Confluence of an extension of combinatory logic by Boolean constants

Czajka, ukasz

Published in:
2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017

DOI:
10.4230/LIPIcs.FSCD.2017.14

Publication date:
2017

Document version
Publisher's PDF, also known as Version of record

Document license:
CC BY

Citation for published version (APA):
Confluence of an Extension of Combinatory Logic by Boolean Constants

Łukasz Czajka

DIKU, University of Copenhagen, Copenhagen, Denmark
luta@di.ku.dk

Abstract

We show confluence of a conditional term rewriting system CL-pc\(^1\), which is an extension of Combinatory Logic by Boolean constants. This solves problem 15 from the RTA list of open problems. The proof has been fully formalized in the Coq proof assistant.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases combinatory logic, conditional linearization, unique normal form property, confluence

Digital Object Identifier 10.4230/LIPIcs.FSCD.2017.14

1 Introduction

Combinatory Logic is a term rewriting system defined by two rules:

\[
\begin{align*}
\text{K}xy & \rightarrow x \\
\text{S}xyz & \rightarrow x(zy) 
\end{align*}
\]

Using only \(S\) and \(K\), it is possible to encode natural numbers via Church numerals. Any computable function may then be represented by a term in the system. However, a conditional \(C\) encoded in this way does not have a desirable property that \(Ct_1t_2t_2 = t_2\) if \(t_1\) encodes neither true nor false. It is therefore interesting to investigate extensions of Combinatory Logic incorporating a conditional directly. Perhaps the most natural such extension is CL-pc:

\[
\begin{align*}
\text{K}xy & \rightarrow x \\
\text{CT}xy & \rightarrow x \\
\text{C}zxy & \rightarrow x \quad \text{if } x = y \\
\text{S}xyz & \rightarrow x(zy) \\
\text{CF}xy & \rightarrow y
\end{align*}
\]

The system CL-pc is known to be not confluent [7]. One may thus try other ways of adding a conditional and Boolean constants to Combinatory Logic.

We show confluence of a conditional term rewriting system CL-pc\(^1\) defined by the rules:

\[
\begin{align*}
\text{K}xy & \rightarrow x \\
\text{CT}xy & \rightarrow x \\
\text{C}xxy & \rightarrow x \leftrightarrow x = y \\
\text{S}xyz & \rightarrow x(zy) \\
\text{CF}xy & \rightarrow y
\end{align*}
\]

Confluence of this system\(^1\) appears as problem 15 on the RTA list of open problems [5].

The equality in the side condition for the third rule for \(C\) in CL-pc\(^1\) refers to equality in the system CL-pc\(^1\) itself, thus the definition is circular. This circularity is an essential property of CL-pc\(^1\) which distinguishes it from CL-pc.

\(^*\) Supported by Marie Skłodowska-Curie action “InfTy”, program H2020-MSCA-IF-2015, number 704111.
\(^1\) Strictly speaking, in the literature the systems CL-pc, CL-pc\(^1\) and CL-pc\(^L\) also contain the rule \(Ix \rightarrow x\). This rule could be added to our definitions without significantly changing the proofs. However, this would increase the number of cases to consider, making the proofs less readable. The formalization of our results uses the definitions from the literature.
A system related to CL-pc$^l$ is CL-pc$^{1\!\!}_L$, which consists of all rules of CL-pc$^1$ plus:

$$C_{zxy} \rightarrow y \iff x = y$$

It is known that CL-pc$^{1\!\!}_L$ is confluent [4]. However, the confluence proof in [4] essentially depends on a “semantic” argument to first establish $T \not\equiv_{CL-pc^1} F$. We provide a “syntactic” proof of confluence of both CL-pc$^1$ and CL-pc$^{1\!\!}_L$.

The systems CL-pc$^1$ and CL-pc$^{1\!\!}_L$ are conditional linearizations of CL-pc. The notion of conditional linearization was introduced in the hope of providing a simpler proof of Chew’s theorem [2, 9] which states that all compatible term rewriting systems have the unique normal form (UN) property. Compatibility imposes certain restrictions on the term rewriting system, but it does not require termination or left-linearity. In particular, Chew’s theorem is applicable to many term rewriting systems which are not confluent. For instance, CL-pc satisfies the conditions of Chew’s theorem, but it is not confluent. As shown in [4], to prove the unique normal form property of a term rewriting system, it suffices to prove confluence of one of its conditional linearizations. The proof of Chew’s theorem in [9] is quite complicated and uses a related but different approach, relying on left-right separated conditional linearizations instead of the more straightforward ones from [4]. The original proof by Chew [2] uses yet another different but related method, but Chew’s proof was later found to contain a gap.

In general, the methods of the present paper are broadly related to the problem of establishing the UN property for classes of term rewriting systems which include non-left-linear non-confluent systems. Aside of Chew’s theorem, some other work in this direction has been carried out in e.g. [8, 12, 13, 10, 6].

In order to increase confidence in the correctness of the main result of this paper, we have formalized our proof of confluence of CL-pc$^1$ in the Coq proof assistant. The formalization is available online$^2$. It follows closely the development presented here. We used the CoqHammer [3] tool and the automated reasoning tactics included with it.

2 Proof overview

In this section we present an informal overview of the proof, trying to convey the underlying intuitions. Section 3 presents formal definitions of the notions informally motivated here, and Section 4 provides details of the proof itself.

We assume familiarity with basic term-rewriting [1, 11]. By $\rightarrow^*$ we denote the transitive-reflexive closure of a relation $\rightarrow$, by $\rightarrow^\equiv$ its reflexive closure, by $\leftrightarrow$ the symmetric closure, and by $=$ the reflexive-transitive-symmetric closure. We use $\equiv$ to denote identity of terms. By $\rightarrow^1$ we denote reduction to normal form, i.e., $t \rightarrow^1 s$ if $t \rightarrow^* s$ and $s$ is in normal form. By $\cdot$ we denote composition of relations, e.g. $t \rightarrow \cdot \leftarrow t'$. We use the standard notions of subterms and subterm occurrences, which could be formally defined by introducing the notion of positions. If $t$ is a redex, i.e. $t \equiv \sigma l$ for some term $l$ and substitution $\sigma$, then a subterm $s$ occurs below a variable position of the redex $t$ if $s$ occurs in a subterm of $t$ occurring at the position of a variable in $l$. The contraction in $t_1 \rightarrow t_2$ occurs at the root if $t_1$ is the contracted redex.

Let $u$ be a normal form w.r.t. a relation $\rightarrow$. The relation $\rightarrow$ (or the underlying rewrite system) is $u$-normal if for every $t$ such that $t = u$ we have $t \rightarrow^* u$.

---

$^2$ http://www.minuw.edu.pl/~lukaszcz/clc.tar.gz
The most difficult part of our confluence proof is to show that CL-pc^1 is F-normal (Lemma 27). The confluence of CL-pc^1 (and CL-pc^2) is then obtained by a relatively simple argument similar to the one used in [4] to derive the confluence of CL-pc^2 from T \neq CL-pc F.

An important observation is that q_1 =\_CL-pc^1 q_2 and q_1 =\_CL-pc q_2 are in fact equivalent (Lemma 2). Hence, we will use =\_CL-pc^1 and =\_CL-pc interchangeably. In particular, we actually prove that for any term q, if q =\_CL-pc F then q \rightarrow^* CL-pc^1 F.

A naive approach to prove this could be to proceed by induction on the length of the conversion q =\_CL-pc F. In the inductive step we would need to prove:
1. if q \rightarrow^* CL-pc^1 F and q \rightarrow CL-pc q’ then q’ \rightarrow^* CL-pc^1 F,
2. if q \rightarrow^* CL-pc^1 F and q CL-pc\leftarrow q’ then q’ \rightarrow^* CL-pc^1 F.

The second part is obvious, but the first one is hard. The difficulty stems from the existence of a non-trivial overlap between the rules for C. If t_1 =\_CL-pc^1 t_2 then CFt_1t_2 \rightarrow CL-pc^1 t_1 by the third rule of CL-pc^1 and CFt_1t_2 \rightarrow CL-pc t_2 by the second rule of CL-pc. We do not know enough about t_1 and t_2 to easily infer that they have a common reduct in CL-pc^1.

One may try to strengthen the inductive hypothesis in the hope of making the first part easier to prove. A naive attempt would be to claim that all reductions starting from q end in F, instead of claiming that some reduction ends in F. This would make the first part trivial, but the second one would not go through as this is false in general, e.g., consider KFO where \Omega \equiv (SI)(SII) and I \equiv SKK.

The idea is to consider, for a given conversion q =\_CL-pc F, a certain set S(q =\_CL-pc F) of reductions, all starting from q. The set S(q =\_CL-pc F) depends on the exact form of q =\_CL-pc F. Then our two parts of the proof for the inductive step become:
1. if S(q =\_CL-pc F) is nonempty and all reductions in it end in F, and q =\_CL-pc q’, then S(q’ =\_CL-pc F) is nonempty and all reductions in it end in F,
2. if S(q =\_CL-pc F) is nonempty and all reductions in it end in F, and q =\_CL-pc q’, then S(q’ =\_CL-pc F) is nonempty and all reductions in it end in F.

The hope is that if we define S(q =\_CL-pc F) appropriately, then showing both parts will become feasible.

Essentially, the set S(q =\_CL-pc F) will be encoded in the labeling of certain constants in q. The labels determine which contractions are permitted when a given constant appears as the leftmost constant in a redex^3. At present the author does not know an “explicit” characterization of the set of reductions S(q =\_CL-pc F) implicitly defined by the labelings described below.

Terms with the leftmost constant labeled will be called “significant”, or s-terms, whereas others will not contain any labels and will be called “insignificant”, or i-terms (cf. Definition 3). Reductions occurring in i-terms will be “insignificant”, or i-reductions. A “significant” contraction, or s-contraction, will be a contraction of a term with the leftmost constant labeled, in a way permitted by the label of the leftmost constant. Contraction of a redex in which the leftmost constant is not labeled is not permitted in s-contractions. See Definition 4. The intuition is that we do not need to care about the expansions and contractions occurring in “insignificant” subterms of a given term, since they cannot influence the s-reductions starting from this term and ending in F.

The set S(q =\_CL-pc F) will be encoded in a labeled variant^4 t of q, and it will consist of all s-reductions starting from t and ending in a normal form (w.r.t. s-contraction). Strictly

---

^3 E.g. in the redex CT_1t_2 the constant C is the leftmost constant.

^4 By a “labeled variant” of a term q we mean a term with certain constants labeled which is identical with q when the labels are “erased.”
Confluence of an Extension of Combinatory Logic by Boolean Constants

speaking, we have just silently shifted from considering contractions in “plain” terms of the system CL-\(pc^1\) to contractions in their labeled variants, in a different rewriting system which we have not yet defined. In particular, we will actually be interested in \(s\)-reductions ending in a labeled variant \(F_1\) of \(F\). However, it will be later shown that \(s\)-reductions defined on labeled terms may be “erased” to appropriate reductions in the system CL-\(pc^1\). In the next section we define the system CL-\(pc^\alpha\) (Definition 4) over labeled terms (Definition 3) which will give precise rules of \(s\)-contraction. In this section we only give informal motivations.

The labels constrain the ways in which \(s\)-redexes may be contracted and encode permissible \(s\)-reductions to \(F_1\). Each term decomposes into a “significant” prefix and an “insignificant” suffix (cf. Definition 8). The “significant” prefix contains all labeled constants and no unlabeled constants. The “insignificant” suffix consists of all “insignificant” subterms. All constants in the “insignificant” suffix are unlabeled. This is analogous to the existence of a needed prefix and a non-needed suffix in orthogonal TRSs [11, Section 9.2.2]. An “insignificant” subterm does not overlap with any needed redexes. In particular, it does not (cf. Definition 8).

1. if
2. if

We write \(t \rightarrow_s t'\) for one-step reduction in CL-\(pc^\alpha\). We use the abbreviation \(s\)-NF for CL-\(pc^\alpha\)-normal form. We write \(t \not\rightarrow_s F_1\) when, among other conditions to be defined later, \(t\) is complete, i.e. terminating and confluent, w.r.t. \(s\)-reductions with \(F_1\) as the normal form (cf. Definition 8).

With the set \(S(q =_{CL-pc} F)\) coded by labels, the two parts of the inductive step become:

1. if \(q\) is a labeled variant of \(q\) such that \(t \not\rightarrow_{F_1} F_1\), and \(q \rightarrow_{CL-pc} q'\), then there exists a labeled variant \(t'\) of \(q'\) such that \(t' \not\rightarrow_{F_1} F_1\) (cf. Corollary 15),

2. if \(q\) is a labeled variant of \(q\) such that \(t \not\rightarrow_{F_1} F_1\), and \(q \leftarrow_{CL-pc} q'\), then there exists a labeled variant \(t'\) of \(q'\) such that \(t' \not\rightarrow_{F_1} F_1\) (cf. Corollary 25).

Now we provide some explanations on how the terms will be labeled. For this purpose we analyze why the second part fails when we take \(S(q =_{CL-pc} F)\) to be the set of all reductions starting from \(q\). We indicate how to introduce the labeled variants so as to make the second part go through while still retaining the feasibility of showing the first part.

Suppose \(q \leftarrow_{CL-pc} q'\) at the root and we have already decided on the labeled variant \(t\) of \(q\). We need to decide on a labeled variant \(t'\) of \(q'\), and assign appropriate meaning to the labels, in such a way that the second part goes through. In short, in \(t'\) we preserve the labelings of the subterms of \(q'\) which are copied to \(q\) in \(q' \rightarrow_{CL-pc} q\), we do not label the new subterms of \(q'\) which are erased in \(q' \rightarrow_{CL-pc} q\) (they become \(i\)-terms), and we ensure that \(i\)-terms and cannot influence any \(s\)-reduction from \(t'\) to \(F_1\). First of all, if \(t\) is an \(i\)-term, i.e., \(t \equiv q\), then we may take \(t' \equiv q'\). So assume \(t\) is an \(s\)-term. Then there are the following possibilities.

- If \(q' \equiv CTq_0q \rightarrow_{CL-pc} q\) then \(q_0\) is a new subterm. We take \(t' \equiv C_1t_1t_q0\). The labeling \(C_1\) of \(C\) will be interpreted as not permitting contraction by the third rule, i.e., in CL-\(pc^\alpha\) we will only have the rules \(C_1T_1x\rightarrow x\) and \(C_1F_1xy \rightarrow y\). This ensures that \(q_0\) gets erased in every \(s\)-reduction of \(t'\) to \(F_1\).

- The case when \(q' \equiv CFq_0q \rightarrow_{CL-pc} q\) is analogous: we take \(t' \equiv C_1F_1t_q0t\).

- If \(q' \equiv Cq_0q \rightarrow_{CL-pc} q\) by the third rule, then \(q_0\) is a new term. We take \(t' \equiv C_2q_0tt\). In

...
For simplicity assume that there is only one position in \( q \) created in an expansion cannot later on appear in place of a "significant" term as a result of the problematic non-trivial overlap occurring in \( \text{CL-pc} \).

By introducing the tuples in the labelings we in essence put constraints on the order of every possible labeling of the right side \( S \). We will thus introduce a new labeling of \( q \) in the system \( \text{CL-pc}^* \).

A solution is to remember both labeled variants of \( q_3 \). So the labeled variant of \( q' \) would be e.g. \( S_1 t_1 (t_3, t_4) \). In \( \text{CL-pc}^* \) the rule for \( S_1 \) would be

\[
S_1 x_1 x_2 (x_3, x_4') \rightarrow x_1 x_3 (x_2 x_4').
\]

However, once we introduce such pairs, terms of the form \( S_1 t_1 t_2 (t_3, t_4') \) may appear in the terms being expanded. This is not a problem for any of the rules of \( \text{CL-pc}^* \) except the rule for \( S \), because the right sides of all other rules are variables.

Consider for instance \( q \equiv q_0 q_1 (S_1 q_2 q_3) \). Now the term \( q_3 \) has three possibly distinct labeled variants, and we need to remember all of them in a tuple. We will thus introduce a new labeling of \( S \) for every possible labeling of the right side \( x_2 (yz) \) of the rule for \( S \) in the system \( \text{CL-pc}^* \).

By introducing the tuples in the labelings we in essence put constraints on the order in which \( s \)-redexes may be contracted (think of all reductions inside a tuple as "really" occurring after the surrounding S-redex is contracted). At present the author does not know a precise "explicit" characterization of these constraints.

Note that by labeling \( C \) differently in \( \text{CF} q_1 q_2 \) and \( C q_0 q q \) we effectively eliminated in \( \text{CL-pc}^* \) the problematic non-trivial overlap occurring in \( \text{CL-pc}^1 \). Now a new "insignificant" term created in an expansion cannot later on appear in place of a "significant" term as a result of an "incompatible" contraction. A redex inside an "insignificant" subterm cannot suddenly become needed in an \( s \)-reduction - it is erased in any \( s \)-reduction to normal form.

We also need to ensure that we can handle the first part of the inductive step when \( q \rightarrow \text{CL-pc} \). Suppose \( t \) is the labeled variant of \( q \). We need to find a labeled variant for \( q' \).

For simplicity assume that there is only one position in \( t \) which corresponds to the position of the contraction in \( q \). If the contraction occurs inside an \( i \)-term in \( t \), then it does not matter and we may label \( q' \) in the same way as \( q \). If an \( s \)-term is contracted in a way permitted for significant contraction, then it is also obvious how to label \( q' \) - just take the labeled variant of \( q' \) to be the reduct of the labeled variant of \( q \). But what if neither of the two holds?

For instance, what if \( t \equiv C_1 t_0 t_1 t_2 \) but \( q \equiv C_0 q q q' \rightarrow \text{CL-pc}^1 \)? This possibility is not problematic, provided that \( t_0 \rightarrow^* \text{T}_1 \) or \( t_0 \rightarrow^* \text{F}_1 \), which will be the case because \( t_0 \) was
Confluence of an Extension of Combinatory Logic by Boolean Constants

“obtained” from $T_1$ or $F_1$ by a conversion with the intermediate terms labeled appropriately (cf. 2 in Definition 8 and 6 in Lemma 9). If e.g. $t_0 \rightarrow^*_T T_1$ then we take $t_1$ to be the labeling of $q'$. We then have $C_1 t_0 t_1 t_2 \rightarrow^*_T C_1 T_1 t_1 t_2$ and the contraction $C_1 t_1 t_2 \rightarrow_s t_1$ is permitted for “significant” contractions.

The last problematic case is when e.g. $t \equiv C_2 F_1 t_2$ is the labeling of $q \equiv CF_1 q'$, and $q \rightarrow_{CL-pc} q'$ by the second rule. However, because $C_2 F_1 t_2$ was “obtained” from $C_2 q' t'$ we will have $|t_1| =_{CL-pc} |t_2|$ (cf. 3 in Definition 8). Then the second rule for $C_2$ in $CL-pc^*$ is applicable and we may take $t_2$ as the labeling of $q'$.

3 Definitions

This section is devoted to fixing notation and introducing definitions of various technical concepts. First, we clarify the formal definition of conditional term rewriting systems. For more background on conditional rewriting see e.g. [11].

Definition 1. A conditional rewrite rule is a rule of the form $l \rightarrow r \Leftarrow P(x_1,\ldots,x_n)$, where $l$ is not a variable, $\text{Var}(r) \subseteq \text{Var}(l)$, $x_1,\ldots,x_n \in \text{Var}(l)$, and $P(x_1,\ldots,x_n)$ is the condition of the rule, with $P$ a fixed predicate on terms. The predicate $P$ may refer to the conversion relation $\equiv$ of the conditional term rewriting system being defined. A term $t$ is a redex (contractum) by this rule if there is a substitution $\sigma$ such that $t \equiv \sigma l (t \equiv \sigma r)$ and $P(\sigma(x_1),\ldots,\sigma(x_n))$. A conditional term rewriting system $R$ is a set of conditional rewrite rules. Because the conditions in the rules may refer to the conversion relation of $R$, the definition is circular. Formally, an $R$-contraction $q \rightarrow_R q'$ is defined in the following way. Define $R_0$ to be the system $R$ but using the equality relation in place of $\equiv$ in the conditions, and $R_{n+1}$ to be the system $R$ with the conversion relation $\equiv_{R_n}$ of $R_n$ used in place of $\equiv$. We then define $q \rightarrow_R q'$ to hold if there is $n \in \mathbb{N}$ with $q \rightarrow_{R_n} q'$. The least such $n$ is called the level of the contraction. If the conditions are continuous w.r.t. $\equiv$ then the relation $\rightarrow_R$ is a fixpoint of the above construction, i.e., it is the contraction relation of the system $R_\infty$ which uses $\equiv_R$ in place of $\equiv$. Let $\sim$ be a binary relation on terms. If for any substitution $\sigma$ such that $P(\sigma(x_1),\ldots,\sigma(x_n))$ holds, and any $\sigma'$ such that $\sigma(x) \sim \sigma'(x)$ for all variables $x$, also $P(\sigma'(x_1),\ldots,\sigma(x_n))$ holds, then the condition $P(x_1,\ldots,x_n)$ is stable under $\sim$.

The following is a simple but crucial observation, which implies that it suffices to consider conversions in $CL-pc$. A generalization of this fact was already shown in [4, Lemma 3.7]. The proof is by induction on the maximum level of the contractions/expansions in $q =_{CL-pc^*} q'$.

Lemma 2. The following are equivalent: $q =_{CL-pc} q'$, $q =_{CL-pc^*} q'$, and $q =_{CL-pc^*} q'$.

Definition 3. We define insignificant terms, or $i$-terms, to be the terms of $CL-pc^i$, i.e., terms over the signature $\Sigma = \{@, C, T, F, K, S\}$ where $@$ is a binary function symbol and the other symbols are constants. We write $t_1 t_2$ instead of $@t_1 t_2$. The set of labeled terms, or $l$-terms, is the set of terms over the signature consisting of the symbols of $\Sigma$, the labeled constants $C_1, C_2, T_1, F_1, K_1$ and $S^{n_0,\ldots,n_k}$ for each $k, n_1,\ldots,n_k \in \mathbb{N}_+$, and an $n$-ary function symbol $P^n$ for each $n \in \mathbb{N}_+$. We write $(t_1,\ldots,t_n)$ instead of $P^n(t_1,\ldots,t_n)$. We adopt the convention $(t) \equiv t$. If $t \equiv (t_1,\ldots,t_n)$ with $n > 1$, then we say that $t$ is a tuple of length $n$. Note that $(t) \equiv t$ is just a notational convention. We say that $(t_1,\ldots,t_n)$ is a tuple only when $n > 1$.

An erasure of an $l$-term is defined as follows:

- an $i$-term is an erasure of itself,
- $C$ is an erasure of $C_1$ and $C_2$; $T$ is an erasure of $T_1$; $F$ is an erasure of $F_1$; $K$ is an erasure of $K_1$; $S$ is an erasure of $S^{n_0,\ldots,n_k}$,
if \( q_1, q_2 \) are erasures of \( t_1, t_2 \), respectively, then \( q_1q_2 \) is an erasure of \( t_1t_2 \).
- if \( q_i \) is an erasure of \( t_i \), for some \( 1 \leq i \leq n \), then \( q_i \) is an erasure of \( t_1, \ldots, t_n \).

The leftmost erasure of \( t \), denoted \( \|t\| \), is the erasure in which we always choose \( i = 1 \) in the last point above. We write \( t \succ q \) if every erasure of \( t \) is identical with \( q \).

We define significant terms, or \( s \)-terms, inductively.
- Any labeled constant is an \( s \)-term.
- If \( t_1 \) is an \( s \)-term and \( t_2 \) is an \( l \)-term, then \( t_1t_2 \) is an \( s \)-term.

In other words, an \( s \)-term is an \( l \)-term whose leftmost constant is labeled.

In what follows \( t, t_1, t_2, r, r_1, r_2, s, s_1, \) etc. stand for \( l \)-terms; and \( q, q_1, q_2, \) etc. stand for \( i \)-terms; unless otherwise qualified. Also, whenever we talk about terms without further qualification, we implicitly assume them to be \( l \)-terms.

**Definition 4.** The system CL-pc\(^s\) is defined by the following significant reduction rules:

\[
\begin{align*}
C_1 T_1 xy & \to x \\
C_2 z xy & \to x \iff |x| =_{\text{CL-pc}\^1} |y| \\
K_1 x y & \to x \\
S^n x(y_1, \ldots, y_k)(\vec{z}_0, \ldots, \vec{z}_k) & \to x(\vec{z}_0)((y_1(\vec{z}_1)), \ldots, (y_k(\vec{z}_k))) \iff \varphi
\end{align*}
\]

where \( \varphi \equiv |z_{i,j}| =_{\text{CL-pc}\^1} |z_{i',j'}| \) for \( i, i' = 0, \ldots, k \), \( j = 1, \ldots, n_i \), \( j' = 1, \ldots, n_{i'} \), and \( |y_i| =_{\text{CL-pc}\^1} |y_j| \) for \( i, j = 1, \ldots, k \),

and \( \vec{n} \) stands for \( n_0, \ldots, n_k \), and \( \vec{z} \) stands for \( z_{1,1}, \ldots, z_{k,n_k} \), for \( i = 0, \ldots, k \). When dealing with terms whose leftmost constant is \( S^{n_0, \ldots, n_k} \), we will often use this kind of vector notation.

Recall the convention \( (t) \equiv t \). Hence, if e.g. \( n_0 = 1 \), then \( \langle \vec{z}_0 \rangle \equiv \langle \vec{z}_0, 1 \rangle \equiv z_0, 1 \) in the above rule.

The condition \( \varphi \) ensures that the leftmost erasures of all \( z_{i,j} \) are convertible in CL-pc\(^1\), and that the leftmost erasures of all \( y_i \) are convertible in CL-pc\(^1\). Some examples of significant reduction rules for \( S^n \) (omitting the conditions) are:

\[
\begin{align*}
S^{1,1} x y_1(\langle z_{0,1}, z_{1,1} \rangle) & \to x z_{0,1}(y_1 z_{1,1}) \\
S^{1,2,1} x y_1(y_1, y_2)(\langle z_{0,1}, z_{1,1}, z_{1,2}, z_{2,1} \rangle) & \to x z_{0,1}(y_1 z_{1,1}, z_{1,2}, y_2 z_{2,1}) \\
S^{2,2} x y_1(\langle z_{0,1}, z_{0,2}, z_{1,1}, z_{1,2} \rangle) & \to x z_{0,1}(z_{0,2}, y_1 z_{1,1}, z_{1,2}) \\
\end{align*}
\]

For instance, the condition for the second of these rules states that \( |y_1| =_{\text{CL-pc}\^1} |y_2| \), \( |z_{0,1}| =_{\text{CL-pc}\^1} |z_{1,1}| \), \( |z_{0,1}| =_{\text{CL-pc}\^1} |z_{1,2}| \), \( |z_{0,1}| =_{\text{CL-pc}\^1} |z_{2,1}| \), \( |z_{1,1}| =_{\text{CL-pc}\^1} |z_{1,2}| \), etc.

Note that the equality \( =_{\text{CL-pc}\^1} \) in the conditions refers to the system CL-pc\(^1\), not CL-pc\(^s\). Note also that all rules of CL-pc\(^s\) are linear, disregarding the side-conditions.

Reduction by a rule in CL-pc\(^s\) is called significant reduction, or \( s \)-reduction. One-step \( s \)-reduction is denoted by \( \to_s \). Analogously, we use the terminology and notation of \( s \)-contraction, \( s \)-expansion, \( s \)-normal form (\( s \)-NF), etc. Note that every \( s \)-redex is an \( s \)-term. We write \( t \to_s t' \) if \( t \to_s t' \) and the \( s \)-contraction is not by the second rule for \( C_2 \) and it does not occur inside a tuple.

An \( i \)-redex is a CL-pc\(^s\)-redex which is also an \( i \)-term. An \( l \)-term \( t_1 \) is said to \( i \)-reduce to \( t_2 \), denoted \( t_1 \to_i t_2 \), if \( t_1 \to_{\text{CL-pc}\^1} t_2 \) and the redex contracted in \( t_1 \) is an \( i \)-term. An \( l \)-term \( t_1 \) is said to \( i \)-expand to \( t_2 \) if \( t_2 \to_i t_1 \). We write \( t_1 \to_{i,s} t_2 \) if \( t_1 \to_i t_2 \) or \( t_1 \to_s t_2 \).

Actually, we will consider mostly \( l \)-terms whose all erasures are identical. For such a term an \( s \)-contraction by a rule for \( S^n \) in CL-pc\(^s\) naturally corresponds to a CL-pc\(^l\)-contraction on its erasure. We could get rid of the side conditions in the rules for \( S^n \) and
consider exclusively terms whose all erasures are identical. But then we would need to require $i/s$-contractions/expansions to always occur “in the same way” (modulo labeling) in all components of a tuple. This would complicate the inductive proofs concerning the relations $\rightarrow_s$, $\rightarrow_i$, etc. Hence, the role of the conditions in the rules for $S^2$ is purely technical.

**Lemma 5.** The system $CL$ is terminating.

**Proof.** The number of labeled constants decreases with each $s$-contraction.

**Lemma 6.** If $t_1 \rightarrow_s t_2$ then $|t_1| =_{CL-pc^1} |t_2|$.

The above simple lemma implies that the conditions in significant reduction rules are stable under $s$-reduction and $s$-expansion. It is obvious that they are also stable under $i$-reduction and $i$-expansion.

**Lemma 7.** If $t \succ q$ and $t \rightarrow_s t'$ then there is $q'$ with $q \rightarrow_{CL-pc^1} q'$ and $t' \succ q'$.

**Proof.** Because all erasures of $t$ are identical and the second rule for $C_2$ is not used, the $s$--reduction may be simulated by a $CL$-pc$^1$-reduction in an obvious way. Because the $s$--contraction does not occur inside a tuple, all erasures of $t'$ are still identical.

In the next definition we introduce the predicate $\downarrow_{F_1}$ and the notion of standard $l$-terms. Intuitively, an $l$-term $t$ is standard if the labelings in $t$ have the meaning we intend to assign them, i.e. if $t$ is a term obtained by the process informally described in the previous section.

**Definition 8.** An $l$-term $t$ is standard if for every subterm $t'$ of $t$ the following hold:
1. $t'$ is either an $i$-term, an $s$-term or a tuple,
2. if $t' \equiv C_1 t_1 t_2$ and $t_0$ is in $s$-NF, then $t_0 \equiv T_1$ or $t_0 \equiv F_1$,
3. if $t' \equiv C_2 t_1 t_2$ then $|t_1| =_{CL-pc^1} |t_2|$,
4. if $t' \equiv S^n t_0 t_1 t_2$ then $t_2$ is a tuple of length $\sum_{i=1}^k n_k$ and if $k > 1$ then $t_1$ is a tuple of length $k$,
5. if $t'$ is an $s$-term and $t' \rightarrow_s t''$, then $t''$ is also an $s$-term,
6. if $t' \equiv \langle t_1, \ldots, t_n \rangle$ with $n > 1$, then none of $t_1, \ldots, t_n$ is a tuple.

An $l$-term $t$ is strongly standard if $t \rightarrow_s t'$ implies that $t'$ is standard. We write $t \downarrow_{F_1}$ if $t$ is strongly standard and has no $s$-NFs other than $F_1$, i.e. if $t \rightarrow_s t'$ then $t' \equiv F_1$.

Point 1 in Definition 8 essentially ensures that a standard term may be decomposed into a “significant” prefix and an “insignificant” suffix. A labeled term which is not standard is e.g. $CT_1$, because it is neither an $s$-term, nor an $i$-term, nor a tuple. Other examples of non-standard terms are: $C_1 TFF$, $C_2 CTF$, $S^{1,1} TTT$, $K_1 FF$, $C_3 CT_1 T$, $S^{1,1} CC(T_1, T_1)$, $K_1(T_1, T_1) T_1$, $\langle \langle C, C, C \rangle, C \rangle$. Examples of standard terms which are not strongly standard are: $S^{1,1}C_1 C(T_1, T_1)$, $S^{1,1}C_1 C(T, T) T$, $\langle K_1, C_1 T \rangle TTT$.

**Lemma 9.**
1. Any $i$-term is standard.
2. Any labeled constant is standard.
3. Every subterm of a standard term is also standard.
4. Every subterm of a term to which some strongly standard term s-reduces, is strongly standard.
5. If $t_1 t_2$ is standard then $t_1$ is not a tuple.
6. If $C_1 t_1 t_2$ is a subterm of a strongly standard term, then $t_0 \rightarrow_s T_1$ or $t_0 \rightarrow^*_s F_1$.

**Proof.** Follows from definitions. For the last point one also needs Lemma 5.
4 Confluence proof

We now give technical details of our confluence proof. As outlined in Section 2, we show:
1. if \( t \succ q \) and \( t \downarrow_{F_1} \), and \( q \rightarrow_{\text{CL-pc}} t' \), then there is \( t' \succ q' \) and \( t' \downarrow_{F_1} \) (Corollary 15),
2. if \( t \succ q \) and \( t \downarrow_{F_1} \), and \( q \rightarrow_{\text{CL-pc}} t' \), then there is \( t' \succ q' \) and \( t' \downarrow_{F_1} \) (Corollary 25).

The first part is proven by showing that CL-pc-reductions in \( q \) may be simulated by \( i \)-reductions and \( s \)-reductions in \( t \), and that \( i/s \)-reductions preserve \( \downarrow_{F_1} \) (Lemma 13 and Lemma 14). For the second part, we show that CL-pc-expansions in \( q \) may be simulated by \( i \)-expansions and \( \alpha \)-expansions (Definition 16) in \( t \). The technical notion of \( \alpha \)-expansion is needed to ensure that the new subterms of \( t' \) are labeled appropriately, in the way outlined in Section 2 (\( s \)-contraction by itself does not put any labeling restrictions on the terms erased in the contraction). Moreover, \( \alpha \)-expansion is also needed to facilitate the proof that \( t' \downarrow_{F_1} \) (see the discussion before Definition 16). Plain \( s \)-expansion does not necessarily preserve \( \downarrow_{F_1} \), while \( \alpha \)-expansion does (Lemma 24).

In other words, we show that CL-pc-reductions (expansions) in unlabeled terms may be simulated by \( i/s \)-reductions (\( i/\alpha \)-reductions) in their labeled variants, and that \( i/s \)-reductions (\( i/\alpha \)-reductions) preserve \( \downarrow_{F_1} \). A conversion \( q =_{\text{CL-pc}} F \) can then be translated into a conversion \( t =_{i,s,a} F_1 \) with no \( s \)-expansions or \( a \)-reductions, and with \( t \succ q \). For instance, a conversion in CL-pc

\[
F \leftarrow C(KF\Omega)FF \leftarrow C(KF\Omega)(CTF(KF\Omega)) \rightarrow \text{CTF}(KF\Omega)) \leftarrow C(KTF)(KF\Omega) \rightarrow C(KTF)FF \rightarrow F \leftarrow KFCF \leftarrow SKCF \rightarrow \text{SKC}(KF\Omega) \rightarrow SKCF
\]

will be translated to

\[
F_1 \leftarrow C_2(KF\Omega)FF_1 \leftarrow C_2(KF\Omega)F_1(C_1T_1F_1(KF\Omega)) \rightarrow, \ C_2FF_1(C_1T_1F_1(KF\Omega)) \leftarrow, \ C_2FF_1(C_1T_1F_1(KF\Omega)) \rightarrow, \ C_1(K_1T_1F)F_1(KF\Omega) \rightarrow, \ C_1(K_1T_1F)F_1F \rightarrow F_1 \leftarrow, \ K_1F_1(F) \leftarrow, \ S^{1,1}K_1C(F_1,F) \leftarrow, \ S^{1,1}K_1C(K_1F_1\Omega,F) \rightarrow, \ K_1F_1(F) \leftarrow, \ S^{1,1}K_1C(F_1,F)
\]

Since \( F_1 \downarrow_{F_1} \) and we prove that \( i/s \)-reductions and \( i/\alpha \)-reductions preserve \( \downarrow_{F_1} \), we may conclude that \( t \downarrow_{F_1} \). Then by the definition of \( \downarrow_{F_1} \) we obtain a significant reduction \( t \rightarrow_{*} F_1 \). In fact, the reduction may be assumed to be a \( s \)-reduction (Lemma 26). By Lemma 7 this reduction \( t \rightarrow_{*} F_1 \) may be translated into a CL-pc\(^{\downarrow} \) -reduction by erasing the labelings. Hence finally \( q =_{\text{CL-pc}} F \) (Lemma 27).

We first show that a CL-pc-contraction may be simulated by \( i \)-reductions and \( s \)-reductions.

\begin{itemize}
  \item \textbf{Lemma 10.} If \( t \) is strongly standard, \( t \succ q \) and \( q \rightarrow_{\text{CL-pc}} t' \), then there exists a term \( t' \) such that \( t \rightarrow_{s,a} t' \) and \( t' \succ q' \).
\end{itemize}

\textbf{Proof.} Induction on the size of \( t \). First assume \( t \) is not a tuple and \( q \) is the CL-pc-redex contracted in \( q \rightarrow_{\text{CL-pc}} t' \). If \( t \equiv q \) then \( t \equiv t' \equiv q' \) and we may take \( t' \equiv q' \). If \( t \neq q \) then \( t \) is not an \( i \)-term because \( t \succ q \). Hence by 1 in Definition 8 we conclude that \( t \) is an \( s \)-term. We have the following possibilities:

\begin{itemize}
  \item If \( q = C_1q_1q_2 \rightarrow_{\text{CL-pc}} q_1 \equiv q' \) then the leftmost constant in \( t \) is either \( C_1 \) or \( C_2 \).
    \begin{itemize}
      \item If \( t \equiv C_1T_1T_2 \) then \( t \rightarrow t_1 \) and \( t_1 \succ q_1 \), so we may take \( t' \equiv t_1 \).
      \item The case \( t \equiv C_1T_1T_2 \) is impossible by 2 in Definition 8.
      \item If \( t \equiv C_2T_1T_2 \) then \( t_1 \succ q_1 \) and \( |t_1| =_{\text{CL-pc}} |t_2| \) by 3 in Definition 8. Thus \( t \rightarrow t_1 \) and we may take \( t' \equiv t_1 \).
    \end{itemize}
  \item If \( q = C_2q_1q_2 \rightarrow_{\text{CL-pc}} q_2 \) then the argument is analogous. Note that the presence of the second rule for \( C_2 \) is necessary here.
\end{itemize}
Confluence of an Extension of Combinatory Logic by Boolean Constants

- If \( q \equiv C[q_0]q_1 \rightarrow_{CL-\text{pc}} q_1 \) then \( t \equiv C[t_0]t_1t_2 \) with \( C' \in \{C_1, C_2\} \), \( t_0 \triangleright q_0 \), \( t_1 \triangleright q_1 \) and \( t_2 \triangleright q_1 \).
- If \( C \equiv C_1 \) then \( t_0 \rightarrow^* T_1 \) or \( t_0 \rightarrow^* F_1 \) by Lemma 9. Hence \( t \rightarrow^*_1 t_1 \) or \( t \rightarrow^*_2 t_2 \). In the first case we may take \( t' \equiv t_1 \), and in the second we take \( t' \equiv t_2 \).
- If \( C' \equiv C_2 \) then \( t \rightarrow s \) \( t_1 \) because \( |t_1| \equiv |t_2| \equiv q_1 \). Thus we take \( t' \equiv t_1 \).
- If \( q \equiv K[q_2] \rightarrow_{CL-\text{pc}} q_1 \) then \( t \equiv K[t_1t_2] \rightarrow s \) \( t_1 \) with \( t_1 \triangleright q_1 \). We take \( t' \equiv t_1 \).
- If \( q \equiv S[q_0]q_1q_2 \rightarrow_{CL-\text{pc}} q_0q_2(q_1q_2) \) then \( t \equiv S^q(s(t_1, \ldots, t_k), q_0, \ldots, q_k) \) where the conventions regarding the vector notation are as in Definition 4, and \( s \triangleright q_0 \), and \( t_1 \triangleright q_1 \) for \( i = 1, \ldots, k \), and \( r_{1,j} \triangleright q_2 \) for \( i = 0, \ldots, k, j = 1, \ldots, i \). Thus

\[
t \rightarrow s r_0(t_1(r_1), \ldots, t_k(r_k)) \triangleright q_0q_2(q_1q_2)
\]

and we may take \( t' \equiv s(r_0(t_1(r_1), \ldots, t_k(r_k))) \).

If \( t \) is a tuple or \( q \) is not the contracted CL-pc-redex, then the claim follows from the inductive hypothesis. \( \blacktriangleleft \)

The following technical lemma shows that \( \leftrightarrow_i \) may be postponed after \( \rightarrow_s \).

**Lemma 11.** If \( t \leftrightarrow_i \rightarrow^* t' \) then \( \rightarrow^*_i \leftrightarrow_i^\equiv t' \).

**Proof.** Suppose \( t \leftrightarrow_i t_1 \rightarrow t_2 \rightarrow t_3 \). We proceed by induction on the definition of \( t_2 \rightarrow t_3 \).

If \( t_3 \) is the contracted s-redex then, because an \( i \)-redex (\( i \)-contractum) is an \( i \)-term, it is easy to see by inspecting Definition 4 that the \( i \)-redex (\( i \)-contractum) in \( t_2 \) must occur below a variable position of the \( s \)-redex. Since significant reduction rules are linear and their conditions are stable under \( i \)-reductions (\( i \)-expansions), the claim holds. Note that we need \( \leftrightarrow_i^\equiv \) instead of \( \leftrightarrow_i \), in the conclusion, because the \( i \)-redex (\( i \)-contractum) may be erased by the \( s \)-contraction.

If \( t_3 \) is not the \( s \)-redex, then \( t_2 \equiv s_1s_2 \) or \( t_2 \equiv (s_1, \ldots, s_n) \) with \( n > 1 \), and the claim is easily established possibly appealing to the inductive hypothesis. \( \blacktriangleleft \)

The next lemmas show that \( i \)-reductions/expansions and \( s \)-reductions preserve \( \| \).

**Lemma 12.** If \( t \) is standard and \( t \leftrightarrow_i t' \) then \( t' \) is standard.

**Proof.** We check that the conditions in Definition 8 hold for every subterm \( s' \) of \( t' \). Note that because \( i \)-reduces and \( i \)-contracta are \( i \)-terms, \( s' \) is an \( i \)-term or there is a subterm \( s \) of \( t \) such that \( s \leftrightarrow_i^\equiv s' \).

1. If \( s' \) is not an \( i \)-term, then there is a subterm \( s \) of \( t \) such that \( s \leftrightarrow_i^\equiv s' \). If \( s \) is an \( i \)-term or a tuple then so is \( s' \). Otherwise, \( s \) is an \( s \)-term by 1 in Definition 8. Then \( s' \) is also an \( s \)-term.

2. Suppose \( s' \equiv C_{i_1}t_1i_1t_2' \) with \( t_0' \) in \( s \)-NF. Since \( s' \) is not an \( i \)-term, there is a subterm \( s \) of \( t \) such that \( s \equiv C_{i_0}t_1t_2 \) and \( t_i \leftrightarrow_i^\equiv t_i' \) for \( i = 0, 1, 2 \). Since \( t_0' \) is in \( s \)-NF and \( t_0 \leftrightarrow_i^\equiv t_0' \), the term \( t_0 \) is also in \( s \)-NF. Thus \( t_0 \equiv T_1 \) or \( t_0 \equiv F_1 \) by 2 in Definition 8. Hence \( t_0' \equiv T_1 \) or \( t_0' \equiv F_1 \).

3. Suppose \( s' \equiv C_{2i_1}t_1i_1t_2' \). Since \( s' \) is not an \( i \)-term, there is a subterm \( s \) of \( t \) such that \( s \equiv C_{i_1}t_1t_2 \) and \( t_i \leftrightarrow_i^\equiv t_i' \) for \( i = 0, 1, 2 \). By 3 in Definition 8 we have \( |t_1| =_{CL-\text{pc}} |t_2| \). Hence also \( |t_0'| =_{CL-\text{pc}} |t_1'| \), because \( t_i \leftrightarrow_i^\equiv t_i' \) implies \( |t_i| =_{CL-\text{pc}} |t_i'| \).

4. Suppose \( s' \equiv S^{n_0, \ldots, n_k}t_1i_1t_2' \). Since \( s' \) is not an \( i \)-term, there is a subterm \( s \) of \( t \) such that \( s \equiv S^{n_0, \ldots, n_k}t_1t_2 \) and \( t_i \leftrightarrow_i^\equiv t_i' \) for \( i = 0, 1, 2 \). By 4 in Definition 8 we conclude that \( t_2 \) is a tuple of length \( n = \sum_{i=0}^k n_i \), and if \( k > 1 \) then \( t_1 \) is a tuple of length \( k \). The same holds for \( t_2' \) and \( t_1' \), because a tuple cannot be an \( i \)-redex or an \( i \)-contractum.
5. Suppose \( s' \) is an s-term. There is a subterm \( s \) of \( t \) such that \( s \leftrightarrow^m_1 s' \). Since \( s' \) is an s-term, so is \( s \). Suppose \( s' \rightarrow^*_s r' \). By Lemma 11 there is \( r \) such that \( s \rightarrow^*_s r \leftrightarrow^m_1 r' \). By 5 in Definition 8, the term \( r \) is an s-term. Hence, \( r' \) is also an s-term.

6. Suppose \( s' \equiv \langle t'_1, \ldots, t'_n \rangle \) with \( n > 1 \). Since \( s' \) is not an \( i \)-term, there is a subterm \( s \) of \( t \) such that \( s \equiv \langle t_1, \ldots, t_n \rangle \) and \( t_i \leftrightarrow^m_1 t'_i \) for \( i = 1, \ldots, n \). By 6 in Definition 8 none of \( t_1, \ldots, t_n \) is a tuple. Hence, none of \( t'_1, \ldots, t'_n \) is a tuple either.

**Lemma 13.** If \( t \Downarrow F_1 \) and \( t \leftrightarrow^*_s t \) then \( t' \Downarrow F_1 \).

**Proof.** Suppose \( t' \rightarrow^*_s t'_0 \). By Lemma 11 there is \( t_0 \) with \( t \rightarrow^*_s t_0 \) and \( t_0 \leftrightarrow^m_1 t'_0 \). Because \( t \) is strongly standard, \( t_0 \) is standard. Hence \( t'_0 \) is standard by Lemma 12. Therefore \( t' \) is strongly standard.

Suppose \( t' \rightarrow^*_s t'_0 \) with \( t'_0 \) in s-NF. By Lemma 11 there is \( t_0 \) with \( t \rightarrow^*_s t_0 \leftrightarrow^m_1 t'_0 \). Since \( t'_0 \) is in s-NF, so is \( t_0 \), because an \( i \)-contraction or an \( i \)-expansion cannot create an s-redex. Since \( t \Downarrow F_1 \), we obtain \( t_0 \equiv F_1 \). Thus \( t'_0 \equiv t_0 \equiv F_1 \).

**Lemma 14.** If \( t \Downarrow F_1 \) and \( t \rightarrow^*_s t' \) then \( t' \Downarrow F_1 \).

**Corollary 15.** If \( t \Downarrow F_1 \), \( t \rightarrow q \) and \( q \rightarrow_{\text{CL-pc}} q' \) then there is \( t' \) with \( t' \rightarrow q' \) and \( t' \Downarrow F_1 \).

**Proof.** Follows from Lemma 10, Lemma 13 and Lemma 14.

With the above corollary we have finished the first half of the proof. Now we need to show an analogous corollary for CL-pc-expansions. First, we want to prove that CL-pc-expansions in unlabeled terms may be simulated by \( i \)-expansions and \( a \)-expansions in their strongly standard labeled variants. We have already shown in Lemma 13 that \( i \)-expansions preserve \( \Downarrow F_1 \). We need to show that \( a \)-expansions also preserve \( \Downarrow F_1 \).

One trivial reason why \( s \)-expansions do not necessarily preserve \( \Downarrow F_1 \), is that if \( t \rightarrow a \) then \( t' \) may be not standard even if \( t \) is, e.g., consider \( F_1 \leftarrow K_1 F_1 (CT_1) \). A more profound reason is that with \( s \)-expansion we do not sufficiently “control” the expansion by a rule for \( C_2 \). E.g. \( F_1 \leftarrow C_2 \Omega F_1 (KF \Omega) \). Then \( C_2 \Omega F_1 (KF \Omega) \rightarrow s, KF \Omega \) but \( KF \Omega \) does not \( s \)-reduce to \( F_1 \).

Hence, we use \( a \)-expansions which put additional restrictions on the \( s \)-redexes, essentially implementing the labeling of expansions described in Section 2. They also allow to “delay” the reductions in a contractum of \( C_2 t_0 t_1 \) to facilitate the proof of an analogon of Lemma 11.

Like in the proof of Lemma 13 we show that if \( t' \rightarrow a \) then any reduction \( t' \rightarrow^* s' \) may be simulated by a reduction \( t \rightarrow^* s \) with \( s' \rightarrow^m_1 s \). The most interesting case is when \( t' \equiv E[C_2 t_0 t_1] \rightarrow a E[t_1] \equiv t \) (where \( E \) is a context), which is obtained from a CL-pc-expansion by the rule \( C_{2yy} \rightarrow y \). We now informally describe the idea for the proof in this case. Thus suppose \( t' \rightarrow^* s' \). If a contracted \( s \)-redex does not overlap with a descendant\(^5\) of \( C_2 t_0 t_1 \), then the \( s \)-reduction is simulated by the same \( s \)-reduction. If a descendant of \( C_2 t_0 t_1 \) occurs inside a contracted \( s \)-redex, but it is different from this redex, then the descendant must occur below a variable position of the \( s \)-redex, because there are no non-root overlaps between the rules of significant reduction. Thus we may simulate this \( s \)-reduction by the same \( s \)-reduction. If a contracted \( s \)-redex occurs inside a descendant \( C_2 t'_0 t'_1 t'_2 \) of \( C_2 t_0 t_1 \), but it is different from this descendant, then it must occur in \( t'_0, t'_1 \) or \( t'_2 \).

---

\(^5\) Note that because the rules of significant reduction are linear there may be at most one descendant.
the simulated s-reduction. Finally, if a descendant $C_2t_0t_1' t_2'$ of $C_2t_0t_1t_1$ is s-contracted, then either $C_2t_0t_1' t_2' \rightarrow_s t_1'$ or $C_2t_0t_1' t_2' \rightarrow_s t_2'$. In any case we can s-reduce $t_1$ to $t_1'$ or $t_2'$. In other words, we defer the choice of the simulated reduction path till the descendant of the a-redex is actually contracted.

**Definition 16.** An l-term $t'$ is an a-redex and t its a-contraction, if t is an s-term and one of the following holds:

- $t' \equiv C_1 T_1 t q$ and $q$ is an i-term, 
- $t' \equiv C_1 F_1 t q$ and $q$ is an i-term, 
- $t' \equiv C_2 q t_1 t_2$, $t \rightarrow_a t_1$, $t \rightarrow_a t_2$ and $q$ is an i-term, 
- $t' \equiv K_1 t q$ and $q$ is an i-term, 
- $t' \equiv S^2 t_0(s_1, \ldots, s_k)\langle\overline{r}_0, \ldots, \overline{r}_k\rangle$ where the conventions regarding vector notation are as in Definition 4, $|s_i|_{\text{CL-pc}} = |s_j|_{\text{CL-pc}}$ for $i, j = 1, \ldots, k$, $|r_{i,j}|_{\text{CL-pc}} = |r_{i',j'}|_{\text{CL-pc}}$ for $i, i' = 0, \ldots, k$, $j = 1, \ldots, n$, $j' = 1, \ldots, n'$, none of the $s_i$ or $r_{i,j}$ is a tuple, and $t \equiv t_0(\overline{r}_0)\langle s_1, \ldots, s_k, \overline{r}_k\rangle$.

Because of the third point, an a-contraction of an a-redex is not unique. The notations $\rightarrow_a$, $\rightarrow_{a^*}$, $\rightarrow_{i,a}$, etc. are used accordingly. Note that any a-redex is an s-redex.

**Lemma 17.** If $t' \rightarrow_a t$ then $t' \rightarrow_s \cdot \cdot \cdot \rightarrow_s t$, and hence $|t'| =_{\text{CL-pc}} |t|$.

The above simple lemma implies that the conditions in significant reduction rules are stable under a-reduction and a-expansion. Note that if $t' \rightarrow_a t$ then not necessarily $t' \rightarrow_s t$ because of the third point in Definition 16.

**Lemma 18.** If t is standard, $t \succ q$ and $q \equiv_{\text{CL-pc}} q'$ then there is $t'$ with $t' \rightarrow_{t,a} t$ and $t' \succ q'$.

**Proof.** Induction on the size of t. First assume t is not a tuple and q is the CL-pc-contraction expanded in $q \equiv_{\text{CL-pc}} q'$. If t is an i-term, then $t \equiv q \leftrightarrow q'$ and we may take $t' \equiv q'$. If t is not an i-term, then it is an s-term by 1 in Definition 8. We have the following possibilities, depending on the rule of CL-pc used in the expansion.

- If $q' \equiv C T_1 q_1 q_2 \rightarrow_{\text{CL-pc}} q_1 \equiv q$ then we take $t' \equiv C_1 T_1 t q_2$ and we have $t' \rightarrow_a t$ and $t' \succ q'$.
- If $q' \equiv C F_1 q_1 q_2 \rightarrow_{\text{CL-pc}} q_2 \equiv q$ then we may take $t' \equiv C_1 F_1 t q_1$. 
- If $q' \equiv C_0 q_1 q_1' \rightarrow_{\text{CL-pc}} q_1$ then we may take $t' \equiv C_0 q_0 t_1$.
- If $q' \equiv K_0 q_1 t \rightarrow_{\text{CL-pc}} q_0 t$ then we may take $t' \equiv K_0 q_1$.
- If $q' \equiv S q_0 q_1 q_2 \rightarrow_{\text{CL-pc}} q_0 q_1 q_2$ then $t' \equiv q_0 q_2(q_1 q_2)$ and t is an s-term. Hence $t \equiv t_0 t_1 t_2 t_3$ with $t_a \succ q_0$, $t_b \succ q_2$ and $t_c \succ q_1 q_2$. Recalling the convention $\langle s \rangle \equiv s$ for any term $s$, we may assume

$$t_b \equiv \langle s_1, \ldots, s_m \rangle, t_c \equiv \langle t_1, \ldots, t_k \rangle, \text{ for } k, m \in \mathbb{N}_+,$$

$$\text{if } k = 1 \text{ then } t_1 \text{ is not a tuple, and if } m = 1 \text{ then } s_1 \text{ is not a tuple.}$$

(\text{(*)})

In other words, if e.g. $t_b$ is a tuple, then $t_b \equiv \langle s_1, \ldots, s_m \rangle$ for some $s_1, \ldots, s_m$. If $t_b$ is not a tuple then we take $s_1 \equiv t_b$ and consider $t_b \equiv (t_b) \equiv \langle s_1 \rangle$. This is chiefly to reduce the number of cases to consider. Let $1 \leq i \leq k$. Because $t_b \succ q_2$, we have $s_i \succ q_2$ for $i = 1, \ldots, m$. Also none of $s_1, \ldots, s_m$ is a tuple, by condition 6 in Definition 8, or by (\text{(*)}) if $m = 1$. Since $t_c \succ q_1 q_2$, we have $t_i \succ q_1 q_2$. Also $t_i$ cannot be a tuple, by condition 6 in Definition 8, or by (\text{(*)}) if $k = 1$. Thus $t_1 \equiv u_i(\overline{r}_i)$ where $\overline{r}_i$ stands for $r_{i,1}, \ldots, r_{i,1,n_i}$, and $u_i \succ q_1$ and $r_{i,j} \succ q_2$ for $j = 1, \ldots, n_i$, where none of the $r_{i,j}$ is a tuple, by definition (if $n_i = 1$) or by condition 6 in Definition 8. By Lemma 9 also none of $u_1, \ldots, u_k$ is a tuple. We may thus take $t' \equiv S^{m,n_1,\ldots,n_k} t_0(\langle u_1, \ldots, u_k \rangle)\langle \overline{r}_0, \overline{r}_1, \ldots, \overline{r}_k \rangle$ where $\overline{r}_0$ stands for $s_1, \ldots, s_m$. We have $t' \rightarrow_a t$ and $t' \succ q'$. 


If $t$ is a tuple or $q$ is not the CL-pc-contractum, then the claim follows from the inductive hypothesis.

\[\blacktriangleright\textbf{Lemma 19.} \text{If } t \xrightarrow{\cdot} \rightarrow_s t' \text{ and } t \text{ is standard then } t \xrightarrow{\cdot} \rightarrow_s t'. \\]

\textbf{Proof.} Suppose $t' \rightarrow_s t'_1$, $t' \rightarrow_a t$ and $t$ is standard. By induction on the definition of $t' \rightarrow_s t'_1$ we show that there is $t_1$ with $t \rightarrow^* t_1$ and $t'_1 \rightarrow^* t_1$. The base case is when the $s$-contraction in $t' \rightarrow_s t'_1$ occurs at the root.

If the $s$-contraction occurs at the root, but the $a$-contraction in $t' \rightarrow_a t$ does not occur at the root, then it is easy to see by inspecting the definitions that the $a$-redex in $t'_1$ must occur below a variable position of the $s$-redex. Since significant reduction rules are linear and their conditions are stable under $a$-reduction, the claim holds in this case.

Assume that both the $s$-contraction and the $a$-contraction occur at the root. If $t' \equiv C_2 q s_1 s_2 \rightarrow_a t$ then $t \rightarrow^* s_1$, $t \rightarrow^* s_2$ and the $s$-contraction of $t'$ yields either $s_1$ or $s_2$. We may thus take either $t_1 \equiv s_1$ or $t_1 \equiv s_2$, and we have $t \rightarrow^* t_1 \equiv t'_1$. If $t' \equiv C_1 t_1 t q \rightarrow_a t$ then the $s$-contraction must be by the first rule of CL-pc, so $t'_1 \equiv t$ and we may take $t_1 \equiv t'_1 \equiv t$.

All other cases are analogous.

If neither the $s$-contraction nor the $a$-contraction occurs at the root, then the claim is easily established, possibly appealing to the inductive hypothesis.

Finally, assume that the $a$-contraction occurs at the root, but the $s$-contraction does not occur at the root. We have the following possibilities.

- If $t' \equiv C_1 t_1 t q \rightarrow_a t$ then the $s$-contraction must occur inside $t$. So $t \rightarrow t_1$ for some term $t_1$. Note that $t$ is an $s$-term by definition of $a$-contraction. Therefore $t_1$ is also an $s$-term, by 5 in Definition 8. Thus $t_1$ satisfies the required conditions.

- If $t' \equiv C_2 q s_1 s_2 \rightarrow_a t$ then $t \rightarrow^* s_1$, $t \rightarrow^* s_2$ and the $s$-contraction must occur inside $s_1$ or $s_2$. We may take $t_1 \equiv t$ and we still have $t'_1 \equiv t_1$.

- The cases $t' \equiv C_1 F_{t q} \rightarrow_a t$ and $t' \equiv K_1 t q \rightarrow_a t$ are analogous to the first case.

- If $t' \equiv S^2 t_0 (s_1, \ldots, s_k) \langle \vec{r}_0, \ldots, \vec{r}_k \rangle$ then $|s_i| =_{CL-pc^1} |s_j|$, $|r_{i,j}| =_{CL-pc^1} |r_{i',j'}|$ for $i, j, i', j'$ as in Definition 16, none of the $s_i$ or $r_{i,j}$ is a tuple, and $t \equiv t_0 (\vec{r}_0) (s_1 (\vec{r}_1), \ldots, s_k (\vec{r}_k))$. The $s$-contraction must occur inside one of the $s_i$ or the $r_{i,j}$, or in $t_0$. For instance, assume $s_1 \rightarrow s_1'$. Since $s_1$ is a subterm of $t$ and it is not a tuple, it cannot $s$-reduce to a tuple by Definition 8. Hence $s_1'$ is not a tuple. Take $t_1 \equiv t_0 (\vec{r}_0) (s_1' (\vec{r}_1), s_2 (\vec{r}_2), \ldots, s_k (\vec{r}_k))$. Note that $t \rightarrow s_1$. Thus $t_1$ is an $s$-term, because $t$ is an $s$-term and it $s$-reduces only to $s$-terms, by 5 in Definition 8.

\[\blacktriangleright\textbf{Corollary 20.} \text{If } t \xrightarrow{\cdot} \rightarrow^*_s t' \text{ and } t \text{ is strongly standard then } t \xrightarrow{\cdot} \rightarrow^*_s t'. \\]

\[\blacktriangleright\textbf{Lemma 21.} \text{If } r \text{ is a strongly standard } a \text{-contractum of an } a \text{-redex } r', \text{ and } s' \text{ is a proper subterm of } r', \text{ then } s' \text{ is standard.} \\]

\textbf{Proof.} It suffices to show that $s'$ is a subterm of some standard term.

- Suppose $r' \equiv C_1 T_1 r_q \rightarrow_a r$ with $q$ an $i$-term. Both $C_1 T_1 r$ and $q$ are standard and $s'$ is a subterm of one of them. The cases $r' \equiv C_1 T_1 r_q$ and $r' \equiv K_1 r_q$ are analogous.

- Suppose $r' \equiv C_2 q r_1 r_2 \rightarrow_a r$ with $q$ an $i$-term. Because $r \rightarrow^*_s r_i$ and $r$ is strongly standard, $r_1, r_2$ are standard. Also $q$ is an $i$-term. This implies that $C_2 q r_1$ is also standard. Since $s'$ occurs in $C_2 q r_1$ or $r_2$, it is standard.

- Suppose $r' \equiv S^2 t_0 (s_1, \ldots, s_k) \langle \vec{r}_0, \ldots, \vec{r}_k \rangle \rightarrow_a t_0 (\vec{r}_0) (s_1 (\vec{r}_1), \ldots, s_k (\vec{r}_k))$. The term $t_0$ and each of $s_i$ and $r_{i,j}$ (with $i, j$ as in Definition 16) is standard. Note that none of $s_i$ or $r_{i,j}$ is a subterm by Definition 16. Since each $s_i$ is also standard, by inspecting Definition 8 we may conclude that $\langle s_1, \ldots, s_k \rangle$ is standard. Similarly $\langle \vec{r}_0, \ldots, \vec{r}_k \rangle$ is standard. Also
Confluence of an Extension of Combinatory Logic by Boolean Constants

\[ S^a t_0(s_1, \ldots, s_k) \text{ is standard. This implies that } s' \text{ is standard, because it occurs in } S^a t_0(s_1, \ldots, s_k) \text{ or } \langle r_0, \ldots, r_k \rangle. \]

**Lemma 22.** If \( s \) is an \( s \)-term and \( s' \rightarrow_a s \) then \( s' \) is also an \( s \)-term.

**Proof.** Induction on the structure of \( s \).

**Lemma 23.** If \( t \) is strongly standard and \( t' \rightarrow_a t \) then \( t' \) is standard.

**Proof.** We check that the conditions in Definition 8 hold for every subterm \( s' \) of \( t' \). We may assume that \( s' \) does not occur in \( t \), as otherwise the claim follows from the fact that \( t \) is standard. Therefore, \( s' \) occurs in the \( a \)-redex contracted in \( t' \rightarrow_a t \), or the \( a \)-redex occurs inside \( s' \). If \( s' \) is a proper subterm of the \( a \)-redex, then our claim holds by Lemma 21. Hence, we may assume that the \( a \)-redex \( r \) is a subterm of \( s' \). Then \( s' \rightarrow_a s \) with \( s \) a subterm of \( t \) (so \( s \) is strongly standard).

1. Suppose \( r \) is the \( a \)-contractum of \( r' \) and \( s' \rightarrow_a s \). By Definition 16, the term \( r \) is an \( s \)-term. Thus \( s \) cannot be an \( i \)-term. If \( s \) is a tuple, then so is \( s' \). Otherwise, \( s \) is an \( s \)-term, by 1 in Definition 8. Hence \( s' \) is also an \( s \)-term by Lemma 22.

2. Suppose \( s' \equiv C_1 t_1 t_2' t_2 \) and \( t_0' \) is in \( s \)-NF. If \( s' \equiv r' \) then \( s' \equiv C_1 t_1 t_2 \) or \( s' \equiv C_1 F_1 t_2' \), hence \( t_0' \equiv T_1 \) or \( t_0' \equiv F_1 \). If \( r' \) is a proper subterm of \( s' \), then \( r' \) must be a subterm of \( t_1' \) or \( t_2' \), because \( a \)-redexes are not in \( s \)-NF. Thus, \( s' \rightarrow_a s \equiv C_1 t_0' t_1 t_2 \) for some terms \( t_1, t_2 \), where \( s \) is a subterm of \( t \). Hence, \( t_0' \equiv T_1 \) or \( t_0' \equiv F_1 \) by 2 in Definition 8.

3. Suppose \( s' \equiv C_2 t_0 t_2' \). If \( s' \equiv r' \) then \( s' \rightarrow_a t_1' \) and \( s' \rightarrow_a t_2' \). Hence \( |t_1'| =_{\text{CL-pc}} |s| =_{\text{CL-pc}} |t_2'| \) by 3 in Definition 8. Thus also \( |t_1'| =_{\text{CL-pc}} |t_2'| \) by Lemma 17.

4. Suppose \( s' \equiv S^a_{s_1, \ldots, s_k} t_1 t_2' \). If \( s' \equiv r' \), then \( s' \equiv S^a_{s_1, \ldots, s_k} t_1 \equiv_{s_1, \ldots, s_k} \langle \rangle \), as in Definition 16, so the claim holds. If \( r' \) is a proper subterm of \( s' \), then \( s' \equiv S^a_{s_1, \ldots, s_k} t_1' t_2' \rightarrow_a s \equiv S^a_{s_1, \ldots, s_k} t_1 t_2 \), where \( t_i' \rightarrow_a t_i \) for \( i = 0, 1, 2 \), and \( s \) is a subterm of \( t \). By 4 in Definition 8, the term \( t_2 \) is a tuple of length \( \sum_{i=0}^k n_i \), and if \( k > 1 \) then \( t_1 \) is a tuple of length \( k \). Since an \( a \)-contractum is an \( s \)-term, and hence not a tuple, \( t_2 \) is not an \( a \)-contractum, and if \( k > 1 \) then \( t_1 \) is not an \( a \)-contractum. Thus we may conclude that \( t_2 \) is a tuple of length \( \sum_{i=0}^k n_i \), and if \( k > 1 \) then \( t_1' \) is a tuple of length \( k \).

5. Suppose \( s' \) is an \( s \)-term and \( s' \rightarrow_a s' \). Because also \( s' \rightarrow_a s \) and \( s \) is strongly standard, by Corollary 20 there is \( s_1 \) with \( s_1' \rightarrow_a \equiv \ s_1 \) and \( s' \rightarrow_a s_1 \). By Definition 16, the term \( s \) is an \( s \)-term, so \( s_1 \) is also an \( s \)-term by 5 in Definition 8. So \( s_1' \) is an \( s \)-term by Lemma 22.

6. Suppose \( s' \equiv (t_1', \ldots, t_n') \) with \( n > 1 \). We have \( s' \rightarrow_a s \equiv (t_1, \ldots, t_n) \) where \( t_i' \rightarrow_a t_i \) for \( i = 1, \ldots, n \). By 6 in Definition 8, none of \( t_1, \ldots, t_n \) is a tuple. Thus it is easy to see by inspecting Definition 16 that none of \( t_1', \ldots, t_n' \) can be a tuple.

**Lemma 24.** If \( t \downarrow_{F_1} \) and \( t' \rightarrow_a t \) then \( t' \downarrow_{F_1} \).

**Proof.** Suppose \( t' \rightarrow_a t_0' \). By Corollary 20 there is \( t_0 \) with \( t \rightarrow_a t_0 \) and \( t_0' \rightarrow_a t_0 \). Since \( t \) is strongly standard, so is \( t_0 \). Therefore, \( t_0' \) is standard by Lemma 23.

Suppose \( t' \rightarrow_a t_0' \) with \( t_0' \) in \( s \)-NF. By Corollary 20 there is \( t_0 \) with \( t \rightarrow_a t_0 \) and \( t_0' \rightarrow_a t_0 \). Since an \( a \)-redex is an \( s \)-redex and \( t_0' \) is in \( s \)-NF, we conclude that \( t_0 \equiv t_0' \). But then \( t_0' \equiv t_0 \equiv F_1 \), because \( t \downarrow_{F_1} \).

**Corollary 25.** If \( t \downarrow_{F_1} \) and \( t \rightarrow_a q \) then \( q \rightarrow_{\text{CL-pc}} q' \) and there is \( t' \) with \( t' \downarrow_{F_1} \) and \( t' \rightarrow q' \).

**Proof.** Follows from Lemma 18, Lemma 13 and Lemma 24.

**Lemma 26.** If \( t \) has no \( s \)-NFs other than \( F_1 \) then \( t \rightarrow_a F_1 \).
Thus, using Corollary 15 and Corollary 25 it is easy to show by induction on the length of $q$ with Lemma 29. Lemma 30. Lemma 2 we have then there is $\text{Theorem 32.}$ 

When reduction are confluent [11, Chapter 4].

reducts inside a tuple cannot erase this tuple or create an $s$-redex outside of it. But since any $s$-NF of $t'$ is an $s$-NF of $t$, this contradicts the fact that $t$ has no $s$-NFs other than $F_1$. ▶

We now have everything we need to show the central lemma of the confluence proof.

Lemma 27. The system $\text{CL-pc}^1$ is $F$-normal, i.e., if $q \equiv_{\text{CL-pc}^1} F$ then $q \rightarrow^*_{\text{CL-pc}^1} F$.

Proof. If $q \equiv_{\text{CL-pc}^1} F$ then by Lemma 2 we have $q \equiv_{\text{CL-pc}} F$. Note that $F_1 \not\equiv_{F_1}$ and $|F_1| \equiv F$. Thus, using Corollary 15 and Corollary 25 it is easy to show by induction on the length of $q \equiv_{\text{CL-pc}} F$ that there is $t$ with $t \rightarrow q$ and $t \not\equiv_{F_1}$. By Lemma 26 we have $t \rightarrow^*_{\text{CL-pc}^1} F_1$. But then, because $t \rightarrow q$, using Lemma 7 it is easy to show by induction on the length of $t \rightarrow^*_{\text{CL-pc}^1} F_1$ that $q \equiv |t| \rightarrow^*_{\text{CL-pc}^1} |F_1| \equiv F$. ▶

It remains to derive the confluence of $\text{CL-pc}^1$ and $\text{CL-pc}^\perp$ from Lemma 27. We use a trick with an auxiliary term rewriting system $R$, in a way similar to how the confluence of $\text{CL-pc}^\perp$ is derived from the condition $T \not\equiv_{\text{CL-pc}^\perp} F$ in [4]. The idea is to eliminate the non-trivial overlap between the rules of $\text{CL-pc}^1$ by imposing additional side conditions.

Definition 28. The term rewriting system $R$ is defined by the following rules:

$K_{xyz} \rightarrow x \\
S_{xyz} \rightarrow xz(yz) \\
C_{xy} \rightarrow x \\
C_{xyz} \rightarrow y \\
C_{zxy} \rightarrow x \iff z \equiv_{\text{CL-pc}^1} F \\
C_{xyz} \rightarrow y \iff z \not\equiv_{\text{CL-pc}^1} F \land x \equiv_{\text{CL-pc}^1} y$

Lemma 29. If $q \rightarrow_{\text{CL-pc}} q' \implies q \rightarrow_R q'$.

Lemma 30. If $q \rightarrow_R q' \implies q \rightarrow_{\text{CL-pc}^1} q'$.

Proof. Follows from definitions and Lemma 27. ▶

Lemma 31. The system $R$ is confluent.

Proof. Because $T \not\equiv_{\text{CL-pc}^1} F$ by Lemma 27, the system $R$ is weakly orthogonal (i.e. it is left-linear and all its critical pairs are trivial). By Lemma 30 the conditions are stable under reduction. Weakly orthogonal conditional term rewriting systems whose conditions are stable under reduction are confluent [11, Chapter 4]. ▶

Theorem 32. The systems $\text{CL-pc}^1$ and $\text{CL-pc}^\perp$ are confluent.

Proof. Since $q_1 \rightarrow_{\text{CL-pc}^1} q_2$ implies $q_1 \rightarrow_{\text{CL-pc}^\perp} q_2$, it suffices to show that if $q_1 \equiv_{\text{CL-pc}^1} q_2$ then there is $q$ with $q_1 \rightarrow^*_{\text{CL-pc}^1} q$ and $q_2 \rightarrow^*_{\text{CL-pc}^\perp} q$. So suppose $q_1 \equiv_{\text{CL-pc}^1} q_2$. Then by Lemma 2 we have $q_1 \equiv_{\text{CL-pc}} q_2$. By Lemma 29 we obtain $q_1 \equiv_R q_2$. By Lemma 31 there is $q$ with $q_1 \rightarrow_R^* q$ and $q_2 \rightarrow_R^* q$. By Lemma 30 we have $q_1 \rightarrow_{\text{CL-pc}^1} q$ and $q_2 \rightarrow_{\text{CL-pc}^1} q$. ▶
Confluence of an Extension of Combinatory Logic by Boolean Constants

References