.
It is a complete lattice with $\bigsqcup \bar{A} = \mathrm{cl}_D(\bigcup \bar{A})$, for $\bar{A} \subseteq \hat{D}$.
There is an embedding $\mathrm{in}_D \colon |D| \to \hat{D}$ defined by $\mathrm{in}_D\, d = \mathrm{cl}_D\{d\} = \{\, x \in D \mid x \sqsubseteq d \,\}$.

Proposition 4.5.5 (Courcelle/Raoult, proof of theorem 1 in [CR80])**.** *The embedding* in_D *is an order embedding, i.e. it is monotonic, injective, and the reverse is monotonic. It maps natural lubs in* D *to lubs in* $\hat{D}$*, i.e. for* $A \to a$ *in* D *it is* $\mathrm{in}_D\, a = \bigsqcup(\mathrm{in}_D^+ A) = \mathrm{cl}_D\, A$*.*

Proof. in_D and its reverse are clearly monotonic.
Now let $A \to a$ in D. It is $A \subseteq \bigcup(\mathrm{in}_D^+ A)$, then $a \in \mathrm{cl}(\bigcup(\mathrm{in}_D^+ A)) = \bigsqcup(\mathrm{in}_D^+ A)$, so $\mathrm{in}_D\, a \subseteq \bigsqcup(\mathrm{in}_D^+ A)$.
For the reverse, it is $\mathrm{in}_D\, b \subseteq \mathrm{in}_D\, a$ for all $b \in A$. □

Proposition 4.5.6. *Let* $(D, P \rightsquigarrow A)$ *be a lub-rule and* $E = (|D|, \sqsubseteq_D, \to^P)$ *be the corresponding lubpo with the set* P *of natural subsets. Let* $\bigsqcup A = a$*.*
$(D, P \rightsquigarrow A)$ *is valid* $\iff a \in \mathrm{cl}_E(A)$*, also written* $a \in \mathrm{cl}_P(A)$*.*

Proof. $\Longrightarrow$: We have the embedding $\mathrm{in}\colon (|E|, \sqsubseteq) \to \text{lub-comp}(E) = (\hat{E}, \sqsubseteq)$.
By proposition 4.5.5, the function in respects the natural lubs of the elements of P. Whence by validity it also respects the lub of A, so $\mathrm{in}(\bigsqcup A) = \bigsqcup(\mathrm{in}^+ A)$.
Further we get $\bigsqcup(\mathrm{in}^+ A) = \mathrm{cl}_E(\bigcup(\mathrm{in}^+ A)) = \mathrm{cl}_E(A)$, thus $a \in \mathrm{cl}_E(A)$.
$\Longleftarrow$: Let $F = (|F|, \sqsubseteq_F)$ be a partial order and $f\colon D \to F$ be a monotonic function that respects the lubs of the elements of P. We have to show: $fa = \bigsqcup f^+A$.
We prove: For all $x \in \mathrm{cl}_E(A)$, for all upper bounds b of f^+A, it is $fx \sqsubseteq b$, by induction on the deduction of $x \in \mathrm{cl}_E(A)$ in the cl-rule system on E.
(1) This is true for $x \in A$.
(2) Let x be deduced from $y \in \mathrm{cl}_E(A)$ by $x \sqsubseteq y$. Then $fx \sqsubseteq fy \sqsubseteq b$.
(3) Let x be deduced from $Y \subseteq \mathrm{cl}_E(A)$ by $Y \to x$. Then $fx = \bigsqcup(f^+Y) \sqsubseteq b$.
It is clear that fa is an upper bound of f^+A. The proved property for $x = a$ shows that it is the least upper bound. □

Proposition 4.5.7. *Let* $\mathcal{R}$ *be a lub-rule system on a partial order* D*. Every derived rule* $P \rightsquigarrow A$ *of* $\mathcal{R}$ *is valid.*

Proof. Let D1 be a deduction of A from P in the lub-rule system $\mathcal{R}$.
We prove by induction on D1 that for every node of D1 labelled with some deduced A', with $\bigsqcup A' = a'$, it is $a' \in \mathrm{cl}_P(A')$.
(1) This is clear for $A' \in P$.
(2) Let A' be deduced from some P' at this node, $P' \rightsquigarrow A'$.
From proposition 4.5.6 follows that $a' \in \mathrm{cl}_{P'}(A')$.
Let D2 be a deduction of a' from A' in the cl-rule system on $(|D|, \sqsubseteq_D, \to^{P'})$.
We prove by induction on D2 that for every node of D2 labelled with some deduced x, it is $x \in \mathrm{cl}_P(A')$.
(2.1) This is clear for $x \in A'$.
(2.2) Let x be deduced from y by $x \sqsubseteq y$. It is $y \in \mathrm{cl}_P(A')$ by induction hypothesis, so $x \in \mathrm{cl}_P(A')$.
(2.3) Let x be deduced from $Y \in P'$ by $Y \to x$.
The elements of P' were deduced in the deduction D1, so by induction hypothesis of the outer induction on D1 it is $x \in \mathrm{cl}_P(A')$.

From the inner induction on D2 follows that $a' \in \mathrm{cl}_P(A')$.
From the outer induction on D1 follows that $\bigsqcup A \in \mathrm{cl}_P(A)$, so $P \rightsquigarrow A$ is valid by proposition 4.5.6. □

The easiest way to get a complete lub-rule class is to take just all valid lub-rules, which can be characterized by proposition 4.5.6:

Definition 4.5.8. The *canonical complete lub-rule class* $\mathcal{C}$ is the class of all valid lub-rules. This is the set of all lub-rules $(D, P{\rightsquigarrow}A)$ with $\bigsqcup A \in \mathrm{cl}_P(A)$.

We are mainly interested in the natural sets that are directed, as continuity is based on them, and regard the undirected natural sets only as intermediary (may be necessary) steps in the deduction of naturality of directed sets, see the discussion in section 4.3. So we go over from lubpos to dlubpos.

Definition 4.5.9. A lubpo that fulfills the singleton axiom S3 is called a *lubpo with singletons*. For a lubpo with singletons we have the *dlubpo for* D, $\delta(D) = (|D|, \sqsubseteq_D, \twoheadrightarrow)$ with $\twoheadrightarrow$ the restriction of $\twoheadrightarrow_D$ to directed subsets.

A lub-rule class $\mathcal{R}$ is *with singletons* if $\mathcal{R}$ has for all partial orders D and all $x \in |D|$ the (valid) lub-rule $(D, \emptyset{\rightsquigarrow}\{x\})$.

A lub-rule $(D, P{\rightsquigarrow}A)$ is *directed* if A and the elements of P are all directed.
For a lub-rule class $\mathcal{R}$ with singletons, $\delta(\mathcal{R})$ is the lub-rule class of all derived lub-rules of $\mathcal{R}$ that are directed.
A lub-rule class $\mathcal{R}$ with singletons *generates* the class of all dlubpos $\delta(D)$ for D a lubpo that fulfills the lub-rules of $\mathcal{R}$. $\mathcal{R}$-**cat** is the full subcategory of this class in **Dlubpo**.

Proposition 4.5.10. *A lub-rule class $\mathcal{R}$ with singletons generates the class of dlubpos that fulfill all lub-rules of $\delta(\mathcal{R})$.*

Proof. We have to prove that very dlubpo D that fulfills all lub-rules of $\delta(\mathcal{R})$ can be extended to a lubpo $D' = (|D|, \sqsubseteq_D, \twoheadrightarrow')$ such that D' fulfills the lub-rules of $\mathcal{R}$ and $\delta(D') = D$.
Take the set P of all (directed) natural sets of D and build the closure P' of P under the lub-rule system induced by $\mathcal{R}$ on $(|D|, \sqsubseteq_D)$. Take $\twoheadrightarrow' = \twoheadrightarrow^{P'}$. □

If we use all valid lub-rules, the deduction of (directed) natural lubs from existing ones is always in one step. Thus we can define the class of dlubpos that is generated by $\mathcal{C}$ by an axiom on directed lubs that tests the condition of proposition 4.5.6:

Definition 4.5.11. A dlubpo $D = (|D|, \sqsubseteq_D, \twoheadrightarrow_D)$ is a *closed dlubpo (cdlubpo)* if it fulfills the axiom:
(S9 closure) If $A \subseteq |D|$ is directed with lub a and $a \in \mathrm{cl}_D(A)$, then $A \twoheadrightarrow_D a$.

Note that this axiom, as it stands, does not describe a complete lub-rule class. But its extension is exactly the class of dlubpos generated by the complete lub-rule class $\mathcal{C}$.

Proposition 4.5.12. *A dlubpo D is a cdlubpo $\iff$ D fulfills all valid directed lub-rules $((|D|, \sqsubseteq_D), P{\rightsquigarrow}A)$.*
The category **Cdlubpo** *of cdlubpos is the category* $\mathcal{C}$-**cat**.

Proof. $\Longrightarrow$: Let $((|D|, \sqsubseteq_D), P{\rightsquigarrow}A)$ be a valid lub-rule of directed sets such that all elements of P are natural in D, and $\bigsqcup A = a$. Then $a \in \mathrm{cl}_P(A)$ by proposition 4.5.6, therefore also $a \in \mathrm{cl}_D(A)$. Thus $A \twoheadrightarrow a$.

$\Longleftarrow$: Let $A \subseteq |D|$ be directed with lub a and $a \in \mathrm{cl}_D(A)$. Let P be the set of all natural (directed) subsets of $|D|$. Then $((|D|, \sqsubseteq_D), P \rightsquigarrow A)$ is valid by proposition 4.5.6. Thus $A \to a$. □

As a preparation for the next sections, we now give some properties of the classes of dlubpos that are generated by (invariant) lub-rule classes.

Definition 4.5.13. Let D be a dlubpo with the set P of natural directed subsets, and $\mathcal{R}$ be a lub-rule class with singletons. The $\mathcal{R}$-completion $\mathcal{R}\text{-comp}(D)$ of D is the dlubpo $(|D|, \sqsubseteq_D, \to^{P'})$ with P' the directed sets of the closure of P under the lub-rule system induced by $\mathcal{R}$.

Proposition 4.5.14. *Let D, E be dlubpos and $\mathcal{R}$ be a lub-rule class with singletons.*

(1) D is in the class of dlubpos generated by $\mathcal{R}$ iff $\mathcal{R}\text{-comp}(D) = D$.

(2) If $\mathcal{R}$ is invariant, then every continuous function $f\colon D \to E$ is also a continuous function $f\colon \mathcal{R}\text{-comp}(D) \to \mathcal{R}\text{-comp}(E)$.

Proof. (1) is clear.
(2) Let $D' = \mathcal{R}\text{-comp}(D)$ and $E' = \mathcal{R}\text{-comp}(E)$.
We have to prove: For every $A \to a$ in D' it is $f^+A \to fa$ in E'.
There is a deduction of A from the set P of natural directed sets of D in the lub-rule system induced by $\mathcal{R}$. We prove by induction on this deduction, that in every node for the deduced A' (with lub a') it is $\bigsqcup f^+A' = fa'$ and f^+A' is in the closure of $f^{++}P$ in the lub-rule system of $\mathcal{R}$ on E.
This is clear for $A' \in P$, as $f\colon D \to E$ is continuous.
Now let A' be deduced from the set of subsets P' by the (valid) lub-rule $(D, P' \rightsquigarrow A')$.
By induction hypothesis f respects the lubs of the elements of P', so also the lub of A': $\bigsqcup f^+A' = fa'$.
By induction hypothesis for all $B \in P'$, f^+B is in the closure of $f^{++}P$. As $\mathcal{R}$ is invariant, $(E, f^{++}P' \rightsquigarrow f^+A')$ is a derived lub-rule in the lub-rule system of $\mathcal{R}$. Therefore also f^+A' is in the closure of $f^{++}P$.

From the proved hypothesis follows that $f^+A \to fa$. □

Theorem 4.5.15. *Let $\mathcal{R}$ be an invariant lub-rule class with singletons.*
*Then $\mathcal{R}$-**cat** is a full reflective subcategory of **Dlubpo**.*
The adjunction is $\langle \mathcal{R}\text{-comp}, \mathcal{R}\text{-inc}, \mathcal{R}\text{-}\eta\rangle\colon \mathbf{Dlubpo} \rightharpoonup \mathcal{R}\text{-}\mathbf{cat}$ with
$\mathcal{R}\text{-comp}\colon \mathbf{Dlubpo} \to \mathcal{R}\text{-}\mathbf{cat}$ the functor mapping each $D \in \mathbf{Dlubpo}$ to $\mathcal{R}\text{-comp}(D)$ and each $f\colon D \to E$ to the same $f\colon \mathcal{R}\text{-comp}(D) \to \mathcal{R}\text{-comp}(E)$,
$\mathcal{R}\text{-inc}\colon \mathcal{R}\text{-}\mathbf{cat} \to \mathbf{Dlubpo}$ the inclusion functor,
$\mathcal{R}\text{-}\eta\colon \mathrm{id}_{\mathbf{Dlubpo}} \to \mathcal{R}\text{-inc} \circ \mathcal{R}\text{-comp}$ the natural transformation with
$\mathcal{R}\text{-}\eta_D\colon D \to \mathcal{R}\text{-inc}(\mathcal{R}\text{-comp}(D))$, $\mathcal{R}\text{-}\eta_D = \mathrm{id}$.

Proof. $\mathcal{R}$-comp is a functor by proposition 4.5.14.
$\mathcal{R}\text{-}\eta$ clearly is a natural transformation.
We have to prove that for every $D \in \mathbf{Dlubpo}$, $E \in \mathcal{R}\text{-}\mathbf{cat}$, $f\colon D \to \mathcal{R}\text{-inc}(E)$, there is a unique $g\colon \mathcal{R}\text{-comp}(D) \to E$ with $f = \mathcal{R}\text{-inc}(g) \circ \mathcal{R}\text{-}\eta_D$. We choose for g the same function f, as $f\colon \mathcal{R}\text{-comp}(D) \to \mathcal{R}\text{-comp}(E) = E$ is continuous by proposition 4.5.14. □

Definition 4.5.16. Let D, E be dlubpos.
We define the *(pointwise) function space dlubpo* $D \Rightarrow E$ by
$|D \Rightarrow E|$ the set of all continuous $f\colon D \to E$,
$f \sqsubseteq_{D\Rightarrow E} g$ iff for all $x \in |D|$ it is $fx \sqsubseteq_E gx$,
if $F \subseteq |D \Rightarrow E|$ is directed and $f \in |D \Rightarrow E|$, then $F \to_{D\Rightarrow E} f$ iff for all $d \in |D|$ it is $Fd = \{\, gd \mid g \in F \,\} \to_E fd$.

In this definition it is $\bigsqcup F = f$. The singleton axiom is fulfilled.

Proposition 4.5.17. *Let $\mathcal{R}$ be an invariant lub-rule class with singletons and $D, E \in \mathcal{R}$-**cat**. Then*

*(1) $\top \in \mathcal{R}$-**cat** is the categorical terminal object,*

*(2) $D \times E \in \mathcal{R}$-**cat** is the categorical product,*

*(3) $D \Rightarrow E \in \mathcal{R}$-**cat** (but need not be the categorical exponent).*

Proof. **(1)** On $(|\top|, \sqsubseteq)$ the only valid lub-rules are $S \rightsquigarrow \{\bot\}$ for $S \subseteq \{\emptyset, \{\bot\}\}$ and they are all fulfilled by $\top$.
(2) Let $((|D \times E|, \sqsubseteq), P \rightsquigarrow A)$ be a derived directed lub-rule of $\mathcal{R}$, such that the elements of P are natural. (We have to show that A is natural.)
Then the elements of $\pi_1^{++}P$ and $\pi_2^{++}P$ are also natural in D resp. E.
$((|D|, \sqsubseteq), \pi_1^{++}P \rightsquigarrow \pi_1^{+}A)$ and $((|E|, \sqsubseteq), \pi_2^{++}P \rightsquigarrow \pi_2^{+}A)$ are derived rules of $\mathcal{R}$, by invariance of $\mathcal{R}$ w.r.t. the continuous functions π_1 resp. π_2.
Thus $\pi_1^{+}A$ and $\pi_2^{+}A$ are natural in D resp. E. Therefore A is natural in $D \times E$, so $D \times E$ fulfills the lub-rule $P \rightsquigarrow A$. We get $D \times E \in \mathcal{R}$-**cat**.
As $\mathcal{R}$-**cat** is a full subcategory of **Dlubpo**, $\mathcal{R}$-**cat** has the projections and pairing functions, so $D \times E$ is the categorical product in $\mathcal{R}$-**cat**.
(3) Let $((|D \Rightarrow E|, \sqsubseteq), P \rightsquigarrow A)$ be a derived directed lub-rule of $\mathcal{R}$, such that the elements of P are natural. (We have to show that A is natural.)
Then the elements of $Px = \{\, Bx \mid B \in P \,\}$, with $Bx = \{\, fx \mid f \in B \,\}$, are also natural in E.
$((|E|, \sqsubseteq), Px \rightsquigarrow Ax)$ is a derived lub-rule of $\mathcal{R}$, by invariance of $\mathcal{R}$ w.r.t. the function $\lambda f.fx$, that respects the lubs of the $B \in P$.
Thus Ax is natural in E for every $x \in |D|$. Therefore A is natural in $D \Rightarrow E$, so $D \Rightarrow E$ fulfills the lub rule $P \rightsquigarrow A$. We get $D \Rightarrow E \in \mathcal{R}$-**cat**. $\square$

4.6 The ccc of S10-dlubpos

In this section we find a large cartesian closed full subcategory of **Dlubpo** whose exponents are the (pointwise) function spaces $D \Rightarrow E$ which are known from natural domains and from definition 4.5.16. The underlying structures are the S10-dlubpos, which fulfill the singleton axiom and a new axiom S10. All natural domains are also S10-dlubpos.

Definition 4.6.1. A dlubpo D is an *S10-dlubpo* if it fulfills the axiom:
(S10) If I is an index set with a directed partial order,
and $\{y_{ij}\}_{i,j\in I}$ is a two-parametric family of elements of $|D|$ that is monotonic in each

parameter i and j,
and for all $i \in I$ it is $\{y_{ij}\}_{j\in I} \to_D x_i$ for some x_i, and $\{x_i\}_{i\in I} \to_D u$ for some u,
and for all $j \in I$ it is $\{y_{ij}\}_{i\in I} \to_D z_j$ for some z_j, and $\{z_j\}_{j\in I} \to_D u$,
then it is $\{y_{ii}\}_{i\in I} \to_D u$.
S10dlubpo is the full subcategory of **Dlubpo** of all S10-dlubpos.

The axiom looks like a cross-over of axioms S4(1⇒) and S4(2) of natural domains. But note that all the natural convergences in the axiom are directed, by the monotonocity of the family.

Proposition 4.6.2. *The axiom S10 follows in a dlubpo from axioms (S4(1⇒) and S4(2)). It also follows from axioms (S5 and S7(transitivity)).*
So every natural domain is an S10-dlubpo. □

Proposition 4.6.3. *The axioms of an S10-dlubpo (i.e. S3(singleton) and S10) form valid lub-rules and an invariant lub-rule class.*
So every cdlubpo is an S10-dlubpo.

Proof. This is clear for the singleton axiom S3.
An instance of axiom S10 translates to a lub-rule $(D, P \rightsquigarrow A)$ for a partial order D with an I and $\{y_{ij}\}_{i,j\in I}$ in D and where P consists of $\{y_{ij}\}_{j\in I}$ for all $i \in I$ (with lub x_i), $\{x_i\}_{i\in I}$ (with lub u), $\{y_{ij}\}_{i\in I}$ for all $j \in I$ (with lub z_j), $\{z_j\}_{j\in I}$ (with lub u). A is $\{y_{ii}\}_{i\in I}$. u is the lub of A.

Let E be another partial order and $f\colon D \to E$ be monotonic and respecting the lubs of the elements of P. In the same way it can be seen that fu is the lub of f^+A, so the lub-rule $(D, P \rightsquigarrow A)$ is valid. It is also clear that $(E, f^{++}P \rightsquigarrow f^+A)$ is a lub-rule for an instance of axiom S10, so that the lub-rule class is invariant. □

Definition 4.6.4. For every directed partial order $(I, \sqsubseteq)$, that will be used as an index set, we construct a dlubpo $\bar{I} = (|\bar{I}|, \sqsubseteq_{\bar{I}}, \to_{\bar{I}})$ as a completion of I by a lub:
If I contains an upper bound of itself, then this will be called t and $|\bar{I}| = I$ and $\sqsubseteq_{\bar{I}} = \sqsubseteq$.
Otherwise, $|\bar{I}| = I \cup \{t\}$ with a new top element t, and $i \sqsubseteq_{\bar{I}} j$ iff $(i, j \in I$ and $i \sqsubseteq j)$ or $(i \in |\bar{I}|$ and $j = t)$.
In every case, the convergence is $\{x\} \to_{\bar{I}} x$ for $x \in |\bar{I}|$, and $I \to_{\bar{I}} t$.

Lemma 4.6.5. *Let $\mathcal{R}$ be an invariant lub-rule class with singletons.*
Let for every directed partial order I be $I^* = \mathcal{R}\text{-comp}(\bar{I})$.
Let E be a dlubpo generated by $\mathcal{R}$. Then the following are equivalent:

(1) E fulfills axiom S10.

(2) For every dlubpo D it is $\mathrm{eval}\colon (D \Rightarrow E) \times D \to E$ continuous.

(3) For every directed partial order I it is $\mathrm{eval}\colon (I^* \Rightarrow E) \times I^* \to E$ continuous.

(4) For every dlubpo D it is $(D \Rightarrow E) = (D \Rightarrow^* E)$.

Proof. **(1)⇒(2)** First, eval is monotonic. Now let $A \subseteq (D \Rightarrow E) \times D$ be directed with $A \to (f, d)$. We have to show that $\mathrm{eval}^+ A \to fd$. It is $\mathrm{eval}^+ A = \{\, (\pi_1 x)(\pi_2 x) \mid x \in A \,\}$.
Let $I = A$ be the index set for axiom S10. For $i, j \in I$ let $y_{ij} = (\pi_1 i)(\pi_2 j)$.

For $i \in I$ it is $\{y_{ij}\}_{j\in I} \to (\pi_1 i)d =: x_i$, as $\{\pi_2 j\}_{j\in I} \to d$ and as $\pi_1 i$ is continuous.
It is $\{x_i\}_{i\in I} \to fd =: u$, as $\{\pi_1 i\}_{i\in I} \to f$ and therefore $\{\pi_1 i\}_{i\in I}\, d \to fd$.
For $j \in I$ it is $\{y_{ij}\}_{i\in I} \to f(\pi_2 j) =: z_j$, as $\{\pi_1 i\}_{i\in I} \to f$.
It is $\{z_j\}_{j\in I} \to fd$, as $\{\pi_2 j\}_{j\in I} \to d$.
Therefore the conditions of axiom S10 are fulfilled and $\{y_{ii}\}_{i\in I} \to fd$, so $\mathrm{eval}^+ A \to fd$.

(2)⇒(4) For $F \subseteq |D \Rightarrow E|$ directed and $f \in |D \Rightarrow E|$ it is $F \to_{D\Rightarrow E} f$ iff $F \to_{D\Rightarrow^* E} f$.

(4)⇒(3) Follows directly from proposition 4.4.6.

(3)⇒(1) Let I be an index set with a directed partial order, and $\{y_{ij}\}_{i,j\in I}$, $\{x_i\}_{i\in I}$, $\{z_j\}_{j\in I}$ and u in E as the axiom S10 describes.
For $i \in I$ we define functions $f_i\colon \bar{I} \to E$ by $f_i j = y_{ij}$ for $j \in I$, and $f_i t = x_i$.
f_i is continuous, as $\{y_{ij}\}_{j\in I} \to x_i$.
As $\mathcal{R}$ is invariant, f_i is also a continuous function $f_i\colon I^* \to E$, by proposition 4.5.14.
We define $f_t\colon \bar{I} \to E$ by $f_t j = z_j$ for $j \in I$, and $f_t t = u$.
f_t is continuous, as $\{z_j\}_{j\in I} \to u$. Again, f_t is also a continuous function $f_t\colon I^* \to E$.
Let $A \subseteq |I^* \Rightarrow E| \times |I^*|$ with $A = \{\, (f_i, i) \mid i \in I \,\}$. A is directed.
We have $\{f_i\}_{i\in I} \to f_t$, as (a) $\{f_i j\}_{i\in I} \to f_t j$ for $j \in I$, as $\{y_{ij}\}_{i\in I} \to z_j$,
and as (b) $\{f_i t\}_{i\in I} \to f_t t$, as $\{x_i\}_{i\in I} \to u$.
And we have $\{i\}_{i\in I} \to t$, so $A \to (f_t, t)$.
As $\mathrm{eval}\colon (I^* \Rightarrow E) \times I^* \to E$ is continuous, we get $\mathrm{eval}^+ A \to \mathrm{eval}(f_t, t) = u$, so $\{y_{ii}\}_{i\in I} \to u$.
□

As a difference to the general function space $D \Rightarrow^* E$, the curried functions for $D \Rightarrow E$ are continuous:

Proposition 4.6.6. *Let C, D, E be dlubpos. For any $f\colon C \times D \to E$, $\mathrm{curry}(f)\colon C \to (D \Rightarrow E)$, with $\mathrm{curry}(f)c = \lambda d \in D. f(c,d)$, is continuous (and has continuous results).*

Proof. Let $A \subseteq |C|$ be directed with $A \to a$.
Then $\{\, \lambda d \in D. f(x,d) \mid x \in A \,\} \to_{D\Rightarrow E} \lambda d \in D. f(a,d)$, as f is continuous in its first argument. □

Theorem 4.6.7.

(1) Let $\mathcal{R}$ be an invariant lub-rule class with singletons.
*The subcategory $\mathcal{R}$-**cat** of **Dlubpo** is a ccc with pointwise exponents $D \Rightarrow E$*
*$\Longleftrightarrow$ every $E \in \mathcal{R}$-**cat** fulfills axiom S10.*
*($\mathcal{R}$-**cat** also has normal terminal and normal products.)*

*(2) From (1) follows that **S10dlubpo**, and the category of natural domains, and **Cdlubpo** are cccs with the pointwise exponent $D \Rightarrow E$.*

*(3) **S10dlubpo** is the largest full sub-ccc of **Dlubpo** which is generated by an invariant lub-rule class and has the pointwise exponents $D \Rightarrow E$.*

Proof. (1) $\Longrightarrow$: By proposition 4.4.9(1) the evaluation function for the exponent in $\mathcal{R}$-**cat** is the normal eval.
For every directed partial order I it is $I^* = \mathcal{R}\text{-comp}(\bar{I})$ in $\mathcal{R}$-**cat**.

Let $E \in \mathcal{R}$-**cat**. It must be eval: $(I^* \Rightarrow E) \times I^* \to E$ continuous, so by lemma 4.6.5(3⇒1), E fulfills S10.

(1) $\Longleftarrow$: From proposition 4.5.17 follows that $\top$ is in $\mathcal{R}$-**cat** the terminal object, $D \times E$ is in $\mathcal{R}$-**cat** the product, and $D \Rightarrow E$ is in $\mathcal{R}$-**cat**.
That $D \Rightarrow E$ is the exponent in $\mathcal{R}$-**cat** follows from lemma 4.6.5(1⇒2) and proposition 4.6.6.

(2) and (3) follow immediately from (1). □

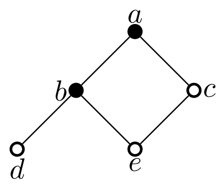
4.7 Example of a natural domain that is no cdlubpo

We give the counter-examples of this section in two stages of increasing complexity. The first example shows that the lub-rule class $\mathcal{S}$ corresponding to the axioms of natural domains is not complete in the sense of Section 4.5. It is simpler, because it refers to general, non-directed natural subsets.

The second example shows a natural domain whose directed natural subsets do not form a cdlubpo. It is more complicated, because it has to refer to *directed* natural subsets. Of course, the second example alone would be enough for all.

Theorem 4.7.1. *The lub-rule class $\mathcal{S}$ corresponding to the axioms of natural domains is not complete. This means that there is a natural domain that does not fulfill axiom S9 (closure) for general (not necessarily directed) subsets.*

Proof. Here is the example, a finite natural domain D:



It has as (non-directed) natural lubs $\{b, c\} \to a$, $\{d, e\} \to b$ and all natural lubs that can be deduced from these by the axioms of natural domains: $\{b, c, d\} \to a$, $\{b, c, e\} \to a$, $\{b, c, d, e\} \to a$ and all trivial natural lubs where the lub is an element of the natural subset.

But by a complete axiom system the further natural lub $\{d, c\} \to a$ could be deduced, because $a \in \mathrm{cl}_{\{\{b,c\},\{d,e\}\}}\{d, c\}$. □

It might still be that the natural domains coincide with the cdlubpos, the closed dlubpos with complete axiom system. But we have the theorem:

Theorem 4.7.2. *There is a natural domain D whose directed natural subsets do not form a cdlubpo.*

Proof. Here is the example, an infinite natural domain D that is no cdlubpo, see figure 4.4.

The elements of D are:
a the top element (i.e. $x \sqsubseteq a$ for all $x \in |D|$).
The elements of $B = \{b_i\}_{i \geq 1}$ with $b_i \sqsubseteq b_j$ for $i \leq j$, it is $B \to a$.
For every $n \geq 1$ there are the elements b_{nw}, where w is a word of numbers ≥ 1 with $1 \leq \mathrm{length}(w) < n$,

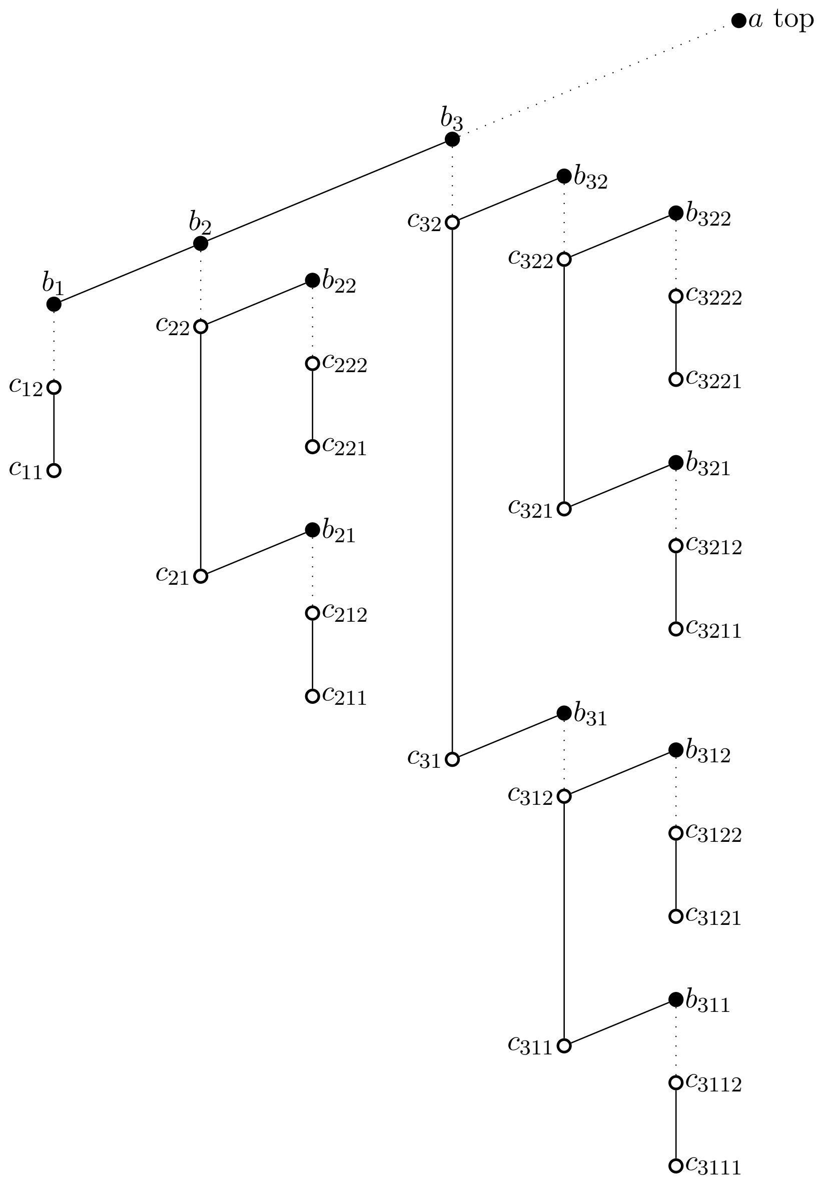


Figure 4.4: natural domain D (partial)

and the elements c_{nw}, where w is a word of numbers ≥ 1 with $1 \leq \text{length}(w) \leq n$.
(Words of numbers will always be denoted by w.)
It is $c_{w1} \sqsubseteq c_{w2} \sqsubseteq \ldots \sqsubseteq b_w$ and $\{c_{w1}, c_{w2}, \ldots\} \to b_w$ and $c_w \sqsubseteq b_w$ for all valid words w.
Let $C = \{\, c_{nw} \mid n \geq 1, \text{length}(w) = n \,\}$, these are the c_{nw} in the rightmost position under each b_n. C is not directed, it has lub a.
Like the example in section 4.3, we erect an "artificial" directed set $\overline{C}$ on C, which is not depicted in the diagram:
Let $\overline{C}$ be the set of all finite subsets of C. All elements of $\overline{C}$ are also elements of D.
For all $A, B \in \overline{C}$: $A \sqsubseteq_D B$ iff $A \subseteq B$, and of course $A \sqsubseteq_D a$.
For all $x \in C$: $x \sqsubseteq_D \{x\} \in \overline{C}$. $\overline{C}$ is directed with lub a.
The natural domain D is the reflexive, transitive closure of the order $\sqsubseteq$ and the closure of $\to$ under the axioms of natural domain.

The intuition of the example is this: To deduce $C \to a$ (and $\overline{C} \to a$) one would start with $B \to a$ and stepwise replace in B elements b_w by $\{c_{wi}\}_{i\geq 1}$ and elements c_w by b_w, working from left to right under each of the b_i, until we reach the elements of C and have replaced all other elements. Under b_i this takes at least $2i-1$ sequential steps, so no deduction can do this for all b_i. (There is no separate deduction of natural lubs b_i. $X \to b_i$ for $X \subseteq \{c_{ij}\}_{j\geq 1}$ infinite are the only non-trivial natural lubs b_i.) Therefore we have:

Proposition 4.7.3. *In D it is not $C \to a$ and not $\overline{C} \to a$.*

Proof. In the following "non-trivial natural lub" means that the natural subset does not contain its own lub. The only non-trivial natural lubs $\neq a$ are the $C' \to b_w$ with a valid word w and where C' contains an infinite subset of $\{c_{wj}\}_{j\geq 1}$. This is so because in such a natural set no c_{wj} can be replaced by b_{wj}, as it is not $b_{wj} \sqsubseteq b_w$.

We will prove that every non-trivial $A \to a$ fulfills this *condition cond*:
There is some $n \geq 0$, and an infinite set N of numbers ≥ 1, and a map $d\colon N \to A$, such that for every $i \in N$ it is $d(i) = b_{iw}$ or $d(i) = c_{iw}$ for some word w with length$(w) \leq n$.

Here is the proof:
Every $A \to a$ must be deduced in a chain $B = A_1 \underset{\leftarrow}{\sqsubseteq} A_2 \underset{\leftarrow}{\sqsubseteq} \dots \underset{\leftarrow}{\sqsubseteq} A_m = A$, where $A_i \to a$ is non-trivial. See the definition 4.3.7 of $\underset{\leftarrow}{\sqsubseteq}$ and proposition 4.3.8 and note that a is the top element.
B fulfills the condition cond. We have to prove that if A fulfills the condition cond, then also A' with $A \underset{\leftarrow}{\sqsubseteq} A'$.

A step $A \underset{\leftarrow}{\sqsubseteq} A'$ means the following:

(1) For every $b_i \in A$ (there is some $b_j \in A'$ with $j \geq i$) or (there is $C' \subseteq A'$ with an infinite $C' \subseteq \{c_{ij}\}_{j\geq 1}$).

(2) For every $b_{iw} \in A$, with length$(w) \geq 1$, (there is $b_{iw} \in A'$) or (there is $C' \subseteq A'$ with an infinite $C' \subseteq \{c_{iwj}\}_{j\geq 1}$).

(3) For every $c_{ij} \in A$, (there is some $c_{ik} \in A'$ with $k \geq j$) or (there is some $b_k \in A'$ with $k \geq i$) or $b_{ij} \in A'$).

(4) For every $c_{iwj} \in A$, with $1 \leq$ length$(w) \leq i-2$, (there is some $c_{iwk} \in A'$ with $k \geq j$) or ($b_{iw} \in A'$) or ($b_{iwj} \in A'$).

(5) For every $c_{iwj} \in A$, with length$(w) = i-1$, (there is some $c_{iwk} \in A'$ with $k \geq j$) or ($b_{iw} \in A'$) or (there is some $F \in A'$ with $F \in \overline{C}$ and $c_{iwj} \in F$).

(6) For every $F \in A$, with $F \in \overline{C}$, there is $F' \in A$ with $F' \in \overline{C}$ and $F \sqsubseteq F'$.

We prove that A' fulfills the condition cond. There are two cases:
(a) $A' \cap B$ is infinite:
Then A' fulfills the condition cond with $n' = 0$, $N' = \{\, i \mid b_i \in A' \cap B \,\}$, $d'(i) = b_i$.

(b) otherwise:
Then A' fulfills the condition cond with the following data:
$n' = n + 1$.
N' is the set of all $i \in N$ with $i \geq n + 1$ that fulfill the following two conditions:

- If $d(i) = b_i$, then $b_i \in A'$ or some $c_{ij} \in A'$.
- If $d(i) = c_{ij}$, then (some $c_{ik} \in A'$ with $k \geq j$) or ($b_{ij} \in A'$) or ($b_i \in A'$).

(These conditions rule out the cases of i where b_i or c_{ij} is replaced in A' only by a b_k with $k > i$. N' is still infinite as the $i \in N$ with $i \geq n + 1$ are infinitely many and the ruled out cases of i can only be finitely many, otherwise we would have case (a).)

$d'(i)$ is defined for $i \in N'$ as follows:

- If $d(i) = b_{iw}$: if $b_{iw} \in A'$ then $d'(i) = b_{iw}$ else $d'(i) = c_{iwj}$ for some j such that $c_{iwj} \in A'$.
- If $d(i) = c_{iwj}$: if there is some $c_{iwk} \in A'$ then $d'(i) = c_{iwk}$
  else (if $b_{iw} \in A'$ then $d'(i) = b_{iw}$ else $d'(i) = b_{iwj}$).
  (Here it is $i \geq n + 1$ and $\text{length}(wj) \leq n$, so $\text{length}(w) < i - 1$. Therefore c_{iwj} is not replaced in A' by some $F \in \overline{C}$.)

It is clear from the meaning of the step $A \underset{\leftarrow}{\sqsubseteq} A'$ above and the choice of N' that $d'(i)$ is an element of A'. The word length of the index of $d'(i)$ stays the same as for $d(i)$ or increases by 1.

So we have proved that every non-trivial $A \to a$ fulfills the condition cond. This proves that $C \to a$ and $\overline{C} \to a$ are not fulfilled in D. □

Proposition 4.7.4. *From the specified natural lubs of D it can be deduced by the closure axiom S9 that $\overline{C} \to a$.*

Proof. Let $C^* = \text{cl}_P(\overline{C})$, where P is the specified set of natural sets of D. We have to show that $a \in C^*$.
First, for every $c \in C$ it is $c \in C^*$, as $c \sqsubseteq_D \{c\} \in \overline{C}$.
For all $n \geq 1$, we now work under b_n:
We have that all $c_{nw} \in C^*$ with $\text{length}(w) = n$, these are the elements from C.
If for some $1 \leq k \leq n$, all $c_{nw} \in C^*$ with $\text{length}(w) = k$, then all $b_{nw} \in C^*$ with $\text{length}(w) = k - 1$.
If for some $1 \leq k \leq n - 1$, all $b_{nw} \in C^*$ with $\text{length}(w) = k$, then all $c_{nw} \in C^*$ with $\text{length}(w) = k$.
As a result of this consecutive process, we get $b_n \in C^*$ for all $n \geq 1$, so $a \in C^*$. □

From the two propositions it is immediate that D is not a cdlubpo. □

Remark 4.7.5. Note that D of the theorem is not a (naturally) algebraic natural domain in the sense of definition 3.2 of [Saz09]. In the next section we show that every algebraic natural domain is a cdlubpo.

Remark 4.7.6. Another question is that for an augmentation of the axioms of natural domains so that they describe cdlubpos. This could be a modification of axiom S8 (under) so that the "limit" L of an ascending $\underset{\leftarrow}{\sqsubseteq}$-chain $\sqsubseteq a$ leads to a natural lub $L \to a$. But this would need a complicated definition of "limit", so we prefer our axiom S9 (closure) instead.

4.8 Algebraic dlubpos

We adapt the definitions of (naturally) finite elements and (naturally) algebraic natural domains from Sazonov [Saz09] to the case of dlubpos. We show that an algebraic dlubpo that fulfills axiom S6 (cofinality) is a cdlubpo. This proves that (naturally) algebraic natural domains and algebraic cdlubpos are the same.

Definition 4.8.1 (Sazonov, definitions 3.1, 3.2 of [Saz09])**.** Let D be a dlubpo.
An element $a \in |D|$ is *finite* if for every (directed) $B \to b$ with $a \sqsubseteq b$ there is $b' \in B$ with $a \sqsubseteq b'$. D^0 is the set of finite elements of D.
For $d \in |D|$ we define $\downarrow^0 d = \{\, a \in D^0 \mid a \sqsubseteq d \,\}$.
D is *algebraic* if for every $d \in |D|$ it is $\downarrow^0 d \to d$. (This includes that $\downarrow^0 d$ is directed.)

Note that we omit Sazonov's prefix "naturally" (finite, algebraic) as this is given by the fact that we are talking about dlubpos.

The next lemma shows that the finiteness property extends to closures.

Lemma 4.8.2. *Let D be a dlubpo.*
Let $A \subseteq |D|$, $a \in D^0$ and $a \in \mathrm{cl}_D\, A$. Then there is some $a' \in A$ with $a \sqsubseteq a'$.

Proof. Let $(N, r, \mathit{lab}, \mathit{pre})$ be a deduction of a from A in the cl-rule system. We construct a path $r = n_1, n_2 \in \mathit{pre}\, n_1, n_3 \in \mathit{pre}\, n_2, \ldots, n_k$ from the root r to a leaf n_k inductively as follows. It is always $a \sqsubseteq \mathit{lab}\, n_i$.
If $\mathit{lab}\, n_i = b$ and $\mathit{pre}\, n_i = \{m\}$ with $b \sqsubseteq \mathit{lab}\, m$, then we choose $n_{i+1} = m$.
If $\mathit{lab}\, n_i = b$ and $\mathit{pre}\, n_i = M$ with $\mathit{lab}^+\, M \to b$, then there is some $m \in M$ with $a \sqsubseteq \mathit{lab}\, m$, as a is finite. We choose $n_{i+1} = m$.
By well-foundedness of the deduction, this process ends with a leaf n_k with $\mathit{lab}\, n_k = a' \in A$ and $a \sqsubseteq a'$. ◻

There is an interesting property of dlubpos connected to algebraicity:

Definition 4.8.3. Let D be a dlubpo. D is *finite-determined* if there is a subset $F \subseteq |D|$ such that for every directed $A \subseteq |D|$ with lub a it is:
$A \to a \iff$ for all $b \in F$ with $b \sqsubseteq a$ there is some $a' \in A$ with $b \sqsubseteq a'$.

The elements of F are finite in the sense of definition 4.8.1. Every algebraic dlubpo D fulfills the direction $\Longrightarrow$ with $F = D^0$. But a finite-determined D need not be algebraic, as it is not stipulated that every element a of D is a (natural) lub of the directed set of finite elements below a. On the other side our definition demands more, namely the direction $\Longleftarrow$, which is not entailed by algebraic dlubpos.

Proposition 4.8.4. *If D is a finite-determined dlubpo and for every $a \in |D|$ it is $a = \bigsqcup {\downarrow}^0 a$ directed, then D is algebraic.* □

Remember that the axiom S6 (cofinality) for directed subsets is:
If $X, Y \subseteq |D|$ directed, $X \to x$ and $X \sqsubseteq Y \sqsubseteq x$, then $Y \to x$.

Proposition 4.8.5 (Reinhold Heckmann, personal communication)**.**
If D is an algebraic dlubpo with axiom S6 for directed subsets, then D is finite-determined with $F = D^0$.

Proof. We must prove the direction $\Longleftarrow$ of the definition.
The right side means: ${\downarrow}^0 a \sqsubseteq A$. It is $A \sqsubseteq a$ and ${\downarrow}^0 a \to a$.
By axiom S6 we get $A \to a$. □

Proposition 4.8.6. *If D is a finite-determined dlubpo, then D is a cdlubpo.*

Proof. Let D be finite-determined by the subset F.
Let $A \subseteq |D|$ be directed with lub a and $a \in \mathrm{cl}_D\, A$. We have to show $A \to a$.
Let $b \in F$ with $b \sqsubseteq a$. Then $b \in D^0$ and $b \in \mathrm{cl}_D\, A$.
By lemma 4.8.2 there is some $a' \in A$ with $b \sqsubseteq a'$. □

Theorem 4.8.7.

(1) If D is an algebraic dlubpo with axiom S6 (cofinality) for directed subsets, then D is a cdlubpo.

(2) For any dlubpo D the following are equivalent:
   (a) D is an algebraic dlubpo that fulfills axiom S6 for directed subsets.
   (b) D is an algebraic cdlubpo.
   (c) D is an algebraic natural domain.

Proof. (1) follows from propositions 4.8.5 and 4.8.6.
(2) is clear. □

4.9 Restricted partial orders and restricted dcpos

We want to establish cdlubpos as the most general canonical structures in which to build non-complete programming language models, corresponding to dcpos as the most general structures to build complete models. (These structures still lack algebraicity/continuity.) We have already seen that cdlubpos are just the dlubpos that are generated by a complete lub-rule class. In this section we see another characterization of cdlubpos that shows their canonicity: they are just the dlubpos that are realized by "restricted partial orders" (rpos), in which they are embedded. This embedding can also be in a "restricted dcpo" (rdcpo), which is a dcpo with a subset designated as the set of "proper" elements.

But before describing the rdcpos we give a property that is sufficient for dlubpos to be cdlubpos.

Proposition 4.9.1. *Let D be a dlubpo and K be a subclass of the class of all dlubpo morphisms with domain D. (The morphisms of K could be given by a subcategory in which D lives.)*
We say that "the natural lubs of D are determined by K"
if for all directed $A \subseteq |D|$ with lub a it is:
if all $f\colon D \to E$ in K respect the lub of A (i.e. $fa = \bigsqcup f^+ A$), then $A \to_D a$.
In this case D is a cdlubpo.

Proof. Let $((|D|, \sqsubseteq_D), P {\leadsto} A)$ be a valid directed lub-rule and let all elements of P be natural in D. All $f\colon D \to E$ in K respect the lubs of the elements of P, so also respect the lub of A, as the lub-rule is valid.
Therefore A is natural in D by the definition, so D fulfills the lub-rule. Thus D is a cdlubpo. □

We now come to the realization of cdlubpos by restricted partial orders and restricted dcpos.

Definition 4.9.2. A structure $E = (|E|, E^{\updownarrow}, \sqsubseteq_E)$ is a *restricted partial order (rpo)* if
$|E|$ is a set of (proper) *elements*,
$E^{\updownarrow}$ is a set of *realizers*, with $|E| \subseteq E^{\updownarrow}$,
$\sqsubseteq_E$ is a partial order on $E^{\updownarrow}$.
We define $E^{\uparrow} = E^{\updownarrow} \setminus |E|$, the set of *blind realizers.*

The idea of a partial order realized by a cpo appears in [Sim95], but there every realizer realizes exactly one proper element (there are no blind realizers), and a proper element may be realized by several realizers.

Definition 4.9.3. An rpo E *realizes* a dlubpo D *by* an order isomorphism $\varphi\colon (|D|, \sqsubseteq_D) \to (|E|, \sqsubseteq_E)$ if for all directed $\bigsqcup A = a$ in D:

$$A \to_D a \Longleftrightarrow \varphi a = \bigsqcup \varphi^+ A.$$

Proposition 4.9.4.

(1) If an rpo E realizes a dlubpo D, then D is a cdlubpo.

(2) Every cdlubpo D is realized by the rpo $(\mathrm{in}^+ |D|, \hat{D}, \sqsubseteq)$ by $\mathrm{in}\colon |D| \to \mathrm{in}^+ |D|$, see def. 4.5.4.

(3) The cdlubpos are exactly the dlubpos realized by rpos.

Proof. **(1)** This is essentially the proof of proposition 4.9.1, when we construe E as a dlubpo with all directed lubs natural.
Let φ be the embedding of D in E. Let $((|D|, \sqsubseteq), P {\leadsto} A)$ be a valid directed lub-rule such that the elements of P are natural in D. φ respects the lubs of the elements of P. As the lub-rule is valid, φ respects also the lub of A, so $\varphi(\bigsqcup A) = \bigsqcup \varphi^+ A$.
As E realizes D by φ, it is $A \to \bigsqcup A$ in D. So D fulfills all valid directed lub-rules, it is a cdlubpo.
(2) Let $A \to a$ in D. Then $\mathrm{in}_D\, a = \bigsqcup(\mathrm{in}_D^+ A) = \mathrm{cl}_D\, A$ by proposition 4.5.5.
For the reverse: It is $a \in \mathrm{cl}_D\, A$, therefore $A \to a$. □

We want to realize our dlubpos in dcpos which can be constructed by a completion process. We define the category of restricted dcpos:

Definition 4.9.5. A structure $E = (|E|, E^{\updownarrow}, \sqsubseteq_E)$ is a *restricted dcpo (rdcpo)* if it is an rpo and $(E^{\updownarrow}, \sqsubseteq_E)$ is a dcpo.
Let D, E be rdcpos. A function $f \colon D^{\updownarrow} \to E^{\updownarrow}$ is *continuous* if it is continuous on the dcpos $(D^{\updownarrow}, \sqsubseteq_D)$ and $(E^{\updownarrow}, \sqsubseteq_E)$, and $f^{+}|D| \subseteq |E|$. In this case we write $f \colon D \to E$.
Rdcpo is the category of all rdcpos and continuous functions, with normal function composition and identity functions.

Proposition 4.9.6. *The cdlubpos are exactly the dlubpos realized by rdcpos.*

Proof. Follows from proposition 4.9.4, as $\hat{D}$ is a dcpo. □

There is an adjunction between the categories **Dlubpo** and **Rdcpo**, which establishes an adjoint equivalence between **Cdlubpo** and a full subcategory **Crdcpo** (closed rdcpos) of **Rdcpo**. The details and further developments in this direction will be found in a sequel paper.

4.10 Outlook

We have introduced lub-rule classes and closed dlubpos corresponding to complete lub-rule classes as a canonical alternative to natural domains. Closed dlubpos can also be characterized as the dlubpos realized by restricted dcpos. We will explore this connection with rdcpos further in a sequel paper. It will turn out that there is an adjunction between the categories which permits the transfer of much of the theory of cccs of algebraic dcpos to closed rdcpos resp. closed dlubpos.

The different approaches to realize partial orders by dcpos, Simpson's [Sim95] and ours, call for a common generalization: in the new concept a proper element may be realized by *several* realizers (Simpson), and a realizer may realize one or none element (as here). This would be a category of dcpos that carry also some kind of *partial* preorder, the correct definition of it is not yet clear. (Simpson has a preorder.)

We indicate further directions of research. The question of the topology of non-complete domains is not yet fully answered, Sazonov made some first observations on this in [Saz09].

In the introduction we posed as an open problem to find categories of abstract incomplete domains with the existence of fixpoints of endofunctions. Also interesting is the question to find such categories of concrete incomplete domains, i.e. domains based on mechanisms like games.

Bibliography

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

[AC98] Roberto M. Amadio and Pierre-Louis Curien. *Domains and Lambda-Calculi.* Cambridge University Press, 1998.

[AJM00] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Information and Computation*, 163:409–470, 2000.

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

[Bar90] F. Barbanera. Combining term rewriting and type assignment systems. *Int. J. of Foundations of Computer Science*, 1(3):165–184, 1990.

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

[BCL85] Gérard Berry, Pierre-Louis Curien, and Jean-Jacques Levy. Full abstraction for sequential languages: The state of the art. In Maurice Nivat and John Reynolds, editors, *Algebraic Methods in Semantics*, pages 59–132. Cambridge University Press, 1985.

[Ber78] Gérard Berry. Stable models of typed λ-calculi. In *5. ICALP'78, LNCS 62*, pages 72–89. Springer, 1978.

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

[Bet03] Inge Bethke. Lambda calculus. In Terese, editor, *Term Rewriting Systems*, pages 548–587. Cambridge University Press, 2003.

[BT88] V. Breazu-Tannen. Combining algebra and higher-order types. In *Proc. Logic in Computer Science*, pages 82–90. IEEE, 1988.

[BTG89] V. Breazu-Tannen and J. Gallier. Polymorphic rewriting conserves algebraic strong normalization and confluence. In *ICALP'89, LNCS 372*, pages 137–150. Springer, 1989.

[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.

[CPW00] Pierre-Louis Curien, Gordon Plotkin, and Glynn Winskel. Bistructures, bidomains and linear logic. In Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language, and Interaction. Essays in Honour of Robin Milner.* MIT Press, 2000.

[CR80] Bruno Courcelle and Jean-Claude Raoult. Completions of ordered magmas. *Fundamenta Informaticae*, 3:105–118, 1980.

[Dou91] D. J. Dougherty. Adding algebraic rewriting to the untyped lambda calculus. In R. V. Book, editor, *Rewriting Techniques and Applications, 4th RTA-91, LNCS 488*, pages 37–48. Springer, 1991.

[EH09] Martin Escardó and Weng Kin Ho. Operational domain theory and topology of sequential programming. *Information and Computation*, 207:411–437, 2009.

[Gir86] Jean-Yves Girard. The system F of variable types, fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

[GTL89] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types.* Cambridge University Press, 1989.

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

[Har15] Fritz Hartogs. Über das Problem der Wohlordnung. *Mathematische Annalen*, 76:438–443, 1915.

[HO00] J. Martin E. Hyland and C.-H. Luke Ong. On full abstraction for PCF. *Information and Computation*, 163:285–408, 2000.

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

[JO91] J.-P. Jouannaud and M. Okada. A computation model for executable higher-order algebraic specification languages. In *Proc. Logic in Computer Science*, pages 350–361. IEEE, 1991.

[Joh87] Peter T. Johnstone. *Notes on Logic and Set Theory.* Cambridge University Press, 1987.

[Klo80] J. W. Klop. *Combinatory Reduction Systems.* Mathematical Centre Tracts 127. Mathematisch Centrum (CWI), Amsterdam, 1980.

[Lai05] Jim Laird. Sequentiality in bounded biorders. *Fundamenta Informaticae*, 65:173–191, 2005.

[Lai07] Jim Laird. On the expressiveness of affine programs with non-local control: The elimination of nesting in SPCF. *Fundamenta Informaticae*, 77:511–531, 2007.

[Lan98] Saunders Mac Lane. *Categories for the Working Mathematician.* Springer, 2nd edition 1998.

[LN15] John Longley and Dag Normann. *Higher-Order Computability.* Springer, 2015.

[Loa01] Ralph Loader. Finitary PCF is not decidable. *Theoretical Computer Science*, 266:341–364, 2001.

[Lon02] John Longley. The sequentially realizable functionals. *Annals of Pure and Applied Logic*, 117:1–93, 2002.

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

[Mil75] Robin Milner. Processes: A mathematical model of computing agents. In H.E. Rose and J.C. Sheperdson, editors, *Logic Colloquium '73*, pages 157–173. North-Holland, 1975.

[Mil77] Robin Milner. Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4:1–22, 1977.

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

[Mül92] Fritz Müller. Confluence of the lambda calculus with left-linear algebraic rewriting. *Information Processing Letters*, 41:293–299, 1992. Please use the electronic version: `http://www.rw.cdl.uni-saarland.de/~mueller/lconfluence.ps.gz`.

[Mül12] Fritz Müller. On Berry's conjectures about the stable order in PCF. *Logical Methods in Computer Science*, 8(4:7):1–39, 2012.

[Nic94] Hanno Nickau. Hereditarily sequential functionals. In *Logical Foundations of Computer Science, LNCS 813*, pages 253–264. Springer, 1994.

[Nip91] T. Nipkow. Higher-order critical pairs. In *Proc. Logic in Computer Science*, pages 342–349. IEEE, 1991.

[Nor06] Dag Normann. On sequential functionals of type 3. *Mathematical Structures in Computer Science*, 16:279–289, 2006.

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

[NS12] Dag Normann and Vladimir Yu. Sazonov. The extensional ordering of the sequential functionals. *Annals of Pure and Applied Logic*, 163:575–603, 2012.

[Ong95] C.-H. Luke Ong. Correspondence between operational and denotational semantics: the full abstraction problem for PCF. In S. Abramsky, D. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science, Volume 4*, pages 269–356. Oxford University Press, 1995.

[OR95] Peter W. O'Hearn and Jon G. Riecke. Kripke logical relations and PCF. *Information and Computation*, 120:107–116, 1995.

[Pao06] Luca Paolini. A stable programming language. *Information and Computation*, 204:339–375, 2006.

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

[Ros73] B. K. Rosen. Tree-manipulating systems and Church-Rosser theorems. *J. of the ACM*, 20(1):160–187, 1973.

[RV80] J.-C. Raoult and J. Vuillemin. Operational and semantic equivalence between recursive programs. *J. of the ACM*, 27(4):772–796, 1980.

[Saz07] Vladimir Sazonov. An inductive definition and domain theoretic properties of fully abstract models for PCF and PCF+. *Logical Methods in Computer Science*, 3:1–50, 2007.

[Saz09] Vladimir Sazonov. Natural non-dcpo domains and f-spaces. *Annals of Pure and Applied Logic*, 159:341–355, 2009.

[Sco93] Dana S. Scott. A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121:411–440, 1993. Originally written and distributed in 1969.

[Sie92] Kurt Sieber. Reasoning about sequential functions via logical relations. In *Applications of Categories in Computer Science*. Cambridge University Press, 1992.

[Sim95] Alex K. Simpson. The convex powerdomain in a category of posets realized by CPOs. In D. Pitt, D. E. Rydeheard, and P. Johnstone, editors, *Category Theory and Computer Science, LNCS 953*, pages 117–145. Springer, 1995.

[Smy83] Michael B. Smyth. The largest cartesian closed category of domains. *Theoretical Computer Science*, 27:109–119, 1983.

[Sto88] Allen Stoughton. Substitution revisited. *Theoretical Computer Science*, 59:317–325, 1988.

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

[Sto94] Allen Stoughton. Mechanizing logical relations. In *Mathematical Foundations of Programming Semantics 1993, LNCS 802*, pages 359–377. Springer, 1994.

[Tay90] Paul Taylor. An algebraic approach to stable domains. *Journal of Pure and Applied Algebra*, 64:171–203, 1990.

[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.

[vOvR94] Vincent van Oostrom and Femke van Raamsdonk. Weak orthogonality implies confluence: the higher-order case. In *Logical Foundations of Computer Science, LNCS 813*, pages 379–392. Springer, 1994.

[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.

[Yan10] Noson S. Yanofsky. Towards a definition of an algorithm. *Journal of Logic and Computation*, 2010. arXiv:0602053v3.