[13] Hardin, T., and Lévy, J.-J. A confluent calculus of substitutions. Tech. Rep. 90-11, C.N.A.M., 292 Rue Saint-Martin, 75141 Paris Cedex 03, France, 1990.

[14] Hindley, J. R., and Seldin, J. P. *Introduction to Combinators and λ-Calculus*. Cambridge University Press, 1986.

[15] Hindley, R. Combinatory reductions and lambda reductions compared. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik 23* (1977).

[16] Hindley, R., and Longo, G. Lambda-calculus models and extensionality. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik 26* (1980).

[17] Jäger, G. Induction in the elementary theory of types and names. In *CSL '87*, E. Börger et al., Ed. Springer, 1988. Proceedings of the 1st Workshop on Computer Science Logic, Karlsruhe, October 12-16, 1987. Lecture Notes in Computer Science 329.

[18] Jäger, G. Type theory and explicit mathematics. In *Logic Colloquium '87*, H.-D. Ebbinghaus, J. Fernandez-Prida, M. Garrido, M. Lascar, and M. R. Artalejo, Eds. North Holland, Amsterdam, 1989.

[19] Martin-Löf, P. Substitition calculus. Unpublished notes, September 1992.

[20] Marzetta, M. A theory of types and names. Contributed paper, Logic Colloquium, Padova 23-30 August 1988.

[21] Marzetta, M. Universes in the theory of types and names. In *CSL '92*, E. Börger et al., Ed. Springer, 1993. Proceedings of the 6th Workshop on Computer Science Logic, San Miniato (Pisa), September 28-October 2, 1992. Lecture Notes in Computer Science ?, to appear.

[22] Moggi, E. *The Partial Lambda Calculus*. PhD thesis, University of Edinburgh, 1988.

[23] Newman, M. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics 43* (1942).

[24] Troelstra, A., and van Dalen, D. *Constructivism in Mathematics*, vol. II. North-Holland, Amsterdam, New York, 1988.

[25] Troelstra, A., and van Dalen, D. *Constructivism in Mathematics*, vol. I. North-Holland, Amsterdam, New York, 1988.

References

[1] Abadi, M., Cardelli, L., Curien, P.-L., and Lévy, J.-J. Explicit substitutions. *Journal of Functional Programming 1*, 4 (1991).

[2] Barendregt, H. P. *The Lambda Calculus*, revised ed. North Holland, 1985.

[3] Beeson, M. J. *Foundations of Constructive Mathematics: Metamathematical Studies*. Springer, Berlin, 1984.

[4] Beeson, M. J. Proving programs and programming proofs. In *Logic, Methodology and Philosophy of Science VII*. North Holland, Amsterdam, 1985.

[5] Curien, P.-L., Hardin, T., and Lévy, J.-J. Confluence properties of weak and strong calculi of explicit substitutions. Tech. Rep. 92-01, C.N.A.M., 292 Rue Saint-Martin, 75141 Paris Cedex 03, France, 1992. Submitted to J.A.C.M.

[6] Feferman, S. A language and axioms for explicit mathematics. In *Algebra and Logic*, J. Crossley, Ed. Springer, Berlin, 1975. Lecture Notes in Mathematics 450.

[7] Feferman, S. Constructive theories of functions and classes. In *Logic Colloquium '78*, M. Boffa, D. van Dalen, and K. McAloon, Eds. North Holland, Amsterdam, 1979.

[8] Feferman, S. Polymorphic typed lambda-calculi in a type-free axiomatic framework. In *Logic and Computation*, W. Sieg, Ed. American Mathematical Society, Providence, 1990. Contemporary Mathematics 106.

[9] Feferman, S. Logics for termination and correctness of functional programs. In *Logic from Computer Science*, Y. N. Moschovakis, Ed. Springer, 1991. Proceedings of a MSRI Workshop held at Berkley, November 13-17, 1989.

[10] Feferman, S. Logics for termination and correctness of functional programs II: Logics of strength PRA. In *Proof Theory*, P. Aczel, H. Simmons, and S. Wainer, Eds. Cambridge University Press, 1992. A selection of papers from the Leeds Proof Theory Programme 1990.

[11] Feferman, S., and Jäger, G. Systems of explicit mathematics with non-constructive μ-operator. Part I. To appear in *Annals of Pure and Applied Logic*.

[12] Hardin, T. Confluence results for the pure strong categorical logic CCL: λ–calculi as subsystems of CCL. *Theoretical Computer Science 65* (1989).

5 Conclusion

We have addressed some defects of the partial λ calculus $\boldsymbol{\lambda}_p$ as a constructive framework for partial functions, and we have proposed a modification $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ of $\boldsymbol{\lambda}_p$ by explicit substitutions. The system $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ is embeddable into partial combinatory logic $\mathbf{CL}_p$ and therefore inherits all the models of the latter. We have studied a reduction relation for $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ and we have established a confluence result. The reduction relation gives rise to direct constructions of term models for $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$. The detailed constructions will be discussed later.

As already mentioned, the theory of explicit substitutions has been treated before, primarily in connection with the implementation of functional programming languages. The main reference on weak calculi of explicit substitution is [5]. In contrast to our approach, not only equality between terms, but also equality between substitutions has been axiomatized in the previous work on explicit substitution. Also this can easily be achieved, the systems have much more axioms, and we think that – especially from a foundational point of view – the real concern is to axiomatize and control the notion of a substitution applied to a term, whereas equality between substitutions can be treated in the metalanguage. Furthermore, the previous systems of explicit substitutions are mainly term rewriting systems, and they do not include partiality. The main purpose of our work, however, was to study questions of substitution in the context of partiality and to design a more perspicuous version of the partial λ calculus, which has natural partial models and is equivalent to partial combinatory logic.

The question arises why one should at all use a fairly complicated system like $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ instead of the formally more simple partial combinatory logic $\mathbf{CL}_p$. We think that $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ has advantages over $\mathbf{CL}_p$. The main reason is that in the system $\mathbf{CL}_p$, the intuitive clearity of the λ notation is completely lost. Additionally, many mistakes in the literature concerning substitution in $\mathbf{CL}_p$ suggest that it is also worth having an explicit treatment of substitution as in $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$. We, therefore, think that $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ can serve as an adequate applicative basis for systems of explicit mathematics.

Acknowledgments

I am indebted to R. Kahle, who pointed out to me that the principle $(\star\star)$ is derivable in $\boldsymbol{\lambda}_p$ though it obviously does not hold in the recursion-theoretic model. I also thank A. S. Troelstra for calling my attention to explicit substitutions as well as S. Feferman, T. Hardin, G. Jäger and R. Stärk for helpful comments on an earlier version of this paper.

Corollary 44 *We have for all terms t and s:*

$$t \rhd^1_\beta s \quad \Longrightarrow \quad \Sigma(t) \rhd_{\beta_n} \Sigma(s).$$

Corollary 45 *We have for all terms t and s:*

$$t \rhd s \quad \Longrightarrow \quad \Sigma(t) \rhd_{\beta_n} \Sigma(s).$$

PROOF Assume $t \rhd s$. Then there is a sequence of terms $t_1, \ldots, t_n$ so that $t_1 = t, t_n = s$ and for all $1 \leq i < n$

$$t_i \rhd^1_\sigma t_{i+1} \quad \text{or} \quad t_i \rhd^1_\beta t_{i+1}.$$

In the first case we have $\Sigma(t_i) = \Sigma(t_{i+1})$ and in the latter, by the previous corollary, $\Sigma(t_i) \rhd_{\beta_n} \Sigma(t_{i+1})$. Alltogether we immediately get $\Sigma(t) \rhd_{\beta_n} \Sigma(s)$, since $\rhd_{\beta_n}$ is transitively closed. $\Box$

We are ready to prove the confluence of $\rhd$.

Theorem 46 $\rhd$ *is Church Rosser.*

PROOF Let t, t_1 and t_2 be $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ terms and assume that

$$t \rhd t_1 \quad \text{and} \quad t \rhd t_2. \tag{1}$$

Then the previous corollary immediately implies

$$\Sigma(t) \rhd_{\beta_n} \Sigma(t_1) \quad \text{and} \quad \Sigma(t) \rhd_{\beta_n} \Sigma(t_2). \tag{2}$$

By Theorem 41 we know that $\rhd_{\beta_n}$ is confluent, hence there is a $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ term s in substitution normal form satisfying

$$\Sigma(t_1) \rhd_{\beta_n} s \quad \text{and} \quad \Sigma(t_2) \rhd_{\beta_n} s. \tag{3}$$

Since $\rhd_{\beta_n} \subset \rhd$, we get from (3)

$$\Sigma(t_1) \rhd s \quad \text{and} \quad \Sigma(t_2) \rhd s, \tag{4}$$

which immediately implies

$$t_1 \rhd \Sigma(t_1) \rhd s \quad \text{and} \quad t_2 \rhd \Sigma(t_2) \rhd s. \tag{5}$$

The claim is established. $\Box$

(4.3) t' is the term $(t'_1 t'_2)$. According to $t'\theta \overset{\beta}{\longmapsto} s$, we can distinguish the following five subcases:

(4.3.1) $t'\theta \overset{\beta}{\longmapsto} s$ is $(t'_1 t'_2)\theta \overset{\beta}{\longmapsto} (t''_1 t'_2)\theta$ and is a consequence of $t'_1 \overset{\beta}{\longmapsto} t''_1$. Then it is $(t'_1 t'_2)\theta \rhd^1_\sigma (t'_1\theta)(t'_2\theta)$ and by the main induction hypothesis we have

$$\Sigma((t'_1 t'_2)\theta) = \Sigma((t'_1\theta)(t'_2\theta)) \rhd_{\beta_n} \Sigma((t''_1\theta)(t'_2\theta)) = \Sigma((t''_1 t'_2)\theta).$$

(4.3.2) $t'\theta \overset{\beta}{\longmapsto} s$ is $(t'_1 t'_2)\theta \overset{\beta}{\longmapsto} (t'_1 t''_2)\theta$ and is a consequence of $t'_2 \overset{\beta}{\longmapsto} t''_2$. This case is treated in the same way as (4.3.1).

(4.3.3) $t'\theta \overset{\beta}{\longmapsto} s$ is $(t'_1 t'_2)\theta \overset{\beta}{\longmapsto} (t'_1 t'_2)\theta'$ and is a consequence of $\theta \overset{\beta}{\longmapsto} \theta'$. Then it is $(t'_1 t'_2)\theta \rhd^1_\sigma (t'_1\theta)(t'_2\theta)$ and by the main induction hypothesis we have $\Sigma(t'_1\theta) \rhd_{\beta_n} \Sigma(t'_1\theta')$ and $\Sigma(t'_2\theta) \rhd_{\beta_n} \Sigma(t'_2\theta')$. This implies

$$\Sigma((t'_1 t'_2)\theta) = \Sigma((t'_1\theta)(t'_2\theta)) = \Sigma(t'_1\theta)\Sigma(t'_2\theta) \rhd_{\beta_n} \Sigma(t'_1\theta')\Sigma(t'_2\theta') = \Sigma((t'_1 t'_2)\theta').$$

(4.3.4) $t'\theta \overset{\beta}{\longmapsto} s$ is $((\lambda x.r)r')\theta \overset{\beta}{\longmapsto} r\{r'/x\}\theta$. First assume that $fvar(\lambda x.r) \cap dom\,\theta \neq \emptyset$. Then we have for $\theta' := \theta^{\lambda x.r}$

$$\Sigma(((\lambda x.r)r')\theta) = (\lambda x.r)\Sigma(\theta')\Sigma(r'\theta) \rhd_{\beta_n} \Sigma(r(\Sigma(r'\theta)/x \cdot \Sigma(\theta'))) = \Sigma(r\{r'/x\}\theta).$$

The case $fvar(\lambda x.r) \cap dom\,\theta = \emptyset$ is treated in a similar way.

(4.3.5) $t'\theta \overset{\beta}{\longmapsto} s$ is $((\lambda x.r)\sigma r')\theta \overset{\beta}{\longmapsto} r(r'/x \cdot \sigma)\theta$. First assume that $fvar(\lambda x.r) \cap dom(\sigma\theta) \neq \emptyset$. Then we have for $\rho := (\sigma\theta)^{\lambda x.r}$

$$\Sigma(((\lambda x.r)\sigma r')\theta) = (\lambda x.r)\Sigma(\rho)\Sigma(r'\theta) \rhd_{\beta_n} \Sigma(r(\Sigma(r'\theta)/x \cdot \Sigma(\rho))) = \Sigma(r(r'/x \cdot \sigma)\theta).$$

The case $fvar(\lambda x.r) \cap dom(\sigma\theta) = \emptyset$ is treated in a similar way.

(4.4) t' is the term $(t''\sigma)$. Since $t''\sigma\theta \rhd^1_\sigma t''(\sigma\theta)$ we can apply the main induction hypothesis to $t''(\sigma\theta)$. According to $t'\theta \overset{\beta}{\longmapsto} s$ we can distinguish the following three subcases:

(4.4.1) $t'\theta \overset{\beta}{\longmapsto} s$ is $t''\sigma\theta \overset{\beta}{\longmapsto} t''\sigma\theta'$ and is a consequence of $\theta \overset{\beta}{\longmapsto} \theta'$. Then, by the main induction hypothesis we have

$$\Sigma(t''\sigma\theta) = \Sigma(t''(\sigma\theta)) \rhd_{\beta_n} \Sigma(t''(\sigma\theta')) = \Sigma(t''\sigma\theta').$$

(4.4.2) $t'\theta \overset{\beta}{\longmapsto} s$ is $t''\sigma\theta \overset{\beta}{\longmapsto} t''\sigma'\theta$ and is a consequence of $\sigma \overset{\beta}{\longmapsto} \sigma'$. This case is treated in a very similar way as (4.4.1).

(4.4.3) $t'\theta \overset{\beta}{\longmapsto} s$ is $t''\sigma\theta \overset{\beta}{\longmapsto} t'''\sigma\theta$ and is a consequence of $t'' \overset{\beta}{\longmapsto} t'''$. Then, by the main induction hypothesis we have

$$\Sigma(t''\sigma\theta) = \Sigma(t''(\sigma\theta)) \rhd_{\beta_n} \Sigma(t'''(\sigma\theta)) = \Sigma(t'''\sigma\theta).$$

This finishes the proof of our claim. $\square$

Since $t \rhd^1_\beta s$ trivially implies $t \overset{\beta}{\longmapsto} s$ we get the following

PROOF We prove the claim by main induction on $\Psi(t)$ and side induction on the complexity of t. According to the structure of t, we can distinguish the following four cases:

(1) t is a variable. Then the claim is trivial.

(2) t is the term $(\lambda x.t')$. The claim is trivial, too.

(3) t is the term $(t_1 t_2)$. According to $t \stackrel{\beta}{\longmapsto} s$ we can distinguish the following four subcases:

(3.1) $t \stackrel{\beta}{\longmapsto} s$ is $t_1 t_2 \stackrel{\beta}{\longmapsto} t_1' t_2$ and is a consequence of $t_1 \stackrel{\beta}{\longmapsto} t_1'$. By the side induction hypothesis we have $\Sigma(t_1) \rhd_{\beta_n} \Sigma(t_1')$. This implies

$$\Sigma(t_1 t_2) = \Sigma(t_1)\Sigma(t_2) \rhd_{\beta_n} \Sigma(t_1')\Sigma(t_2) = \Sigma(t_1' t_2).$$

(3.2) $t \stackrel{\beta}{\longmapsto} s$ is $t_1 t_2 \stackrel{\beta}{\longmapsto} t_1 t_2'$ and is a consequence of $t_2 \stackrel{\beta}{\longmapsto} t_2'$. This case is treated in the same way as (3.1).

(3.3) $t \stackrel{\beta}{\longmapsto} s$ is $(\lambda x.r)r' \stackrel{\beta}{\longmapsto} r\{r'/x\}$. Then we have

$$\Sigma((\lambda x.r)r') = \Sigma(\lambda x.r)\Sigma(r') = (\lambda x.r)\Sigma(r') \rhd_{\beta_n} \Sigma(r\{\Sigma(r')/x\}) = \Sigma(r\{r'/x\}).$$

(3.4) $t \stackrel{\beta}{\longmapsto} s$ is $(\lambda x.r)\theta r' \stackrel{\beta}{\longmapsto} r(r'/x \cdot \theta)$. First assume that $fvar(\lambda x.r) \cap dom\,\theta \neq \emptyset$. Then we have

$$\Sigma((\lambda x.r)\theta r') = (\lambda x.r)\Sigma(\theta^{\lambda x.r})\Sigma(r') \rhd_{\beta_n} \Sigma(r(\Sigma(r')/x \cdot \Sigma(\theta^{\lambda x.r}))) = \Sigma(r(r'/x \cdot \theta)).$$

The case $fvar(\lambda x.r) \cap dom\,\theta = \emptyset$ is treated in a similar way.

(4) t is the term $(t'\theta)$. According to the structure of t' we can distinguish the following four subcases:

(4.1) t' is the variable x. Then s is of the form $x\theta'$, where $\theta \stackrel{\beta}{\longmapsto} \theta'$. If $x \notin dom\,\theta$ then $\Sigma(x\theta) = x = \Sigma(x\theta')$ and there is nothing to prove. Therefore, assume $t/x \in \theta$, $t'/x \in \theta'$ and $t \stackrel{\beta}{\longmapsto} t'$ for some terms t and t'. Then it is $x\theta \rhd_\sigma^1 t$ and by the main induction hypothesis we have

$$\Sigma(x\theta) = \Sigma(t) \rhd_{\beta_n} \Sigma(t') = \Sigma(x\theta').$$

(4.2) t' is the term $(\lambda x.t'')$. Then s is of the form $(\lambda x.t'')\theta'$, where $\theta \stackrel{\beta}{\longmapsto} \theta'$. If $fvar(\lambda x.t'') \cap dom\,\theta = \emptyset$ then $\Sigma((\lambda x.t'')\theta) = \lambda x.t'' = \Sigma((\lambda x.t'')\theta')$ and there is nothing to prove. Otherwise, by the side induction hypothesis we have

$$\Sigma((\lambda x.t'')\theta) = (\lambda x.t'')\theta^{(\lambda x.t'')} \rhd_{\beta_n} (\lambda x.t'')\theta'^{(\lambda x.t'')} = \Sigma((\lambda x.t'')\theta').$$

Theorem 41 *$\rhd_{\beta_n}$ is Church Rosser.*

PROOF By the previous lemma, $\overset{\beta_n}{\longmapsto}$ is Church Rosser. By Lemma 38, $\rhd_{\beta_n}$ is the transitive closure of $\overset{\beta_n}{\longmapsto}$, which, by a simple diagram chase, implies that $\rhd_{\beta_n}$ is confluent, too. $\Box$

The last step towards the Church Rosser property of $\rhd$ is to show that

$$t \rhd^1_\beta s \quad \Longrightarrow \quad \Sigma(t) \rhd_{\beta_n} \Sigma(s)$$

holds for all $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ terms t and s. Then the confluence of $\rhd$ will be an immediate consequence. In order to establish the above claim, we have to define another intermediate relation, the reduction relation $\overset{\beta}{\longmapsto}$.

Definition 42 The relation $\overset{\beta}{\longmapsto}$ between $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ terms and $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ substitutions is given by the following clauses (1)–(8):

A. TERMS

(1) $t \overset{\beta}{\longmapsto} t$

(2) $(\lambda x.t)s \overset{\beta}{\longmapsto} t\{s/x\}$

(3) $(\lambda x.t)\theta s \overset{\beta}{\longmapsto} t(s/x \cdot \theta)$

(4) $\theta \overset{\beta}{\longmapsto} \theta' \quad \Longrightarrow \quad r\theta \overset{\beta}{\longmapsto} r\theta'$

(5) $t \overset{\beta}{\longmapsto} s \quad \Longrightarrow \quad t\theta \overset{\beta}{\longmapsto} s\theta$

(6) $t \overset{\beta}{\longmapsto} s \quad \Longrightarrow \quad rt \overset{\beta}{\longmapsto} rs$

(7) $t \overset{\beta}{\longmapsto} s \quad \Longrightarrow \quad tr \overset{\beta}{\longmapsto} sr$

B. SUBSTITUTIONS

(8) $t_1 \overset{\beta}{\longmapsto} s_1, \ldots, t_n \overset{\beta}{\longmapsto} s_n \quad \Longrightarrow \quad \{t_1/x_1, \ldots, t_n/x_n\} \overset{\beta}{\longmapsto} \{s_1/x_1, \ldots, s_n/x_n\}$

Notice that $\overset{\beta}{\longmapsto}$ is not transitively closed. $\overset{\beta}{\longmapsto}$ can be considered as an extended form of one step β reduction, where substitutions can be reduced in parallel.

Lemma 43 *We have for all terms t and s:*

$$t \overset{\beta}{\longmapsto} s \quad \Longrightarrow \quad \Sigma(t) \rhd_{\beta_n} \Sigma(s).$$

$\theta' \stackrel{\beta_n}{\longmapsto} \theta'''$, $\theta'' \stackrel{\beta_n}{\longmapsto} \theta'''$. If we take $t_3 := \Sigma(r(s'''/x \cdot \theta'''))$ then obviously $t_2 \stackrel{\beta_n}{\longmapsto} t_3$. By the fact that $s'/x \cdot \theta' \stackrel{\beta_n}{\longmapsto} s'''/x \cdot \theta'''$ and the previous lemma we also have $t_1 \stackrel{\beta_n}{\longmapsto} t_3$.

(3.2) $t \stackrel{\beta_n}{\longmapsto} t_2$ is $(\lambda x.r)\theta s \stackrel{\beta_n}{\longmapsto} \Sigma(r(s''/x \cdot \theta''))$ and is a consequence of $s \stackrel{\beta_n}{\longmapsto} s''$ and $\theta \stackrel{\beta_n}{\longmapsto} \theta''$. By the induction hypothesis there are s''' and θ''' with $s' \stackrel{\beta_n}{\longmapsto} s'''$, $s'' \stackrel{\beta_n}{\longmapsto} s'''$ and $\theta' \stackrel{\beta_n}{\longmapsto} \theta'''$, $\theta'' \stackrel{\beta_n}{\longmapsto} \theta'''$. Let $t_3 := \Sigma(r(s'''/x \cdot \theta'''))$. Then, by the previous lemma we have $t_1 \stackrel{\beta_n}{\longmapsto} t_3$ and $t_2 \stackrel{\beta_n}{\longmapsto} t_3$, since $s'/x \cdot \theta' \stackrel{\beta_n}{\longmapsto} s'''/x \cdot \theta'''$ and $s''/x \cdot \theta'' \stackrel{\beta_n}{\longmapsto} s'''/x \cdot \theta'''$.

(4) $t \stackrel{\beta_n}{\longmapsto} t_1$ is $r\theta \stackrel{\beta_n}{\longmapsto} r\theta'$ and is a consequence of $\theta \stackrel{\beta_n}{\longmapsto} \theta'$. Then $t \stackrel{\beta_n}{\longmapsto} t_2$ is $r\theta \stackrel{\beta_n}{\longmapsto} r\theta''$ and is a consequence of $\theta \stackrel{\beta_n}{\longmapsto} \theta''$. By the induction hypothesis there is a θ''' with $\theta' \stackrel{\beta_n}{\longmapsto} \theta'''$ and $\theta'' \stackrel{\beta_n}{\longmapsto} \theta'''$. The claim obviously holds for $t_3 := r\theta'''$.

(5) $t \stackrel{\beta_n}{\longmapsto} t_1$ is $rs \stackrel{\beta_n}{\longmapsto} r's'$ and is a consequence of $r \stackrel{\beta_n}{\longmapsto} r'$ and $s \stackrel{\beta_n}{\longmapsto} s'$. According to $t \stackrel{\beta_n}{\longmapsto} t_2$ we can distinguish the following three subcases:

(5.1) $t \stackrel{\beta_n}{\longmapsto} t_2$ is $rs \stackrel{\beta_n}{\longmapsto} r''s''$ and is a consequence of $r \stackrel{\beta_n}{\longmapsto} r''$ and $s \stackrel{\beta_n}{\longmapsto} s''$. By the induction hypothesis there are terms r''' and s''' with $r' \stackrel{\beta_n}{\longmapsto} r'''$, $r'' \stackrel{\beta_n}{\longmapsto} r'''$ and $s' \stackrel{\beta_n}{\longmapsto} s'''$, $s'' \stackrel{\beta_n}{\longmapsto} s'''$. The claim immediately follows for $t_3 := r'''s'''$.

(5.2) $t \stackrel{\beta_n}{\longmapsto} t_2$ is $(\lambda x.r'')s \stackrel{\beta_n}{\longmapsto} \Sigma(r''\{s''/x\})$ and is a consequence of $s \stackrel{\beta_n}{\longmapsto} s''$. By the induction hypothesis there is an s''' with $s' \stackrel{\beta_n}{\longmapsto} s'''$ and $s'' \stackrel{\beta_n}{\longmapsto} s'''$. Furthermore, it is clear that $r = r' = (\lambda x.r'')$. If we take $t_3 := \Sigma(r''\{s'''/x\})$ then obviously $t_1 \stackrel{\beta_n}{\longmapsto} t_3$. By the previous lemma we also have $t_2 \stackrel{\beta_n}{\longmapsto} t_3$.

(5.3) $t \stackrel{\beta_n}{\longmapsto} t_2$ is $(\lambda x.r'')\theta s \stackrel{\beta_n}{\longmapsto} \Sigma(r''\{s''/x \cdot \theta''\})$ and is a consequence of $s \stackrel{\beta_n}{\longmapsto} s''$ and $\theta \stackrel{\beta_n}{\longmapsto} \theta''$. By the induction hypothesis there is an s''' with $s' \stackrel{\beta_n}{\longmapsto} s'''$ and $s'' \stackrel{\beta_n}{\longmapsto} s'''$. Furthermore, $r \stackrel{\beta_n}{\longmapsto} r'$ has the form $(\lambda x.r'')\theta \stackrel{\beta_n}{\longmapsto} (\lambda x.r'')\theta'$ and is a consequence of $\theta \stackrel{\beta_n}{\longmapsto} \theta'$. By the induction hypothesis there is a θ''' with $\theta' \stackrel{\beta_n}{\longmapsto} \theta'''$ and $\theta'' \stackrel{\beta_n}{\longmapsto} \theta'''$. If we take $t_3 := \Sigma(r''(s'''/x \cdot \theta'''))$ then obviously $t_1 \stackrel{\beta_n}{\longmapsto} t_3$. By the previous lemma we also have $t_2 \stackrel{\beta_n}{\longmapsto} t_3$.

This finishes the proof of 1. The proof of 2. is straightforward. Assume $\theta \stackrel{\beta_n}{\longmapsto} \theta_1$ and $\theta \stackrel{\beta_n}{\longmapsto} \theta_2$, where

$$\begin{aligned} \theta &= \{t_1/x_1, \ldots, t_n/x_n\}, \\ \theta_1 &= \{s_1/x_1, \ldots, s_n/x_n\}, \\ \theta_2 &= \{s'_1/x_1, \ldots, s'_n/x_n\} \end{aligned}$$

and $t_i \stackrel{\beta_n}{\longmapsto} s_i$, $t_i \stackrel{\beta_n}{\longmapsto} s'_i$ for $1 \leq i \leq n$. By the induction hypothesis there are terms r_i with $s_i \stackrel{\beta_n}{\longmapsto} r_i$ and $s'_i \stackrel{\beta_n}{\longmapsto} r_i$ for $1 \leq i \leq n$. If we take $\theta_3 := \{r_1/x_1, \ldots, r_n/x_n\}$ then we immediately get $\theta_1 \stackrel{\beta_n}{\longmapsto} \theta_3$ and $\theta_2 \stackrel{\beta_n}{\longmapsto} \theta_3$. $\square$

The proof of the Church Rosser property of $\rhd_{\beta_n}$ is complete.

Lemma 38 *$\rhd_{\beta_n}$ is the transitive closure of $\stackrel{\beta_n}{\longmapsto}$.*

PROOF Obvious. $\square$

The following lemma is essential in the confluence proof for $\stackrel{\beta_n}{\longmapsto}$ below. The proof of the lemma is not difficult but long and tedious, so we leave it to the reader.

Lemma 39

1. *If $t \stackrel{\beta_n}{\longmapsto} t'$ then we have $\Sigma(t\theta) \stackrel{\beta_n}{\longmapsto} \Sigma(t'\theta')$ for all $\theta \stackrel{\beta_n}{\longmapsto} \theta'$.*
2. *If $\theta \stackrel{\beta_n}{\longmapsto} \theta'$ then we have $\Sigma(\theta\sigma) \stackrel{\beta_n}{\longmapsto} \Sigma(\theta'\sigma')$ for all $\sigma \stackrel{\beta_n}{\longmapsto} \sigma'$.*

PROOF 1. and 2. are proved simultaneously by induction on the complexity of t and θ respectively. $\square$

The next lemma says that $\stackrel{\beta_n}{\longmapsto}$ is Church Rosser on terms and substitutions.

Lemma 40

1. *If $t \stackrel{\beta_n}{\longmapsto} t_1$ then for all $t \stackrel{\beta_n}{\longmapsto} t_2$ there exists a t_3 such that $t_1 \stackrel{\beta_n}{\longmapsto} t_3$ and $t_2 \stackrel{\beta_n}{\longmapsto} t_3$.*
2. *If $\theta \stackrel{\beta_n}{\longmapsto} \theta_1$ then for all $\theta \stackrel{\beta_n}{\longmapsto} \theta_2$ there exists a θ_3 such that $\theta_1 \stackrel{\beta_n}{\longmapsto} \theta_3$ and $\theta_2 \stackrel{\beta_n}{\longmapsto} \theta_3$.*

PROOF We prove 1. and 2. simultaneously by induction on $t \stackrel{\beta_n}{\longmapsto} t_1$ and $\theta \stackrel{\beta_n}{\longmapsto} \theta_1$ respectively. Let us first prove 1. According to $t \stackrel{\beta_n}{\longmapsto} t_1$ we can distinguish the following five cases:

(1) $t \stackrel{\beta_n}{\longmapsto} t_1$ is $t \stackrel{\beta_n}{\longmapsto} t$. Then we can choose $t_3 := t_2$.

(2) $t \stackrel{\beta_n}{\longmapsto} t_1$ is $(\lambda x.r)s \stackrel{\beta_n}{\longmapsto} \Sigma(r\{s'/x\})$ and is a consequence of $s \stackrel{\beta_n}{\longmapsto} s'$. According to $t \stackrel{\beta_n}{\longmapsto} t_2$ we can distinguish the following two subcases:

(2.1) $t \stackrel{\beta_n}{\longmapsto} t_2$ is $(\lambda x.r)s \stackrel{\beta_n}{\longmapsto} (\lambda x.r)s''$ and is a consequence of $s \stackrel{\beta_n}{\longmapsto} s''$. By the induction hypothesis there is a term s''' with $s' \stackrel{\beta_n}{\longmapsto} s'''$ and $s'' \stackrel{\beta_n}{\longmapsto} s'''$. If we take $t_3 := \Sigma(r\{s'''/x\})$ then obviously $t_2 \stackrel{\beta_n}{\longmapsto} t_3$. By the previous lemma we also have $t_1 \stackrel{\beta_n}{\longmapsto} t_3$.

(2.2) $t \stackrel{\beta_n}{\longmapsto} t_2$ is $(\lambda x.r)s \stackrel{\beta_n}{\longmapsto} \Sigma(r\{s''/x\})$ and is a consequence of $s \stackrel{\beta_n}{\longmapsto} s''$. By the induction hypothesis there is an s''' with $s' \stackrel{\beta_n}{\longmapsto} s'''$ and $s'' \stackrel{\beta_n}{\longmapsto} s'''$. Let $t_3 := \Sigma(r\{s'''/x\})$. Using the previous lemma one easily verifies $t_1 \stackrel{\beta_n}{\longmapsto} t_3$ and $t_2 \stackrel{\beta_n}{\longmapsto} t_3$.

(3) $t \stackrel{\beta_n}{\longmapsto} t_1$ is $(\lambda x.r)\theta s \stackrel{\beta_n}{\longmapsto} \Sigma(r(s'/x \cdot \theta'))$ and is a consequence of $s \stackrel{\beta_n}{\longmapsto} s'$ and $\theta \stackrel{\beta_n}{\longmapsto} \theta'$. According to $t \stackrel{\beta_n}{\longmapsto} t_2$ we can distinguish the following two subcases:

(3.1) $t \stackrel{\beta_n}{\longmapsto} t_2$ is $(\lambda x.r)\theta s \stackrel{\beta_n}{\longmapsto} (\lambda x.r)\theta'' s''$ and is a consequence of $\theta \stackrel{\beta_n}{\longmapsto} \theta''$ and $s \stackrel{\beta_n}{\longmapsto} s''$. By the induction hypothesis there are s''' and θ''' with $s' \stackrel{\beta_n}{\longmapsto} s'''$, $s'' \stackrel{\beta_n}{\longmapsto} s'''$ and

Definition 36 The relation $\rhd_{\beta_n}$ between $\boldsymbol{\lambda_p\sigma}$ terms in substitution normal form is generated by the following clauses (1)–(7):

(1) $t \rhd_{\beta_n} t$

(2) $(\lambda x.t)s \rhd_{\beta_n} \Sigma(t\{s/x\})$

(3) $(\lambda x.t)\theta s \rhd_{\beta_n} \Sigma(t(s/x \cdot \theta))$

(4) $t \rhd_{\beta_n} s \quad \Longrightarrow \quad r(t/x \cdot \theta) \rhd_{\beta_n} r(s/x \cdot \theta)$

(5) $t \rhd_{\beta_n} s \quad \Longrightarrow \quad rt \rhd_{\beta_n} rs$

(6) $t \rhd_{\beta_n} s \quad \Longrightarrow \quad tr \rhd_{\beta_n} sr$

(7) $t \rhd_{\beta_n} s,\ s \rhd_{\beta_n} r \quad \Longrightarrow \quad t \rhd_{\beta_n} r$

As we restrict ourselves to terms with $\Sigma(t) = t$, a clause of the form

$$t \rhd_{\beta_n} s \quad \Longrightarrow \quad t\theta \rhd_{\beta_n} s\theta.$$

is redundant.

In order to show that $\rhd_{\beta_n}$ satisfies the diamond property, we define a "parallel" version $\stackrel{\beta_n}{\longmapsto}$ of $\rhd_{\beta_n}$ such that the transitive closure of $\stackrel{\beta_n}{\longmapsto}$ is $\rhd_{\beta_n}$. Then the Church Rosser property of $\rhd_{\beta_n}$ follows from the Church Rosser property of $\stackrel{\beta_n}{\longmapsto}$ by a well known diagram chase.

For notational convenience we define $\stackrel{\beta_n}{\longmapsto}$ on substitutions, too.

Definition 37 The relation $\stackrel{\beta_n}{\longmapsto}$ between $\boldsymbol{\lambda_p\sigma}$ terms and $\boldsymbol{\lambda_p\sigma}$ substitutions in substitution normal form is simultaneously generated by the following clauses (1)–(6):

A. Terms

(1) $t \stackrel{\beta_n}{\longmapsto} t$

(2) $s \stackrel{\beta_n}{\longmapsto} s' \quad \Longrightarrow \quad (\lambda x.t)s \stackrel{\beta_n}{\longmapsto} \Sigma(t\{s'/x\})$

(3) $s \stackrel{\beta_n}{\longmapsto} s',\ \theta \stackrel{\beta_n}{\longmapsto} \theta' \quad \Longrightarrow \quad (\lambda x.t)\theta s \stackrel{\beta_n}{\longmapsto} \Sigma(t(s'/x \cdot \theta'))$

(4) $\theta \stackrel{\beta_n}{\longmapsto} \theta' \quad \Longrightarrow \quad r\theta \stackrel{\beta_n}{\longmapsto} r\theta'$

(5) $t \stackrel{\beta_n}{\longmapsto} t',\ s \stackrel{\beta_n}{\longmapsto} s' \quad \Longrightarrow \quad ts \stackrel{\beta_n}{\longmapsto} t's'$

B. Substitutions

(6) $t_1 \stackrel{\beta_n}{\longmapsto} s_1, \ldots, t_n \stackrel{\beta_n}{\longmapsto} s_n \quad \Longrightarrow \quad \{t_1/x_1, \ldots, t_n/x_n\} \stackrel{\beta_n}{\longmapsto} \{s_1/x_1, \ldots, s_n/x_n\}$

PROOF by a straightforward induction on the length of a derivation of $t \rhd^1_\sigma s$. Let us exemplary discuss the substitution reduction (6), i.e. $t = (r\theta)\sigma$ and $s = r(\theta\sigma)$. Then we have

$$\begin{aligned}
\Psi((r\theta)\sigma) &= \Psi(r\theta) \cdot (\Psi(\sigma) + 1) \\
&= \Psi(r) \cdot (\Psi(\theta) + 1) \cdot (\Psi(\sigma) + 1) \\
&= \Psi(r) \cdot [(\Psi(\sigma) + 1) \cdot \Psi(\theta) + (\Psi(\sigma) + 1)] \\
&> \Psi(r) \cdot [(\Psi(\sigma) + 1) \cdot \Psi(\theta) + 1]. \qquad (1)
\end{aligned}$$

According to the previous lemma we have

$$(\Psi(\sigma) + 1) \cdot \Psi(\theta) > \Psi(\theta\sigma), \qquad (2)$$

hence

$$\Psi(r) \cdot [(\Psi(\sigma) + 1) \cdot \Psi(\theta) + 1] > \Psi(r) \cdot [\Psi(\theta\sigma) + 1]. \qquad (3)$$

Since $\Psi(r) \cdot [\Psi(\theta\sigma) + 1] = \Psi(r(\theta\sigma))$, (1) and (3) immediately imply

$$\Psi((r\theta)\sigma) > \Psi(r(\theta\sigma)). \qquad (4)$$

The claim is proved. □

Corollary 33 *$\rhd^1_\sigma$ is wellfounded.*

Alltogether we have established the following

Theorem 34 *$\rhd_\sigma$ is Church Rosser.*

PROOF The theorem is immediate from Lemma 29 and Corollary 33 and a result by Newman [23] saying that a reduction relation, which is weakly Church Rosser and wellfounded satisfies the full Church Rosser property. □

Corollary 35 *Every $\boldsymbol{\lambda_p\sigma}$ term has a unique substitution normal form.*

In the following we denote the substitution normal form of a term t with $\Sigma(t)$. A substitution $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ is in substitution normal form, if for all $1 \leq i \leq n$ the term t_i is in substitution normal form. Analogously, $\Sigma(\theta)$ denotes the substitution normal form of a substitution θ.

As a further step towards the Church Rosser property of $\rhd$ we define a relation $\rhd_{\beta_n}$, which corresponds to β reduction on terms in substitution normal form, i.e. terms satisfying $\Sigma(t) = t$.

due to Hardin, which was identified in [12]. This method has subsequently been used several times in order to show confluence for systems of explicit substitutions.

A first step towards the proof of the Church Rosser property for $\rhd$ is to show that $\rhd_\sigma$ is Church Rosser. According to an old result by Newman [23] it is enough to establish that $\rhd_\sigma^1$ is weakly Church Rosser and wellfounded.

Lemma 29 *$\rhd_\sigma^1$ is weakly Church Rosser, i.e. we have for all terms t, t_1, t_2: If $t \rhd_\sigma^1 t_1$ and $t \rhd_\sigma^1 t_2$ then there exists a term t_3 such that $t_1 \rhd_\sigma t_3$ and $t_2 \rhd_\sigma t_3$.*

PROOF One shows by a straightforward but tedious induction on the length of a derivation of $t \rhd_\sigma^1 t_1$ that for all $t \rhd_\sigma^1 t_2$ there exists a t_3 such that $t_1 \rhd_\sigma t_3$ and $t_2 \rhd_\sigma t_3$. $\Box$

In order to prove that $\rhd_\sigma^1$ is wellfounded we define a measure function Ψ from the $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ terms and $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ substitutions to ω.

Definition 30 $(\Psi(t); \Psi(\theta))$

1. If t is a variable then $\Psi(t) := 1$.
2. If t is the term $(\lambda x.s)$ then $\Psi(t) := 1$.
3. If t is the term $(t_1 t_2)$ then $\Psi(t) := \Psi(t_1) + \Psi(t_2) + 1$.
4. If t is the term $(s\theta)$ then $\Psi(t) := \Psi(s) \cdot (\Psi(\theta) + 1)$.
5. If θ is the substitution $\{t_1/x_1, \ldots, t_n/x_n\}$ then $\Psi(\theta) := \Psi(t_1) + \cdots + \Psi(t_n) + 1$.

Notice that $\Psi(t) > 0$ for all terms t and $\Psi(\theta) > 0$ for all substitutions θ, e.g. we have $\Psi(\varepsilon) = 1$.

The composition $(\theta\sigma)$ of two substitutions θ and σ is bounded as follows w.r.t. Ψ.

Lemma 31 *We have for all substitutions θ and σ:*

$$\Psi(\theta\sigma) < \Psi(\theta) \cdot (\Psi(\sigma) + 1).$$

PROOF by an easy calculation. $\Box$

We are ready to prove that Ψ is strictly decreasing on $\rhd_\sigma^1$.

Lemma 32 *We have for all terms t and s:*

$$t \rhd_\sigma^1 s \quad \Longrightarrow \quad \Psi(t) > \Psi(s).$$

(7) $t(s/x \cdot \theta) \rhd t\theta \qquad (x \notin fvar(t) \cup dom\,\theta)$

(8) $t\varepsilon \rhd t$

D. Structural Rules

(9) $t \rhd s \implies r(t/x \cdot \theta) \rhd r(s/x \cdot \theta)$

(10) $t \rhd s \implies t\theta \rhd s\theta$

(11) $t \rhd s \implies rt \rhd rs$

(12) $t \rhd s \implies tr \rhd sr$

E. Transitivity

(13) $t \rhd s,\ s \rhd r \implies t \rhd r$

Notice that we have to state two clauses for β reduction, since (2) is no longer derivable from (3) in the context of reductions. Furthermore, it should be observed again that we do not have the rule (ξ),

$$(\xi) \quad \frac{t \rhd s}{\lambda x.t \rhd \lambda x.s}$$

among the structural rules (D).

Remark 28 It should be stressed that the reduction relation $\rhd$ does not take into consideration partiality in $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$. Hence $\rhd$ rather corresponds to a total version of $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$. This is, however, in complete analogy to the system CL_p, where truly partial term models are constructed using special reduction strategies of a *total* reduction relation (cf. p.7), and partiality is reflected by non-terminating reduction sequences. Summarizing, $\rhd$ provides a general term reduction framework giving rise to partial and total term models for $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$.

In the following $\rhd_\sigma$ denotes the restriction of $\rhd$ to substitution reductions, i.e. $t \rhd_\sigma s$ holds if and only if there is a derivation of $t \rhd s$ according to the above clauses, which does not use (2) and (3). Analogously, $\rhd_\beta$ is the restriction of $\rhd$ to β reductions. Finally, we write $t \rhd^1 s$ if $t \rhd s$ is derivable from (2)–(12), i.e. $\rhd^1$ denotes one step reduction. The relations $\rhd^1_\sigma$ and $\rhd^1_\beta$ are defined in the same way.

In the sequel we want to show that $\rhd$ satisfies the Church Rosser property. As usual, this will guarantee that all terminating rewrite sequences yield identical results, i.e. we will have uniqueness of normal forms. All attempts to find a direct proof for the confluence of $\rhd$ failed. Instead we will make use of an interpretation technique

One easily verifies that $(\mathsf{k}^\lambda)^\bullet = \mathsf{k}$ and $(\mathsf{s}^\lambda)^\bullet = \mathsf{s}$. As an immediate consequence we have $(t^\lambda)^\bullet = t$ and $(A^\lambda)^\bullet = A$ for all CL_p terms t and all CL_p formulas A, respectively. Furthermore, Lemma 8 holds for $\lambda^\bullet$, too. In particular, we have $\lambda^\bullet x.t\downarrow$ for all CL_p terms t. We can, therefore, establish Theorem 19 for $(\cdot)^\bullet$ instead of $(\cdot)^{CL}$. Hence we have

$$\boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash A \quad \Longrightarrow \quad \mathsf{CL}_p \vdash A^\bullet$$

for all $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ formulas A. Now the claim of the lemma immediately follows from the fact that $(\cdot)^\bullet$ is the inverse of $(\cdot)^\lambda$, as we have mentioned above. $\square$

Here is the final embedding theorem.

Theorem 26 *We have for all* CL_p *formulas* A*:*

$$\mathsf{CL}_p \vdash A \quad \Longleftrightarrow \quad \boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash A^\lambda.$$

PROOF Immediate from the previous two lemmas. $\square$

4 Confluent reductions

Once we have introduced the system $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ it is natural to study the corresponding reduction relation on $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ terms. In the following we define a binary relation $\rhd$ on $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ terms, which reflects a directed equality relation for the system $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$. $\rhd$ is defined in an inductive way.

Definition 27 The relation $\rhd$ between $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ terms is generated by the following clauses (1)-(13).

A. IDENTITY

(1) $t \rhd t$

B. β REDUCTIONS

(2) $(\lambda x.t)s \rhd t\{s/x\}$

(3) $(\lambda x.t)\theta s \rhd t(s/x \cdot \theta)$

C. SUBSTITUTION REDUCTIONS

(4) $x\theta \rhd t \qquad (t/x \in \theta)$

(5) $(ts)\theta \rhd (t\theta)(s\theta)$

(6) $(t\theta)\sigma \rhd t(\theta\sigma)$

PROOF We only prove 1. The proof of 2. and 3. is similar. As we will see, essential use will be made of the *extended* β axiom (15). First of all we have

$$(\lambda uv.u)x \simeq (\lambda v.u)\{x/u\}, \tag{1}$$

which by axiom (15) immediately implies

$$(\lambda uv.u)xy \simeq (\lambda v.u)\{x/u\}y \simeq u\{x/u, y/v\}. \tag{2}$$

Of course,

$$u\{x/u, y/v\} \simeq x. \tag{3}$$

We are done. $\square$

Lemma 24 *We have for all* CL_p *formulas* A*:*

$$\mathsf{CL}_p \vdash A \quad \Longrightarrow \quad \boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash A^{\lambda}.$$

PROOF by induction on the length of a proof of A in CL_p. Again the propositional axioms are trivial. The translation of the quantifier axioms is provable in $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ by Corollary 22. The same corollary also helps in establishing the equality axiom (6). The strictness axioms (9) and the axioms for a partial combinatory algebra are already treated in Lemma 23. The inference rules of CL_p readily translate into inference rules of $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$. $\square$

The converse of the above lemma also holds.

Lemma 25 *We have for all* CL_p *formulas* A*:*

$$\boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash A^{\lambda} \quad \Longrightarrow \quad \mathsf{CL}_p \vdash A.$$

PROOF We define a modification $(\cdot)^{\bullet}$ of the translation $(\cdot)^{CL}$ from $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ into CL_p. $(\cdot)^{\bullet}$ is defined in the same way as $(\cdot)^{CL}$, except that it uses the more complicated coding of λ abstraction $\lambda^{\bullet}$ instead of λ^{*}. The term $\lambda^{\bullet}x.t$ is inductively defined as follows.

1. If t is the variable x then $\lambda^{\bullet}x.t := \mathsf{skk}$.
2. If t is a variable different from x or a constant then $\lambda^{\bullet}x.t := \mathsf{k}t$.
3. If t is the term (sx), where $s \in \{y, \mathsf{k}, \mathsf{s}, \mathsf{s}y\}$ and $y \neq x$ then $\lambda^{\bullet}x.t := s$.
4. If t is the term (t_1t_2), and if 3. does not apply then $\lambda^{\bullet}x.t := \mathsf{s}(\lambda^{\bullet}x.t_1)(\lambda^{\bullet}x.t_2)$.

8. The substitution axioms are easily verified, too. The extended β axiom (15) is treated in exactly the same way as in the proof of Lemma 9. Finally, it is trivial to check the inference rules. $\square$

In the sequel we show that $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ also includes CL_p by giving an embedding $(\cdot)^\lambda$ from CL_p into $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$. We first give the translation $(\cdot)^\lambda$ for terms of CL_p.

Definition 20 (t^λ)

1. If t is a variable then $t^\lambda := t$.
2. If t is the constant k then $t^\lambda := \lambda uv.u$.
3. If t is the constant s then $t^\lambda := \lambda uvw.uw(vw)$.
4. If t is the term $(t_1 t_2)$ then $t^\lambda := (t_1^\lambda t_2^\lambda)$.

For formulas, $(\cdot)^\lambda$ is given in the obvious way, i.e. $(\cdot)^\lambda$ commutes with $=$, $\downarrow$, the logical connectives and the quantifiers. Again it is obvious that $\mathit{fvar}(t^\lambda) = \mathit{fvar}(t)$ and $\mathit{fvar}(A^\lambda) = \mathit{fvar}(A)$.

Lemma 21 *We have for all CL_p terms t and s:*

$$\boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash t[s/x]^\lambda \simeq t^\lambda\{s^\lambda/x\}.$$

PROOF by straightforward induction on the complexity of t. Essential use is made of the substitution axioms, in particular axiom (13). $\square$

Corollary 22 *We have for all CL_p formulas A and all CL_p terms t:*

$$\boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash A[t/x]^\lambda \leftrightarrow A^\lambda\{t^\lambda/x\}.$$

PROOF by an easy induction on the complexity of A using the above lemma. $\square$

Lemma 23

1. $\boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash (\mathsf{k}xy \simeq x)^\lambda$.
2. $\boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash (\mathsf{s}xyz \simeq xz(yz))^\lambda$.
3. $\boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash (\mathsf{s}xy\downarrow)^\lambda$.

2. $\boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash s\downarrow \to (\lambda x.t)\theta s \simeq t(s/x \cdot \theta)$.

3. $\boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash t\theta \simeq t \quad (dom\,\theta \cap fvar(t) = \emptyset)$.

PROOF by easy reasoning in $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$. $\square$

In the following we give an embedding $(\cdot)^{CL}$ of the system $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ into partial combinatory logic $\mathbf{CL}_p$. This embedding is made possible by a careful concept of substitution in the system $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ corresponding to substitution in $\mathbf{CL}_p$. As an immediate consequence of this interpretation we get that the recursion theoretic model PRO and the normal term model CNT *are* models of $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$. This makes $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ into a system with a reasonable computational and constructive meaning.

Let us first define a $\mathbf{CL}_p$ term t^{CL} for each $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ term t. The definition is by induction on the complexity of t.

Definition 18 (t^{CL})

1. If t is a variable then $t^{CL} := t$.
2. If t is the term $(\lambda x.s)$ then $t^{CL} := (\lambda^* x.s^{CL})$.
3. If t is the term $(t_1 t_2)$ then t^{CL} is the term $(t_1^{CL} t_2^{CL})$.
4. If t is the term $(s\theta)$ where $\theta = \{s_1/x_1, \ldots, s_n/x_n\}$ then t^{CL} is the term $s^{CL}[s_1^{CL}/x_1, \ldots, s_n^{CL}/x_n]$.

Once $(\cdot)^{CL}$ is defined for terms of $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$, the translation for formulas is uniquely determined by the requirement that it commutes with $=$, $\downarrow$, the logical connectives and the quantifiers. Hence we have a $\mathbf{CL}_p$ formula A^{CL} for each $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ formula A. From the definition of $(\cdot)^{CL}$ it is immediate that $fvar(t^{CL}) = fvar(t)$, $fvar(A^{CL}) = fvar(A)$ and $A\{t_1/x_1, \ldots, t_n/x_n\}^{CL} = A^{CL}[t_1^{CL}/x_1, \ldots, t_n^{CL}/x_n]$.

We are ready to state the embedding theorem.

Theorem 19 *We have for all* $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ *formulas* A*:*

$$\boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash A \quad \Longrightarrow \quad \mathbf{CL}_p \vdash A^{CL}.\ ^2$$

PROOF by induction on the length of a proof of A in $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$. The propositional axioms do not cause any problems, of course. Also the quantifier axioms are handled easily using the properties of $(\cdot)^{CL}$ mentioned above. The equality axioms are trivial, and in order to establish the translation of $(\lambda x.t)\downarrow$ one makes immediate use of Lemma

[2] The converse of this theorem does not hold. For example, take for A the formula $x = y \to \lambda z.x = \lambda z.y$.

(5) $t_1 = s_1 \wedge t_2 = s_2 \to t_1 t_2 \simeq s_1 s_2$

(6) $\theta \simeq \theta' \to t\theta \simeq t\theta'$

D. Strictness Axioms

(7) $x\!\downarrow$

(8) $t_1 = t_2 \to t_1\!\downarrow \wedge\, t_2\!\downarrow \qquad t_1 t_2\!\downarrow \to t_1\!\downarrow \wedge\, t_2\!\downarrow$

(9) $(\lambda x.t)\!\downarrow$

E. Substitution Axioms

(10) $x\theta \simeq t \qquad (t/x \in \theta)$

(11) $(ts)\theta \simeq (t\theta)(s\theta)$

(12) $(t\theta)\sigma \simeq t(\theta\sigma)$

(13) $t(s/x \cdot \theta) \simeq t\theta \qquad (x \notin fvar(t) \cup dom\,\theta)$

(14) $t\varepsilon \simeq t$

F. β Axiom

(15) $(\lambda x.t)\theta y \simeq t(y/x \cdot \theta)$

G. Rules of Inference

(16) $\dfrac{A \quad A \to B}{B}$

(17) $\dfrac{A \to B}{\exists x A \to B} \qquad \dfrac{A \to B}{A \to \forall x B}$

In the inference rules (16) x does not appear free in the conclusion.

It should be observed that among the substitution axioms (E) we do *not* have an axiom, which allows us to push a substitution θ inside an abstraction $(\lambda x.t)$, of course. This is exactly what we want to prevent. Terms of the form $(\lambda x.t)\theta$ can only be resolved if applied to another object, say y. This is reflected in the extended β axiom (15), where an interleaving substitution θ is allowed. If θ is the empty substitution ε then we have the usual β axiom. Axioms similar to (15) are also stated in [5] and [19].

Lemma 17 *We have for all terms t, s and all substitutions θ:*

1. $\boldsymbol{\lambda}_p\boldsymbol{\sigma} \vdash s\!\downarrow \to (\lambda x.t)s \simeq t\{s/x\}$.

Definition 14 $(A\theta)$

1. If A is the formula $(s = t)$ then $A\theta$ is the formula $(s\theta = t\theta)$.
2. If A is the formula $t\!\downarrow$ then $A\theta$ is the formula $t\theta\!\downarrow$.
3. If A is the formula $\neg B$, $(B \vee C)$, $(B \wedge C)$ or $(B \to C)$ then $A\theta$ is the formula $\neg(B\theta)$, $(B\theta \vee C\theta)$, $(B\theta \wedge C\theta)$ or $(B\theta \to C\theta)$ respectively.
4. If A is the formula $\exists x B$ or $\forall x B$ then $A\theta$ is the formula $\exists y\,(B[y/x]\theta)$ or $\forall y\,(B[y/x]\theta)$ respectively, where y is a "fresh" variable.[1]

The *composition* of two substitutions θ and σ is defined in the usual way.

Definition 15 Let $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ and $\sigma = \{s_1/y_1, \ldots, s_m/y_m\}$ be substitutions. The substitution $\theta\sigma$ is obtained by deleting all variable bindings of the form s_i/y_i in the set

$$\{t_1\sigma/x_1, \ldots, t_n\sigma/x_n, s_1/y_1, \ldots, s_m/y_m\},$$

such that $y_i = x_j$ for a $1 \leq j \leq n$.

Now we are ready to give the exact formulation of the system $\boldsymbol{\lambda_p\sigma}$. The logic of $\boldsymbol{\lambda_p\sigma}$ is an adaptation of Beeson's logic of partial terms to the framework of explicit substitutions. The novel point of this axiom system compared to the system $\boldsymbol{\lambda_p}$ are the substitution axioms (E), which incorporate rules to evaluate substitutions step by step. Furthermore, an extended form of the β axiom in the context of explicit substitutions is given.

Definition 16 The system $\boldsymbol{\lambda_p\sigma}$ is formulated in the language of $\boldsymbol{\lambda_p\sigma}$ and contains the following list of axioms and rules of inference.

A. Propositional Logic

(1) Some complete axiom schemes of classical propositional logic

B. Quantifier Axioms

(2) $\forall\vec{x}A \wedge \theta\!\downarrow \to A\theta$

(3) $A\theta \wedge \theta\!\downarrow \to \exists\vec{x}A \qquad (\vec{x} = x_1, \ldots, x_n;\ dom\,\theta = \{\vec{x}\})$

C. Equality Axioms

(4) $x = x \qquad t = s \to s = t \qquad t = s \wedge s = r \to t = r$

[1] $B[y/x]$ is the formula B, where each free occurrence of x is replaced by y in the *usual* sense. The exact definition of $B[y/x]$ is straightforward but tedious.

We again want to stress the difference between $t[s/x]$ and $t\{s/x\}$. In the first expression substitution is an abbreviation in the metalanguage for the term t with all free occurrences of x replaced by s. $t\{s/x\}$, however, is a purely syntactical object where the substitution $\{s/x\}$ can only be evaluated by means of appropriate axioms to be described below.

We will often use the following abbreviations.

Definition 12 Let $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ and $\theta' = \{s_1/x_1, \ldots, s_n/x_n\}$ be substitutions and let r be an arbitrary term. Then we define:

1. $dom\,\theta \;:=\; \{x_1, \ldots, x_n\}$.
2. $\theta\downarrow \;:=\; t_1\downarrow \wedge \cdots \wedge t_n\downarrow$.
3. $\theta \simeq \theta' \;:=\; t_1 \simeq s_1 \wedge \cdots \wedge t_n \simeq s_n$.
4. $t/x \cdot \theta \;:=\; \{t/x\} \cup \theta^{-x}$, where θ^{-x} is θ with a possible binding for x deleted, i.e. $t/x \cdot \theta$ is the extension of $\{t/x\}$ by θ.
5. $\varepsilon \;:=\; \{\}$ (the empty substitution).

The set of free variables $fvar(t)$ of a $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ term t is computed in the obvious way. For reasons of completeness we give the exact inductive definition below.

Definition 13 ($fvar(t)$)

1. If t is the variable x then $fvar(t) \;:=\; \{x\}$.
2. If t is the term $(\lambda x.s)$ then $fvar(t) \;:=\; fvar(s) \setminus \{x\}$.
3. If t is the term $(t_1 t_2)$ then $fvar(t) \;:=\; fvar(t_1) \cup fvar(t_2)$.
4. If t is the term $(s\theta)$ and $\theta = \{s_1/x_1, \ldots, s_n/x_n\}$ then

$$fvar(t) \;:=\; (fvar(s) \setminus dom\,\theta) \;\cup \bigcup_{i \in I} fvar(s_i),$$

where $I = \{i : 1 \leq i \leq n \text{ and } x_i \in fvar(s)\}$.

Once $fvar(t)$ is defined, we get $fvar(A)$ as the set of free variables of a $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ formula A in the usual way.

We will sometimes use the notation θ^t for the substitution θ with all variable bindings s/y deleted for $y \notin fvar(t)$.

In the sequel we define a $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ formula $A\theta$ for each $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ formula A and each $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ substitution θ. The definition is by induction on the complexity of A.

Lemma 9 *We have for all* CL_p *terms* t, s *and all variables* x, y *such that* $x \neq y$ *and* $x \notin fvar(s)$*:*

$$\mathsf{CL}_p + (\text{Ext}) \vdash s\downarrow \to (\lambda^* x.t)[s/y] = \lambda^* x.t[s/y].$$

PROOF Assume $s\downarrow$. By Lemma 8 we have $(\lambda^* x.t)x \simeq t$ and $(\lambda^* x.t[s/y])x \simeq t[s/y]$. Since $s\downarrow$ and $x \neq y$ we can conclude that

$$(\lambda^* x.t)[s/y]x \simeq t[s/y] \simeq (\lambda^* x.t[s/y])x,$$

which by (Ext) immediately implies the claim of the lemma. □

As an immediate consequence of this lemma we get the following theorem, which is also stated in Moggi [22].

Theorem 10 *The systems* $\mathsf{CL}_p + (\text{Ext})$ *and* $\boldsymbol{\lambda}_p + (\text{Ext})$ *are equivalent with respect to the standard embeddings.*

3 The system $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$

In the following we introduce the system $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$, which is a modification of the system $\boldsymbol{\lambda}_p$ by explicit substitutions. The language of $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ is an extension of the language of $\boldsymbol{\lambda}_p$ by the set brackets $\{$, $\}$, the slash / and commas. The new symbols will be used in order to form finite sets of variable bindings, i.e. *substitutions*.

The terms (in the metalanguage: $r, s, t, \ldots$) and substitutions (in the metalanguage: $\theta, \sigma, \tau, \ldots$) of $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ are given by a simultaneous inductive definition.

Definition 11 ($\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ *terms and* $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ *substitutions*)

1. Every variable is a $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ term.
2. If t is a $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ term then $(\lambda x.t)$ is a $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ term.
3. If s and t are $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ terms then $(s \circ t)$ is a $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ term.
4. If t is a $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ term and if θ is a $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ substitution then $(t\theta)$ is a $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ term.
5. If $t_1, \ldots, t_n$ are $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ terms and if $x_1, \ldots, x_n$ are variables with $x_i \neq x_j$ for $1 \leq i < j \leq n$ then $\{t_1/x_1, \ldots, t_n/x_n\}$ is a $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ substitution.

In the following we will often write $t\theta\sigma$ instead of $(t\theta)\sigma$.

The formulas of $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ are defined in exactly the same way as the formulas of $\boldsymbol{\lambda}_p$, e.g. we have an atomic formula $t\downarrow$ for each $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ term t. In the following we often speak of terms, substitutions and formulas instead of $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ terms, $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ substitutions and $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ formulas.

2. $\mathbf{CL}_p \vdash \lambda^* x.t\downarrow$.

3. $\mathbf{CL}_p \vdash (\lambda^* x.t)x \simeq t$.

4. $\mathbf{CL}_p \vdash s\downarrow \to (\lambda^* x.t)s \simeq t[s/x]$.

PROOF 1.-3. are proved by induction on the complexity of t. 4. is a direct consequence of 3. $\square$

In the context of a total logic one normally defines $\lambda^* x.t := \mathbf{k}t$, if $x \notin fvar(t)$. So we have e.g. $\lambda^* x.(yz) = \mathbf{k}(yz)$. The example shows that $\lambda^* x.t\downarrow$ does not hold for the usual definition of λ abstraction.

As already mentioned, the λ abstraction of Definition 7 behaves very bad as far as substitution in λ expressions is concerned. For a usual λ abstraction we have

$$(\lambda x.t)[s/y] = \lambda x.t[s/y], \qquad (\star)$$

provided that $x \neq y$ and $x \notin fvar(s)$. This property fails for the λ^*defined above: We have e.g. $(\lambda^* x.y)[zz/y] = \mathbf{k}(zz)$, but $\lambda^* x.zz = \mathbf{s}(\mathbf{k}z)(\mathbf{k}z)$.

The fact that the substitution property $(\star)$ does not hold for the λ abstraction of Definition 7 is not just a technical inconvenience but has rather strong consequences for the system $\boldsymbol{\lambda}_p$ as a constructive framework for partial functions. Since $(\star)$ trivially holds in $\boldsymbol{\lambda}_p$, the standard embedding of $\boldsymbol{\lambda}_p$ into $\mathbf{CL}_p$ fails. As a consequence the two partial models of $\mathbf{CL}_p$ described above are no longer models of $\boldsymbol{\lambda}_p$. For example, it is easy to see that $(\star)$ fails in the recursion theoretic model.

An illustrative consequence of the stronger substitution concept of $\boldsymbol{\lambda}_p$ is a very weak form of (ξ), which is derivable in $\boldsymbol{\lambda}_p$. Let s, t be $\boldsymbol{\lambda}_p$ terms and x be a variable with $x \notin fvar(s) \cup fvar(t)$. Then it is easy to see that the principle

$$s = t \to \lambda x.s = \lambda x.t \qquad (\star\star)$$

is derivable in $\boldsymbol{\lambda}_p$ *only* from (β) and the fact that equality respects application. For $(s = t)$ implies $(\lambda y.\lambda x.y)s \simeq (\lambda y.\lambda x.y)t$ and hence we can conclude by the β axiom

$$\lambda x.s = (\lambda y.\lambda x.y)s = (\lambda y.\lambda x.y)t = \lambda x.t.$$

Note that in the argument above, we pushed the substitutions $[s/y]$ and $[t/y]$ inside the abstraction $(\lambda x.y)$. Obviously $(\star\star)$ does not hold in the recursion theoretic model. It is also clear that $(\star\star)$ has to be rejected from a strongly intensional point of view.

It should be stressed that the problems described above completely disappear in the presence of the extensionality axiom (Ext). In particular $(\star)$ holds in $\mathbf{CL}_p + (\mathrm{Ext})$.

The recursion theoretic model PRO. The universe of the model *PRO* of partial recursive operations consists of the set of natural numbers ω. Application $\circ$ is interpreted as partial recursive function application, i.e. $x \circ y$ means $\{x\}(y)$ in *PRO*, where $\{x\}$ is a standard enumeration of the partial recursive functions. It is easy to find appropriate interpretations of $\mathbf{k}$ and $\mathbf{s}$ such that the axioms of a partial combinatory algebra are satisfied. *PRO* provides a natural example of a domain where objects may be programs as well as inputs to programs. The model underlines the constructive and operational character of applicative theories.

The normal term model CNT. This model is based on standard notions of term reduction for combinatory logic, i.e. $\mathbf{k}t_1t_2$ reduces to t_1 and $\mathbf{s}t_1t_2t_3$ reduces to $t_1t_3(t_2t_3)$. The universe of *CNT* consists of all closed $\mathbf{CL}_p$ terms in normal form, $\mathbf{k}$ and $\mathbf{s}$ are interpreted by themselves and $t_1 \circ t_2$ means $\mathit{InFirst}(t_1t_2)$. $\mathit{InFirst}(t_1t_2)$ denotes the uniquely determined normal term s provided that t_1t_2 can be reduced to s according to the *leftmost minimal strategy*, $\mathit{InFirst}(t_1t_2)$ is undefined otherwise. Using the leftmost minimal strategy, at each stage of a reduction the leftmost minimal redex is contracted, where a redex is called *minimal*, if it does not contain any other redexes. It is necessary to use the leftmost minimal strategy in order to be consistent with the strictness axioms of $\mathbf{CL}_p$. The model *CNT* provides us with another interesting operational semantics of $\mathbf{CL}_p$. For a detailed description of *CNT* the reader is referred to Beeson [3], p. 119 ff.

As we will see below, the models just described cannot be made into models of the system $\boldsymbol{\lambda}_p$. This is due to a stronger concept of substitution, which is inherent in the partial λ calculus $\boldsymbol{\lambda}_p$.

Our next aim is to code λ abstraction in $\mathbf{CL}_p$. We have to be careful in defining it in the context of the logic of partial terms, because we want $\lambda x.t$ to be defined for each term t. As we will see below this modified λ abstraction will have very unpleasant properties as far as substitution is concerned.

Definition 7 ($\lambda^* x.t$)
For each term t of $\mathbf{CL}_p$ a term $\lambda^* x.t$ is defined by induction on the complexity of t.

1. If t is the variable x then $\lambda^* x.t := \mathbf{skk}$.
2. If t is a variable different from x or a constant then $\lambda^* x.t := \mathbf{k}t$.
3. If t is the term (t_1t_2) then $\lambda^* x.t := \mathbf{s}(\lambda^* x.t_1)(\lambda^* x.t_2)$.

Lemma 8 *We have for all* $\mathbf{CL}_p$ *terms* t *and* s*:*

1. $\mathit{fvar}(\lambda^* x.t) = \mathit{fvar}(t) \setminus \{x\}$.

D. Strictness Axioms

(6) $x\downarrow$

(7) $t_1 = t_2 \to t_1\downarrow \wedge\, t_2\downarrow \qquad t_1t_2\downarrow \to t_1\downarrow \wedge\, t_2\downarrow$

(8) $\mathsf{k}\downarrow \qquad \mathsf{s}\downarrow \qquad \mathsf{s}xy\downarrow$

E. Partial Combinatory Algebra

(9) $\mathsf{k}xy \simeq x$

(10) $\mathsf{s}xyz \simeq xz(yz)$

F. Rules of Inference

(11) $\dfrac{A \quad A \to B}{B}$

(12) $\dfrac{A \to B}{\exists x A \to B} \qquad \dfrac{A \to B}{A \to \forall x B}$

In the inference rules (12) x does not appear free in the conclusion.

Definition 6 The system $\boldsymbol{\lambda}_p$ is formulated in the language of $\boldsymbol{\lambda}_p$ and contains the same axioms and rules of inference as CL_p except that the strictness axioms (8) are replaced by

$$(\lambda x.t)\downarrow$$

and the axioms of a partial combinatory algebra are replaced by the β axiom

$$(\beta) \quad (\lambda x.t)y \simeq t[y/x].$$

It is important to notice that in the system $\boldsymbol{\lambda}_p$ we do not have the partial analogue of the rule (ξ),

$$(\xi) \quad \frac{t \simeq s}{\lambda x.t = \lambda x.s}.$$

In the presence of (ξ) it is a priori hopeless to embed λ calculus into combinatory logic. Nevertheless, we will study fully extensional versions of CL_p and $\boldsymbol{\lambda}_p$ respectively, i.e. we will consider the strong extensionality axiom (Ext),

$$(\text{Ext}) \quad \forall x(fx \simeq gx) \to f = g.$$

Let us briefly sketch some models of CL_p. As we are mainly interested in partiality, we will only discuss truly partial models. Of course, there are many models of CL_p where application is a total operation, e.g. each model of the λ calculus is a model of CL_p. In the following we give two partial models of CL_p.

The formulas of $\boldsymbol{\lambda}_p$ are defined in exactly the same way as the $\mathbf{CL}_p$ formulas.

If t is a term of $\mathbf{CL}_p$ or $\boldsymbol{\lambda}_p$ then $fvar(t)$ denotes the set of free variables of t. As usual, $t[s_1/x_1, \ldots, s_n/x_n]$ is the term t in which the free occurrences of $x_1, \ldots, x_n$ are simultaneously substituted by $s_1, \ldots, s_n$. If t is a $\boldsymbol{\lambda}_p$ term then in the case of variable collisions bound variables have to be renamed. Analogously, $fvar(A)$ and $A[s_1/x_1, \ldots, s_n/x_n]$ are defined.

Definition 4

1. $(A \leftrightarrow B) \;:=\; ((A \to B) \wedge (B \to A))$.
2. $(t \simeq s) \;:=\; ((t\!\downarrow \vee\, s\!\downarrow) \to t = s)$.

Together with the axioms stated below it will become clear that $(t \simeq s)$ is equivalent to

$$(t\!\downarrow \wedge\, s\!\downarrow \wedge\, t = s) \vee (\neg t\!\downarrow \wedge\, \neg s\!\downarrow),$$

i.e. $\simeq$ is a partial equality relation as it is used e.g. in a recursion-theoretic framework.

In the following we give the axioms and rules of inference of the systems $\mathbf{CL}_p$ and $\boldsymbol{\lambda}_p$ respectively. The logic of both $\mathbf{CL}_p$ and $\boldsymbol{\lambda}_p$ is the classical logic of partial terms due to Beeson [4] and [3]. It corresponds to $\mathbf{E}^+$ logic as it is discussed in Troelstra and van Dalen [25]. All our results also hold if *intuitionistic* logic is chosen as a basis of $\mathbf{CL}_p$ and $\boldsymbol{\lambda}_p$ respectively.

Definition 5 The system $\mathbf{CL}_p$ is formulated in the language of $\mathbf{CL}_p$ and contains the following list of axioms and rules of inference.

A. Propositional Logic

(1) Some complete axiom schemes of classical propositional logic

B. Quantifier Axioms

(2) $\forall x A \wedge t\!\downarrow \to A[t/x]$

(3) $A[t/x] \wedge t\!\downarrow \to \exists x A$

C. Equality Axioms

(4) $x = x \qquad t = s \to s = t \qquad t = s \wedge s = r \to t = r$

(5) $t_1 = s_1 \wedge t_2 = s_2 \to t_1 t_2 \simeq s_1 s_2$

2 The systems CL_p and $\boldsymbol{\lambda}_p$

Let us first define partial combinatory logic CL_p and the usual partial λ calculus $\boldsymbol{\lambda}_p$, without the rule ξ. The language of CL_p includes an infinite list of object variables (in the metalanguage: $x, y, z, f, g, h, u, v, w, \ldots$), constants k and s (partial combinatory algebra), the binary function symbol $\circ$ (application), the equality symbol $=$, the symbol $\downarrow$ (defined) and the usual propositional connectives and first order quantifiers. The language of $\boldsymbol{\lambda}_p$ contains the same symbols except that k and s are replaced by the abstractor λ.

The terms of CL_p and $\boldsymbol{\lambda}_p$ (in the metalanguage: $r, s, t, \ldots$) are given by the following definitions.

Definition 1 (CL_p *terms*)

1. Every variable is a CL_p term.
2. k and s are CL_p terms.
3. If s and t are CL_p terms then $(s \circ t)$ is a CL_p term.

Definition 2 ($\boldsymbol{\lambda}_p$ *terms*)

1. Every variable is a $\boldsymbol{\lambda}_p$ term.
2. If t is a $\boldsymbol{\lambda}_p$ term then $(\lambda x.t)$ is a $\boldsymbol{\lambda}_p$ term.
3. If s and t are $\boldsymbol{\lambda}_p$ terms then $(s \circ t)$ is a $\boldsymbol{\lambda}_p$ term.

In the following we write (st) for $(s \circ t)$. Additionally, we adopt the convention of association to the left, i.e. $t_1 t_2 t_3 \ldots t_n$ stands for $(\ldots((t_1 t_2)t_3)\ldots t_n)$. Finally, we often write $(\lambda x_1 \ldots x_n.t)$ instead of $\lambda x_1.(\lambda x_2.(\ldots(\lambda x_n.t)\ldots))$.

The formulas of CL_p (in the metalanguage: $A, B, C, \ldots$) are defined in the obvious way.

Definition 3 (CL_p *formulas*)

1. If s and t are CL_p terms then $(s = t)$ is a CL_p formula.
2. If t is a CL_p term then $t\downarrow$ is a CL_p formula.
3. If A and B are CL_p formulas then $\neg A$, $(A \vee B)$, $(A \wedge B)$ and $(A \to B)$ are CL_p formulas.
4. If A is a CL_p formula then $\exists x A$ and $\forall x A$ are CL_p formulas.

In the following we propose a modification of the partial λ calculus, which can be embedded into partial combinatory logic via a natural embedding. As a consequence, this weakened form of the partial λ calculus has all the partial models, which we have for partial combinatory logic. In particular, it is possible to determine upper bounds of weak systems of explicit mathematics based on our modified version of the partial λ calculus using the recursion-theoretic model.

The novel point of our system will be the use of *explicit substitutions*. According to this approach substitution is no longer a notion of the metalanguage, but an operation axiomatized in the theory under consideration itself. If $t, s_1, \ldots, s_n$ are terms and θ is the substitution $\{s_1/x_1, \ldots, s_n/x_n\}$ then $t\theta$ is no longer an abbreviation in the metalanguage for the term t with the variables x_i simultaneously replaced by the terms s_i, but a purely syntactical object. The evaluation of θ has to be described by appropriate axioms or rules. So it is possible to provide a very controlled process of substitution. In particular, substitution can be axiomatized in a way, which is consistent with the recursion-theoretic model and partial combinatory logic, respectively. Hence a substitution θ can no longer be pushed inside an abstraction $(\lambda x.t)$.

The theory of explicit substitutions has been treated in the literature before, but from a different point of view. The main work has been done in the context of implementation of functional programming languages. A key reference is the paper by Abadi, Cardelli, Curien and Lévy [1]. Further investigations are presented in Hardin and Lévy [13] and Curien, Hardin and Lévy [5].

Very recently, Martin-Löf introduced a calculus of explicit substitutions in connection with his intuitionistic theory of types ([19]).

Let us briefly sketch the procedure of these investigations. In Section 2 we first introduce partial combinatory logic $\mathbf{CL}_p$ and the (usual) partial λ calculus (without ξ) $\boldsymbol{\lambda}_p$. In particular, we recapitulate Beeson's logic of partial terms. After having sketched some interesting partial models of $\mathbf{CL}_p$, we discuss the substitution problems, which prevent the embedding of $\boldsymbol{\lambda}_p$ into $\mathbf{CL}_p$. In Section 3 we give a detailed formulation of the system $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$, which is a modification of $\boldsymbol{\lambda}_p$ by explicit substitutions. The system incorporates an adaptation of Beeson's logic of partial terms to the framework of explicit substitutions and rules to evaluate substitutions, of course. We further show that $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ is embeddable into $\mathbf{CL}_p$ via a natural embedding and that $\mathbf{CL}_p$ is also contained in $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ via the standard embedding. In Section 4, finally, we study the reduction relation on $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$ terms corresponding to the system $\boldsymbol{\lambda}_p\boldsymbol{\sigma}$. We give a long and tedious proof for the Church Rosser property of this relation.

account to Bishop's style of constructive mathematics (BCM). More recently Feferman's systems of explicit mathematics were used to develop a unitary axiomatic framework for representing programs, stating properties of programs and proving properties of programs. The programs considered are taken from functional programming languages, which are mainly based on the untyped λ calculus. Important references for the use of systems of explicit mathematics in the context of functional programming are Feferman [8], [9], [10], Jäger [18], [17] and Marzetta [20], [21].

As far as the explicit representation of a theory with partial self-application in the previous literature is concerned, people either took partial combinatory logic or the partial λ calculus (without the rule ξ) as the applicative basis. The former possibility was chosen in [3], [6], [7], [11], [18], [17], [21], [24], the latter in [8], [9], [10] and [20]. At first sight, these two approaches seem to be completely equivalent, and sometimes they are treated as such in the literature. But they are only equivalent in the presence of a *total* logic, since then λ calculus (without ξ) can be embedded into combinatory logic and vice versa. For a detailed discussion of the total case also in the context of reductions, the reader is referred to Hindley [15] and Hindley and Longo [16].

The situation changes drastically if one considers a *partial* application operation. Then the partial λ calculus (without ξ) is no longer embeddable into partial combinatory logic. This is due to the fact that the coding of λ abstraction in the context of partial combinatory logic is more complicated than usual. The modified definition of λ does not permit to push a substitution θ inside an abstraction $(\lambda x.t)$, a principle, which is valid for the partial λ calculus for obvious reasons. For example, the terms $(\lambda x.y)[zz/y]$ and $(\lambda x.zz)$ are not equal in partial combinatory logic. Hence a stronger concept of substitution in the partial λ calculus makes the embedding of the latter into partial combinatory logic fail. For the same reason, the standard recursion-theoretic model of partial combinatory logic is no longer a model of the partial λ calculus.

As a consequence, it is not possible to determine proof-theoretical upper bounds of weak applicative theories based on the partial λ calculus by means of the recursion-theoretic model, as it can be done for the corresponding systems based on partial combinatory logic. Although the upper bounds of those systems can be determined by formalizing a – perhaps less natural – *total* term model, the adequacy of the partial λ calculus as a constructive framework for partial functions is seriously put into question. The system simply does not seem to have any reasonable models with a perspicuous constructive meaning that are truly partial. Not only the recursion-theoretic model but also other partial models of partial combinatory logic do not have their counterparts as models of the partial λ calculus.

Partial Applicative Theories and Explicit Substitutions

Thomas Strahm
Institut für Informatik und angewandte Mathematik, Universität Bern
Länggassstrasse 51, CH–3012 Bern, Switzerland
⟨strahm@iam.unibe.ch⟩

June 24, 1993

Abstract

Systems based on theories with partial self-application are relevant to the formalization of constructive mathematics and as a logical basis for functional programming languages. In the literature, they are either presented in the form of partial combinatory logic or the partial λ calculus, and sometimes these two approaches are erroneously considered to be equivalent.
In this paper we address some defects in the partial λ calculus as a constructive framework for partial functions. In particular, the partial λ calculus is not embeddable into partial combinatory logic and it lacks the standard recursion-theoretic model. The main reason is a concept of substitution, which is not consistent with a strongly intensional point of view. We design a weakening of the partial λ calculus, which can be embedded into partial combinatory logic. As a consequence, the natural numbers with partial recursive function application *are* a model of our system. The novel point will be the use of *explicit substitutions*, which have previously been studied in the literature in connection with the implementation of functional programming languages.

Keywords: explicit mathematics, logic of partial terms, partial lambda calculus, partial combinatory logic, explicit substitutions

1 Introduction

Partial applicative theories form the basis of various formal systems for constructive mathematics and functional programming. It was Feferman in [6] and [7] who introduced partial applicative theories of operations and classes in order to give a logical