

Proving Failure in Functional Logic Programs (Extended Version)*

F. J. López-Fraguas and J. Sánchez-Hernández

Dep. Sistemas Informáticos y Programación, Univ. Complutense de Madrid
{fraguas,jaime}@sip.ucm.es

Abstract. How to extract negative information from programs is an important issue in logic programming. Here we address the problem for functional logic programs, from a proof-theoretic perspective. The starting point of our work is *CRWL* (Constructor based ReWriting Logic), a well established theoretical framework for functional logic programming, whose fundamental notion is that of non-strict non-deterministic function. We present a proof calculus, *CRWLF*, which is able to deduce negative information from *CRWL*-programs. In particular, *CRWLF* is able to prove ‘finite’ failure of reduction within *CRWL*.

1 Introduction

We address in this paper the problem of extracting negative information from functional logic programs. The question of negation is a main topic of research in the logic programming field, and the most common approach is *negation as failure*, as an easy effective approximation to the *CWA* (*closed world assumption*), which is a simple, but uncomputable, way of deducing negative information from positive programs (see e.g. [1] for a survey on negation in logic programming).

On the other hand, functional logic programming (*FLP* for short) is a powerful programming paradigm trying to combine the nicest properties of functional and logic programming (see [4] for a now ‘classical’ survey on *FLP*). *FLP* subsumes *pure* logic programming: predicates can be defined as functions returning the value ‘true’, for which definite clauses can be written as conditional rewrite rules. In some simple cases it is enough, to handle negation, just to define predicates as two-valued boolean functions returning the values ‘true’ or ‘false’. But negation as failure is far more expressive, and it is then of clear interest to investigate a similar notion for the case of *FLP*. Failure in logic programs, when seen as functional logic programs, corresponds to failure of reduction to ‘true’. This generalizes to a natural notion of failure in *FLP*, which is ‘failure of reduction’.

As technical setting for our work we have chosen *CRWL* [3], a well established theoretical framework for *FLP*. The fundamental notion in *CRWL* is that of non-strict non-deterministic function, for which *CRWL* provides a firm logical basis, as mentioned for instance in [5]. Instead of equational logic, *CRWL* considers

* The authors have been partially supported by the Spanish CICYT (project TIC 98-0445-C03-02 ‘TREND’)

a Constructor based ReWriting Logic, presented by means of a proof calculus, which determines what statements can be deduced from a given program. In addition to the proof-theoretic semantics, [3] develop a model theoretic semantics for *CRWL*, with existence of distinguished free term models for programs, and a sound and complete lazy narrowing calculus as operational semantics. The *CRWL* framework (with many extensions related to types, HO and constraints) has been implemented in the system *TCOY* [8].

Here we are interested in extending the proof-theoretic side of *CRWL* to cope with failure. More concretely, we look for a proof calculus, which will be called *CRWLF* (*CRWL* with failure), which is able to prove failure of reduction in *CRWL*. Since reduction in *CRWL* is expressed by proving certain statements, our calculus will provide proofs of unprovability within *CRWL*. As for the case of *CWA*, unprovability is not computable, which means that our calculus can only give an approximation, corresponding to cases which can be intuitively described as ‘finite failures’.

There are very few works about negation in *FLP*. In [10] the work of Stuckey about *constructive negation* [12] is adapted to the case of *FLP* with strict functions and innermost narrowing as operational mechanism. In [11] a similar work is done for the case of non-strict functions and lazy narrowing. The approach is very different of the proof-theoretic view of our work. The fact that we also consider non-deterministic functions makes a significant difference.

The proof-theoretic approach, although not very common, has been followed sometimes in the logic programming field, as in [6], which develops for logic programs (with negation) a framework which resembles, in a very general sense, *CRWL*: a program determine a deductive system for which deducibility, validity in a class of models, validity in a distinguished model and derivability by an operational calculus are all equivalent. Our work attempts to be the first step of what could be a similar programme for *FLP* extended with the use of failure as a programming construct.

The rest of the paper is organized as follows. In Section 2 we give the essentials of *CRWL* which are needed for our work. Section 3 presents the *CRWLF*-calculus, preceded by some illustrative examples. Section 4 contains the results about *CRWLF*. Most of the results are technically involved, and their proofs have been skipped because of the lack of space (full details can be found in [9]). Section 5 contains some conclusions.

2 The *CRWL* Framework

We give here a short summary of *CRWL*, in its proof-theoretic face. Model theoretic semantics and lazy narrowing operational semantics are not considered here. Full details can be found in [3].

2.1 Technical Preliminaries

We assume a signature $\Sigma = DC_\Sigma \cup FS_\Sigma$ where $DC_\Sigma = \bigcup_{n \in \mathbb{N}} DC_\Sigma^n$ is a set of *constructor* symbols and $FS_\Sigma = \bigcup_{n \in \mathbb{N}} FS_\Sigma^n$ is a set of *function* symbols, all

of them with associated arity and such that $DC_\Sigma \cap FS_\Sigma = \emptyset$. We also assume a countable set \mathcal{V} of *variable symbols*. We write $Term_\Sigma$ for the set of (total) *terms* (we say also *expressions*) built up with Σ and \mathcal{V} in the usual way, and we distinguish the subset $CTerm_\Sigma$ of (total) constructor terms or (total) *c-terms*, which only make use of DC_Σ and \mathcal{V} . The subindex Σ will usually be omitted. Terms intend to represent possibly reducible expressions, while c-terms represent data values, not further reducible.

We will need sometimes to use the signature Σ_\perp which is the result of extending Σ with the new constant (0-arity constructor) \perp , that plays the role of the undefined value. Over Σ_\perp , we can build up the sets $Term_\perp$ and $CTerm_\perp$ of (partial) terms and (partial) c-terms respectively. Partial c-terms represent the result of partially evaluated expressions; thus, they can be seen as approximations to the value of expressions.

As usual notations we will write X, Y, Z, \dots for variables, c, d, \dots for constructor symbols, f, g, \dots for functions, e, e', \dots for terms and s, t, \dots for c-terms.

We will use the sets of substitutions $CSubst = \{\theta : \mathcal{V} \rightarrow CTerm\}$ and $CSubst_\perp = \{\theta : \mathcal{V} \rightarrow CTerm_\perp\}$. We write $e\theta$ for the result of applying θ to e .

Given a set of constructor symbols S we say that the terms t and t' have an S -clash if they have different constructor symbols of S at the same position.

2.2 The Proof Calculus for CRWL

A *CRWL*-program \mathcal{P} is a set of conditional rewrite rules of the form:

$$\underbrace{f(t_1, \dots, t_n)}_{\text{head}} \rightarrow \underbrace{e}_{\text{body}} \Leftarrow \underbrace{C_1, \dots, C_n}_{\text{condition}}$$

where $f \in FS^n$; (t_1, \dots, t_n) is a linear tuple (each variable in it occurs only once) with $t_1, \dots, t_n \in CTerm$; $e \in Term$ and each C_i is a constraint of the form $e' \bowtie e''$ (*joinability*) or $e' \diamond e''$ (*divergence*) where $e', e'' \in Term$. The reading of the rule is: $f(t_1, \dots, t_n)$ reduces to e if the conditions C_1, \dots, C_n are satisfied. We write \mathcal{P}_f for the set of defining rules of f in \mathcal{P} .

From a given program \mathcal{P} , the proof calculus for *CRWLF* can derive three kinds of statements:

- *Reduction or approximation statements:* $e \rightarrow t$, with $e \in Term_\perp$ and $t \in CTerm_\perp$. The intended meaning of such statement is that e can be reduced to t , where reduction may be done by applying rewriting rules of \mathcal{P} or by replacing subterms of e by \perp . If $e \rightarrow t$ can be derived, t represents one of the possible values of the denotation of e .
- *Joinability statements:* $e \bowtie e'$, with $e, e' \in Term_\perp$. The intended meaning in this case is that e and e' can be both reduced to some common totally defined value, that is, we can prove $e \rightarrow t$ and $e' \rightarrow t$ for some $t \in CTerm$.
- *Divergence statements:* $e \diamond e'$, with $e, e' \in Term_\perp$. The intended meaning now is that e and e' can be reduced to some (possibly partial) c-terms t and t' such that they have a *DC*-clash.

Table 1. Rules for *CRWL*-provability

(1)	$\frac{}{e \rightarrow \perp}$	
(2)	$\frac{}{X \rightarrow X}$	$X \in \mathcal{V}$
(3)	$\frac{e_1 \rightarrow t_1, \dots, e_n \rightarrow t_n}{c(e_1, \dots, e_n) \rightarrow c(t_1, \dots, t_n)}$	$c \in DC^n, \quad t_i \in CTerm_{\perp}$
(4)	$\frac{e_1 \rightarrow s_1, \dots, e_n \rightarrow s_n \quad C \quad e \rightarrow t}{f(e_1, \dots, e_n) \rightarrow t}$	if $t \neq \perp, R \in \mathcal{P}_f$ ($f(s_1, \dots, s_n) \rightarrow e \Leftarrow C$) $\in [R]_{\perp}$
(5)	$\frac{e \rightarrow t \quad e' \rightarrow t}{e \bowtie e'}$	if $t \in CTerm$
(6)	$\frac{e \rightarrow t \quad e' \rightarrow t'}{e \diamond e'}$	if $t, t' \in CTerm_{\perp}$ and have a <i>DC</i> -clash

It must be mentioned that the *CRWL* framework as presented in [3] does not consider divergence conditions. They have been incorporated to *CRWL* in [7] as a useful and expressive resource for programming.

When using function rules to derive statements, we will need to use what are called *c*-instances of such rules: the set of *c*-instances of a program rule R is defined as $[R]_{\perp} = \{R\theta \mid \theta \in CSubst_{\perp}\}$. This allows, in particular, to express parameter passing.

Table 1 shows the proof calculus for *CRWL*. We write $\mathcal{P} \vdash_{CRWL} \varphi$ for expressing that the statement φ is provable from the program \mathcal{P} .

The rule 4 allows to use *c*-instances of program rules to prove approximations. These *c*-instances may contain \perp and rule (1) allows to reduce any expression to \perp . This reflects a non-strict semantics.

A distinguished feature of *CRWL* is that functions can be *non-deterministic*. For example, assuming the constructors z (zero) and s (successor) for natural numbers, a non-deterministic function *coin* can be defined by the rules $coin \rightarrow z$ and $coin \rightarrow s(z)$. The use of *c*-instances in rule (4) instead of general instances corresponds to *call time choice* semantics for non-determinism (see [3]). As an example, if in addition to *coin* we consider the function definition $mkpair(X) \rightarrow pair(X, X)$ (*pair* is a constructor), it is possible to build a *CRWL*-proof for $mkpair(coin) \rightarrow pair(z, z)$ and also for $mkpair(coin) \rightarrow pair(s(z), s(z))$, but not for $mkpair(coin) \rightarrow pair(z, s(z))$.

Observe that \diamond is not the logical negation of \bowtie . They are not even incompatible: due to non-determinism, two expressions e, e' can satisfy both $e \bowtie e'$ and $e \diamond e'$ (although this cannot happen if e, e' are *c*-terms). In the ‘coin’ example, we can derive both $coin \bowtie z$ and $coin \diamond z$.

We can define the *denotation* of an expression e as the set of *c*-terms to which e can be reduced according to this calculus: $\llbracket e \rrbracket = \{t \in CTerm_{\perp} \mid \mathcal{P} \vdash_{CRWL} e \rightarrow t\}$

3 The CRWLF Framework

We now address the problem of failure in *CRWL*. Our primary interest is to obtain a calculus able to prove that a given expression fails to be reduced. Since reduction corresponds in *CRWL* to approximation statements $e \rightarrow t$, we can reformulate our aim more precisely: we look for a calculus able to prove that a given expression e has no possible reduction (other than the trivial $e \rightarrow \perp$) in *CRWL*, i.e., $\llbracket e \rrbracket = \{\perp\}$.

Of course, we cannot expect to achieve that with full generality since, in particular, the reason for having $\llbracket e \rrbracket = \{\perp\}$ can be non-termination of the program as rewrite system, which is uncomputable. Instead, we look for a suitable computable approximation to the property $\llbracket e \rrbracket = \{\perp\}$, corresponding to cases where failure of reduction is due to ‘finite’ reasons, which can be constructively detected and managed.

Previous to the formal presentation of the calculus, which will be called *CRWLF* (for ‘*CRWL* with failure’) we give several simple examples for a preliminary understanding of some key aspects of it, and the reasons underlying some of its technicalities.

3.1 Some Illustrative Examples

Consider the following functions, in addition to *coin*, defined in Sect. 2:

$$f(z) \rightarrow f(z) \quad g(s(s(X))) \rightarrow z \quad \begin{array}{l} h \rightarrow s(z) \\ h \rightarrow s(h) \end{array} \quad k(X) \rightarrow z \Leftarrow X \bowtie s(z)$$

The expressions $f(z)$ and $f(s(z))$ fail to be reduced, but for quite different reasons. In the first case $f(z)$ does not terminate. The only possible proof accordingly to *CRWL* is $f(z) \rightarrow \perp$ (by rule 1); any attempt to prove $f(z) \rightarrow t$ with $t \neq \perp$ would produce an ‘infinite derivation’. In the second case, the only possible proof is again $f(s(z)) \rightarrow \perp$, but if we try to prove $f(s(z)) \rightarrow t$ with $t \neq \perp$ we have a kind of ‘finite failure’: rule 4 needs to solve the parameter passing $s(z) \rightarrow z$, that could be finitely checked as failed, since no rule of the *CRWL*-calculus is applicable. The *CRWLF*-calculus does not prove non-termination of $f(z)$, but will be able to detect and manage the failure for $f(s(z))$. In fact it will be able to perform a *constructive proof* of this failure.

Consider now the expression $g(\textit{coin})$. Again, the only possible reduction is $g(\textit{coin}) \rightarrow \perp$ and it is intuitively clear that this is another case of finite failure. But this failure is not as simple as in the previous example for $f(s(z))$: in this case the two possible reductions for *coin* to defined values are $\textit{coin} \rightarrow z$ and $\textit{coin} \rightarrow s(z)$. Both of z and $s(z)$ fail to match the pattern $s(s(X))$ in the rule for g , but none of them can be used separately to detect the failure of $g(\textit{coin})$. A suitable idea is to collect the set of defined values to which a given expression can be reduced. In the case of *coin* that set is $\{z, s(z)\}$. The fact that \mathcal{C} is the collected set of values of e is expressed in *CRWLF* by means of the statement $e \triangleleft \mathcal{C}$. In our example, *CRWLF* will prove $\textit{coin} \triangleleft \{z, s(z)\}$. Statements $e \triangleleft \mathcal{C}$

generalize the approximation statements $e \rightarrow t$ of *CRWL*, and in fact can replace them. Thus, *CRWLF* will not need to use explicit $e \rightarrow t$ statements.

How far should we go when collecting values? The idea of collecting all values (and to have them completely evaluated) works fine in the previous example, but there are problems when the collection is infinite. For example, according to its definition above, the expression h can be reduced to any positive natural number, so the corresponding set would be $\{s(z), s(s(z)), s(s(s(z))), \dots\}$. Then, what if we try to reduce the expression $f(h)$? From an intuitive point of view it is clear that the value z will not appear in this set, because all the values in it have the form $s(\dots)$. We can represent all this values by the set $\{s(\perp)\}$. Here we can understand \perp as an *incomplete information*: we know that all the possible values for h are successor of ‘something’; we do not know what is this ‘something’, but in fact, we do not need to know it. Anyway the set does not contain the value z , so $f(h)$ fails. Notice that all the possible values for h are represented (not present) in the set $\{s(\perp)\}$, and this information is sufficient to prove the failure of $f(h)$. The *CRWLF*-calculus will be able to prove the statement $h \triangleleft \{s(\perp)\}$, and we say that $\{s(\perp)\}$ is a *Sufficient Approximation Set (SAS)* for h .

In general, an expression will have multiple *SAS*'s. Any expression has $\{\perp\}$ as its simplest *SAS*. And, for example, the expression h has an infinite number of *SAS*'s: $\{\perp\}$, $\{s(\perp)\}$, $\{s(z), s(s(\perp))\}$, ... The *SAS*'s obtained by the calculus for *coin* are $\{\perp\}$, $\{\perp, s(\perp)\}$, $\{\perp, s(z)\}$, $\{z, \perp\}$, $\{z, s(\perp)\}$ and $\{z, s(z)\}$. The *CRWLF*-calculus provides appropriate rules for working with *SAS*'s. The derivation steps will be guided by these *SAS*'s in the same sense that *CRWL* is guided by approximation statements.

Failure of reduction is due in many cases to failure in proving the conditions in the program rules. The calculus must be able to prove those failures. Consider for instance the expression $k(z)$. In this case we would try to use the c-instance $k(z) \rightarrow z \Leftarrow z \bowtie s(z)$ that allows to perform parameter passing. But the condition $z \bowtie s(z)$ is clearly not provable, so $k(z)$ must fail. For achieving it we must be able to give a proof for ‘ $z \bowtie s(z)$ cannot be proved with respect to *CRWL*’. For this purpose we introduce a new constraint $e \not\bowtie e'$ that will be true if we can build a *proof of non-provability* for $e \bowtie e'$. In our case, $z \not\bowtie s(z)$ is clear simply because of the clash of constructors. In general the proof for a constraint $e \not\bowtie e'$ will be guided by the corresponding *SAS*'s for e and e' as we will see in the next section. As our initial *CRWL* framework also allows constraints of the form $e \triangleright e'$, we need still another constraint $\not\triangleright$ for expressing ‘failure of \triangleright ’.

There is another important question to justify: we use an explicit representation for failure by means of the new constant symbol \mathbb{F} . Let us examine some examples involving failures. First, consider the expression $g(s(f(s(z))))$; for reducing it we would need to do parameter passing, i.e., matching $s(f(s(z)))$ with some c-instance of the pattern $s(s(X))$ of the definition of g . As $f(s(z))$ fails to be reduced the parameter passing must also fail. If we take $\{\perp\}$ as an *SAS* for $f(s(z))$ we have not enough information for detecting the failure (nothing can be said about the matching of $s(s(X))$ and $s(\perp)$). But if we take $\{\mathbb{F}\}$ as an *SAS* for $f(s(z))$, this provides enough information to ensure that $s(\mathbb{F})$ cannot match

any c-instance of the pattern $s(s(X))$. Notice that we allow the value \perp to appear inside the term $s(\perp)$. It could appear that the information $s(\perp)$ is essentially the same of \perp (for instance, \perp also fails to match any c-instance of $s(s(X))$), but this is not true in general. For instance, the expression $g(s(s(f(s(z)))))$ is reducible to z . But if we take the *SAS* $\{\perp\}$ for $f(s(z))$ and we identify the expression $s(s(f(s(z))))$ with \perp , matching with the rule for g would not succeed, and the reduction of $g(s(s(f(s(z)))))$ would fail.

We can now proceed with the formal presentation of the *CRWLF*-calculus.

3.2 Technical Preliminaries

We introduce the new constant symbol \perp into the signature Σ to obtain $\Sigma_{\perp, \perp} = \Sigma \cup \{\perp, \perp\}$. The sets $Term_{\perp, \perp}$, $CTerm_{\perp, \perp}$ are defined in the natural way and we will use the set $CSubst_{\perp, \perp} = \{\theta : \mathcal{V} \rightarrow CTerm_{\perp, \perp}\}$.

A natural *approximation ordering* \sqsubseteq over $Term_{\perp, \perp}$ can be defined as the least partial ordering over $Term_{\perp, \perp}$ satisfying the following properties:

- $\perp \sqsubseteq e$ for all $e \in Term_{\perp, \perp}$,
- $h(e_1, \dots, e_n) \sqsubseteq h(e'_1, \dots, e'_n)$, if $e_i \sqsubseteq e'_i$ for all $i \in \{1, \dots, n\}$, $h \in DC \cup FS$

The intended meaning of $e \sqsubseteq e'$ is that e is less defined or has less information than e' . Two expressions $e, e' \in Term_{\perp, \perp}$ are *consistent* if they can be refined to obtain the same information, i.e., if there exists $e'' \in Term_{\perp, \perp}$ such that $e \sqsubseteq e''$ and $e' \sqsubseteq e''$.

Notice that the only relations satisfied by \perp are $\perp \sqsubseteq \perp$ and $\perp \sqsubseteq \perp$. In particular, \perp is maximal. This is reasonable, since \perp represents ‘failure of reduction’ and this gives a no further refinable information about the result of the evaluation of an expression. This contrasts with the status given to failure in [11], where \perp is chosen to verify $\perp \sqsubseteq t$ for any t different from \perp .

The class of programs that we consider in the following is less general than in the *CRWL* framework. Rules of functions have the same form, but they must not contain *extra variables*, i.e., for any rule $(f(\vec{t}) \rightarrow e \Leftarrow \vec{C}) \in \mathcal{P}$ all the variables appearing in e and \vec{C} must also appear in the head $f(\vec{t})$, i.e., $var(e) \cup \mathcal{V}(\vec{C}) \subseteq var(\vec{t})$. In *FLP* with non-deterministic functions this is not *as* restrictive as it could appear: function nesting can replace the use (typical in logic programming) of variables as repositories of intermediate values, and in many other cases where extra variables represent unknown values to be computed by search, they can be successfully replaced by non-deterministic ‘lazy generating’ functions (see [3] for some examples).

We will frequently use the following notation: given $e \in Term_{\perp, \perp}$, \hat{e} stands for the result of replacing by \perp all the occurrences of \perp in e (notice that $\hat{e} \in Term_{\perp}$, and $e = \hat{e}$ iff $e \in Term_{\perp}$).

3.3 The Proof Calculus for *CRWLF*

In *CRWLF* five kinds of statements can be deduced:

- $e \triangleleft \mathcal{C}$, intended to mean ‘ \mathcal{C} is an SAS for e ’.
- $e \boxtimes e'$, $e \diamond e'$, with the same intended meaning as in *CRWL*.
- $e \not\boxtimes e'$, $e \not\diamond e'$, intended to mean failure of $e \boxtimes e'$ and $e \diamond e'$ respectively.

We will sometimes speak of \boxtimes , \diamond , $\not\boxtimes$, $\not\diamond$ as ‘constraints’, and use the symbol \diamond to refer to any of them. The constraints $\not\boxtimes$ and \boxtimes are called the *complementary* of each other; the same holds for $\not\diamond$ and \diamond , and we write $\tilde{\diamond}$ for the complementary of \diamond .

When proving a constraint $e \diamond e'$ the calculus *CRWLF* will evaluate an SAS for the expressions e and e' . These SAS’s will consist of c-terms from $CTerm_{\perp, \mathbb{F}}$, and provability of the constraint $e \diamond e'$ depends on certain syntactic (hence decidable) relations between those c-terms. Actually, the constraints \boxtimes , \diamond , $\not\boxtimes$ and $\not\diamond$ can be seen as the result of generalizing to expressions the relations \downarrow , \uparrow , $\not\downarrow$ and $\not\uparrow$ on c-terms, which we define now.

Definition 1 (Relations over $CTerm_{\perp, \mathbb{F}}$).

- $t \downarrow t' \Leftrightarrow_{def} t = t', t \in CTerm$
- $t \uparrow t' \Leftrightarrow_{def} t$ and t' have a DC-clash
- $t \not\downarrow t' \Leftrightarrow_{def} t$ or t' contain \mathbb{F} as subterm or they have a DC-clash
- $\not\uparrow$ is defined as the least symmetric relation over $CTerm_{\perp, \mathbb{F}}$ satisfying:
 - $X \not\uparrow X$, for all $X \in \mathcal{V}$
 - $\mathbb{F} \not\uparrow t$, for all $t \in CTerm_{\perp, \mathbb{F}}$
 - if $t_1 \not\uparrow t'_1, \dots, t_n \not\uparrow t'_n$ then $c(t_1, \dots, t_n) \not\uparrow c(t'_1, \dots, t'_n)$ for all $c \in DC^n$

The relations \downarrow and \uparrow do not take into account the presence of \mathbb{F} , which behaves in this case as \perp . The relation \downarrow is *strict* equality, i.e., equality restricted to total c-terms. It is the notion of equality used in lazy functional or functional-logic languages as the suitable approximation to ‘true’ equality ($=$) over $CTerm_{\perp}$. The relation \uparrow is a suitable approximation to ‘ $\neg =$ ’, and hence to ‘ $\neg \downarrow$ ’ (where \neg stands for logical negation). The relation $\not\downarrow$ is also an approximation to ‘ $\neg \downarrow$ ’, but in this case using failure information ($\not\downarrow$ can be read as ‘ \downarrow fails’). Notice that $\not\downarrow$ does not imply ‘ $\neg =$ ’ anymore (we have, for instance, $\mathbb{F} \not\downarrow \mathbb{F}$). Similarly, $\not\uparrow$ is also an approximation to ‘ $\neg \uparrow$ ’ which can be read as ‘ \uparrow fails’.

The following proposition reflects these and more good properties of \downarrow , \uparrow , $\not\downarrow$, $\not\uparrow$.

Proposition 1. *The relations \downarrow , \uparrow , $\not\downarrow$, $\not\uparrow$ verify*

- For all $t, t', s, s' \in CTerm_{\perp, \mathbb{F}}$
 - $t \downarrow t' \Leftrightarrow \hat{t} \downarrow \hat{t}'$ and $t \uparrow t' \Leftrightarrow \hat{t} \uparrow \hat{t}'$
 - $t \uparrow t' \Rightarrow t \not\downarrow t' \Rightarrow \neg(t \downarrow t')$
 - $t \downarrow t' \Rightarrow t \not\uparrow t' \Rightarrow \neg(t \uparrow t')$
- \downarrow , \uparrow , $\not\downarrow$, $\not\uparrow$ are monotonic, i.e., if $t \sqsubseteq s$ and $t' \sqsubseteq s'$ then: $t \mathfrak{R} t' \Rightarrow s \mathfrak{R} s'$, where $\mathfrak{R} \in \{\downarrow, \uparrow, \not\downarrow, \not\uparrow\}$. Furthermore $\not\downarrow_B$ and $\not\uparrow_B$ are the greatest monotonic approximations to $\neg \downarrow_B$ and $\neg \uparrow_B$ respectively, where \mathfrak{R}_B is the restriction of \mathfrak{R} to the set of basic (i.e., without variables) c-terms from $CTerm_{\perp, \mathbb{F}}$.
- \downarrow and $\not\uparrow$ are closed under substitutions from $CSubst$; $\not\downarrow$ and \uparrow are closed under substitutions from $CSubst_{\perp, \mathbb{F}}$

By (b), we can say that $\downarrow, \uparrow, \Downarrow, \Uparrow$ behave well with respect to the information ordering: if they are true for some terms, they remain true if we refine the information contained in the terms. Furthermore, (b) states that \Downarrow, \Uparrow are defined in the best way, at least for basic c-terms. For c-terms with variables, we must take care: for instance, given the constructor z , we have $\neg(X \downarrow z)$, but not $X \Downarrow z$. Actually, to have $X \Downarrow z$ would violate a basic intuition about free variables in logical statements: if the statement is true, it should be true for any value (taken from an appropriate range) substituted for its free variables. The part (c) shows that the definitions of $\downarrow, \uparrow, \Downarrow, \Uparrow$ respect such principle. Propositions 2 and 3 of the next section show that monotonicity and closedness by substitutions are preserved when generalizing $\downarrow, \uparrow, \Downarrow, \Uparrow$ to $\boxtimes, \diamond, \boxtimes, \diamond$.

Table 2 contains the *CRWLF*-calculus. Some of the rules use a generalized notion of c-instances of a rule R : $[R]_{\perp, \mathbb{F}} = \{R\theta \mid \theta \in CSubst_{\perp, \mathbb{F}}\}$. We will use the notation $\mathcal{P} \vdash_{CRWLF} \varphi$ ($\mathcal{P} \not\vdash_{CRWLF} \varphi$ resp.) for expressing that the statement φ is provable (is not provable resp.) with respect to the calculus *CRWLF* and the program \mathcal{P} .

The first three rules are analogous to those of the *CRWL*-calculus, now dealing with *SAS*'s instead of simple approximations (notice the cross product of *SAS*'s in rule 3). In rule 4, for evaluating an expression $f(\bar{e})$ we produce *SAS*'s for the arguments e_i and then, for each combination of values in these *SAS*'s and each program rule for f , a part of the whole *SAS* is produced; all of them are unioned to obtain the final *SAS* for $f(\bar{e})$. This is quite different from rule 4 in *CRWL*: there we could use any c-instance of any rule for f ; here we need to consider simultaneously the contribution of each rule to achieve 'complete' information about the values to which the expression can be evaluated. We use the notation $f(\bar{t}) \triangleleft_R \mathcal{C}$ to indicate that only the rule R is used to produce \mathcal{C} .

Rules 5 to 8 consider all the possible ways in which a concrete rule R can contribute to the *SAS* of a call $f(\bar{t})$, where the arguments \bar{t} are all in $CTerm_{\perp, \mathbb{F}}$ (come from the evaluation of the arguments of a previous call $f(\bar{e})$). Rules 5 and 6 can be viewed as *positive* contributions. The first one obtains the trivial *SAS* and 6 works if there is a c-instance of the rule R with a head identical to the head of the call (parameter passing); in this case, if the constraints of this c-instance are provable, then the resulting *SAS* is generated by the body of the c-instance. Rules 7 and 8 consider the *negative* or *failed* contributions. Rule 7 applies when parameter passing can be done, but it is possible to prove the complementary $e_i \Downarrow e'_i$ of one of the constraints $e_i \Downarrow e'_i$ in the condition of the used c-instance. In this case the constraint $e_i \Downarrow e'_i$ (hence the whole condition in the c-instance) fails. Finally, rule 8 considers the case in which parameter passing fails because of a $DC \cup \{\mathbb{F}\}$ -clash between one of the arguments in the call and the corresponding pattern in R .

We remark that for given $f(\bar{t})$ and R , the rule 5 and exactly one of rules 6 to 8 are applicable. This fact, although intuitive, is far from being trivial to prove and constitutes in fact an important technical detail in the proofs of the results in the next section. We also remark that, for the sake of a better reading of rule 4, we have written ordinary set union for collecting *SAS*'s. This could

Table 2. Rules for *CRWLF*-provability

(1)	$\frac{}{e \triangleleft \{\perp\}}$	
(2)	$\frac{}{X \triangleleft \{X\}}$	$X \in \mathcal{V}$
(3)	$\frac{e_1 \triangleleft \mathcal{C}_1 \quad \dots \quad e_n \triangleleft \mathcal{C}_n}{c(e_1, \dots, e_n) \triangleleft \{c(t_1, \dots, t_n) \mid \bar{t} \in \mathcal{C}_1 \times \dots \times \mathcal{C}_n\}}$	$c \in DC^n \cup \{\mathbb{F}\}$
(4)	$\frac{e_1 \triangleleft \mathcal{C}_1 \quad \dots \quad e_n \triangleleft \mathcal{C}_n \quad \dots \quad f(\bar{t}) \triangleleft_R \mathcal{C}_{R, \bar{t}} \quad \dots}{f(e_1, \dots, e_n) \triangleleft \bigcup_{R \in \mathcal{P}_f, \bar{t} \in \mathcal{C}_1 \times \dots \times \mathcal{C}_n} \mathcal{C}_{R, \bar{t}}}$	$f \in FS^n$
(5)	$\frac{}{f(\bar{t}) \triangleleft_R \{\perp\}}$	
(6)	$\frac{e \triangleleft \mathcal{C} \quad \bar{\mathcal{C}}}{f(\bar{t}) \triangleleft_R \mathcal{C}}$	$(f(\bar{t}) \rightarrow e \Leftarrow \bar{\mathcal{C}}) \in [R]_{\perp, \mathbb{F}}$
(7)	$\frac{e_i \tilde{\diamond} e'_i}{f(\bar{t}) \triangleleft_R \{\mathbb{F}\}}$	$(f(\bar{t}) \rightarrow e \Leftarrow \dots, e_i \tilde{\diamond} e'_i, \dots) \in [R]_{\perp, \mathbb{F}}$, where $i \in \{1, \dots, n\}$
(8)	$\frac{}{f(t_1, \dots, t_n) \triangleleft_R \{\mathbb{F}\}}$	$R \equiv (f(s_1, \dots, s_n) \rightarrow e \Leftarrow \bar{\mathcal{C}})$, t_i and s_i have a $DC \cup \{\mathbb{F}\}$ -clash for some $i \in \{1, \dots, n\}$
(9)	$\frac{e \triangleleft \mathcal{C} \quad e' \triangleleft \mathcal{C}'}{e \bowtie e'}$	$\exists t \in \mathcal{C}, t' \in \mathcal{C}' \quad t \downarrow t'$
(10)	$\frac{e \triangleleft \mathcal{C} \quad e' \triangleleft \mathcal{C}'}{e \diamond e'}$	$\exists t \in \mathcal{C}, t' \in \mathcal{C}' \quad t \uparrow t'$
(11)	$\frac{e \triangleleft \mathcal{C} \quad e' \triangleleft \mathcal{C}'}{e \not\bowtie e'}$	$\forall t \in \mathcal{C}, t' \in \mathcal{C}' \quad t \not\downarrow t'$
(12)	$\frac{e \triangleleft \mathcal{C} \quad e' \triangleleft \mathcal{C}'}{e \not\diamond e'}$	$\forall t \in \mathcal{C}, t' \in \mathcal{C}' \quad t \not\uparrow t'$

be modified in such a way that \mathbb{F} is excluded from the union if it contains some other c-term different from \mathbb{F} . For example, if we obtain the *SAS*'s $\{z\}$ and $\{\mathbb{F}\}$ from two function rules, we could take $\{z\}$ as the final *SAS* for the call instead of $\{\mathbb{F}, z\}$. All the results of the next section are valid with this modification.

Rules 9 to 12 deal with constraints. With the use of the relations $\downarrow, \uparrow, \not\downarrow, \not\uparrow$ introduced in Sect. 3.3 the rules are easy to formulate. For $e \bowtie e'$ it is sufficient to find two c-terms in the *SAS*'s verifying the relation \downarrow , what in fact is equivalent to find a common totally defined c-term such that both expressions e and e' can be reduced to it (observe the analogy with rule 5 of *CRWL*). For the complementary constraint $\not\bowtie$ we need to use all the information of *SAS*'s in order to check the relation $\not\downarrow$ over all the possible pairs. The explanation of rules 11 and 12 is quite similar.

The next example shows a derivation of failure using the *CRWLF*-calculus.

4 Properties of CRWLF

In this section we explore some properties of the *CRWLF*-calculus and its relation with *CRWL*. In the following we assume a fixed program \mathcal{P} .

The non-determinism of the *CRWLF*-calculus allows to obtain different *SAS*'s for the same expression. As the *SAS* for an expression is a finite approximation to the denotation of the expression it is expected some kind of consistency between *SAS*'s for the same expression. Given two of them, we cannot ensure that one *SAS* must be more defined than the other in the sense that all the elements of the first are more defined than all of the second. For instance, two *SAS*'s for *coin* are $\{\perp, s(z)\}$ and $\{z, \perp\}$. The kind of consistency for *SAS*'s that we can expect is the following:

Definition 2 (Consistent Sets of c-terms). *Two sets $\mathcal{C}, \mathcal{C}' \subset CTerm_{\perp, F}$ are consistent iff for all $t \in \mathcal{C}$ there exists $t' \in \mathcal{C}'$ (and vice versa, for all $t' \in \mathcal{C}'$ there exists $t \in \mathcal{C}$) such that t and t' are consistent.*

Our first result states that two different *SAS*'s for the same expression must be consistent.

Theorem 1 (Consistency of SAS). *Given $e \in Term_{\perp, F}$, if $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}$ and $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}'$, then \mathcal{C} and \mathcal{C}' are consistent.*

This result is a trivial corollary of part a) of the following lemma.

Lemma 1 (Consistency). *For any $e, e', e_1, e_2, e'_1, e'_2 \in Term_{\perp, F}$*

- a) *If e, e' are consistent, $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}$ and $\mathcal{P} \vdash_{CRWLF} e' \triangleleft \mathcal{C}'$, then \mathcal{C} and \mathcal{C}' are consistent.*
- b) *If e_1, e'_1 are consistent and e_2, e'_2 are also consistent, then: $\mathcal{P} \vdash_{CRWLF} e_1 \diamond e_2 \Rightarrow \mathcal{P} \not\vdash_{CRWLF} e'_1 \tilde{\diamond} e'_2$*

As a trivial consequence of part b) we have:

Corollary 1. *$\mathcal{P} \vdash_{CRWLF} e \diamond e' \Rightarrow \mathcal{P} \not\vdash_{CRWLF} e \tilde{\diamond} e'$, for all $e, e' \in Term_{\perp, F}$*

This supports our original idea about \bowtie and \blacktriangleright as computable approximations to the negations of \bowtie and \blacktriangleright .

Another desirable property of our calculus is *monotonicity*, that we can informally understand in this way: the information that can be extracted from an expression can not decrease when we add information to the expression itself. This also reflects in the fact that if we can prove a constraint and we consider more defined terms in both sides of it, the resulting constraint must be also provable. Formally:

Proposition 2 (Monotonicity of CRWLF). *For $e, e', e_1, e_2, e'_1, e'_2 \in Term_{\perp, F}$*

- a) *If $e \sqsubseteq e'$ and $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}$, then $\mathcal{P} \vdash_{CRWLF} e' \triangleleft \mathcal{C}$*
- b) *If $e_1 \sqsubseteq e'_1$, $e_2 \sqsubseteq e'_2$ and $\mathcal{P} \vdash_{CRWLF} e_1 \diamond e_2$ then $\mathcal{P} \vdash_{CRWLF} e'_1 \diamond e'_2$, where $\diamond \in \{\bowtie, \bowtie, \blacktriangleright, \blacktriangleright\}$*

Monotonicity, as stated here, refers to the degree of evaluation of expression, and does not contradict the well known fact that negation as failure is a non-monotonic reasoning rule. In our setting it is also clearly true that, if we ‘define more’ the functions (i.e, we refine the program, not the evaluation of a given expression), an expression can become reducible when it was previously failed.

The next property says that what is true for free variables is also true for any possible (totally defined) value, i.e., provability in *CRWLF* is closed under total substitutions.

Proposition 3. *For any $\theta \in CSubst$, $e, e' \in Term_{\perp, F}$*

- a) $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C} \Rightarrow \mathcal{P} \vdash_{CRWLF} e\theta \triangleleft \mathcal{C}\theta$
- b) $\mathcal{P} \vdash_{CRWLF} e \diamond e' \Rightarrow \mathcal{P} \vdash_{CRWLF} e\theta \diamond e'\theta$

4.1 *CRWLF* related to *CRWL*

The *CRWLF*-calculus has been built as an extension of *CRWL* for dealing with failure. Here we show that our aims have been achieved.

We recall that a *CRWLF*-program is a *CRWL*-program not containing extra variables in rules. The following results are all referred to *CRWLF*-programs.

The next result shows that the *CRWLF*-calculus indeed extends *CRWL*. Parts a) and b) show that statements $e \triangleleft \mathcal{C}$ generalize approximation statements $e \rightarrow t$ of *CRWL*. Parts c) and d) show that *CRWLF* and *CRWL* are able to prove exactly the same joinabilities and divergences (if \mathbb{F} is ignored for the comparison).

Proposition 4. *For any $e, e' \in Term_{\perp, F}$*

- a) $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C} \Rightarrow \forall t \in \mathcal{C}, \mathcal{P} \vdash_{CRWL} \hat{e} \rightarrow \hat{t}$
- b) $\mathcal{P} \vdash_{CRWL} \hat{e} \rightarrow t \Rightarrow \exists \mathcal{C}$ such that $t \in \mathcal{C}$ and $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}$
- c) $\mathcal{P} \vdash_{CRWLF} e \bowtie e' \Leftrightarrow \mathcal{P} \vdash_{CRWL} \hat{e} \bowtie \hat{e}'$
- d) $\mathcal{P} \vdash_{CRWLF} e \diamond e' \Leftrightarrow \mathcal{P} \vdash_{CRWL} \hat{e} \diamond \hat{e}'$

We can revise within *CRWLF* the notion of denotation of an expression, and define $\llbracket e \rrbracket^F = \{t \in CTerm_{\perp, F} \mid e \triangleleft \mathcal{C}, t \in \mathcal{C}\}$, for any $e \in Term_{\perp, F}$. As a consequence of the previous proposition we have $\llbracket e \rrbracket \subseteq \llbracket e \rrbracket^F$ for any $e \in Term_{\perp}$ and $\llbracket \widehat{e} \rrbracket^F = \llbracket \hat{e} \rrbracket$ for any $e \in Term_{\perp, F}$, where, given a set S , \hat{S} is defined in the natural way $\hat{S} = \{\hat{t} \mid t \in S\}$.

All the previous results make easy the task of proving that we have done things right with respect to failure. We will need a result stronger than Prop. 4, which does not provide enough information about the relation between the denotation of an expression and each of its calculable *SAS*'s.

Proposition 5. *Given $e \in Term_{\perp, F}$, if $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}$ and $t \in \llbracket \hat{e} \rrbracket$, then there exists $s \in \mathcal{C}$ such that s and t are consistent.*

Proof. Assume $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}$. If we take $t \in CTerm_{\perp}$ such that $\mathcal{P} \vdash_{CRWL} \hat{e} \rightarrow t$, then by part b) of Prop. 4 there exists \mathcal{C}' such that $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}'$ with $t \in \mathcal{C}'$.

By Theorem 1 it follows that \mathcal{C} and \mathcal{C}' are consistent. By definition of consistent *SAS*'s, as $t \in \mathcal{C}'$, then there exist $s \in \mathcal{C}$ such that t and s are consistent. \square

We easily arrive now at our final result.

Theorem 2. *Given $e \in Term_{\perp, \mathbb{F}}$, if $\mathcal{P} \vdash_{CRWLF} e \triangleleft \{\mathbb{F}\}$ then $[[\hat{e}]] = \{\perp\}$*

Proof. If $t \in [[\hat{e}]]$, we know from Prop. 5 that \mathbb{F} and t must be consistent. As \mathbb{F} is consistent only with \perp and itself, and $t \in CTerm_{\perp}$, we conclude that $t = \perp$. \square

5 Conclusions and Future Work

We have proposed the proof calculus *CRWLF* (Constructor based ReWriting Logic with Failure), which allows to deduce negative information from a wide class of functional logic programs. In particular, the calculus provides proofs of failure of reduction, a notion that can be seen as the natural *FLP* counterpart of negation as failure in logic programming.

The starting point for *CRWLF* has been the proof calculus of *CRWL* [3], a well established theoretical framework for *FLP*. The most remarkable insight has been to replace the statements $e \rightarrow t$ of *CRWL* (representing a single reduction of e to an approximated value t) by $e \triangleleft \mathcal{C}$ (representing a whole, somehow complete, set \mathcal{C} of approximations to e). With the aid of \triangleleft we have been able to cover all the derivations in *CRWL*, as well as to prove failure of reduction and, as auxiliary notions, failure of joinability and divergence, the two other kinds of statements that *CRWL* was able to prove.

It is interesting to remark that \triangleleft provide, at the level of logical descriptions, a finer control over reduction than \rightarrow . Two examples: $e \triangleleft \{t\}, t \in CTerm$ expresses the property that e is reducible to the unique totally defined value t ; $e \triangleleft \mathcal{C}, e' \triangleleft \mathcal{C}$, with \mathcal{C} consisting only of total c-terms, expresses that e and e' reduce to exactly the same (totally defined) values. The same properties, if expressed by means of \rightarrow , would require the use of universal quantification, which is out of the scope of *CRWL*. Observe that, although the side conditions ‘ $t \in CTerm$ ’ and ‘ \mathcal{C} consisting only of total c-terms’ of the examples are not statements of *CRWLF*, they are purely syntactical conditions.

The idea of collecting into an *SAS* values coming from different reductions for a given expression e presents some similarities with abstract interpretation which, within the *FLP* field, has been used in [2] for detecting unsatisfiability of equations $e = e'$ (something similar to failure of our $e \bowtie e'$). We can mention some differences between our work and [2]:

- Programs in [2] are much more restrictive: they must be confluent, terminating, satisfy a property of stratification on conditions, and define strict and total functions.
- In our setting, each *SAS* for an expression e consists of (down) approximations to the denotation of e , and the set of *SAS*'s for e determines in a precise sense (Props. 4 and 5) the *exact* denotation of e . In the abstract interpretation approach one typically obtains, for an expression e , an abstract term representing a *superset* of the denotation of all the instances of e . But some of the rules of the *CRWLF*-calculus (like (9) or (10)) are not valid if we replace *SAS*'s by such supersets. To be more concrete, if we adopt an abstract

interpretation view of our *SAS*'s, it would be natural to see \perp as standing for the set of all constructor terms (since \perp is refinable to any value), and therefore to identify an *SAS* like $\mathcal{C} = \{\perp, z\}$ with $\mathcal{C}' = \{\perp\}$. But from $e \triangleleft \mathcal{C}$ we can deduce $e \bowtie z$, while it is not correct to do the same from $e \triangleleft \mathcal{C}'$. Therefore, the good properties of *CRWLF* with respect to *CRWL* are lost.

We see our work as the first step in the research of a whole framework for dealing with failure in *FLP*. Some natural (but not small!) future steps are: to enlarge the class of considered programs by allowing extra variables; to consider 'general' programs which make use of failure information, and to develop model theoretic and operational semantics for them.

Acknowledgements: We thank the anonymous referees for their useful comments.

References

- [1] K.R. Apt and R. Bol. Logic programming and negation: A survey. *Journal of Logic Programming*, 19&20:9–71, 1994.
- [2] D. Bert and R. Echahed. Abstraction of conditional term rewriting systems. In *Proc. ILPS'95*, pages 162–176. MIT Press, 1995.
- [3] J.C. González-Moreno, T. Hortalá-González, F.J. López-Fraguas, and M. Rodríguez-Artalejo. An approach to declarative programming based on a rewriting logic. *Journal of Logic Programming*, 40(1):47–87, 1999.
- [4] M. Hanus. The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming*, 19&20:583–628, 1994.
- [5] M. Hanus (ed.). Curry: An integrated functional logic language. Available at <http://www-i2.informatik.rwth-aachen.de/~hanus/curry/report.html>, February 2000.
- [6] G. Jäger and R.F. Stärk. A proof-theoretic framework for logic programming. In S.R. Buss (ed.), *Handbook of Proof Theory*, pages 639–682. Elsevier, 1998.
- [7] F.J. López-Fraguas and J. Sánchez-Hernández. Disequalities may help to narrow. In *Proc. APPIA-GULP-PRODE'99*, pages 89–104, 1999.
- [8] F.J. López-Fraguas and J. Sánchez-Hernández. *TOY*: A multiparadigm declarative system. In *Proc. RTA'99, Springer LNCS 1631*, pages 244–247, 1999.
- [9] F.J. López-Fraguas and J. Sánchez-Hernández. Proving failure in functional logic programs (extended version). Tech. Rep. SIP 00/100-00, UCM Madrid, 2000.
- [10] J.J. Moreno-Navarro. Default rules: An extension of constructive negation for narrowing-based languages. In *Proc. ICLP'95*, pages 535–549. MIT Press, 1994.
- [11] J.J. Moreno-Navarro. Extending constructive negation for partial functions in lazy functional-logic languages. In *Proc. ELP'96*, pages 213–227. Springer LNAI 1050, 1996.
- [12] P.J. Stuckey. Constructive negation for constraint logic programming. In *Proc. LICS'91*, pages 328–339, 1991.

Appendix: Proofs of the results

Proposition 1. *The relations $\downarrow, \uparrow, \Downarrow, \Upsilon$ verify*

- a) For all $t, t', s, s' \in CTerm_{\perp, \mathfrak{F}}$
 - i) $t \downarrow t' \Leftrightarrow \hat{t} \downarrow \hat{t}'$ and $t \uparrow t' \Leftrightarrow \hat{t} \uparrow \hat{t}'$
 - ii) $t \uparrow t' \Rightarrow t \Downarrow t' \Rightarrow \neg(t \downarrow t')$
 - iii) $t \downarrow t' \Rightarrow t \Upsilon t' \Rightarrow \neg(t \uparrow t')$
- b) $\downarrow, \uparrow, \Downarrow, \Upsilon$ are monotonic, i.e., if $t \sqsubseteq s$ and $t' \sqsubseteq s'$ then: $t \mathfrak{R} t' \Rightarrow s \mathfrak{R} s'$, where $\mathfrak{R} \in \{\downarrow, \uparrow, \Downarrow, \Upsilon\}$. Furthermore \Downarrow_B and Υ_B are the greatest monotonic approximations to $\neg \downarrow_B$ and $\neg \uparrow_B$ respectively, where \mathfrak{R}_B is the restriction of \mathfrak{R} to the set of basic (i.e., without variables) c-terms from $CTerm_{\perp, \mathfrak{F}}$.
- c) \downarrow and Υ are closed under substitutions from $CSubst$; \Downarrow and \uparrow are closed under substitutions from $CSubst_{\perp, \mathfrak{F}}$

Proof. We prove each property separately:

- a) i) $t \downarrow t' \Leftrightarrow \hat{t} \downarrow \hat{t}'$: two terms satisfying the relation \downarrow can not contain \perp neither \mathfrak{F} , what means that they remain identical under the hat, so $t = \hat{t}$ and $t' = \hat{t}'$, and both implications are clear.
 $t \uparrow t' \Leftrightarrow \hat{t} \uparrow \hat{t}'$: the relation \uparrow satisfies when the terms have a *DC*-clash at some position p ; the hat does not affect to constructors of *DC*, so t and \hat{t} (t' and \hat{t}' resp.) have the same constructor at position p and then we have the same *DC*-clash in both sides of the implication.
- ii) The implication $t \uparrow t' \Rightarrow t \Downarrow t'$ is clear from definitions of \uparrow and \Downarrow . For $t \Downarrow t' \Rightarrow \neg(t \downarrow t')$: if $t \Downarrow t'$ then it can be because \mathfrak{F} appears in t or t' , or because t and t' have a *DC*-clash. In both cases it can not be $t \downarrow t'$.
- iii) For $t \downarrow t' \Rightarrow t \Upsilon t'$: if $t \downarrow t'$ then $t = t'$ with $t \in CTerm$ and we have $t \Upsilon t'$ by applying repeatedly i) and iii) of the definition of Υ . For $t \Upsilon t' \Rightarrow \neg(t \uparrow t')$ let us assume $t \Upsilon t'$ and proceed by induction on the depth d of t :
 $d = 0$: if $t = \perp$ or $t = \mathfrak{F}$ then t and t' can not have any *DC*-clash and then $t \uparrow t'$ is not true. If $t = X$ or $t = c \in DC^0$ then it has to be $t' = \mathfrak{F}$ or $t' = t$ for having $t \Upsilon t'$, therefore t and t' can not have any *DC*-clash and $t \uparrow t'$ is not true.
 $d + 1$: if $t = c(t_1, \dots, t_n)$, it can be $t' = \mathfrak{F}$ and then $t \uparrow t'$ is not true, or $t' = c(t'_1, \dots, t'_n)$ with $t_i \Upsilon t'_i$ for all $i \in \{1, \dots, n\}$; in this case, by i.h. there is not a pair (t_i, t'_i) with a *DC*-clash, therefore t and t' neither have *DC*-clashes, so $t \uparrow t'$ is not true.
- b) We prove monotonicity for each relation:
 - For \downarrow : by definition of \downarrow , if $t \downarrow t'$ then $t, t' \in CTerm$ (they are maximal with respect to \sqsubseteq), so it has to be $s = t$ and $s' = t'$ and then it is clear $s \downarrow s'$
 - For \uparrow : if $t \uparrow t'$ then t and t' have a *DC*-clash at some position. As $t \sqsubseteq s$ and $t' \sqsubseteq s'$, then s and s' will have the same *DC*-clash at the same position, so $s \uparrow s'$.

- For $\not\sqsubseteq$: if t and t' have a DC -clash, s and s' will contain the same DC -clash, as in *ii*). If one of them have ε as subterm, by definition of \sqsubseteq it is clear that s or s' will also contain ε , so $s \not\sqsubseteq s'$.
- For $\not\sqsupset$: Here we proceed by induction on the depth d of the term t :
 $d = 0$: let us check the possibilities for t . If $t = X$ or $t = c \in DC^0$ it must be $t' = t$ or $t' = \varepsilon$ for having $t \not\sqsupset t'$; anyway, as they are maximal with respect to \sqsubseteq , it will be $s = t$ and $s' = t'$, so we will also have $s \not\sqsupset s'$. If $t = \varepsilon$ then $s = \varepsilon$ and then it is clear that $s \not\sqsupset s'$. If $t = \perp$ then it must be $t' = \varepsilon$, so $s' = \varepsilon$ and it is clear that $s \not\sqsupset s'$.
 $d + 1$: it has to be $t = c(t_1, \dots, t_n)$ then it could be $t' = \varepsilon$, what implies $s' = \varepsilon$ and then $s \not\sqsupset s'$, or it could be $t' = c(t'_1, \dots, t'_n)$ with $t_i \not\sqsupset t'_i$ for all $i \in \{1, \dots, n\}$. From $t \sqsubseteq s$ and $t' \sqsubseteq s'$ follows that $s = c(s_1, \dots, s_n)$ and $s' = c(s'_1, \dots, s'_n)$ and by i.h. we have $s_i \not\sqsupset s'_i$ for all $i \in \{1, \dots, n\}$ what implies $s \not\sqsupset s'$.

Now we will see that $\not\sqsubseteq_B$ and $\not\sqsupset_B$ are the greatest monotonic approximations to $\neg \downarrow_B$ and $\neg \uparrow_B$ respectively:

- For $\not\sqsubseteq_B$ the property can be enunciated as follows: let $R \subseteq (BCTerm_{\perp, \varepsilon} \times BCTerm_{\perp, \varepsilon})$ such that:

$$\left. \begin{array}{l} tRt' \Rightarrow \neg(t \downarrow_B t') \\ t \sqsubseteq s, t' \sqsubseteq s', tRt' \Rightarrow sRs' \end{array} \right\} \Rightarrow (tRt' \Rightarrow t \not\sqsubseteq_B t')$$

Assume tRt' and $\neg(t \not\sqsubseteq_B t')$, then t and t' can not have a DC -clash by definition of $\not\sqsubseteq_B$. Then there must exist $s \in BCTerm$ (totally defined) such that $t \sqsubseteq s$ and $t' \sqsubseteq s$. By monotonicity of R we have sRs what implies $\neg(s \downarrow_B s)$ being $s \in BCTerm$; this is a contradiction due to the assumption $\neg(t \not\sqsubseteq_B t')$, so $t \not\sqsubseteq_B t'$.

- For $\not\sqsupset_B$ in a similar way as in the previous point: let $R \subseteq (BCTerm_{\perp, \varepsilon} \times BCTerm_{\perp, \varepsilon})$ such that:

$$\left. \begin{array}{l} tRt' \Rightarrow \neg(t \uparrow_B t') \\ t \sqsubseteq s, t' \sqsubseteq s', tRt' \Rightarrow sRs' \end{array} \right\} \Rightarrow (tRt' \Rightarrow t \not\sqsupset_B t')$$

If tRt' then $\neg(t \uparrow_B t')$, so t and t' can not have any DC -clash. They could contain ε as subterm but, by *ii*) and *iii*) of the definition of $\not\sqsupset$, we will have $t \not\sqsupset_B t'$.

- c) The property is clear for \downarrow : if we replace in a c -term all the occurrences of a variable by a totally defined c -term, we will obtain a totally defined c -term. For \uparrow , such substitution preserves the DC -clash of the original c -terms. For $\not\sqsubseteq$, if some of the original c -terms had ε as a subterm, the substitution preserves this occurrence of ε . On the other hand, if they had a DC -clash, then it is clear that this clash will also be present under the substitution. For $\not\sqsupset$, suppose $t \not\sqsupset t'$ and $\theta \in CSusbt_{\perp, \varepsilon}$; we proceed by induction on the depth d of the term t :
 $d = 0$: if $t = \varepsilon$, then $t\theta = \varepsilon$ and it is clear that $t\theta \not\sqsupset t'\theta$. For the cases $t = X$ and $t = c \in DC^0$ we have two possibilities for t' : $t' = \varepsilon$ or $t' = t$; if $t' = \varepsilon$ the result is clear. If we have $t = t' = X$ it is not difficult to prove that $X\theta \not\sqsupset X\theta$ by applying repeatedly *i*) and *iii*) of definition of $\not\sqsupset$. The last case, if $t = t' = c \in DC^0$ is trivial because θ do not change the terms.

$d+1$: it must be $t = c(t_1, \dots, t_n)$. If $t' = \mathbb{F}$ the proof is as in the base case, else it has to be $t' = c(t'_1, \dots, t'_n)$ with $t_i \not\gamma t'_i$ for all $i \in \{1, \dots, n\}$. By i.h. we have $t_i \theta \not\gamma t'_i \theta$ and then, by *iii*) of the definition of γ we will have $t \theta \not\gamma t' \theta$. \square

Lemma 1 (Consistency). *For any $e, e', e_1, e_2, e'_1, e'_2 \in Term_{\perp, \mathbb{F}}$*

- a) *If e, e' are consistent, $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}$ and $\mathcal{P} \vdash_{CRWLF} e' \triangleleft \mathcal{C}'$, then \mathcal{C} and \mathcal{C}' are consistent.*
- b) *If e_1, e'_1 are consistent and e_2, e'_2 are also consistent, then: $\mathcal{P} \vdash_{CRWLF} e_1 \diamond e_2 \Rightarrow \mathcal{P} \not\vdash_{CRWLF} e'_1 \tilde{\diamond} e'_2$*

Proof. For proving the consistency lemma we will split b) into b.1), b.2) and also strengthen the lemma with a new part c):

- b) If e_1, e'_1 are consistent and e_2, e'_2 are also consistent, then:
 - b.1) $\mathcal{P} \vdash_{CRWLF} e_1 \bowtie e_2 \Rightarrow \mathcal{P} \not\vdash_{CRWLF} e'_1 \bowtie e'_2$
 - b.2) $\mathcal{P} \vdash_{CRWLF} e_1 \diamond e_2 \Rightarrow \mathcal{P} \not\vdash_{CRWLF} e'_1 \diamond e'_2$
- c) Given $\bar{t}, \bar{t}' \in CTerm_{\perp, \mathbb{F}} \times \dots \times CTerm_{\perp, \mathbb{F}}$ pairwise consistent and $R \in \mathcal{P}_f$, if $\mathcal{P} \vdash_{CRWLF} f(\bar{t}) \triangleleft_R \mathcal{C}, f(\bar{t}') \triangleleft_R \mathcal{C}'$, then \mathcal{C} and \mathcal{C}' are consistent.

Now we will prove a), b) and c) simultaneously by induction on the length l of the proof for $e \triangleleft \mathcal{C}$ in a), $e_1 \bowtie e_2$ in b.1), $e_1 \diamond e_2$ in b.2) and $f(\bar{t}) \triangleleft_R \mathcal{C}$ in c).

$l = 0$

- a) The possible proofs in 0 steps are:
 - $e \triangleleft \{\perp\}$. This SAS is consistent with any other;
 - $X \triangleleft \{X\}$. It could be $e' = X$ or $e' = \perp$, so the possibilities for \mathcal{C}' are $\{X\}$ or $\{\perp\}$, both consistent with $\{X\}$;
 - $c \triangleleft \{c\}$, being $c \in DC^0 \cup \{\mathbb{F}\}$. In this case e' must be c or \perp whose possible SAS's are $\{c\}$ and $\{\perp\}$, that are consistent with $\{c\}$.
- b) There are no possible proofs of the form $e \bowtie e'$ or $e \diamond e'$ in 0 steps.
- c) The possible proofs of the form $f(\bar{t}) \triangleleft_R \mathcal{C}$ are:
 - $f(\bar{t}) \triangleleft_R \{\perp\}$. This SAS is consistent with any other;
 - $f(\bar{t}) \triangleleft_R \{\mathbb{F}\}$, by means of rule 8, i.e., there exist some $R \equiv (f(\bar{s}) \rightarrow e \Leftarrow \bar{C}) \in \mathcal{P}_f$ and some i such that s_i and t_i have a $DC \cup \{\mathbb{F}\}$ -clash at some position p . The SAS \mathcal{C}' for $f(\bar{t}')$ using the function rule R must be done by one of the rules 5 to 8:
 - * by rule 5 it would be $\mathcal{C}' = \{\perp\}$ that is consistent with \mathcal{C}' ;
 - * by 6 it is not possible: t_i and t'_i are consistent because \bar{t} and \bar{t}' are pairwise consistent; then t'_i at position p must have the same constructor symbol that t_i , and then the clash with s_i remains, or t'_i at p or some of its ancestor positions have \perp . In both cases it is clear that there is not any c-instance of R for using rule 6;
 - * by rules 7 or 8 the SAS is $\{\mathbb{F}\}$ that is consistent with the initial one $\{\mathbb{F}\}$.

$l + 1$

a) In $l + 1$ steps the possible proofs for $e \triangleleft \mathcal{C}$ are:

- $\frac{e_1 \triangleleft \mathcal{C}_1 \quad \dots \quad e_n \triangleleft \mathcal{C}_n}{e = c(e_1, \dots, e_n) \triangleleft \{c(t_1, \dots, t_n) \mid \bar{t} \in \mathcal{C}_1 \times \dots \times \mathcal{C}_n\}}$ by rule 3, being $c \in DC^n$ ($n > 0$). Then it can be $e' = \perp$, whose only possible SAS is $\{\perp\}$, that is consistent with any other, or $e' = c(e'_1, \dots, e'_n)$ and the SAS is produced by rule 3: $\frac{e_1 \triangleleft \mathcal{C}'_1 \quad \dots \quad e_n \triangleleft \mathcal{C}'_n}{e' = c(e'_1, \dots, e'_n) \triangleleft \{c(t'_1, \dots, t'_n) \mid \bar{t}' \in \mathcal{C}'_1 \times \dots \times \mathcal{C}'_n\}}$
By i.h. \mathcal{C}'_i is consistent with \mathcal{C}_i for all $i \in \{1, \dots, n\}$ and then it is clear that \mathcal{C} and \mathcal{C}' are also consistent.
- $\frac{e_1 \triangleleft \mathcal{C}_1 \quad \dots \quad e_n \triangleleft \mathcal{C}_n \quad f(\bar{t}) \triangleleft_R \mathcal{C}_{R, \bar{t}}}{e = f(e_1, \dots, e_n) \triangleleft \bigcup_{R \in \mathcal{P}_f, \bar{t} \in \mathcal{C}_1 \times \dots \times \mathcal{C}_n} \mathcal{C}_{R, \bar{t}}}$ by rule 4. It can be $e' = \perp$ whose only possible SAS is $\{\perp\}$ that is consistent with any other, or $e' = f(e'_1, \dots, e'_n)$ with e_i, e'_i consistent for all $i \in \{1, \dots, n\}$. If the SAS for e' is generated by rule 1 of the calculus, the result would be clear and for rule 4 we have $\frac{e'_1 \triangleleft \mathcal{C}'_1 \quad \dots \quad e'_n \triangleleft \mathcal{C}'_n \quad f(\bar{t}') \triangleleft_R \mathcal{C}_{R, \bar{t}'}}{e' = f(e'_1, \dots, e'_n) \triangleleft \bigcup_{R \in \mathcal{P}_f, \bar{t}' \in \mathcal{C}'_1 \times \dots \times \mathcal{C}'_n} \mathcal{C}_{R, \bar{t}'}}$
By i.h. \mathcal{C}_i and \mathcal{C}'_i are consistent for all $i \in \{1, \dots, n\}$, what means that for each $\bar{t} \in \mathcal{C}_1 \times \dots \times \mathcal{C}_n$ there exists $\bar{t}' \in \mathcal{C}'_1 \times \dots \times \mathcal{C}'_n$ consistent with \bar{t} . Again by i.h. we have that each SAS $\mathcal{C}_{R, \bar{t}}$ is consistent with $\mathcal{C}_{R, \bar{t}'}$ and it can be easily proved that $\mathcal{C} = \bigcup_{R \in \mathcal{P}_f, \bar{t} \in \mathcal{C}_1 \times \dots \times \mathcal{C}_n} \mathcal{C}_{R, \bar{t}}$ is consistent with $\mathcal{C}' = \bigcup_{R \in \mathcal{P}_f, \bar{t}' \in \mathcal{C}'_1 \times \dots \times \mathcal{C}'_n} \mathcal{C}_{R, \bar{t}'}$.

- b.1) If we have a proof for $e_1 \bowtie e_2$ by rule 9, there exist two SAS's \mathcal{C}_{e_1} and \mathcal{C}_{e_2} such that $e \triangleleft \mathcal{C}_{e_1}, e_2 \triangleleft \mathcal{C}_{e_2}$ and there exist $t \in \mathcal{C}_{e_1}, t' \in \mathcal{C}_{e_2}$ with $t \downarrow t'$.
Now, assume that we can prove $e'_1 \bowtie e'_2$ being e'_1 consistent with e_1 and e'_2 consistent with e_2 . Then we can prove $e'_1 \triangleleft \mathcal{C}_{e'_1}, e'_2 \triangleleft \mathcal{C}_{e'_2}$ such that for all $s \in \mathcal{C}_{e'_1}, s' \in \mathcal{C}_{e'_2}$ it will be $s \not\downarrow s'$.
By i.h. \mathcal{C}_{e_1} is consistent with $\mathcal{C}_{e'_1}$, what implies that there exists $u \in \mathcal{C}_{e'_1}$ consistent with t , and then there exists v such that $v \supseteq u, v \supseteq t$. In a similar way, there exists $u' \in \mathcal{C}_{e'_2}$ consistent with t' , so there exists v' such that $v' \supseteq u', v' \supseteq t'$.
As $u \in \mathcal{C}_{e'_1}$ and $u' \in \mathcal{C}_{e'_2}$ we would have $u \not\downarrow u'$; by monotonicity of $\not\downarrow$ we have $v \not\downarrow v'$, that implies $\neg(v \downarrow v')$; again by monotonicity of \downarrow (negation of the implication) we have $\neg(t \downarrow t')$, what is a contradiction; so it is not provable $e'_1 \bowtie e'_2$ being e'_1 consistent with e_1 and e'_2 consistent with e_2 .
- b.2) If we have a proof for $e_1 \diamond e_2$ using rule 10, there exist two SAS's \mathcal{C}_{e_1} and \mathcal{C}_{e_2} such that $e_1 \triangleleft \mathcal{C}_{e_1}, e_2 \triangleleft \mathcal{C}_{e_2}$ and there exist $t \in \mathcal{C}_{e_1}, t' \in \mathcal{C}_{e_2}$ with $t \uparrow t'$.
Assume we can prove $e'_1 \diamond e'_2$ being e'_1 consistent with e_1 and e'_2 consistent with e_2 . Then we can prove two SAS's $e'_1 \triangleleft \mathcal{C}_{e'_1}, e'_2 \triangleleft \mathcal{C}_{e'_2}$ such that for all $s \in \mathcal{C}_{e'_1}, s' \in \mathcal{C}_{e'_2}$ it will be $s \not\uparrow s'$.
By i.h. \mathcal{C}_{e_1} is consistent with $\mathcal{C}_{e'_1}$, what implies that there exists $u \in \mathcal{C}_{e'_1}$ consistent with t , and this implies that there exists v such that $v \supseteq u, v \supseteq t$. In a similar way there exists $u' \in \mathcal{C}_{e'_2}$ consistent with t' , so there exists v' such that $v' \supseteq u', v' \supseteq t'$.

As $u \in \mathcal{C}_{e'_1}$ and $u' \in \mathcal{C}_{e'_2}$ we would have $u \not\gamma u'$; by monotonicity of γ we have $v \not\gamma v'$, that implies $\neg(v \uparrow v')$; again by monotonicity of \uparrow (negation of the implication) we have $\neg(t \uparrow t')$, what is a contradiction; so $e'_1 \not\triangleleft e'_2$, being e'_1 consistent with e_1 and e'_2 consistent with e_2 , is not provable.

c) In $l+1$ steps the possible proofs for $f(\bar{t}) \triangleleft_R \mathcal{C}$ being $R \equiv (f(\bar{s}) \rightarrow e \leftarrow \bar{C})$ are:

- $\frac{e\theta \triangleleft \mathcal{C} \quad \bar{C}\theta}{f(\bar{t}) \triangleleft_R \mathcal{C}}$ by rule 6, using the c-instance $R\theta$ ($\theta \in CSubst_{\perp, \mathbb{F}}$), such that $\bar{t} = \bar{s}\theta$. The proof $f(\bar{t}') \triangleleft_R \mathcal{C}'$ must be done by one of the rules 5 to 8:
 - if $f(\bar{t}') \triangleleft_R \{\perp\}$ by rule 5, it is clear that this SAS is consistent with \mathcal{C} ;
 - if the proof is done by rule 6, it will have the form $\frac{e\theta' \triangleleft \mathcal{C}' \quad \bar{C}'\theta'}{f(\bar{t}') \triangleleft_R \mathcal{C}'}$ using a c-instance $R\theta'$ of R . In particular, we have $\bar{t}' = \bar{s}\theta'$ and we also had $\bar{t} = \bar{s}\theta$. As \bar{t} and \bar{t}' are pairwise consistent, and $var(e) \subseteq var(\bar{s})$ it is not difficult to see that $e\theta$ and $e\theta'$ must be consistent. Then by i.h. (part a)) we will have that \mathcal{C} and \mathcal{C}' are consistent SAS's.
 - rule 7 is not applicable: suppose that we have the proof $\frac{\widetilde{C}_i\theta'}{f(\bar{t}') \triangleleft_R \{\mathbb{F}\}}$ using a c-instance $R\theta'$ of R and being $C_i\theta'$ a constraint of $\bar{C}'\theta'$. Analogously to the previous case, we have that both members of $C_i\theta'$ are consistent with the corresponding of $C_i\theta$; as $C_i\theta$ is provable then by i.h. (part b)), $\widetilde{C}_i\theta'$ is not provable, what means that rule 7 is can not be applied.
 - rule 8 is not applicable: there can not be a pair of \bar{t}' and \bar{s} with a $DCU\{\mathbb{F}\}$ -clash because then the corresponding pair of \bar{t}' and $\bar{s}\theta = \bar{t}$ would have the same clash (the substitution θ can not make disappear the clash).
- $\frac{\widetilde{C}_i\theta}{f(\bar{t}) \triangleleft_R \{\mathbb{F}\}}$ by rule 7, being $R\theta$ a c-instance of the rule R such that $\bar{t} = \bar{s}\theta$. The proof $f(\bar{t}') \triangleleft_R \mathcal{C}'$ can be done by one of the rules 5 to 8:
 - by rule 5, the SAS is $\{\perp\}$ that is consistent with any other;
 - it is not possible to use rule 6 because we would need to prove a constraint $C_i\theta'$ of a c-instance $R\theta'$ of R . As $\bar{s}\theta = \bar{t}$ and $\bar{s}\theta' = \bar{t}'$ are pairwise consistent and $var(C_i) \subseteq var(\bar{s})$, both members of $C_i\theta$ and $C_i\theta'$ will be also consistent. Then by i.h. (part b)), as $C_i\theta$ is provable, $C_i\theta'$ will not be provable.
 - if 7 or 8 applies we will have $\mathcal{C}' = \{\mathbb{F}\}$ that is consistent with $\mathcal{C} = \{\mathbb{F}\}$ (in fact, 8 would not be applicable). \square

Proposition 2 (Monotonicity of CRWLF). For $e, e', e_1, e_2, e'_1, e'_2 \in Term_{\perp, \mathbb{F}}$

- a) If $e \sqsubseteq e'$ and $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}$, then $\mathcal{P} \vdash_{CRWLF} e' \triangleleft \mathcal{C}$
- b) If $e_1 \sqsubseteq e'_1$, $e_2 \sqsubseteq e'_2$ and $\mathcal{P} \vdash_{CRWLF} e_1 \diamond e_2$ then $\mathcal{P} \vdash_{CRWLF} e'_1 \diamond e'_2$, where $\diamond \in \{\bowtie, \triangleright, \triangleleft, \triangleleft\}$

Proof. Again we need to strengthen the result with a new part c)

- c) Given $\bar{t}, \bar{t}' \in CTerm_{\perp, \mathbb{F}} \times \dots \times CTerm_{\perp, \mathbb{F}}$ such that $t_i \sqsubseteq t'_i$ for all $i \in \{1, \dots, n\}$ and $R \in \mathcal{P}_f$, if $f(\bar{t}) \triangleleft_R \mathcal{C}$ then $f(\bar{t}') \triangleleft_R \mathcal{C}$

We will prove parts a), b) and c) simultaneously by induction on the length l of the proof for $e \triangleleft \mathcal{C}$ in a), $e_1 \diamond e_2$ in b) and $f(\bar{t}) \triangleleft_R \mathcal{C}$ in c):

$l = 0$

- a) The proof $e \triangleleft \mathcal{C}$ in 0 steps can be:
- $e \triangleleft \{\perp\}$, it is clear that $e' \triangleleft \perp$
 - $X \triangleleft \{X\}$ (being $e = X$), it must be $e' = X$
 - $c \triangleleft \{c\}$, $c \in DC^0$, it must be $e = e' = c$
- b) For $e_1 \diamond e_2$ there are not possible proofs in 0 steps.
- c) For $f(\bar{t}) \triangleleft_R \mathcal{C}$ the proofs can be:
- $f(\bar{t}) \triangleleft_R \{\perp\}$ and for all \bar{t}' we have $f(\bar{t}') \triangleleft_R \{\perp\}$
 - $f(\bar{t}) \triangleleft_R \{\mathbb{F}\}$ being $R \equiv (f(\bar{s}) \rightarrow e \Leftarrow \bar{C})$ and \bar{t} and \bar{s} have a DC -clash in some component. If $\bar{t} \sqsubseteq \bar{t}'$ then \bar{t}' and \bar{s} have the same clash.

$l + 1$

- a) The proof $e \triangleleft \mathcal{C}$ can be:
- By rule 3, assume $e = c(e_1, \dots, e_n)$ and $e \sqsubseteq e'$, then $e' = c(e'_1, \dots, e'_n)$ with $e_i \sqsubseteq e'_i$. The proof will be:
$$\frac{e_1 \triangleleft \mathcal{C}_1 \dots e_n \triangleleft \mathcal{C}_n}{c(e_1, \dots, e_n) \triangleleft \{c(t_1, \dots, t_n) \mid \bar{t} \in \mathcal{C}_1 \times \dots \times \mathcal{C}_n\}}$$
 By i.h. we have $e'_1 \triangleleft \mathcal{C}_1, \dots, e'_n \triangleleft \mathcal{C}_n$ and with the same rule 3 we can build the same SAS for $c(\bar{e}')$.
 - By rule 4 the proof would be:
$$\frac{e_1 \triangleleft \mathcal{C}_1 \dots e_n \triangleleft \mathcal{C}_n \quad f(\bar{t}) \triangleleft_R \mathcal{C}_{R, \bar{t}}}{f(e_1, \dots, e_n) \triangleleft \bigcup_{R \in \mathcal{P}_f, \bar{t} \in \mathcal{C}_1 \times \dots \times \mathcal{C}_n} \mathcal{C}_{R, \bar{t}}}$$
 with $e = f(e_1, \dots, e_n)$. It must be $e' = f(e'_1, \dots, e'_n)$ with $e_i \sqsubseteq e'_i$. By i.h. we will have $e'_1 \triangleleft \mathcal{C}_1, \dots, e'_n \triangleleft \mathcal{C}_n$ and then we have the same tuples \bar{t} , the same SAS 's $\mathcal{C}_{R, \bar{t}}$ and then the same SAS for $f(\bar{e}')$.
- b) The proof $e_1 \diamond e_2$ with $\diamond \in \{\bowtie, \not\bowtie, \diamond, \not\diamond\}$ will be done by generating the SAS 's $e_1 \triangleleft \mathcal{C}_1$ and $e_2 \triangleleft \mathcal{C}_2$. By i.h. we have $e'_1 \triangleleft \mathcal{C}_1, e'_2 \triangleleft \mathcal{C}_2$ and then it is clear that $e'_1 \diamond e'_2$ is provable.
- c) By rule 6 the proof would be:
$$\frac{e\theta \triangleleft \mathcal{C} \quad \bar{C}\theta}{f(t_1, \dots, t_n) \triangleleft_R \mathcal{C}}$$
 being $R \equiv (f(s_1, \dots, s_n) \rightarrow e \Leftarrow \bar{C})$ and $\theta \in Subst_{\perp, \mathbb{F}}$ such that $\bar{s}\theta = \bar{t}$.

We will show that the same rule 6 is applicable for generating an SAS for $f(\bar{t}')$ being $t_i \sqsubseteq t'_i$ for all $i \in \{1, \dots, n\}$. The idea is that if $t_i \sqsubseteq t'_i$ then t'_i is the result of replacing some subterms \perp of t_i by c -terms more defined than \perp . As $s_i \in CTerm$ then the corresponding positions or some ancestors must have variables in s_i . Then we can get a substitution $\theta' \in CSubst_{\perp, \mathbb{F}}$ such that $s_i\theta' = t'_i$. A formal justification of this fact may be done by induction on the syntactic structure of t_i and, as \bar{s} is a linear tuple, the result can be extended in such a way that $\bar{s}_i\theta' = \bar{t}'$.

We also have that $e\theta \sqsubseteq e\theta'$, so by i.h. we have $e\theta' \triangleleft \mathcal{C}$. As the constraints $\bar{C}\theta$ are provable, then by i.h. b), the constraints $\bar{C}\theta'$ will also be provable. So we can build a proof for $f(\bar{t}') \triangleleft_R \mathcal{C}$ by rule 6.

- By rule 7 we would have $\frac{\tilde{C}_i\theta}{f(\bar{t}) \triangleleft_R \{F\}}$ being $R \equiv (f(\bar{s}) \rightarrow e \Leftarrow C_1, \dots, C_n), i \in \{1, \dots, n\}$ and $\theta \in CSubst_{\perp, F}$ such that $\bar{s}\theta = \bar{t}$.
As $\bar{t} \sqsubseteq \bar{t}'$, in a similar way as before there exists θ' such that $\bar{s}\theta' = \bar{t}'$ and by i.h. we can prove $\tilde{C}_i\theta'$, what implies that we can build the proof for $f(\bar{t}') \triangleleft_R \{F\}$, using rule 7. \square

Proposition 3. For any $\theta \in CSubst$, $e, e' \in Term_{\perp, F}$

- $\mathcal{P} \vdash_{CRWLF} e \triangleleft C \Rightarrow \mathcal{P} \vdash_{CRWLF} e\theta \triangleleft C\theta$
- $\mathcal{P} \vdash_{CRWLF} e \diamond e' \Rightarrow \mathcal{P} \vdash_{CRWLF} e\theta \diamond e'\theta$

Proof. Again we need to strengthen the result, with a new part c):

- Given $\bar{t}, \bar{t}' \in CTerm_{\perp, F} \times \dots \times CTerm_{\perp, F}$ we have $f(\bar{t}) \triangleleft_R C \Rightarrow f(\bar{t}')\theta \triangleleft_R C\theta$

We prove simultaneously the three parts by induction on the length l of the proofs.

$l = 0$ In 0 steps we can have the proofs $e \triangleleft \{\perp\}, c \triangleleft \{c\}$ ($c \in DC^0$) and $X \triangleleft \{X\}$. The property is clear for the first two and the third follows from the fact that if $t \in CTerm_{\perp, F}$ then $t \triangleleft \{t\}$ is provable (it can be proved by induction on the depth of the term t). Notice that $X\theta \in CTerm \subset CTerm_{\perp, F}$, so $X\theta \triangleleft \{X\theta\}$.

$l + 1$ Now we can have the following proofs:

- by rule 3 we have $\frac{e_1 \triangleleft C_1 \quad \dots \quad e_n \triangleleft C_n}{c(e_1, \dots, e_n) \triangleleft \{c(t_1, \dots, t_n) \mid \bar{t} \in C_1 \times \dots \times C_n\}}$ By i.h. we have $e_i\theta \triangleleft C_i\theta$ for all $i \in \{1, \dots, n\}$ and again by rule 3 we can build a proof for $c(\bar{e}) \triangleleft \{c(t_1, \dots, t_n)\theta \mid \bar{t}\theta \in C_1\theta \times \dots \times C_n\theta\}$
- by 4 we have $\frac{e_1 \triangleleft C_1 \quad \dots \quad e_n \triangleleft C_n \quad \dots \quad f(\bar{t}) \triangleleft_R C_{R, \bar{t}} \quad \dots}{f(e_1, \dots, e_n) \triangleleft \bigcup_{R \in \mathcal{P}_f, \bar{t} \in C_1 \times \dots \times C_n} C_{R, \bar{t}}}$ By i.h. we have $e_i\theta \triangleleft C_i\theta$ for all $i \in \{1, \dots, n\}$ and $f(\bar{t})\theta \triangleleft_R C_{R, \bar{t}}\theta$ for each $\bar{t}\theta \in C_1\theta \times \dots \times C_n\theta$ and each rule $R \in \mathcal{P}_f$. So we can build a proof for $f(\bar{e})\theta \triangleleft \bigcup_{R \in \mathcal{P}_f, \bar{t}\theta \in C_1\theta \times \dots \times C_n\theta} C_{R, \bar{t}\theta}$
- by 5 we have $\frac{}{f(\bar{t}) \triangleleft_R \{\perp\}}$ and it is clear $f(\bar{t})\theta \triangleleft_R \{\perp\}$
- by 6 we have $\frac{e\theta' \triangleleft C \quad \overline{C}\theta'}{f(\bar{t}) \triangleleft_R C}$ being $(f(\bar{s}) \rightarrow e \Leftarrow \overline{C}) \in R$ and $\theta' \in CSubst_{\perp, F}$ such that $f(\bar{t}) = f(\bar{s})\theta'$. For the call $f(\bar{t})\theta$ we can get the appropriate c-instance by composing θ' and θ , so $f(\bar{t})\theta = f(\bar{s})\theta'\theta$. By i.h. we have $e\theta'\theta \triangleleft C\theta$ and $\overline{C}\theta'\theta$, what follows $f(\bar{t})\theta \triangleleft_R C\theta$ by the same rule 6.
- by 7 we have $\frac{e_i\theta' \diamond e'_i\theta'}{f(\bar{t}) \triangleleft_R \{F\}}$ being $(f(\bar{s}) \rightarrow e \Leftarrow \overline{C}) \in R$ and $\theta' \in CSubst_{\perp, F}$ such that $f(\bar{t}) = f(\bar{s})\theta'$. As before, for the call $f(\bar{t})\theta$ we can get the appropriate c-instance by composing θ' and θ , so $f(\bar{t})\theta = f(\bar{s})\theta'\theta$. By i.h. we have $e_i\theta'\theta \diamond e'_i\theta'\theta$, from follows $f(\bar{t})\theta \triangleleft_R \{F\}$

- by 8 we have $\frac{}{f(t_1, \dots, t_n) \triangleleft_R \{F\}}$ being $R \equiv (f(s_1, \dots, s_n) \rightarrow e \leftarrow \overline{C})$ and such that t_i and s_i have a $DC \cup \{F\}$ -clash for some $i \in \{1, \dots, n\}$. It is clear that $t_i\theta$ and s_i will have the same clash so $f(\overline{t})\theta \triangleleft_r \{F\}$
- by 9 to 12, the proof would have the form $e \diamond e'$. By i.h. we have $e\theta \triangleleft C\theta$ $e'\theta \triangleleft C'\theta$. Now, if we take $t \in C, t' \in C'$ and the corresponding $t\theta \in C\theta, t'\theta \in C'\theta$, by 1 if we $t\Re t'$ then $t\theta\Re t'\theta$ (where $\Re \in \{\downarrow, \uparrow, \downarrow, \uparrow\}$). So, the relations between terms of C and C' remains under θ and then $e\theta \diamond e'\theta$. \square

Proposition 4. For any $e, e' \in \text{Term}_{\perp, F}$

- a) $\mathcal{P} \vdash_{CRWLF} e \triangleleft C \Rightarrow \forall t \in C, \mathcal{P} \vdash_{CRWL} \hat{e} \rightarrow \hat{t}$
- b) $\mathcal{P} \vdash_{CRWLF} \hat{e} \rightarrow t \Rightarrow \exists C \text{ such that } t \in C \text{ and } \mathcal{P} \vdash_{CRWLF} e \triangleleft C$
- c) $\mathcal{P} \vdash_{CRWLF} e \bowtie e' \Leftrightarrow \mathcal{P} \vdash_{CRWL} \hat{e} \bowtie \hat{e}'$
- d) $\mathcal{P} \vdash_{CRWLF} e \diamond e' \Leftrightarrow \mathcal{P} \vdash_{CRWL} \hat{e} \diamond \hat{e}'$

We will split this property in two and prove them separately. The first one contains the part a), the implications to the right of part c) and d) and a new part e):

Proposition 4. (Part I) Let \mathcal{P} a CRWLF-program. Then:

- a) $\mathcal{P} \vdash_{CRWLF} e \triangleleft C \Rightarrow \forall t \in C, \mathcal{P} \vdash_{CRWL} \hat{e} \rightarrow \hat{t}$
- c) $\mathcal{P} \vdash_{CRWLF} e \bowtie e' \Rightarrow \mathcal{P} \vdash_{CRWL} \hat{e} \bowtie \hat{e}'$
- d) $\mathcal{P} \vdash_{CRWLF} e \diamond e' \Rightarrow \mathcal{P} \vdash_{CRWL} \hat{e} \diamond \hat{e}'$
- e) Given $\overline{t} \in C\text{Term}_{\perp, F} \times \dots \times C\text{Term}_{\perp, F}$ and $R \in \mathcal{P}_f: \mathcal{P} \vdash_{CRWLF} f(\overline{t}) \triangleleft_R C \Rightarrow \forall t \in C, \mathcal{P} \vdash_{CRWL} \widehat{f(\overline{t})} \rightarrow \hat{t}$

Proof. We will prove simultaneously all the parts by induction on the length l of the corresponding proof in each case:

$l = 0$ The proof can be:

- $\mathcal{P} \vdash_{CRWLF} e \triangleleft \{\perp\}$, and we have $\mathcal{P} \vdash_{CRWL} \hat{e} \rightarrow \perp$
- $\mathcal{P} \vdash_{CRWLF} X \triangleleft \{X\}$, we have $\hat{X} = X$ and $\mathcal{P} \vdash_{CRWL} X \rightarrow X$
- $\mathcal{P} \vdash_{CRWLF} c \triangleleft \{c\}$, where $c \in DC^0$ and we have $\hat{c} = c$ and $\mathcal{P} \vdash_{CRWL} c \rightarrow c$
- $\mathcal{P} \vdash_{CRWLF} F \triangleleft \{F\}$, we have $\hat{F} = \perp$ and $\mathcal{P} \vdash_{CRWL} \perp \rightarrow \perp$
- $\mathcal{P} \vdash_{CRWLF} f(\overline{t}) \triangleleft_R \{\perp\}$, and we have $\mathcal{P} \vdash_{CRWL} \widehat{f(\overline{t})} \rightarrow \perp$
- $\mathcal{P} \vdash_{CRWLF} f(\overline{t}) \triangleleft_R \{F\}$, and we have $\mathcal{P} \vdash_{CRWL} \widehat{f(\overline{t})} \rightarrow \perp$

$l + 1$ the proof can be:

- $\mathcal{P} \vdash_{CRWLF} c(e_1, \dots, e_n) \triangleleft \{\dots, c(t_1, \dots, t_n), \dots\}$, then by the rule 3 of CRWLF, it must be $\mathcal{P} \vdash_{CRWLF} e_i \triangleleft \{\dots, t_i, \dots\}$. By i.h. we have $\mathcal{P} \vdash_{CRWL} \hat{e}_i \rightarrow \hat{t}_i$ and then we can build the proof $\mathcal{P} \vdash_{CRWL} \widehat{c(e_1, \dots, e_n)} \rightarrow \widehat{c(t_1, \dots, t_n)}$, by the rule 3 of CRWL.

- $\mathcal{P} \vdash_{CRWLF} f(e_1, \dots, e_n) \triangleleft \{\dots, t, \dots\}$. This proof must use the rule 4 of $CRWLF$, and then we will have the proofs $\mathcal{P} \vdash_{CRWLF} e_i \triangleleft \mathcal{C}_i$ for all $i \in \{1, \dots, n\}$ and $f(\bar{t}) \triangleleft_R \mathcal{C}_{R, \bar{t}}$. It must be $t \in \mathcal{C}_{R, \bar{t}}$ for some $\bar{t} \in \mathcal{C}_1 \times \dots \times \mathcal{C}_n$ and $R \in \mathcal{P}_f$. By i.h. we will have $\mathcal{P} \vdash_{CRWL} \widehat{f(\bar{t})} \rightarrow \hat{t}$
- $\mathcal{P} \vdash_{CRWLF} f(\bar{t}) \triangleleft_R \mathcal{C}$ by rule 6 of $CRWLF$, for which we will take $R \equiv (f(\bar{s}) \rightarrow e \Leftarrow \bar{C}), \theta \in CSubst_{\perp, \mathbb{F}}$ such that $\bar{s}\theta = \bar{t}$. We can define $\theta' \in CSubst_{\perp}$ as $X\theta' = \perp$ if $X\theta = \mathbb{F}$ and $X\theta' = X\theta$, in other case. So we have $\bar{s}\theta' = \hat{\bar{t}}, e\theta' = \widehat{e\theta}$ and $\bar{C}\theta' = \widehat{\bar{C}\theta}$. Now we can take $(f(\bar{s}) \rightarrow e \Leftarrow \bar{C})\theta' \in [\mathcal{P}]_{\perp}$. We also have $e\theta \triangleleft \mathcal{C}$ and if $t \in \mathcal{C}$ by i.h. we have $\widehat{e\theta} \rightarrow \hat{t}$, or what is the same, $e\theta' \rightarrow \hat{t}$. Also by i.h. $\bar{C}\theta' = \widehat{\bar{C}\theta}$ is provable.
- $\mathcal{P} \vdash_{CRWLF} f(\bar{t}) \triangleleft_R \{\mathbb{F}\}$ and we have $\mathcal{P} \vdash_{CRWL} \widehat{f(\bar{t})} \rightarrow \perp$
- $\mathcal{P} \vdash_{CRWLF} e \bowtie e'$ using the rule 9 of $CRWLF$. Then we will have $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}, \mathcal{P} \vdash_{CRWLF} e' \triangleleft \mathcal{C}'$ and there exists $t \in \mathcal{C}, t' \in \mathcal{C}'$ such that $t \downarrow t'$ (by definition of \downarrow it is easy to see that $t = t'$). By i.h. we have $\mathcal{P} \vdash_{CRWL} \hat{e} \rightarrow \hat{t}$ and $\mathcal{P} \vdash_{CRWL} \hat{e}' \rightarrow \hat{t}$ and by rule 5 of $CRWL$ we have $\mathcal{P} \vdash_{CRWL} \hat{e} \bowtie \hat{t}$.
- $\mathcal{P} \vdash_{CRWLF} e \diamond e'$ using the rule 10 of $CRWLF$. Then we will have $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}, \mathcal{P} \vdash_{CRWLF} e' \triangleleft \mathcal{C}'$ and there exists $t \in \mathcal{C}, t' \in \mathcal{C}'$ such that $t \uparrow t'$. By definition of \uparrow , t and t' have a conflict of constructors of DC . By i.h. we have $\mathcal{P} \vdash_{CRWL} \hat{e} \rightarrow \hat{t}$ and $\mathcal{P} \vdash_{CRWL} \hat{e}' \rightarrow \hat{t}$ and by rule 5 of $CRWL$ we have $\mathcal{P} \vdash_{CRWL} \hat{e} \diamond \hat{t}$. \square

Now the second part of Proposition 4, in which we will prove parts b) and the implications to the left of part c) and d).

Proposition 4. (Part II) For any $e, e' \in Term_{\perp, \mathbb{F}}$

- b) $\mathcal{P} \vdash_{CRWL} \hat{e} \rightarrow t \Rightarrow \exists \mathcal{C}$ such that $t \in \mathcal{C}$ and $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}$
- c) $\mathcal{P} \vdash_{CRWL} \hat{e} \bowtie \hat{e}' \Rightarrow \mathcal{P} \vdash_{CRWLF} e \bowtie e'$
- d) $\mathcal{P} \vdash_{CRWL} \hat{e} \diamond \hat{e}' \Rightarrow \mathcal{P} \vdash_{CRWLF} e \diamond e'$

Proof. We will prove the three parts simultaneously by by induction on the length l of the proof:

$l = 0$ the proof can be:

- $\mathcal{P} \vdash_{CRWL} \hat{e} \rightarrow \perp$ and it is clear that $\mathcal{P} \vdash_{CRWLF} e \triangleleft \{\perp\}$
- $\mathcal{P} \vdash_{CRWL} X \rightarrow X$ and it is clear that $\mathcal{P} \vdash_{CRWLF} X \triangleleft \{X\}$
- $\mathcal{P} \vdash_{CRWL} c \rightarrow c$ with $c \in DC^0$ and it is clear that $\mathcal{P} \vdash_{CRWLF} c \triangleleft \{c\}$

$l + 1$ the proof can have four forms:

- $\mathcal{P} \vdash_{CRWL} c(\widehat{e_1, \dots, e_n}) \rightarrow c(t_1, \dots, t_n)$ by rule 3 of $CRWL$ and then we have $\mathcal{P} \vdash_{CRWL} \hat{e}_i \rightarrow t_i$ for all $i \in \{1, \dots, n\}$. By i.h. we have $\mathcal{P} \vdash_{CRWLF} e_i \triangleleft \mathcal{C}_i$ with $t_i \in \mathcal{C}_i$ and by rule 3 of $CRWLF$ we have $\mathcal{P} \vdash_{CRWLF} c(e_1, \dots, e_n) \triangleleft \mathcal{C}$ with $c(t_1, \dots, t_n) \in \mathcal{C}$.

- $\mathcal{P} \vdash_{CRWL} f(\widehat{e_1, \dots, e_n}) \rightarrow t$, then there must exist a rule $R = (f(\vec{s}) \rightarrow e \Leftarrow \overline{C}) \in \mathcal{P}$ and $\theta \in CSubst_{\perp}$ such that by rule 4 of *CRWL* we will have the proof $\frac{\hat{e}_1 \rightarrow s_1\theta \dots \hat{e}_n \rightarrow s_n\theta \quad e\theta \rightarrow t \quad \overline{C}\theta}{f(\widehat{e_1, \dots, e_n}) \rightarrow t}$ By i.h. we have:
 - i) there exists \mathcal{C}_i such that $\mathcal{P} \vdash_{CRWLF} e_i \triangleleft \mathcal{C}_i$ with $s_i\theta \in \mathcal{C}_i$
 - ii) there exists \mathcal{C}' such that $\mathcal{P} \vdash_{CRWLF} e\theta \triangleleft \mathcal{C}'$ with $t \in \mathcal{C}'$
 - iii) $\mathcal{P} \vdash_{CRWLF} \overline{C}\theta$
 From ii) and iii), by rule 6 of *CRWLF* we can build the proof $\mathcal{P} \vdash_{CRWLF} f(s_i\theta) \triangleleft_R \mathcal{C}'$ using the c-instance $R\theta$. With this proof and i) we have $\mathcal{P} \vdash_{CRWLF} f(\vec{e}) \triangleleft \mathcal{C}$ such that $\mathcal{C}' \subseteq \mathcal{C}$, so $t \in \mathcal{C}$.
- $\mathcal{P} \vdash_{CRWL} \hat{e} \bowtie \hat{e}'$, using the rule 5 of *CRWL*, that follows $\mathcal{P} \vdash_{CRWL} \hat{e} \rightarrow t$ and $\mathcal{P} \vdash_{CRWL} \hat{e}' \rightarrow t$ being $t \in CTerm$. By i.h. $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}$ and $\mathcal{P} \vdash_{CRWL} e \triangleleft \mathcal{C}'$ where $t \in \mathcal{C} \cap \mathcal{C}'$. Taking into account that $t \downarrow t$ for all $t \in CTerm$, by rule 9 of *CRWLF* we can build a proof for $\mathcal{P} \vdash_{CRWLF} e \bowtie e'$.
- $\mathcal{P} \vdash_{CRWL} \hat{e} \diamond \hat{e}'$, using the rule 6 of *CRWL*, that follows $\mathcal{P} \vdash_{CRWL} \hat{e} \rightarrow t$ and $\mathcal{P} \vdash_{CRWL} \hat{e}' \rightarrow t'$ where $t, t' \in CTerm_{\perp}$ and they *DC*-clash. By i.h. $\mathcal{P} \vdash_{CRWLF} e \triangleleft \mathcal{C}$ and $\mathcal{P} \vdash_{CRWL} e \triangleleft \mathcal{C}'$ where $t \in \mathcal{C}$ and $t' \in \mathcal{C}'$. By definition of \uparrow and by rule 11 of *CRWLF* we can build a proof for $\mathcal{P} \vdash_{CRWLF} e \diamond e'$. \square