Termination of ELAN strategies by simplification - Extended version -

Olivier Fissore, Isabelle Gnaedig, Hélène Kirchner

Termination of ELAN strategies by simplification – Extended version –

Olivier Fissore, Isabelle Gnaedig, Hélène Kirchner

LORIA-INRIA & LORIA-CNRS
BP 239 F-54506 Vandœuvre-lès-Nancy Cedex
e-mail: fissore@loria.fr, gnaedig@loria.fr, Helene.Kirchner@loria.fr

Abstract. We propose a transformation based method for proving termination of ELAN strategies. We first give a sufficient criterion for ELAN strategies to terminate, only lying on rewrite rules involved in the strategy. We then give a simplification process of strategies, itself described by rewriting, to empower the previous criterion. This simplification, beyond easing termination proof of strategies, can both facilitate elaboration of specifications and ease proofs of other program properties.

1 Strategy-guided evaluation in rule-based languages

Rule-based languages like ASF+SDF [18], Maude [8], Cafe-OBJ [11], Stratego [24] or ELAN [4, 3] are now used for various applications like constraint solving, protocol verification, modelization of biological or chemical systems, and more. Nowadays, these systems deserve all the more interest as they are backed up by efficient compilers or interpreters. The declarative aspects of programming brought by rewrite rules can be enforced by a declarative language for expressing control as for instance in [15]. In a more general context, methods for controlling non-deterministic computations [20, 21, 16] are used to reduce backtracking.
The ELAN system, in particular, provides an environment for specifying and prototyping deduction systems by allowing to combine rules with strategy operators. Evaluating a data in the rewriting context consists in exploring all possible rewriting derivations. The strategy operators combining the rules are used to refine the rewriting process while preserving its non-determinism. They restrict the search space by describing the form of the best derivations to get results.

In this context, as in programming in general, termination is a key property. It warrants, in particular, the existence of a result for every evaluation of a program. Our purpose here is to study termination of ELAN-like strategies. The idea is to transform the termination problem of a strategy, very difficult to tackle directly, into the termination study of a conditional rewriting system (CRS in short). The transformation we propose is a simplification in the sense that operators of the strategy are suppressed, as well as redundancy of rules implicated in the program. The structure of the resulting program is then simpler ; it has in general significantly less rules than the initial one, and so is in general more readable.

For related work, R. Kieburtz, in [17], defines a programming logic for rewriting Stratego-like strategies [24]. His study consists in characterizing the largest set of terms verifying a certain property P once transformed by a strategy S. However, no termination result emerges from this work, and recursion is explicitly assumed terminating.
Some of the simplifications of this paper have been inspired from [19], where many laws on ELAN-like strategies (called *tactics*) are established. The notion of simplification of strategies also occurs in [25], but, as we will see, ours is the first one that preserves both semantics and termination behaviour, may the strategies terminate or not. This work can also be of interest for rule-based languages like Stratego [24], Maude [8] and Cafe-OBJ [11], previously cited, and for declarative languages [21, 16], since they have strategy features similar to the ones described in this work.

The paper is organized as follows. In Section 2, we present the background and introduce the strategies studied in the paper. In Section 3, we introduce a method for proving termination of these strategies. In Section 4, a simplification process empowering this method is proposed. Illustrative examples are given.

2 Rule-based programs with strategies

We assume that the reader is familiar with the basic definitions and notations of term rewriting given for instance in [7]. In the following, we denote $\mathcal{X}$ a (potentially infinite) set of variables. $\mathcal{T}(\mathcal{F},\mathcal{X})$ is the set of terms built from a given finite set $\mathcal{F}$ of function symbols f having (possibly variable) arity $n \in \mathbb{N}$ (which is denoted $f:n$), and the set $\mathcal{X}$. Given a term $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$, $\mathcal{V}ar(t)$ denotes the set of variables in t. The normal form of a term t w.r.t. a TRS $\mathcal{R}$ is denoted $t{\downarrow_{\mathcal{R}}}$. A substitution is an assignment from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F},\mathcal{X})$, written $\sigma = (x \mapsto t)\ldots(y \mapsto u)$, and identified with the finite set of equations $(x = t) \wedge \ldots \wedge (y = u)$. It uniquely extends to an endomorphism of $\mathcal{T}(\mathcal{F},\mathcal{X})$. The result of applying σ to a term $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ is written $\sigma(t)$ or σt. The domain of σ, denoted $Dom(\sigma)$ is the finite subset of $\mathcal{X}$ such that $\sigma x \neq x$. The range of σ, denoted $Ran(\sigma)$, is defined by $Ran(\sigma) = \bigcup_{x \in Dom(\sigma)} Var(\sigma x)$. We have in addition $Dom(\sigma) \cap Ran(\sigma) = \emptyset$. The restriction of a substitution σ to a set of variables $\mathcal{X}$ is denoted $\sigma_{\mathcal{X}}$, and is defined by $Dom(\sigma_{\mathcal{X}}) \subseteq \mathcal{X}$, and $\sigma_{\mathcal{X}} x = \sigma x$ for every $x \in \mathcal{X}$.

From the programmation point of view, a rule-based program with strategies enables to specify system calculi based on multi-sorted rewriting theories ; each theory is described by a program consisting of a signature, a set of rewrite rules for calculus and execution strategies for control.

Given a program E describing a theory, we denote $\mathcal{F}_E$ the set of symbols of the signature, where the subscript E may be omitted if there is no ambiguity. We denote $\mathcal{L}_E$ the set of its rules, assumed labelled to enable strategy constructors to use them later on. Several rewrite rules may have the same label ; hence a labelled rule is a set of conditional rewrite rules built over terms of $\mathcal{T}(\mathcal{F}_E,\mathcal{X})$. The syntax of a rewrite rule is

$$[label]\ l \to r \text{ if } c$$

where *label* is the label of the rule, $(l \to r \text{ if } c)$ is a conditional rewrite rule, where $l, r, c \in \mathcal{T}(\mathcal{F}_E,\mathcal{X})$ are respectively the left-hand side, the right-hand side and the boolean conditional part of the rule. By convention, we have $\mathcal{V}ar(r) \cup \mathcal{V}ar(c) \subseteq \mathcal{V}ar(l)$.

In the following, to fix ideas, we will consider the strategy language offered by the ELAN system. Strategies consist of terms built on $\mathcal{F}_E \cup \mathcal{S}_E$, with $\mathcal{S}_E = \mathcal{L}_E \cup \{id, fail, dk, dc, first, ";", repeat^*\}$. The *dk* operator is an abbreviation of *dont know choose*, and selects non-deterministically all strategies given as parameter. The *dc* operator, for *dont care choose*, chooses in a non-deterministic way one strategy that does not fail. The $first$ operator selects the first strategy that does not fail. It is similar to the classical structure *if-then-else-orelse* in other languages. The ";" operator expresses composition of strategies, and the $repeat^*$ operator composes a strategy with itself while it is not failing. Note that this is the case of recursion we consider in our study. Id is the identity strategy, and $fail$ a strategy that always fails. Given a strategy $S \in \mathcal{T}(\mathcal{F} \cup \mathcal{S},\mathcal{X})$, for $i \geq 0$, we inductively define S^i by $S^0 = id$ and $S^{i+1} = S^i; S$. The semantics of the strategy operators is given in Table 1, and borrowed from [2].

Strategies apply to terms to give sets of terms, empty if the application fails. Application of a strategy S to a ground term $t \in \mathcal{T}(\mathcal{F})$ is denoted $S(t)$, and computed with rules of Table 1. For readibility, we may also use the notation $[S](t)$. Labelled rewrite rules apply to a term at the top position ; however, a *congruence* strategy enables the application of a strategy deeper in a term. We denote $\mapsto\!\!\!\!\to$ the rewriting relation on strategy applications induced by rules of Table 1. Its reflexive transitive closure is denoted $\mapsto\!\!\!\!\to^*$. Normal forms of this rewriting are sets of terms. Due to the non-determinism of some strategy operators, we may have two sets

Table 1. The ELAN strategies : atoms and operators

Label/Operator	**Notation**	**Functional semantics / Definition**
Identity	id	$[id](t) \mapsto\!\!\!\!\rightarrow \{t\}$
		makes no transformation, but never fails.
Failure	$fail$	$[fail](t) \mapsto\!\!\!\!\rightarrow \emptyset$
		always fails.
Labelled rule	$[S]\begin{cases} u_1 \to v_1 \text{ if } c_1 \\ \vdots \\ u_n \to v_n \text{ if } c_n \end{cases}$	$[S](t) \mapsto\!\!\!\!\rightarrow \{\alpha v_i \mid \alpha u_i = t \wedge \alpha c_i \to^* true\}$
		gives all one-step reductions of t at the top position with the set of conditional rewrite rules of S
Composition	$S_1; S_2$	$[S_1; S_2](t) \mapsto\!\!\!\!\rightarrow \bigcup_{t' \in [S_1](t)} [S_2](t')$
		returns all results (maybe none) of S_2 applied to the results of S_1.
Congruence	$f(S_1, \ldots, S_n)$	$[f(S_1, \ldots, S_n)](g(t_1, \ldots, t_m)) \mapsto\!\!\!\!\rightarrow \begin{cases} \bigcup_{u_1 \in [S_1](t_1), \ldots, u_n \in [S_n](t_n)} f(u_1, \ldots, u_n) & \text{if } f = g \text{ and } \forall i : [S_i](t_i) \neq \emptyset \\ \emptyset & \text{if } f = g \text{ and } \exists i : [S_i](t_i) = \emptyset \\ \emptyset & \text{if } f \neq g \end{cases}$
		enables the application of a strategy deeper into a term.
Don't know	$dk(S_1, \ldots, S_n)$	$[dk(S_1, \ldots, S_n)](t) \mapsto\!\!\!\!\rightarrow \bigcup_{i=1}^{n} [S_i](t)$
		returns all results of $S_1, \ldots, S_n$. Whatever the term t, every strategy S_i is tried in all possible ways on t.
Don't care	$dc(S_1, \ldots, S_n)$	$[dc(S_1, \ldots, S_n)](t) \mapsto\!\!\!\!\rightarrow \begin{cases} \emptyset & \text{if } \bigcup_{i=1}^{n} [S_i](t) = \emptyset \\ [S_j](t) & \text{if } [S_j](t) \neq \emptyset \end{cases}$
		chooses non-deterministically a successful strategy among $S_1, \ldots, S_n$ and returns all its results.
First	$first(S_1, \ldots, S_n)$	$[first(S_1, \ldots, S_n)](t) \mapsto\!\!\!\!\rightarrow \begin{cases} \emptyset & \text{if } \bigcup_{i=1}^{n} [S_i](t) = \emptyset \\ [S_j](t) & \text{if } \bigcup_{i=1}^{j-1} [S_i](t) = \emptyset \text{ and } [S_j](t) \neq \emptyset \end{cases}$
		selects in a sequential way the first strategy that does not fail, and returns all its results. If S_i is selected, then $S_1, \ldots, S_{i-1}$ have failed.
Repeat	$repeat^*(S)$	$[repeat^*(S)](t) \mapsto\!\!\!\!\rightarrow [S^j](t) \text{ if } [S^{j+1}](t) = \emptyset \bigwedge_{i=0}^{j} [S^i](t) \neq \emptyset$
		iterates the strategy S until it fails and then returns the last result. Remark that if S fails, then $repeat^*(S)$ is equivalent to the id strategy.

of terms $E_1 \neq E_2$ such that $S(t) \mapsto\!\!\!\!\twoheadrightarrow^* E_1$ and $S(t) \mapsto\!\!\!\!\twoheadrightarrow^* E_2$. We write $S(t) \subseteq S'(t)$ iff for every set of terms E such that $S(t) \mapsto\!\!\!\!\twoheadrightarrow^* E$, there exists an evaluation of $S'(t)$ such that $S'(t) \mapsto\!\!\!\!\twoheadrightarrow^* E'$, with $E \subseteq E'$. We write $S(t) = S'(t)$ iff $S(t) \subseteq S'(t)$ and $S'(t) \subseteq S(t)$. We say that S and S' have the same semantics iff $S(t) = S'(t)$ for every ground term t, that $S(t)$ terminates (or is terminating) if every evaluation of $S(t)$ terminates and that S terminates (or is terminating) if $S(t)$ terminates for every ground term t. We say that two strategies S and S' have the same termination behaviour if for every ground term t, $S(t)$ terminates iff $S'(t)$ terminates, and that two strategies S and S' are equivalent iff they have both same semantics and termination behaviour.
Remark that two strategies S and S' may have the same semantics without having the same termination behaviour ; if we consider a terminating strategy S and a strategy S_∞ such that there exists no terminating evaluation of $[S_\infty](t)$ for every ground term t, then the strategies $dc(S, S_\infty)$ and S have the same semantics, but the first is not terminating, while the second is.
Remark also that given a ground term $t \in \mathcal{T}(\mathcal{F})$ and a strategy S, we can infer from the rules of Table 1 that if $[S](t) \mapsto\!\!\!\!\twoheadrightarrow^* \emptyset$, then every evaluation of $[S](t)$ fails. By abuse of notation, we may denote $[S](t) = \emptyset$ for $[S](t) \mapsto\!\!\!\!\twoheadrightarrow^* \emptyset$.

Table 2. Collection of TRSs used for termination

$$\mathcal{LR} = \begin{cases} Rules(id) & \to \{x \to x\} \\ Rules(fail) & \to \emptyset \\ Rules(\{\ldots\}) & \to \{\ldots\} \\ Rules(dk(S_1, \ldots, S_n)) & \to \bigcup_{i=1}^n Rules(S_i) \\ Rules(dc(S_1, \ldots, S_n)) & \to \bigcup_{i=1}^n Rules(S_i) \\ Rules(first(S_1, \ldots, S_n)) & \to \bigcup_{i=1}^n Rules(S_i) \\ Rules(f(S_1, \ldots, S_n)) & \to \bigcup_{i=1}^n Rules(S_i) \text{ if } f \in \mathcal{F} \\ Rules(S_1; S_2) & \to Rules(S_1) \cup Rules(S_2) \\ Rules(repeat^*(S)) & \to Rules(S) \cup \{x \to x\} \end{cases}$$

$$\mathcal{T}ermin = \begin{cases} TERMIN(S) & \to T(true, S) \\ T(b, id) & \to true \\ T(b, fail) & \to true \\ T(b, \{\ldots\}) & \to true \\ T(b, dk(S_1, \ldots, S_n)) & \to \bigwedge_{i=1}^n T(b, S_i) \\ T(b, dc(S_1, \ldots, S_n)) & \to \bigwedge_{i=1}^n T(b, S_i) \\ T(b, first(S_1, \ldots, S_n)) & \to T(b, S_1) \bigwedge_{i=2}^n T(false, S_i) \\ T(b, f(S_1, \ldots, S_n)) & \to \bigwedge_{i=1}^n T(b, S_i) \text{ if } f \in \mathcal{F} \\ T(b, S_1; S_2) & \to T(b, S_1) \wedge T(false, S_2) \\ T(false, repeat^*(S)) & \to \begin{cases} true \text{ if } Rules(S)\downarrow_{\mathcal{LR}} \text{ is terminating or} \\ \qquad \text{is } \varepsilon\text{-terminating and } S \text{ contains no congruence strategy} \\ \sharp \quad \text{otherwise} \end{cases} \\ T(true, repeat^*(\{\ldots\})) & \to \begin{cases} true & \text{if } \{\ldots\} \text{ is } \varepsilon\text{-terminating} \\ false & \text{if } \{\ldots\} \text{ is not } \varepsilon\text{-terminating} \\ \sharp & \text{if } \{\ldots\} \text{ cannot be proved } \varepsilon\text{-terminating} \end{cases} \\ T(true, repeat^*(S)) & \to \begin{cases} true & \text{if } S \neq \{\ldots\} \text{ and } Rules(S)\downarrow_{\mathcal{LR}} \text{ is terminating or} \\ & \qquad \text{is } \varepsilon\text{-terminating and } S \text{ contains no congruence strategy} \\ \sharp & \text{if } S \neq \{\ldots\} \text{ and the above condition is false} \end{cases} \end{cases}$$

3 Termination of strategies

Given a strategy, we propose a way of extracting a set of labelled rules whose termination ensures termination of the strategy. A first, naive approach for proving termination of a strategy S is

to prove termination of the set of all labelled rewrite rules involved in S. Indeed, given a set of labelled rewrite rules, strategy operators are used to constrain the rewriting, and hence restrict the possible applications of the rules. If termination of all applications is proved, then, in particular, so is termination of a strategy restricting these applications. By decomposing the strategy term and gathering the involved set of rules, the rewriting system $\mathcal{LR}$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{S} \cup \{Rules:1\}, \mathcal{X})$ given in Table 2 rewrites $Rules(S)$ into this set.
Remark the particular rewriting of $Rules(repeat^*(S))$, evaluating in $Rules(S)$ to which id is added. Indeed, id is hidden in the strategy $repeat^*(S)$, and applies if S fails. In particular, a strategy $repeat^*(repeat^*(S))$ cannot be terminating, even if S is.

Before using $\mathcal{LR}$ to prove termination of strategies, let us remark that since labelled rewrite rules only apply at the top position, we can restrict their termination study to the particular case of ε-rewriting, that is rewriting at the top position.

Definition 1. *Given a CRS $\mathcal{R}$ built on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and two terms $t, t' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, we say that t ε-rewrites into t' iff $t \longrightarrow^{\epsilon}_{\mathcal{R}} t'$. We say that $\mathcal{R}$ is ε-terminating iff there exists no infinite ε-rewriting chain in $\mathcal{R}$.*

We can now state a first sufficient condition for termination of strategies. The complete proofs of the results presented in this paper can be found in the appendix.

Proposition 1. *Given a strategy S, the reduction of the term $Rules(S)$ with $\mathcal{LR}$ terminates and results in a set of rules such that : if $Rules(S)\downarrow_{\mathcal{LR}}$ is ε-terminating, then S terminates.*

Classical methods for proving termination of CRSs [6, 14] rely on simplification orderings and then show general termination, hence in particular ε-termination. In Section 4.1, we propose a sufficient condition for proving ϵ-termination of a set of labelled rewrite rules when classical methods fail, in some cases.

Proposition 1 can be refined, since it considers all rules of a strategy, while only some of these rules may cause non termination. For instance, let us consider the strategy $S = \{f(a) \to g(a)\}; repeat^*(\{g(a) \to f(a)\})$. Obviously, S is terminating : for any ground term t, the rule $f(a) \to g(a)$ fails or applies and, if it applies, the rule $g(a) \to f(a)$ can apply only once. However, we have $Rules(S)\downarrow_{\mathcal{LR}} = \{f(a) \to g(a), g(a) \to f(a)\}$, which is obviously not ε-terminating.

The intuition suggests that non termination can be caused only by recursion. We then use Proposition 1 for strategies encapsulated by a $repeat^*$ operator. Moreover, if the encapsulated strategy is a set of rules S, then we have *equivalence* between ε-termination of S and termination of $repeat^*(S)$, and in this particular case, non termination of a strategy may be shown. If S is not a set of rules, then ε-termination of the set of rules $Rules(S)\downarrow_{\mathcal{LR}}$ is just a sufficient condition for termination of S.
For strategies that are not headed by a $repeat^*$ operator, we infer from the semantics given in Table 1 and the definition of a terminating strategy the following conditions for termination :

- if $S = id, fail$ or $\{\ldots\}$, then its application consists of a finite number of one-step rewritings, and hence terminates ;
- if $S = f(S_1, \ldots, S_n)$, for $f \in \mathcal{F}$, then S terminates iff $S_1, \ldots, S_n$ are terminating ;
- if $S = dk(S_1, \ldots, S_n)$, then S terminates iff $S_1, \ldots, S_n$ are terminating ;
- if $S = dc(S_1, \ldots, S_n)$, application of S may evaluate into application of S_i, for some $i \in \{1, \ldots, n\}$; therefore S terminates iff $S_1, \ldots, S_n$ are terminating ;
- if $S = first(S_1, \ldots, S_n)$, application of S may never evaluate into application of $S_j, \ldots, S_n$, for $j > 1$; this is why termination of $S_1, \ldots, S_n$ is just a sufficient condition for S to terminate. However, if S_1 does not terminate, then neither does S ;
- if $S = S_1; S_2$, a sufficient condition for S to terminate is that S_1 and S_2 are terminating. However, like for the previous point, if S_1 does not terminate, then neither does S.

The above conditions for termination induce a transformation that preserves termination and can even sometimes ensure non termination of a strategy. This transformation is defined by a

TRS $\mathcal{T}ermin$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{S} \cup \{TERMIN:1, T:2, true:0, false:0, \sharp:0\}, \mathcal{X})$ such that given a strategy S, S terminates if the term $TERMIN(S)$ built on S with the new symbol $TERMIN$ rewrites into $true$ and does not terminate if $TERMIN(S)$ rewrites into $false$. Termination of S is undetermined if $TERMIN(S)$ rewrites into $\sharp$.
Reduction of the term $TERMIN(S)$ is achieved in a classical way through reduction of the term $T(b, S)$ where b is a boolean value, initially $true$, indicating whether the reduction will give a necessary and sufficient condition or just a sufficient condition. The value $\sharp$ extends the boolean values $true$ and $false$ by $\sharp \wedge \sharp = \sharp$, $\sharp \wedge true = \sharp$, $\sharp \wedge false = false$. See Table 2 for $\mathcal{T}ermin$, whose correctness is ensured by the following theorem.

Theorem 1. *Let S be a strategy. Then the reduction of $TERMIN(S)$ with $\mathcal{T}ermin$ terminates and its normal form is either $true$, $false$ or $\sharp$, and :*

- *if $TERMIN(S)\downarrow_{\mathcal{T}ermin} = true$, then S terminates ;*
- *if $TERMIN(S)\downarrow_{\mathcal{T}ermin} = false$, then S does not terminate.*

A corollary of Theorem 1 is that any strategy containing no $repeat^*$ operator is terminating.

Example 1. Let us come back to the strategy S motivating the refinement of Proposition 1 :

$$\{f(a) \to g(a)\}; repeat^*(\{g(a) \to f(a)\}).$$

With $\mathcal{T}ermin$, we have the reduction :
$TERMIN(S)$
$\longrightarrow T(true, \{f(a) \to g(a)\}; repeat^*(\{g(a) \to f(a)\}))$
$\longrightarrow T(true, \{f(a) \to g(a)\})$
$\quad \wedge\, T(false, repeat^*(\{g(a) \to f(a)\}))$
$\longrightarrow true \wedge T(false, repeat^*(\{g(a) \to f(a)\}))$
$\longrightarrow true \wedge true$
$\longrightarrow true.$
Consequently, by Theorem 1, S is terminating.

Our termination criterion is still too coarse, for we have not taken into account the way strategy operators make their arguments interact. For instance, let us consider the strategy S :

$$repeat^*(first(\{f(a) \to g(b)\}, \{f(x) \to g(x), g(a) \to f(a)\})).$$

Theorem 1 proves that S terminates if $\{f(a) \to g(b), f(x) \to g(x), g(a) \to f(a)\}$ is ε-terminating, which is not the case, because of the infinite ε-derivation

$$f(a) \longrightarrow g(a) \longrightarrow f(a) \longrightarrow \ldots$$

Yet, the semantics of the $first$ operator ensures that the rule $f(x) \to g(x)$ cannot apply with the instance $x = a$, because of the first argument of the $first$ strategy. In fact, S terminates.
We then need to exploit the semantics of the strategy operators for proving termination of strategies in a finer way than previously. An appropriate way consists in simplifying, in some sense defined later on, a strategy into another one whose termination is easier to prove and ensures termination of the original strategy.

4 Simplification of strategies

We then propose to exploit the particular definition of strategy operators to define a reduction mechanism on strategies. We come up with rewrite rules transforming strategy terms into equivalent ones, i.e. having both same semantics and termination behaviour. To distinguish rewriting of strategies from rewriting applied to data t of programs, we will denote the reduction

relation on strategies by $\hookrightarrow$. Defining the "size" of a strategy by its number of labelled rewrite rules and strategy operators, the main idea is to make this size decrease when simplifying a strategy S into a strategy S'. We propose a simplification process empowering the termination criterion described in Section 3, by removing labelled rules and strategy operators.

Starting from very simple observations, we can easily, in a first step, introduce simplification rules dealing with properties of the failure and success of a strategy, i.e. of the particular operators $fail$ and id. They are rules (1–9) of the system $\mathcal{SIMPL}$ given in Table 5 at the end of Section 4, where all simplification rules of the paper are gathered.
Remark that one would also expect the simplification rule $S; fail \rightarrow fail$ in the list of rules above, but this rules does not preserve the termination behaviour of the left-hand side strategy. Indeed, if S is not terminating then neither is $S; fail$, while the strategy $fail$ always terminates. The rules (6) and (7) express that the $fail$ strategy is never selected in the list of arguments. On the opposite, the rules (8) and (9) express that if a strategy S cannot fail, then the strategies occuring after S in the list of arguments of a $first$ strategy will never be selected.

Now, rules expressing the distributivity of the composition can also simplify strategy expressions. See rules (10–12) in Table 5. Note however that though the ";" operator is distributive over congruence and dk strategies, it is not over dc and $first$ strategies.
To illustrate the non left-distributivity over the dc strategies, let us consider the application of the strategy $\mathbf{dc}(\{\mathbf{a} \rightarrow \mathbf{b}\}, \{\mathbf{a} \rightarrow \mathbf{c}\}); \{c \rightarrow d\}$ to the term a (for conveniency, we sometimes use rewrite rules instead of their label). It results either in the empty set or in $\{d\}$, while the application $[\mathbf{dc}(\{\mathbf{a} \rightarrow \mathbf{b}\}; \{c \rightarrow d\}, \{\mathbf{a} \rightarrow \mathbf{c}\}; \{c \rightarrow d\})](a)$ always results in $\{d\}$, and hence never fails. The right distributivity is not correct either, such as illustrated by $[dk(\{a \rightarrow b\}, \{a \rightarrow c\}); \mathbf{dc}(\{\mathbf{b} \rightarrow \mathbf{d}\}, \{\mathbf{c} \rightarrow \mathbf{e}\})](a) \mapsto\!\!\!\!\gg^* \{d, e\}$, while $[\mathbf{dc}(dk(\{a \rightarrow b\}, \{a \rightarrow c\}); \{\mathbf{b} \rightarrow \mathbf{d}\}, dk(\{a \rightarrow b\}, \{a \rightarrow c\}); \{\mathbf{c} \rightarrow \mathbf{e}\})](a) \mapsto\!\!\!\!\gg^* \{d\}$ or $\{e\}$. Similar counter-examples can be found to prove the non-distributivity of the composition over the $first$ strategies.

When encapsulation of arguments by a strategy combinator is superfluous, we propose a way of flattening strategies with rules (13–20). For the rule (20), we assume that $f \in \mathcal{F}$ and $\mathcal{V}ar(l_i) \cap \mathcal{V}ar(l_j) = \emptyset, \forall i \neq j$.

Finally, we deal with redundancy of arguments in $first$ and dc strategies by removing arguments that are syntactically equal to other ones. See rules (21),(22).
Let us now tackle more complex simplifications.

4.1 Composition of labelled rewrite rules

We now focus on the simplification of the composition of sets of labelled rewrite rules. As said above, the main purpose of the simplification is to reduce the number of rewrite rules occuring in a strategy, which has a direct impact on its termination study with our termination criterion.

We recall that labelled rewrite rules apply only at the top position ; given a ground term t, the application of the composition of two labelled rewrite rules $l_1 \rightarrow r_1$ if $c_1; l_2 \rightarrow r_2$ if c_2 to t can then either fail or result in a singleton. Putting the conditional parts aside, the previous composition succeeds if there exist two ground substitutions α_1, α_2 such that $t = \alpha_1 l_1$ and $\alpha_1 r_1 = \alpha_2 l_2$. Assuming there is no variable shared by both rules, the domains of α_1 and α_2 are disjoint, and the last condition is equivalent to the existence of a unifier of r_1 and l_2. The following Proposition formalizes this result and extends it to the conditional case.

Proposition 2. *Let $l_1 \rightarrow r_1$ if $c_1, l_2 \rightarrow r_2$ if c_2 be two labelled rewrite rules, and let us assume that any rewriting step only occurs at the top position. Then, for any ground term $t \in \mathcal{T}(\mathcal{F})$, we have :*

- $[l_1 \rightarrow r_1$ *if* $c_1; l_2 \rightarrow r_2$ *if* $c_2](t) = \emptyset$ *if r_1 and l_2 are not unifiable.*
- $[l_1 \rightarrow r_1$ *if* $c_1; l_2 \rightarrow r_2$ *if* $c_2](t) = [\mu l_1 \rightarrow \mu r_2$ *if* $\mu c_1 \wedge \mu c_2](t)$ *if μ is a most general unifier of r_1 and l_2.*

We now infer from Proposition 2 the two-rule TRS $\mathcal{C}omp$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{S} \cup \{comp{:}2\}, \mathcal{X})$ given in Table 3 such that given two labelled rewrite rules $(l_1 \to r_1 \text{ if } c_1), (l_2 \to r_2 \text{ if } c_2)$, the term $comp(l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2)$ rewrites into a set of rules equivalent to the composition of the two rules. This set of rules is either the empty set, or a singleton. Rule (23), in Table 5, simplifies the composition of two sets of labelled rewrite rules $\{\ldots\}_1$ and $\{\ldots\}_2$ into a new (possibly empty) set of rewrite rules, by using $\mathcal{C}omp$. Let us give an example illustrating how efficient this simplification may be.

Example 2. Let us consider the strategy

$$S = \{f(x) \to g(x), f(x) \to h(0,x)\};\\ \{h(1,x) \to f(1), h(x,y) \to y, f(1) \to h(1,1)\}$$

By using the simplification rule (23), S rewrites in the union of the following six sets of rules :
$comp(f(x_1) \to g(x_1), h(1,x_2) \to f(1))\downarrow_{\mathcal{C}omp}$
$comp(f(x_1) \to g(x_1), h(x_2,y_2) \to y_2)\downarrow_{\mathcal{C}omp}$
$comp(f(x_1) \to g(x_1), f(1) \to h(1,1))\downarrow_{\mathcal{C}omp}$
$comp(f(x_1) \to h(0,x_1), h(1,x_2) \to f(1))\downarrow_{\mathcal{C}omp}$
$comp(f(x_1) \to h(0,x_1), h(x_2,y_2) \to y_2)\downarrow_{\mathcal{C}omp}$
$comp(f(x_1) \to h(0,x_1), f(1) \to h(1,1))\downarrow_{\mathcal{C}omp}$
The fifth set is reduced into $\{f(x) \to x\}$ by using the second rule of $\mathcal{C}omp$ with the unifying substitution $\mu = (x_2 = 0 \wedge x_1 = x \wedge y_2 = x)$. The other sets are reduced into $\emptyset$ by using the first rule of $\mathcal{C}omp$.
Then S simplifies into $S' = \{f(x) \to x\}$.

Given a set $\{\ldots\}$ of labelled rewrite rules, rule (23) enables to simplify $\{\ldots\}^i = \{\ldots\};\ldots;\{\ldots\}$, for $i > 0$, into a (possibly empty) set of rules. As mentioned in Section 3, we can induce from this simplification a sufficient condition for ε-termination.

Proposition 3. *Given a CRS $\mathcal{R}$ consisting of a set of rules $\{\ldots\}$ built on $\mathcal{T}(\mathcal{F},\mathcal{X})$, $\mathcal{R}$ is ε-terminating if $\exists i > 0 : \{\ldots\}^i \hookrightarrow^*_{\mathcal{SIMPL}} \emptyset$.*

Proposition 3 is of high interest in practice, since it allows to prove specific ε-termination of a non terminating (in the classical sense) set of rules, which often arises for programs with labelled rules.

4.2 Constraining rewrite rules

The idea now is to suppress redundancy in the application of the rewrite rules occuring in a strategy by constraining them with extra conditions. On the one hand, the conditional part of a rewrite rule may be so constrained that it can be shown to be unsatisfiable. In this case, the rewrite rule is equivalent to $fail$, and the strategy can be simplified by the rules given at the beginning of Section 4. On the other hand, constraining rewrite rules restricts their application domain, and hence may improve their termination behaviour. For instance, the rewriting system $\{f(x) \to g(x), g(a) \to f(a)\}$ is not terminating, but terminates if the first rule is constrained by the condition $x \neq a$.

We propose here a way of constraining rewrite rules occuring in a strategy by using the assumption that other strategies fail. This is of particular interest for strategies like $first(S_1,\ldots,S_n)$ applying to a ground term t, which evaluates into $[S_i](t)$ only if $[S_1](t),\ldots,[S_{i-1}](t)$ fail.
As labelled rewrite rules apply to a term at the top position, the main idea is the following : given a rewrite rule $l_1 \to r_1$ and a ground term t such that $[l_1 \to r_1](t) = \emptyset$, we also have $[l_2 \to r_2](t) = \emptyset$ for any rewrite rule $l_2 \to r_2$ such that l_2 is an instance of l_1. On the opposite, if l_1 is an instance σ of l_2 ($l_1 = \sigma l_2$), then we can exclude this instance when applying $l_2 \to r_2$ to t, that is $[l_2 \to r_2](t) = [l_2 \to r_2 \text{ if } \overline{\sigma}](t)$. To ensure that the variables of the condition occur in l_2, we must restrict to the case where σ is ground. Proposition 4 generalizes this idea to the case where l_1 and l_2 are unifiable, and extends it to the conditional case.

Proposition 4. *Let $l_1 \to r_1$ if c_1 and $l_2 \to r_2$ if c_2 two conditional rewrite rules, $t \in \mathcal{T}(\mathcal{F})$ such that $l_1 \to r_1$ if c_1 fails on t. If there exists a unifying substitution μ such that $\mu l_1 = \mu l_2$ and $Ran(\mu) \subseteq \mathcal{V}ar(l_2)$, then we have $[l_2 \to r_2 \text{ if } c_2](t) = [l_2 \to r_2 \text{ if } c_2 \wedge not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)})](t)$.*

The conditional part $not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)})$ excludes instances of l_2 equal to l_1 and satisfying c_1. For consistency, we denote the empty substitution by the boolean value *true* ; hence, if $Dom(\mu) \cap \mathcal{V}ar(l_2) = \emptyset$, the boolean expression $not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)})$ is equivalent to $not(\mu c_1)$. Note that the condition $Ran(\mu) \subseteq \mathcal{V}ar(l_2)$, together with the usual assumption $Dom(\mu) \cap Ran(\mu) = \emptyset$, ensures that for every variable $x \in Dom(\mu) \cap \mathcal{V}ar(l_2)$, μx is a ground term. Thus the set of variables of the new condition is still included in the set of variables of l_2, such as required by the definition of the conditional rewrite rules.

Table 3. Collection of TRSs used for simplification : composing and constraining rewrite rules

$$\mathcal{C}omp = \begin{cases} comp(l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2) \hookrightarrow \emptyset & \text{if } r_1 \text{ and } l_2 \text{ are not unifiable} \\ comp(l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2) \hookrightarrow \{\mu l_1 \to \mu r_2 \text{ if } \mu c_1 \wedge \mu c_2\} & \text{if } \mu = mgu(r_1, l_2) \end{cases}$$

$$\mathcal{FR} = \begin{cases} failureRules(\{\ldots\}) & \hookrightarrow \{\ldots\} \\ failureRules(id) & \hookrightarrow \emptyset \\ failureRules(fail) & \hookrightarrow \emptyset \\ failureRules(dk(S_1, \ldots, S_n)) & \hookrightarrow failureRules(S_1) \cup \ldots \cup failureRules(S_n) \\ failureRules(dc(S_1, \ldots, S_n)) & \hookrightarrow failureRules(S_1) \cup \ldots \cup failureRules(S_n) \\ failureRules(first(S_1, \ldots, S_n)) & \hookrightarrow failureRules(S_1) \cup \ldots \cup failureRules(S_n) \\ failureRules(S_1; S_2) & \hookrightarrow \emptyset \\ failureRules(repeat^*(S_1)) & \hookrightarrow \emptyset \\ failureRules(f(S_1, \ldots, S_n)) & \hookrightarrow f(failureRules(S_1), \ldots, failureRules(S_n)) \text{ if } f \in \mathcal{F} \\ f(\{\ldots\}_1, \ldots, \{\ldots\}_n) & \hookrightarrow \bigcup_{(l_i \to r_i \text{ if } c_i) \in \{\ldots\}_i} \{f(l_1, \ldots, l_n) \to f(r_1, \ldots, r_n) \text{ if } \bigwedge_{i=1}^n c_i\} \end{cases}$$

$$\mathcal{CO} = \begin{cases} constrain(\{\ldots\}, l \to r \text{ if } c) & \hookrightarrow \{l \to r \text{ if } c \bigwedge_{(\mu_i, c_i) \in E} not(\mu_i c_i \wedge \mu_i \mathcal{V}ar(l))\} \\ constrain(\{\ldots\}, id) & \hookrightarrow id \\ constrain(\{\ldots\}, fail) & \hookrightarrow fail \\ constrain(\{\ldots\}_1, \{\ldots\}_2) & \hookrightarrow \bigcup_{(l \to r \text{ if } c) \in \{\ldots\}_2} constrain(\{\ldots\}_1, l \to r \text{ if } c) \\ constrain(\{\ldots\}, dk(S_1, \ldots, S_n)) & \hookrightarrow dk(constrain(\{\ldots\}, S_1), \ldots, constrain(\{\ldots\}, S_n)) \\ constrain(\{\ldots\}, dc(S_1, \ldots, S_n)) & \hookrightarrow dc(constrain(\{\ldots\}, S_1), \ldots, constrain(\{\ldots\}, S_n)) \\ constrain(\{\ldots\}, first(S_1, \ldots, S_n)) & \hookrightarrow first(constrain(\{\ldots\}, S_1), \ldots, constrain(\{\ldots\}, S_n)) \\ constrain(\{\ldots\}, S_1; S_2) & \hookrightarrow constrain(\{\ldots\}, S_1); S_2 \\ constrain(\{\ldots\}, repeat^*(S)) & \hookrightarrow \begin{cases} id & \text{if } constrain(\{\ldots\}, S) \longrightarrow^*_{\mathcal{CO}} fail \\ repeat^*(S) & \text{otherwise} \end{cases} \\ constrain(\{\ldots\}, f(S_1, \ldots, S_n)) & \hookrightarrow f(S_1, \ldots, S_n) \text{ if } f \in \mathcal{F} \\ constrain(S_1, S_2) & \hookrightarrow constrain(failureRules(S_1)\downarrow_{\mathcal{FR}}, S_2) \text{ if } S_1 \text{ is not a set of rules} \end{cases}$$

with $E = \{(\mu_i, c_i) | \exists (l_i \to r_i \text{ if } c_i) \in \{\ldots\}$ such that $\mu_i l = \mu_i l_i$ and $Ran(\mu_i) \subseteq \mathcal{V}ar(l)\}$.

We illustrate on an example how to use Proposition 4 to constrain a rewrite rule.

Example 3. Let us assume that the comparison between integers reduces into *true* or *false*. Given $rule_1 = (g(x_1, 1) \to h(x_1)$ if $x_1 > 1)$, a ground term $t \in \mathcal{T}(\mathcal{F})$ such that $[rule_1](t) = \emptyset$, let us constrain the rewrite rule $rule_2 = (g(x_3, x_4) \to f(x_3)$ if $x_3 > 2)$ in the application $[rule_2](t)$.

The intuition suggests that $rule_2$ can be constrained by adding the condition $x_4 \neq 1$. In this way, we exclude possible instances of the left-hand side of the failing rule $rule_1$. Let us apply the constraining suggested by Proposition 4 to $rule_2$, by taking $\mu = (x_1 = x_3 \wedge x_4 = 1)$. With such a substitution, the conditions $\mu(g(x_1, 1)) = \mu(g(x_3, x_4))$ and $Ran(\mu) \subseteq \mathcal{V}ar(g(x_3, x_4))$ are fulfilled.

Proposition 4 then states that, since $[rule_1](t) = \emptyset$, $[rule_2](t)$ is equivalent to the application $[g(x_3, x_4) \to f(x_3)$ if $x_3 > 2 \wedge not(x_3 > 1 \wedge x_4 = 1)](t)$. Assuming that the instances x_3 and x_4 normalizes into integers, the condition is equivalent to $x_3 > 2 \wedge (x_3 \leq 1 \vee x_4 \neq 1)$, that is to $(x_3 > 2 \wedge x_4 \neq 1)$, such as suggested by the intuition.

We then define a TRS $\mathcal{CO}$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{S} \cup \{constrain\!:\!2\}, \mathcal{X})$ such that given two strategies S_1 and S_2, the term $constrain(S_1, S_2)$ rewrites into a constrained form S_2' of S_2 such that $[S_2'](t) = [S_2](t)$ for every ground term $t \in \mathcal{T}(\mathcal{F})$ on which S_1 fails. Thanks to an easy extension of the failing rule in Proposition 4 to a set of failing rules, we get the first rewrite rule of $\mathcal{CO}$ (the complete TRS is given in Table 3).

Let us now extend the constraint mechanism for a rewrite rule, expressed in the first rule of $\mathcal{CO}$ introduced above, to a constraint mechanism for any strategy S knowing the failure of a set of rules $\{\ldots\}$, using the following facts :

- If $S = id$ or $fail$, then there is no rule to constrain.
- If S is a set of rewrite rules, then we can try to constrain each rule of S.
- If $S = op(S_1, \ldots, S_n)$, with $op \in \{dk, dc, first\}$, then we look for the rules to constrain in each strategy $S_1, \ldots, S_n$.
- If $S = S_1; S_2$, then we look for the rules to constrain in S_1 only. We cannot deduce anything on S_2, since S_2 applies to the results of S_1, and not to the term on which the set of rules $\{\ldots\}$ fails.
- If $S = repeat^*(S_1)$, for the same reason as the composition above, we cannot constrain S_1, which is repeatedly applied. However, S_1 is not repeatedly applied if its first application fails. We can then test whether its constraint form is $fail$; if it is the case, then S can be simplified in id.
- If $S = f(S_1, \ldots, S_n)$, with $f \in \mathcal{F}$, the strategies $S_1, \ldots, S_n$ apply to the subterms of the term on which the set of rules $\{\ldots\}$ fails, and then we cannot deduce any constraint mechanism.

Hence the 9 following rules in Table 3.
We then have a way to constrain a strategy according to the failure of a set of rules. We finally extend this failure case to the failure of any strategy. To do so, we define a TRS $\mathcal{FR}$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{S} \cup \{constrain\} \cup \{failureRules\!:\!1\}, \mathcal{X})$, given in Table 3, such that $failureRules(S)$ rewrites into a set of rewrite rules that fail on every term t on which S fails. The rule $failureRules(repeat^*(S_1)) \hookrightarrow \emptyset$ of $\mathcal{FR}$ is implied by the fact that a $repeat^*$ strategy never fails. The last rule consists of simplifying a congruence strategy when all arguments are sets of rules. The following Lemma ensures the correction of the transformation induced by $\mathcal{FR}$.

Lemma 1. *Given a strategy S, we have :*

1. *the reduction of the term $failureRules(S)$ with $\mathcal{FR}$ terminates and its normal form is a (possibly empty) set of rewrite rules ;*
2. *every rule of $failureRules(S)\downarrow_{\mathcal{FR}}$ fails on any ground term $t \in \mathcal{T}(\mathcal{F})$ such that $[S](t) = \emptyset$.*

Thanks to $\mathcal{FR}$, we can link the constraint mechanism of a strategy according to the failure of another strategy to the previous constraint mechanism of a strategy according to the failure of a set of rules, hence the last rule of $\mathcal{CO}$.

Lemma 2. *For any strategies S_1 and S_2, the reduction of the term $constrain(S_1, S_2)$ with $\mathcal{CO}$ terminates, and its normal form S_2' is a strategy such that for any ground term $t \in \mathcal{T}(\mathcal{F})$ on which S_1 fails :*

(correctness) $[S_2'](t) \subseteq [S_2](t)$.
(completeness) $[S_2](t) \subseteq [S_2'](t)$.
(termination) $[S_2](t)$ *terminates iff* $[S_2'](t)$ *terminates.*

Let us now study how to exploit $\mathcal{CO}$ for simplifying strategies. As said before, constraining a strategy with respect to the failure of another one is of particular interest for $first$ strategies. We recall that the $first$ operator behaves like the classical *if-then-else-orelse* in other languages. So, given a ground term t, the application $[first(S_1, \ldots, S_n)](t)$ evaluates into $[S_i](t)$ only if $S_1, \ldots, S_{i-1}$ fail on t. According to the semantics of the dk operator, the latter condition is equivalent to the failure of $[dk(S_1, \ldots, S_{i-1})](t)$. The constrained form of each S_i is then given by the normal form with $\mathcal{CO}$ of the term $constrain(dk(S_1, \ldots, S_{i-1}), S_i)$. The $first$ strategy may then be simplified by the rule (24) of Table 5. With this rule, we can get an infinite reduction of a $first$ strategy :
$S = first(S_1, S_2, \ldots, S_n)$
$\hookrightarrow^{(24)} S' = first(S_1, S'_2, \ldots, S'_n)$
$\hookrightarrow^{(24)} S'' = first(S_1, S''_2, \ldots, S''_n)$
$\hookrightarrow^{(24)} \ldots$
But the rule has no effect anymore from the term S'. This is why we require this rule to be applied only once on the same strategy term.

Remark that we can also use $\mathcal{CO}$ to simplify the particular case of the composition of a $repeat^*$ strategy with another strategy (see rule (25) in Table 5). Indeed, the strategy $repeat^*(S_1)$ iterates the application of S_1 until S_1 fails. Henceforth, in the particular case of $repeat^*(S_1); S_2$, the strategy S_2 applies to a set of terms on which S_1 has just failed, and can then be constrained. For the same reason as above, rule (25) is required to be applied only once on the same strategy term.

4.3 Exclusive strategies

The constraint-based simplification studied in Section 4.2 may make strategy arguments of the $first$ operator mutually exclusive. Intuitively, two strategies S and S' are mutually exclusive if there exists no ground term on which both S and S' apply.

Definition 2. *Let S, S' be two strategies. We say that S excludes S' iff $\forall t \in \mathcal{T}(\mathcal{F}) : (\exists E \neq \emptyset : [S](t) \twoheadrightarrow^* E) \Rightarrow ([S'](t)$ terminates and evaluates into $\emptyset)$. We say that S and S' are mutually exclusive iff S excludes S' and S' excludes S.*

To illustrate how important exclusive strategies are for simplification, let us consider the strategy $S = first(rule_1, rule_2)$, where $s_1 = g(x_1, 1) \rightarrow h(x_1)$ if $x_1 > 1$ and $s_2 = g(x_3, x_4) \rightarrow f(x_3)$ if $x_3 > 2$. Both s_1 and s_2 apply on ground terms $g(n, 1)$, with $n > 2$. We showed in Example 3 that S can be simplified into $S' \equiv first(rule_1, rule'_2)$, with $rule'_2 = g(x_3, x_4) \rightarrow f(x_3)$ if $x_3 > 2 \wedge x_4 \neq 1$. Now $rule_1$ and $rule'_2$ are mutually exclusive. As a consequence, we can show that the strategy $first(rule_1, rule'_2)$ is equivalent to $dk(rule_1, rule'_2)$, which simplifies into $\{rule_1\} \cup \{rule'_2\}$, that is a set of rewrite rules.
Proposition 5 generalizes the use of mutually exclusive strategies sketched above to transform $first$ and dc strategies into dk strategies, whose simplification is in general easier, and particularly interesting if arguments of the strategy are sets of rules.

Proposition 5. *Let $S_1, \ldots, S_n$ be strategies such that $\forall i, j \in \{1, \ldots, n\}, i \neq j : S_i$ and S_j are mutually exclusive. Then we have :*

1. *$dc(S_1, \ldots, S_n)$ is equivalent to $dk(S_1, \ldots, S_n)$.*
2. *$first(S_1, \ldots, S_n)$ is equivalent to $dk(S_1, \ldots, S_n)$.*

Using Proposition 5 to effectively transform dc or $first$ strategies into dk strategies requires a way of proving that two strategies S and S' are mutually exclusive. We then define a TRS $\mathcal{E}xcl$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{S} \cup \{constrain, failureRules\} \cup \{mutex{:}2, true{:}0, \sharp{:}0\}, \mathcal{X})$ such that $mutex(S, S')$ rewrites with $\mathcal{E}xcl$ into $true$ if the strategies S and S' are mutually exclusive, and into $\sharp$ if we do not deduce anything on mutual exclusion of S and S'. The definition of $\mathcal{E}xcl$ relies on the following sufficient conditions :

Table 4. Collection of TRSs used for simplification : exclusive strategies

$$
\mathcal{E}xcl = \left\{
\begin{array}{ll}
mutex(l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2) & \hookrightarrow \left\{ \begin{array}{ll} \sharp & \text{if } \exists \mu = mgu(l_1, l_2) : \mu \text{ satisfies } c_1 \wedge c_2 \\ true & \text{otherwise} \end{array} \right. \\
mutex(id, S') & \hookrightarrow \left\{ \begin{array}{ll} true & \text{if } S' = fail \\ \sharp & \text{otherwise} \end{array} \right. \\
mutex(fail, S') & \hookrightarrow true \\
mutex(\{\ldots\}_1, \{\ldots\}_2) & \hookrightarrow \bigwedge_{(l_i \to r_i \text{ if } c_i) \in \{\ldots\}_i} mutex(l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2) \\
mutex(\{\ldots\}_1, S') & \hookrightarrow mutex(S', \{\ldots\}_1) \text{ if } S' \text{ is not a set of rules} \\
mutex(dk(S_1, \ldots, S_n), S') & \hookrightarrow \bigwedge_{i=1}^{n} mutex(S_i, S') \\
mutex(dc(S_1, \ldots, S_n), S') & \hookrightarrow \bigwedge_{i=1}^{n} mutex(S_i, S') \\
mutex(first(S_1, \ldots, S_n), S') & \hookrightarrow \bigwedge_{i=1}^{n} mutex(S_i, S') \\
mutex(f(S_1, \ldots, S_n), \{\ldots\}_2) & \hookrightarrow \left\{ \begin{array}{ll} true & \text{if } \forall (l_2 \to r_2 \text{ if } c_2) \in \{\ldots\}_2 : top(l_2) \neq f \\ \sharp & \text{otherwise} \end{array} \right. \\
mutex(f(S_1, \ldots, S_n), g(S'_1, \ldots, S'_m)) & \hookrightarrow \left\{ \begin{array}{ll} true & \text{if } f \neq g \\ \bigvee_{i=1}^{n} mutex(S_i, S'_i) & \text{if } f = g \end{array} \right. \\
mutex(f(S_1, \ldots, S_n), S') & \hookrightarrow \sharp \text{ if } S' \text{ is neither a set of rules nor a congruence strategy} \\
mutex(S_1; S_2, S') & \hookrightarrow mutex(S_1, S') \\
mutex(repeat^*(S), S') & \hookrightarrow \left\{ \begin{array}{ll} true & \text{if } S' = fail \\ \sharp & \text{otherwise} \end{array} \right.
\end{array}
\right.
$$

- if the left-hand sides of two rewrite rules are not unifiable, then these rules are mutually exclusive ;
- if the left-hand sides of two rewrite rules are unifiable and their mgu makes one of the conditional parts of the rules unsatisfiable, then these rules are also mutually exclusive ;
- since id never fails, id and a strategy S' are mutually exclusive iff $S' = fail$;
- two sets of rules are mutually exclusive if each rule of the first set is exclusive with each rule of the second ;
- a strategy $op(S_1, \ldots, S_n)$, with $op \in \{dk, dc, first\}$, and a strategy S' are mutually exclusive if for each S_i, $i \in \{1, \ldots, n\}$, S_i and S' are mutually exclusive ;
- a congruence strategy $f(S_1, \ldots, S_n)$ and a strategy S' are mutually exclusive if :
  - S' is a set of rules and no top symbol of left-hand side of rule of S' is equal to f, or
  - S' is another congruence strategy $g(S'_1, \ldots, S'_m)$ with $f \neq g$, or
  - S' is another congruence strategy $f(S'_1, \ldots, S'_n)$ and each S_i, S'_i, for $i \in \{1, \ldots, n\}$, are mutually exclusive.

  We do not deduce any mutual exclusion if S' is neither a set of rules nor a congruence ;
- if S_1 and S_2 are mutually exclusive, then so are $S_1; S'_1$ and $S_2; S'_2$, for any strategies S'_1, S'_2. Indeed, let us consider a ground term t such that $[S_1; S'_1](t) \neq \emptyset$. In particular, we have $[S_1](t) \neq \emptyset$, and then, if S_1 and S_2 are mutually exclusive, we get $[S_2](t) = \emptyset$. Consequently, we have $[S_2; S'_2](t) = \emptyset$. By a symmetrical reasoning, we get that $S_1; S'_1$ and $S_2; S'_2$ are mutually exclusive ;
- since a $repeat^*$ strategy never fails, $repeat^*(S)$ and a strategy S' are mutually exclusive iff $S' = fail$.

The TRS $\mathcal{E}xcl$ is given in Table 4.

Lemma 3. *Let S, S' be two strategies. We have :*

1. *The reduction of $mutex(S, S')$ with $\mathcal{E}xcl$ terminates and its normal form is either $true$ or $\sharp$.*
2. *If $mutex(S, S') \hookrightarrow^*_{\mathcal{E}xcl} true$, then S, S' are mutually exclusive.*

We infer from Proposition 5 and Lemma 3 simplification rules (26) and (27) in Table 5.

Table 5. Strategy simplification rules – system $\mathcal{SIMPL}$

$$fail; S \hookrightarrow fail \tag{1}$$
$$id; S \hookrightarrow S \tag{2}$$
$$S; id \hookrightarrow S \tag{3}$$
$$repeat^*(fail) \hookrightarrow id \tag{4}$$
$$dk(S_1, \ldots, fail, \ldots, S_n) \hookrightarrow dk(S_1, \ldots, S_n) \tag{5}$$
$$first(S_1, \ldots, fail, \ldots, S_n) \hookrightarrow first(S_1, \ldots, S_n) \tag{6}$$
$$dc(S_1, \ldots, fail, \ldots, S_n) \hookrightarrow dc(S_1, \ldots, S_n) \tag{7}$$
$$first(S_1, \ldots, id, \ldots, S_n) \hookrightarrow first(S_1, \ldots, id) \tag{8}$$
$$first(S_1, \ldots, repeat^*(S_i), \ldots, S_n) \hookrightarrow first(S_1, \ldots, repeat^*(S_i)) \tag{9}$$
$$f(S_1, \ldots, S_n); g(S'_1, \ldots, S'_m) \hookrightarrow \begin{cases} \emptyset & \text{if } f \neq g \\ f(S_1; S'_1, \ldots, S_n; S'_n) & \text{if } f = g \end{cases} \tag{10}$$
$$dk(S_1, \ldots, S_n); S \hookrightarrow dk(S_1; S, \ldots, S_n; S) \tag{11}$$
$$S; dk(S_1, \ldots, S_n) \hookrightarrow dk(S; S_1, \ldots, S; S_n) \tag{12}$$
$$dk(S) \hookrightarrow S \tag{13}$$
$$first(S) \hookrightarrow S \tag{14}$$
$$dc(S) \hookrightarrow S \tag{15}$$
$$dk(S_1, \ldots, dk(S'_1, \ldots, S'_m), \ldots, S_n) \hookrightarrow dk(S_1, \ldots, S'_1, \ldots, S'_m, \ldots, S_n) \tag{16}$$
$$first(S_1, \ldots, first(S'_1, \ldots, S'_m), \ldots, S_n) \hookrightarrow first(S_1, \ldots, S'_1, \ldots, S'_m, \ldots, S_n) \tag{17}$$
$$dc(S_1, \ldots, dc(S'_1, \ldots, S'_m), \ldots, S_n) \hookrightarrow dc(S_1, \ldots, S'_1, \ldots, S'_m, \ldots, S_n) \tag{18}$$
$$dk(\{\ldots\}_1, \ldots, \{\ldots\}_n) \hookrightarrow \bigcup_{i=1}^{n} \{\ldots\}_i \tag{19}$$
$$f(\{\ldots\}_1, \ldots, \{\ldots\}_n) \hookrightarrow \bigcup_{(l_i \to r_i \text{ if } c_i) \in \{\ldots\}_i} \{f(l_1, \ldots, l_n) \to f(r_1, \ldots, r_n) \text{ if } \bigwedge_{i=1}^{n} c_i\} \tag{20}$$
$$first(\ldots, S_i, \ldots, S_j, \ldots) \hookrightarrow first(\ldots, S_i, \ldots, \ldots) \text{ if } S_i \equiv S_j \tag{21}$$
$$dc(\ldots, S_i, \ldots, S_j, \ldots) \hookrightarrow dc(\ldots, S_i, \ldots, \ldots) \text{ if } S_i \equiv S_j \tag{22}$$
$$\{\ldots\}_1; \{\ldots\}_2 \hookrightarrow \bigcup_{(l_i \to r_i \text{ if } c_i) \in \{\ldots\}_i} comp(l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2)\downarrow_{\mathcal{C}omp} \tag{23}$$
$$first(S_1, \ldots, S_n) \hookrightarrow first(S_1, \ldots, constrain(dk(S_1, \ldots, S_{n-1}), S_n)\downarrow_{\mathcal{CO}}) \tag{24}$$
$$repeat^*(S_1); S_2 \hookrightarrow repeat^*(S_1); constrain(S_1, S_2)\downarrow_{\mathcal{CO}} \tag{25}$$
$$first(S_1, \ldots, S_n) \hookrightarrow dk(S_1, \ldots, S_n) \text{ if } \bigwedge_{i \neq j} mutex(S_i, S_j)\downarrow_{\mathcal{E}xcl} \tag{26}$$
$$dc(S_1, \ldots, S_n) \hookrightarrow dk(S_1, \ldots, S_n) \text{ if } \bigwedge_{i \neq j} mutex(S_i, S_j)\downarrow_{\mathcal{E}xcl} \tag{27}$$

(24) and (25) have to be applied once

4.4 A correct and complete simplification

We now have obtained simplification rules for all strategy operators. They are gathered in the rewriting system $\mathcal{SIMPL}$, on $\mathcal{T}(\mathcal{F} \cup \mathcal{S}, \mathcal{X})$ given in Table 5. Remark that although $\mathcal{C}omp$, $\mathcal{CO}$ and $\mathcal{E}xcl$ introduce symbols that are not in $\mathcal{F} \cup \mathcal{S}$, these symbols do not appear in $\mathcal{SIMPL}$. Indeed, only normal forms for $\mathcal{C}omp$, $\mathcal{CO}$ and $\mathcal{E}xcl$ are involved in $\mathcal{SIMPL}$, and these normal forms do not contain these symbols.
The following theorem states that the transformation induced by the simplification rules preserves both the semantics of the strategies and their termination behaviour.

Theorem 2. *Let S, S' be two strategies such that $S \hookrightarrow^*_{\mathcal{SIMPL}} S'$. Then S and S' are equivalent.*

As our simplification preserves the semantics of the initial strategy, it can be used not only for proving its termination, but also as an assitance for writing programs : obtaining a clearer, simpler and more concise form of the program enables a better control on it and facilitates proofs of its properties. Moreover, as the simplified form of the programs often is a set of rewrite rules, properties like confluence, sufficient completeness of the simplified program, reachability test of a given value and characterization of computed values can be studied and proved with classical methods of the rewriting theory. In addition, as the simplified form of the programs computes the same results as the initial one, reachability tests and characterization of computed forms also hold for the initial program.

4.5 Efficiently computing the simplification

The efficiency of the previously defined simplification of strategies is closely related itself to the strategy the rules of $\mathcal{SIMPL}$ are applied with. For instance, to simplify $first(S)$, it is recommended to apply the rule (1) at the top position before trying to simplify S. We now propose an efficient rewriting strategy to simplify a strategy with $\mathcal{SIMPL}$. The strategy consists in attaching to each operator a strategy annotation [22, 13, 11], with the notation $0_{(i)}$ for an attempt at rewriting at the top position *without using the rule* (i).

$$
\begin{array}{ll}
first & : [0_{(24)}, 1, 0_{(24)}, 2, \ldots, 0_{(24)}, n, 0] \\
\text{";"} & : [0, 1, 0, 2, 0] \\
dk & : [0, 1, 0, 2, \ldots, 0, n, 0] \\
dc & : [0, 1, 0, 2, \ldots, 0, n, 0] \\
repeat^* & : [0, 1, 0] \\
f & : [1, \ldots, n, 0] \text{ for } f \in \mathcal{F}
\end{array}
$$

The above strategies attached to each strategy operator rely on the idea that we first try to rewrite at the top position, then simplify an argument if rewriting at the top is not possible, then try again to rewrite at the top position, and so on. This strategy is somehow similar to the lazy strategy, but we need strategy annotations for a better control of the application of the rules, such as motivated by the simplification of $first$ strategies.
In particular, the strategy attached to the $first$ operator guarantees that the rule (24) is tried on a $first$ strategy S only when S has previously been simplified by all other possible rules. In fact, these other rules may remove arguments in S, while the rule (24) attempts at constraining each argument. In addition to the strategy annotations of $first$ and $repeat^*$, we respectively require rules (24) and (25) to be applied only once, such as noticed at the end of Section 4.2.

For the strategy attached to the ";", the normalization of the first argument is required before reducing the second argument. Indeed, $S_1; S_2$ can be simplified by rules (1) or (2) if S_1 simplifies respectively to $fail$ or id, and by rule (3) if S_2 simplifies to id. Therefore $S_1; S_2$ has more chance to be reduced by simplifying S_1 than by simplifying S_2.

But there is no clear preference for the order in which arguments of *dk* or *dc* strategies have to be normalized. We arbitrarily chose the leftmost order, but we could have chosen any permutation of the arguments.

Let us come back to the example introduced at the end of Section 3, where Theorem 1 failed to prove termination of $S = repeat^*(first(\{f(a) \to g(b)\}, \{f(x) \to g(x), g(a) \to f(a)\}))$, because the extracted set of rewrite rules $\{f(a) \to g(b), f(x) \to g(x), g(a) \to f(a)\}$ was not ε-terminating. Simplification of S with $\mathcal{SIMPL}$ results in $S' = repeat^*(\{f(a) \to g(b), f(x) \to g(x)$ if $x \neq a, g(a) \to f(a)\})$. Theorem 1 succeeds in proving termination of S', for the extracted set of rules to show termination of is $\{f(a) \to g(b), f(x) \to g(x)$ if $x \neq a, g(a) \to f(a)\}$, which can be shown terminating by Proposition 3. Note that no simplification ordering enables to show termination of the set of conditional rules above. For details, see examples in the appendix.

4.6 Examples

Let us show on other examples how simplification can both ease termination proof and make syntactic expression of strategies much clearer. We already showed in Example 2 that the strategy

$$S = repeat^* \ (\{f(x) \to g(x), f(x) \to h(0,x)\}; \\ \{h(1,x) \to f(1), h(x,y) \to y, f(1) \to h(1,1)\})$$

simplifies with $\mathcal{SIMPL}$ into

$$S' = repeat^*(\{f(x) \to x\}).$$

Theorem 1 fails to prove termination of S, because the extracted set of rules contains the rules $h(1,x) \to f(1)$ and $f(1) \to h(1,1)$, and hence is not ε-terminating. However, ε-termination of S' is obvious, and ensures termination of S. Remark that S' is syntactically much simpler than S, though equivalent.

Example 4. Let us consider the following three labelled rules :

$$[s_1]\ f(x,y) \to g(x,y) \\ [s_2]\ g(x,y) \to y \\ [s_3]\ g(x,y) \to f(y,x).$$

The strategy

$$S_1 = first(first(s_2, s_3), first(s_3, s_2, s_1)).$$

rewrites with $\mathcal{SIMPL}$ into

$$S_1' = s_1 \cup s_2.$$

Then, while the strategy $repeat^*(S_1)$ built on S_1 cannot be shown terminating by Theorem 1, the equivalent strategy $repeat^*(S_1')$ built on S_1' can, by Theorem 1. Let us now consider the strategy

$$S_2 = first(first(s_3, s_2), first(s_3, s_2, s_1)),$$

which rewrites with $\mathcal{SIMPL}$ into

$$s_1 \cup s_3.$$

Then, though Theorem 1 fails to say anything about termination of the strategy $repeat^*(S_2)$ built on S_2, it enables to show non termination of the strategy $repeat^*(S'2)$ and hence non termination of S_2.

The example above illustrates a case where non termination of a strategy can be shown. Let us mention that the library provided with ELAN offers a strategy implementing a matching algorithm. The simplification has made possible to show, in combination with Theorem 1, that this strategy was not terminating. The bug has been fixed since then.

Example 5. The following strategy

$$\begin{aligned} S = repeat^*(first(&\{f(x_1) \to g(x_1) \text{ if } x_1 > 3\};\\ &first(\{g(x_2) \to h(x_2) \text{ if } x_2 > 1\},\\ &\{g(4) \to g(4)\}, \{g(x_4) \to f(x_4) \text{ if } x_4 > 2\}),\\ &dk(\{f(x) \to g(x), f(x) \to h(0,x) \text{ if } x > 4\};\\ &\{h(1,x) \to f(1), h(x,y) \to y, f(1) \to h(1,1)\},\\ &f(x) \to f(f(x)) \text{ if } x > 5))) \end{aligned}$$

simplifies with $\mathcal{SIMPL}$ into

$$S' = repeat^*(\{f(x) \to h(x) \text{ if } x > 3\}).$$

Theorem 1 fails to prove termination of S, while it enables to prove termination of S' built on S'. Remark the important syntactic simplification of the strategy.

5 Conclusion

We have presented a criterion for proving termination of strategies lying on a simplification process of these strategies. This simplification is defined by a rewriting mechanism removing most of the operators in the strategy, and suppressing redundancy in application of its rewrite rules. Beyond the impact on the termination study, since a simplified strategy is syntactically much clearer than the original one, the simplification process can be used as a helpful verification tool for writing specifications.

Another interesting aspect of our transformation is that it enables to infer for strategies other classical properties than termination. Indeed, when a strategy is shown, by simplification, to be equivalent to a set of rules, then one can deal with classical rewriting properties like sufficient completeness or confluence on the simplified form of the program, and even with reachability and characterization of normal forms of the initial program.

A first interesting perspective is to extend this work to the full expressivity of ELAN. Indeed, in all its generality, an ELAN program is not only a combination of labelled rules, but a composition of labelled rules, together with a set of unlabelled rules, assumed to be confluent and terminating, and whose leftmost-innermost normalization strategy is predefined. A data is normalized with the unlabelled rules between two rewriting steps determined by the user defined strategy.

Since the unlabelled rules are evaluated leftmost-innermost, existing methods for proving innermost termination of rewriting [1, 9, 10] can be used. Their combination with labelled rules, under study, will be dealt with equational rewriting [23], by rewriting with labelled rules modulo rewriting with unlabelled rules.

Another perspective is the use of our simplification of the program at run time. We will now study in which cases it will be more efficient to replace the direct execution of the ELAN strategies, as ELAN has provided till now, by the rewriting process with the set of rules obtained by simplification.

References

1. Th. Arts and J. Giesl. A collection of examples for termination of term rewriting using dependency pairs. Technical Report AIB-2001-09, RWTH Aachen, Germany, September 2001.
2. P. Borovanský, C. Kirchner, H. Kirchner, and Ch. Ringeissen. Rewriting with strategies in ELAN: a functional semantics. *International Journal of Foundations of Computer Science*, 1999.
3. Peter Borovanský, Claude Kirchner, Hélène Kirchner, and Pierre-Etienne Moreau. Elan from a rewriting logic point of view. *Theoretical Computer Science*, 285:155–185, August 2002.

4. Peter Borovanský, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, and Christophe Ringeissen. An overview of ELAN. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the second International Workshop on Rewriting Logic and Applications*, volume 15, `http://www.elsevier.nl/locate/entcs/volume15.html`, Pont-à-Mousson (France), September 1998. Electronic Notes in Theoretical Computer Science. Report LORIA 98-R-316.
5. H. Cirstea. *Le Rho Calcul: Fondements et Applications*. PhD thesis, Université Henri Poincaré - Nancy I, 2000.
6. N. Dershowitz, M. Okada, and G. Sivakumar. Canonical conditional rewrite systems. In E. Lusk and R. Overbeek, editors, *Proceedings of the 9th International Conference on Automated Deduction*, volume 310 of *Lecture Notes in Computer Science*, pages 538–549, 1988.
7. Nachum Dershowitz and Jean-Pierre Jouannaud. *Handbook of Theoretical Computer Science*, volume B, chapter 6: Rewrite Systems, pages 244–320. Elsevier Science Publishers B. V. (North-Holland), 1990. Also as: Research report 478, LRI.
8. M. Clavel et al. Maude: specification and programming in rewriting logic. *Theoretical Computer Science*, 285:187–243, August 2002.
9. O. Fissore, I. Gnaedig, and H. Kirchner. Termination of rewriting with local strategies. In M. P. Bonacina and B. Gramlich, editors, *Electronic Notes in Theoretical Computer Science*, volume 58. Elsevier Science Publishers, 2001. Also available as Technical Report A01-R-177, LORIA, Nancy (France).
10. O. Fissore, I. Gnaedig, and H. Kirchner. CARIBOO : An induction based proof tool for termination with strategies. In *Proceedings of the Fourth International Conference on Principles and Practice of Declarative Programming (PPDP)*, Pittsburgh, USA, October 2002. ACM Press.
11. K. Futatsugi and A. Nakagawa. An overview of CAFE specification environment – an algebraic approach for creating, verifying, and maintaining formal specifications over networks. In *Proceedings of the 1st IEEE Int. Conference on Formal Engineering Methods*, 1997.
12. I. Gnaedig. Termination of order-sorted rewriting. In Hélène Kirchner and G. Levi, editors, *Proceedings 3rd International Conference on Algebraic and Logic Programming, Volterra (Italy)*, volume 632 of *Lecture Notes in Computer Science*, pages 37–52. Springer-Verlag, September 1992.
13. Joseph A. Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. In Joseph A. Goguen and Grant Malcolm, editors, *Software Engineering with OBJ: Algebraic Specification in Action*, chapter 1, pages 3–167. Kluwer, Boston, 2000.
14. Bernhard Gramlich. On termination and confluence properties of disjoint and constructor-sharing conditional rewrite systems. *Theoretical Computer Science*, 165(1):97–131, September 1996.
15. M. Hanus. Reduction strategies for declarative programming. In B. Gramlich and S. Lucas, editors, *Electronic Notes in Theoretical Computer Science*, volume 57. Elsevier Science Publishers, 2001. Also available from `http://www.elsevier.nl/locate/entcs/volume57.html`.
16. M. Hanus and F. Steiner. Controlling search in declarative programs. In *Principles of Declarative Programming (Proc. Joint International Symposium PLILP/ALP'98)*, pages 374–390. Springer LNCS 1490, 1998.
17. R. Kieburtz. A Logic for Rewriting Strategies. In M. P. Bonacina and B. Gramlich, editors, *Electronic Notes in Theoretical Computer Science*, volume 58. Elsevier Science Publishers, 2001.
18. P. Klint. A meta-environment for generating programming environments. *ACM Transactions on Software Engineering and Methodology*, 2:176–201, 1993.
19. A.P. Martin, P. H. B. Gardiner, and J. C. P Woodcock. A tactic calculus – full version. *Formal Aspects of Computing*, 8(E):244–285, 1996.
20. Christian Schulte and Gert Smolka. Encapsulated search in higher-order concurrent constraint programming. In Maurice Bruynooghe, editor, *Logic Programming: Proceedings of the 1994 International Symposium*, pages 505–520, Ithaca, New York, USA, November 1994. MIT-Press.
21. Gert Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin, 1995.
22. J. van de Pol. Just-in-time: on strategy annotations. In B. Gramlich and S. Lucas, editors, *Proceedings of the first International Workshop pn Reduction Strategies in Rewriting and Programming, WRS'01*, volume 2359, pages 39–58, 2001.
23. Patrick Viry. Equational rules for rewriting logic. *Theoretical Computer Science*, 285:487–517, 2002.
24. Eelco Visser and Zine el Abidine Benaissa. A core language for rewriting. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the second International Workshop on Rewriting Logic and*

Applications, volume 15, Pont-à-Mousson (France), September 1998. Electronic Notes in Theoretical Computer Science.
25. M. Vittek. *ELAN: Un cadre logique pour le prototypage de langages de programmation avec contraintes*. Thèse de Doctorat d'Université, Université Henri Poincaré - Nancy 1, octobre 1994.

A Proofs

A.1 Composition (Section 4.1)

Proposition 2 *Let $l_1 \to r_1$ if $c_1, l_2 \to r_2$ if c_2 be two labelled rewrite rules, and let us assume that any rewriting step only occurs at the top position. Then, for any ground term $t \in \mathcal{T}(\mathcal{F})$, we have :*

- $[l_1 \to r_1$ *if* $c_1; l_2 \to r_2$ *if* $c_2](t) = \emptyset$ *if* r_1 *and* l_2 *are not unifiable.*
- $[l_1 \to r_1$ *if* $c_1; l_2 \to r_2$ *if* $c_2](t) = [\mu l_1 \to \mu r_2$ *if* $\mu c_1 \wedge \mu c_2](t)$ *if* μ *is a most general unifier of* r_1 *and* l_2.

Proof: We recall that we assume $\mathcal{V}ar(l_1) \cap \mathcal{V}ar(l_2) = \emptyset$, which can be ensured by an appropriate renaming of variables.

Let $(l_1 \to r_1$ if $c_1)$ and $(l_1 \to r_2$ if $c_2)$ two ELAN rewrite rules such that r_1 and l_2 are not unifiable, and let $t \in \mathcal{T}(\mathcal{F})$. Let us show $[l_1 \to r_1$ if $c_1; l_2 \to r_2$ if $c_2](t) = \emptyset$.

If $[l_1 \to r_1$ if $c_1](t) = \emptyset$, then the property is trivial.

Let $t' = [l_1 \to r_1$ if $c_1](t)$. By definition of the rewriting, there exists a matching ground substitution α_1 such that $Dom(\alpha_1) = \mathcal{V}ar(l_1)$ and $\alpha_1 r_1 = t'$. Let us assume the existence of a matching ground substitution α_2 such that $Dom(\alpha_2) = \mathcal{V}ar(l_2)$ and $\alpha_2 l_2 = t'$. Then we would have $\alpha_2 l_2 = \alpha_1 r_1$, with $\mathcal{V}ar(l_2) \cap \mathcal{V}ar(r_1) = \emptyset$. Therefore $(\alpha_1 \wedge \alpha_2)(l_2) = \alpha_2 l_2 = \alpha_1 r_1 = (\alpha_1 \wedge \alpha_2)(r_1)$, which is in contradiction with the hypothesis according to which r_1 and l_2 are not unifiable. Consequently, there exists no matching substitution α_2 such that $\alpha_2 l_2 = t'$, and then $[l_2 \to r_2$ if $c_2](t') = \emptyset$, and therefore $[l_1 \to r_1$ if $c_1; l_2 \to r_2$ if $c_2](t) = \emptyset$.

Let us now assume $\exists \mu = mgu(l_2, r_1)$, and let $t \in \mathcal{T}(\mathcal{F})$.

We first show the inclusion $[l_1 \to r_1$ if $c_1; l_2 \to r_2$ if $c_2](t) \subseteq [\mu l_1 \to \mu r_2$ if $\mu c_1 \wedge \mu c_2](t)$.

If $[l_1 \to r_1$ if $c_1; l_2 \to r_2$ if $c_2](t) = \emptyset$, then the inclusion is trivial.

Let $\{t'\} = [l_1 \to r_1$ if $c_1; l_2 \to r_2$ if $c_2](t)$, and show that $[\mu l_1 \to \mu r_2$ if $\mu c_1 \wedge \mu c_2](t) = \{t'\}$.

By definition of the composition and of the conditional rewriting, we have :

- $\exists t'' \in \mathcal{T}(\mathcal{F}) : [l_1 \to r_1$ if $c_1](t) = \{t''\}$ and $[l_2 \to r_2$ if $c_2](t'') = \{t'\}$.
- $\exists \alpha_1$ ground substitution such that $Dom(\alpha_1) = \mathcal{V}ar(l_1)$ and $\alpha_1 l_1 = t_1$ and $\alpha_1 c_1 \longrightarrow^* true$ and $t'' = \alpha_1 r_1$.
- $\exists \alpha_2$ ground substitution such that $Dom(\alpha_2) = \mathcal{V}ar(l_2)$ and $\alpha_2 l_2 = t''$ and $\alpha_2 c_2 \longrightarrow^* true$ and $t' = \alpha_2 r_2$.

Our goal is to exhibit a ground substitution α such that $\alpha\mu l_1 = t$ and $\alpha(\mu c_1 \wedge \mu c_2) \longrightarrow^* true$ and $t' = \alpha\mu r_2$. We are going to build α so that $\alpha\mu = \alpha_1[Dom(\alpha_1)]$ and $\alpha\mu = \alpha_2[Dom(\alpha_2)]$.

For $x \in Dom(\alpha_1) \setminus Dom(\mu)$, we state $\alpha x = \alpha_1 x$ (1).

Let $x \in Dom(\alpha_1) \cap Dom(\mu)$. We have $Dom(\alpha_1) = \mathcal{V}ar(l_1)$ and $\mathcal{V}ar(l_1) \cap \mathcal{V}ar(l_2) = \emptyset$, therefore $x \notin \mathcal{V}ar(l_2)$. In addition, $x \in Dom(\mu)$ and $\mu = mgu(r_1, l_2)$, therefore, since $x \notin \mathcal{V}ar(l_2)$, we have $x \in \mathcal{V}ar(r_1)$.

By hypothesis, we have $\alpha_2 l_2 = t''$ and $t'' = \alpha_1 r_1$, hence $\alpha_1 r_1 = \alpha_2 l_2$. Since $\mathcal{V}ar(r_1) \cap Dom(\alpha_2) = \emptyset$ and $\mathcal{V}ar(l_2) \cap Dom(\alpha_1) = \emptyset$ the ground substitution $\alpha_1 \wedge \alpha_2$ unifies r_1 and l_2. Since $\mu = mgu(l_2, r_1)$, $(\alpha_1 \wedge \alpha_2)(x)$ is a ground instance of $\mu(x)$. Since $x \notin \mathcal{V}ar(l_2)$ and $Dom(\alpha_2) = \mathcal{V}ar(l_2)$, we get $x \notin Dom(\alpha_2)$ and hence $\alpha_1 \wedge \alpha_2(x) = \alpha_1(x)$. We then showed that $\alpha_1(x)$ is a ground instance of $\mu(x)$. We can then state $\alpha_1(x) = \alpha\mu(x)$ for $x \in Dom(\alpha_1) \cap Dom(\mu)$ (2).

From (1) and (2), we infer $\alpha\mu = \alpha_1[Dom(\alpha_1)]$ (3).

By a symetrical reasoning, we build α such that $\alpha\mu = \alpha_2[Dom(\alpha_2)]$ (4). We now show $\alpha\mu l_1 = t$ and $\alpha(\mu c_1 \wedge \mu c_2) \longrightarrow^* true$ and $t' = \alpha\mu r_2$.

By definition of α_1, we have $Dom(\alpha_1) = \mathcal{V}ar(l_1)$ and $\alpha_1 l_1 = t$; we then infer from (3) that $\alpha\mu l_1 = t$ (5).

By definition of a conditional rewrite rule, we have $\mathcal{V}ar(c_1) \subseteq \mathcal{V}ar(l_1) = Dom(\alpha_1)$ and, by construction of α_1, we have $\alpha_1 c_1 \longrightarrow^* true$. From (3) we then infer $\alpha\mu c_1 \longrightarrow^* true$ (6).

By definition of a conditional rewrite rule, we have $\mathcal{V}ar(r_2) \subseteq \mathcal{V}ar(l_2)$ and, by construction of α_2, we have $\mathcal{V}ar(l_2) = Dom(\alpha_2)$ and $t' = \alpha_2 r_2$. From (4) we then infer $\alpha\mu r_2 = t'$ (7).
By definition of a conditional rewrite rule, we have $\mathcal{V}ar(c_2) \subseteq \mathcal{V}ar(l_2) = Dom(\alpha_2)$ and, by construction of α_2, we have $\alpha_2 c_2 \longrightarrow^* true$. From (4) we then infer $\alpha\mu c_2 \longrightarrow^* true$ (8).
Finally, from (5), (6), (7) and (8) we infer $\alpha\mu l_1 = t$ and $\alpha(\mu c_1 \wedge \mu c_2) \longrightarrow^* true$ and $t' = \alpha\mu r_2$, and then $\{t'\} = [\mu l_1 \rightarrow \mu r_2 \text{ if } \mu c_1 \wedge \mu c_2](t)$.
Let us now show the inclusion $[\mu l_1 \rightarrow \mu r_2 \text{ if } \mu c_1 \wedge \mu c_2](t) \subseteq [l_1 \rightarrow r_1 \text{ if } c_1; l_2 \rightarrow r_2 \text{ if } c_2](t)$.
If $[\mu l_1 \rightarrow \mu r_2 \text{ if } \mu c_1 \wedge \mu c_2](t) = \emptyset$, then the inclusion is trivial.
Let $t' = [\mu l_1 \rightarrow \mu r_2 \text{ if } \mu c_1 \wedge \mu c_2](t)$. By definition, there exists a ground substitution α such that $\alpha\mu l_1 = t$ and $\alpha(\mu c_1 \wedge \mu c_2) \longrightarrow^* true$ and $t' = \alpha\mu r_2$.
Let us consider α_1 and α_2 so that $\alpha_1 = \alpha\mu[\mathcal{V}ar(l_1)]$ and $\alpha_2 = \alpha\mu[\mathcal{V}ar(l_2)]$. We then have :

- $\alpha_1 l_1 = \alpha\mu l_1 = t$ (1)
- by definition of a conditional rewrite rule, we have $\mathcal{V}ar(c_1) \subseteq \mathcal{V}ar(l_1)$, therefore $\alpha\mu c_1 = \alpha_1 c_1$. Since we have, by hypothesis, $\alpha(\mu c_1 \wedge \mu c_2) \longrightarrow^* true$, we have in particular $\alpha\mu c_1 \longrightarrow^* true$, hence $\alpha_1 c_1 \longrightarrow^* true$ (2)
- by definition of a conditional rewrite rule, we have $\mathcal{V}ar(c_2) \subseteq \mathcal{V}ar(l_2)$, therefore $\alpha\mu c_2 = \alpha_2 c_2$. Since we have, by hypothesis, $\alpha(\mu c_1 \wedge \mu c_2) \longrightarrow^* true$, we have in particular $\alpha\mu c_2 \longrightarrow^* true$, hence $\alpha_2 c_2 \longrightarrow^* true$ (3)
- $\mathcal{V}ar(r_2) \subseteq \mathcal{V}ar(l_2)$, therefore $\alpha_2 r_2 = \alpha\mu r_2 = t'$ (4)
- by definition of μ, we have $\mu r_1 = \mu l_2$, therefore $\alpha\mu r_1 = \alpha\mu l_2$, hence $\alpha_1 r_1 = \alpha_2 l_2$ (5)

Denoting $t'' = \alpha_1 r_1$, we have shown :

- $\alpha_1 l_1 = t$ and $\alpha_1 c_1 \longrightarrow^* true$ and $t'' = \alpha_1 r_1$, therefore $\{t''\} = [l_1 \rightarrow r_1 \text{ if } c_1](t)$.
- $\alpha_2 l_2 = t''$ and $\alpha_2 c_2 \longrightarrow^* true$ and $t' = \alpha_2 r_2$, therefore :

$$\begin{aligned} t' &= [l_2 \rightarrow r_2 \text{ if } c_2](t'') \\ &= [l_2 \rightarrow r_2 \text{ if } c_2]([l_1 \rightarrow r_1 \text{ if } c_1](t)) \\ &= [l_1 \rightarrow r_1 \text{ if } c_1; l_2 \rightarrow r_2 \text{ if } c_2](t) \end{aligned}$$

□

A.2 Constraining strategies (Section 4.2)

Proposition 4 *Let $l_1 \rightarrow r_1$ if c_1 and $l_2 \rightarrow r_2$ if c_2 two conditional rewrite rules, $t \in \mathcal{T}(\mathcal{F})$ such that $l_1 \rightarrow r_1$ if c_1 fails on t. If there exists a unifying substitution μ such that $\mu l_1 = \mu l_2$ and $Ran(\mu) \subseteq \mathcal{V}ar(l_2)$, then we have $[l_2 \rightarrow r_2$ if $c_2](t) = [l_2 \rightarrow r_2$ if $c_2 \wedge not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)})](t)$.*

Proof: Let $t \in \mathcal{T}(\mathcal{F})$.
Let us first show the inclusion $[l_2 \rightarrow r_2 \text{ if } c_2](t) \subseteq [l_2 \rightarrow r_2 \text{ if } c_2 \wedge not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)})](t)$.
If $[l_2 \rightarrow r_2 \text{ if } c_2](t) = \emptyset$, then the inclusion is trivial. Let us assume $\exists t' \in \mathcal{T}(\mathcal{F}) : [l_2 \rightarrow r_2 \text{ if } c_2](t) = \{t'\}$. By definition, there exists a ground substitution α such that $\alpha l_2 = t$ and $\alpha c_2 \longrightarrow^* true$ and $\wedge t' = \alpha r_2$. Let us show that α is also the matching substitution enabling the application of $l_2 \rightarrow r_2$ if $c_2 \wedge not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)})$ to t. Thus we would have $[l_2 \rightarrow r_2 \text{ if } c_2 \wedge not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)})](t) = \{t'\}$, since $\alpha r_2 = t'$.
We already have the properties $\alpha l_2 = t$ and $\alpha c_2 \longrightarrow^* true$. It remains to show that $\alpha not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)}) \longrightarrow^* true$. Let us proceed by contradiction : we assume that $\alpha\mu c_1 \wedge \alpha\mu_{\mathcal{V}ar(l_2)} \longrightarrow^* true$, and we find a ground substitution α_1 such that $\alpha_1 l_1 = t$ and $\alpha_1 c_1 \longrightarrow^* true$, which contradicts the failure of the application of the rule $l_1 \rightarrow r_1$ if c_1 to t.
Let us denote $\alpha' = \alpha[\mathcal{V}ar(l_2) \setminus Dom(\mu)]$ such that $Dom(\alpha') = \mathcal{V}ar(l_2) \setminus Dom(\mu)$. Let us now consider the ground substitution α_1 such that $\alpha_1 = \alpha'\mu[\mathcal{V}ar(l_1) \cup \mathcal{V}ar(l_2)]$.
We first show that $\alpha_1 l_1 = t$. By definition of α_1, we have $\alpha_1 l_1 = \alpha'\mu l_1$. By definition of μ, we have $\mu l_1 = \mu l_2$, therefore $\alpha_1 l_1 = \alpha'\mu l_2$ (0).
We now show that $\alpha'\mu l_2 = \alpha l_2$. For $x \in \mathcal{V}ar(l_2) \cap Dom(\mu)$, we have $\mu x = \alpha x$, for we

know by hypothesis that $\alpha\mu_{\mathcal{V}ar(l_2)} \longrightarrow^* true$. Since α is a ground substitution, we get $\alpha'\mu x = \alpha'\alpha x = \alpha x$ (1). For $x \in \mathcal{V}ar(l_2) \setminus Dom(\mu)$, we have $\mu x = x$ and, by definition of α', $\alpha' x = \alpha x$, and therefore $\alpha'\mu x = \alpha' x = \alpha x$ (2). From (1) and (2), we get $\alpha'\mu x = \alpha x$ for any $x \in \mathcal{V}ar(l_2)$, and then $\alpha'\mu l_2 = \alpha l_2$. From (0), we then get $\alpha_1 l_1 = \alpha l_2$ and then $\alpha_1 t_1 = t$ by definition of α.
Let us now show that $\alpha_1 c_1 \longrightarrow^* true$, by showing that $\alpha_1 c_1 = \alpha\mu c_1$.
Let $x_1 \in \mathcal{V}ar(c_1)$. By hypothesis, we have $\mathcal{V}ar(c_1) \subseteq \mathcal{V}ar(l_1)$, therefore $x_1 \in \mathcal{V}ar(l_1)$. Let $x_2 \in \mathcal{V}ar(\mu x_1)$. Since, by definition of μ, we have $Ran(\mu) \subseteq \mathcal{V}ar(l_2)$, we get $x_2 \in \mathcal{V}ar(l_2)$. Moreover, $x_2 \in \mathcal{V}ar(\mu x_1)$, therefore $x_2 \notin Dom(\mu)$, and finally $x_2 \in \mathcal{V}ar(l_2) \setminus Dom(\mu)$. By definition of α', we then get $\alpha' x_2 = \alpha x_2$ for any $x_2 \in \mathcal{V}ar(\mu x_1)$, and hence $\alpha'\mu x_1 = \alpha\mu x_1$, therefore $\alpha_1 x_1 = \alpha\mu x_1$ for any $x_1 \in \mathcal{V}ar(c_1)$. We then get $\alpha_1 c_1 = \alpha\mu c_1$, and therefore, by hypothesis, $\alpha_1 c_1 \longrightarrow^* true$.
We have shown the existence of a substitution α_1 such that $\alpha_1 l_1 = t$ and $\alpha_1 c_1 \longrightarrow^* true$, which is in contradiction with the failure of the application of the rule $l_1 \to r_1$ if c_1 to t. Therefore the property $\alpha\mu c_1 \wedge \alpha\mu_{\mathcal{V}ar(l_2)} \longrightarrow^* true$ is false, and then we have $\alpha not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)}) \longrightarrow^* true$, and henceforth $[l_2 \to r_2 \text{ if } c_2 \wedge not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)})](t) = \{\alpha r_2\} = \{t'\}$.
Let us now show the inclusion $[l_2 \to r_2 \text{ if } c_2 \wedge not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)})](t) \subseteq [l_2 \to r_2 \text{ if } c_2](t)$.
If $[l_2 \to r_2 \text{ if } c_2 \wedge not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)})](t) = \emptyset$, then the inclusion is trivial. Let us assume $\exists t' \in \mathcal{T}(\mathcal{F}) : [l_2 \to r_2 \text{ if } c_2 \wedge not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)})](t) = \{t'\}$, and let show that $[l_2 \to r_2 \text{ if } c_2](t) = \{t'\}$. By hypothesis, there exists a ground substitution α such that $\alpha l_2 = t$ and $\alpha(c_2 \wedge not(\mu c_1 \wedge \mu_{\mathcal{V}ar(l_2)})) \longrightarrow^* true$ and $\alpha r_2 = t'$. In particular, α is such that $\alpha c_2 \longrightarrow^* true$, and then $[l_2 \to r_2 \text{ if } c_2](t) = \{t'\}$ with the matching substitution α.
□

Lemma 1 *Given a strategy S, we have :*

1. *the reduction of the term $failureRules(S)$ with $\mathcal{FR}$ terminates and its normal form is a (possibly empty) set of rewrite rules ;*
2. *every rule of $failureRules(S)\downarrow_{\mathcal{FR}}$ fails on any ground term $t \in \mathcal{T}(\mathcal{F})$ such that $[S](t) = \emptyset$.*

Proof: Let us show Lemma 1 by structural induction on S.
We first consider the case where S is an atom :

- If $S = id$ or $S = fail$, then $failureRules(S) \longrightarrow_{\mathcal{FR}} \emptyset$, and $failureRules(S)\downarrow_{\mathcal{FR}} = \emptyset$. Consequently we have $[failureRules(S)\downarrow_{\mathcal{FR}}](t) = \emptyset$ for any t, and Lemma 1 holds for S.
- If $S = \{l_1 \to r_1, \ldots, l_n \to r_n\}$, then $failureRules(S) \longrightarrow_{\mathcal{FR}} S$, and $failureRules(S)\downarrow_{\mathcal{FR}} = S$. Consequently Lemma 1 trivially holds.

Let us now assume the properties true for $S_1, \ldots, S_n$, and let us show that Lemma 1 holds for $repeat^*(S_1)$ and $op(S_1, \ldots, S_n)$, $op \in \mathcal{F} \cup \{dk, dc, first\}$.

- If $S = op(S_1, \ldots, S_n)$, with $op \in \{dk, dc, first\}$, then $failureRules(S) \longrightarrow_{\mathcal{FR}} \bigcup_{i=1}^n failureRules(S_i)$ and $failureRules(S)\downarrow_{\mathcal{FR}} = \bigcup_{i=1}^n failureRules(S_i)\downarrow_{\mathcal{FR}}$.
  Let us show the two properties of Lemma 1.
  1. By induction hypothesis, for each $i \in \{1, \ldots, n\}$, the evaluation of $failureRules(S_i)$ with $\mathcal{FR}$ terminates and $failureRules(S_i)\downarrow_{\mathcal{FR}}$ is a set of rewrite rules. Therefore $failureRules(S)\downarrow_{\mathcal{FR}} = \bigcup_{i=1}^n failureRules(S_i)\downarrow_{\mathcal{FR}}$ is a set of rewrite rules.
  2. Let t be a ground term such that $[S](t) = \emptyset$. By definition of the dk, dc and $first$ operators (see Table 1), we have $[S](t) = \emptyset$ iff $[S_i](t) = \emptyset$ for each $i \in \{1, \ldots, n\}$. By induction hypothesis, since $[S](t) = \emptyset$, we have $[failureRules(S_i)\downarrow_{\mathcal{FR}}](t) = \emptyset$ for each $i \in \{1, \ldots, n\}$, and therefore $[failureRules(S)\downarrow_{\mathcal{FR}}](t) = \bigcup_{i=1}^n [failureRules(S_i)\downarrow_{\mathcal{FR}}](t) = \emptyset$.

– If $S = f(S_1, \ldots, S_n)$, with $f \in \mathcal{F}$, then $failureRules(S) \longrightarrow_{\mathcal{FR}} f(failureRules(S_1), \ldots, failureRules(S_n))$, and hence $failureRules(S)\downarrow_{\mathcal{FR}} = f(failureRules(S_1)\downarrow_{\mathcal{FR}}, \ldots, failureRules(S_n)\downarrow_{\mathcal{FR}})\downarrow_{\mathcal{FR}}$.
Let t be a ground term such that $[S](t) = \emptyset$. By induction hypothesis , each $failureRules(S_i)\downarrow_{\mathcal{FR}}$ is a set of rules $\{\ldots\}_i$ such that $[\{\ldots\}_i](t) = \emptyset$.
Therefore $failureRules(S)\downarrow_{\mathcal{FR}} = \bigcup_{(l_i \to r_i \text{ if } c_i) \in \{\ldots\}_1} \{f(l_1, \ldots, l_n) \to f(r_1, \ldots, r_n) \text{ if } \bigwedge_{i=1}^{n} c_i\}$ is a set of rules.
Let us show that $[failureRules(S)\downarrow_{\mathcal{FR}}](t) = \emptyset$. By definition of t, we have $[f(S_1, \ldots, S_n)](t) = \emptyset$, that is either $top(t) \neq f$ or $t = f(t_1, \ldots, t_n)$ and $\exists j \in \{1, \ldots, n\} : [S_j](t_j) = \emptyset$.
If $top(t) \neq f$, then we obviously have $[failureRules(S)\downarrow_{\mathcal{FR}}](t) = \emptyset$.
Let us now assume $t = f(t_1, \ldots, t_n)$. Then $\exists j \in \{1, \ldots, n\} : [S_j](t_j) = \emptyset$. By induction hypothesis, we have $[\{\ldots\}_j](t_j) = \emptyset$, i.e. $[l \to r \text{ if } c](t_j) = \emptyset$ for any rule $(l \to r \text{ if } c) \in \{\ldots\}_j$.
Consequently, for any rule $(l_j \to r_j \text{ if } c_j) \in \{\ldots\}_j$, we have :
$[f(l_1, \ldots, l_j, \ldots, l_n) \to f(r_1, \ldots, r_j, \ldots, r_n) \text{ if } c_j \bigwedge_{i \neq j} c_i](t_j) = \emptyset$,
and hence $[failureRules(S)\downarrow_{\mathcal{FR}}](f(t_1, \ldots, t_n)) = \emptyset$.
– If $S = S_1; S_2$ or $S = repeat^*(S_1)$, then $failureRules(S) \longrightarrow_{\mathcal{FR}} \emptyset$ and hence $failureRules(S)\downarrow_{\mathcal{FR}} = \emptyset$. Consequently, $[failureRules(S)\downarrow_{\mathcal{FR}}](t) = \emptyset$ for any ground term t.

□

Lemma 2 *For any strategies S_1 and S_2, the reduction of the term $constrain(S_1, S_2)$ with $\mathcal{CO}$ terminates, and its normal form S_2' is a strategy such that for any ground term $t \in \mathcal{T}(\mathcal{F})$ on which S_1 fails :*

(correctness) $[S_2'](t) \subseteq [S_2](t)$.
(completeness) $[S_2](t) \subseteq [S_2'](t)$.
(termination) $[S_2](t)$ *terminates iff* $[S_2'](t)$ *terminates.*

Proof: If S_1 is not a set of rules, then the first rewriting of $constrain(S_1, S_2)$ with $\mathcal{CO}$ is $constraint(S_1, S_2) \longrightarrow_{\mathcal{CO}} constrain(S_1\downarrow_{\mathcal{FR}}, S_2)$. By Lemma 1, there exists a set of rules $\{\ldots\}_1 = S_1\downarrow_{\mathcal{FR}}$ such that $[\{\ldots\}_1](t) = \emptyset$ for any ground term $t \in \mathcal{T}(\mathcal{F})$ such that $[S_1](t) = \emptyset$.
If S_1 is a set of rewrite rules, let us denote $\{\ldots\}_1 = S_1$. In any case, the normal form of $constrain(S_1, S_2)$ with $\mathcal{NF}$ exists iff the normal form of $constrain(\{\ldots\}_1, S_2)$ with $\mathcal{NF}$ exists and, in case of existence, we have $constrain(S_1, S_2)\downarrow_{\mathcal{CO}} = constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}$.
Let $t \in \mathcal{T}(\mathcal{F})$ such that $[S_1](t) = \emptyset$. By construction of $\{\ldots\}_1$, t is also such that $[\{\ldots\}_1](t) = \emptyset$. Proving Lemma 2 then comes down to showing the following four properties :
(P1) $constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}$ exists and is an ELAN strategy.
(P2) $[S_2](t) \subseteq [constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t)$.
(P3) $[constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t) \subseteq [S_2](t)$.
(P4) $[S_2](t)$ terminates iff $[constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t)$ terminates.
We proceed by structural induction on S_2.
We first prove the property when S_2 is an atom.
If $S_2 = id$ or $S_2 = fail$ or $S_2 = \emptyset$, then $constrain(\{\ldots\}_1, S_2) \to S_2$, and the properties are trivially true.
If $S_2 = l \to r$ if c, then we have :

$$constrain(\{\ldots\}_1, S_2) \longrightarrow_{\mathcal{CO}} \{l \to r \text{ if } c \bigwedge_{(\mu_i, c_i) \in E} not(\mu_i c_i \wedge \mu_i \mathcal{V}ar(l))\}$$

with $E = \{(\mu_i, c_i) | \exists (l_i \to r_i \text{ if } c_i) \in \{\ldots\}_1$ such that $\mu_i l = \mu_i l_i$ and $Ran(\mu_i) \subseteq \mathcal{V}ar(l)\}$.
Then the evaluation of $constrain(\{\ldots\}_1, S)$ with $\mathcal{CO}$ terminates and its normal form is a rewrite rule, therefore an ELAN strategy. Since S_2 is a rewrite rule, $[S_2](t)$ terminates and

we only have to show $[l \to r \text{ if } c](t) = [l \to r \text{ if } c \bigwedge_{(\mu_i,c_i)\in E} not(\mu_i c_i \wedge \mu_i \mathcal{V}ar(l))](t)$, which is straightforward from Proposition 4.
If $S_2 = \{l_1 \to r_1 \text{ if } c_1, \ldots, l_n \to r_n \text{ if } c_n\}_2$, then we have :

$$\begin{aligned} constrain(\{\ldots\}_1, S_2) &\longrightarrow_{\mathcal{CO}} \bigcup_{(l\to r \text{ if } c)\in\{\ldots\}_2} constrain(\{\ldots\}_1, l \to r \text{ if } c) \\ &\longrightarrow_{\mathcal{CO}} \bigcup_{(l\to r \text{ if } c)\in\{\ldots\}_2} \{l \to r \text{ if } c \bigwedge_{(\mu_i,c_i)\in E} not(\mu_i c_i \wedge \mu_i \mathcal{V}ar(l))\} \end{aligned}$$

with $E = \{(\mu_i, c_i) | \exists (l_i \to r_i \text{ if } c_i) \in \{\ldots\}_1$ such that $\mu_i l = \mu_i l_i$ and $Ran(\mu_i) \subseteq \mathcal{V}ar(l)\}$.
Then the evaluation of $constrain(\{\ldots\}_1, S_2)$ with $\mathcal{CO}$ terminates and its normal form is a union of rewrite rules, therefore an ELAN strategy. Since S_2 is a set of rewrite rules, $[S_2](t)$ terminates and we only have to show the properties (P2) and (P3), which is straightforward from Proposition 4.
Finally the properties hold for any atom S_2. Let us now assume that the properties hold for $S'_1, \ldots, S'_n$, and let us show them for any combination S_2 of $S'_1, \ldots, S'_n$.
If $\mathbf{S_2} = \mathbf{dk}(\mathbf{S'_1}, \ldots, \mathbf{S'_n})$, we have the following rewriting step :

$$constrain(\{\ldots\}_1, S_2) \longrightarrow_{\mathcal{CO}} dk(constrain(\{\ldots\}_1, S'_1), \ldots, constrain(\{\ldots\}_1, S'_n)).$$

Let us show the property $(P1)$. By induction hypothesis , for $i \in \{1, \ldots, n\}$, the evaluation of $constrain(\{\ldots\}_1, S'_i)$ with $\mathcal{CO}$ terminates and its normal form is an ELAN strategy.
Therefore $constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}} = dk(constrain(\{\ldots\}_1, S'_1)\downarrow_{\mathcal{CO}}, \ldots, constrain(\{\ldots\}_1, S'_n)\downarrow_{\mathcal{CO}})$ exists and is an ELAN strategy (**dk0**).
Let us now show (P2) and (P3). By definition of the dk operator, we have $[S_2](t) = \bigcup_{i=1}^{n} [S'_i](t)$ and, by induction hypothesis , we have $[S'_i](t) = [constrain(\{\ldots\}_1, S'_i)\downarrow_{\mathcal{CO}}](t)$ $\forall i \in \{1, \ldots, n\}$. Then we get

$$[S_2](t) = \bigcup_{i=1}^{n} [constrain(\{\ldots\}_1, S'_i)\downarrow_{\mathcal{CO}}](t) \quad (\mathbf{dk1})$$

From (dk0) and the definition of the dk operator, we have

$$[constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t) = \bigcup_{i=1}^{n} [constrain(\{\ldots\}_1, S'_i)\downarrow_{\mathcal{CO}}](t) \quad (\mathbf{dk2})$$

From (dk1) and (dk2) we get

$$[S_2](t) = [constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t)$$

Let us now show (P4). By definition of the dk operator, we have $[S_2](t) \mapsto\!\!\!\!\!\rightarrow \bigcup_{i=1}^{n} [S'_i](t)$ and then $[S_2](t)$ terminates iff $[S'_i](t)$ terminates $\forall i \in \{1, \ldots, n\}$ (**dk3**).
From (dk0) and the definition of the dk operator we have :

$$[constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t) \mapsto\!\!\!\!\!\rightarrow \bigcup_{i=1}^{n} [constrain(\{\ldots\}_1, S'_i)\downarrow_{\mathcal{CO}}](t)$$

and then $[constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t)$ terminates iff $[constrain(\{\ldots\}_1, S'_i)\downarrow_{\mathcal{CO}}](t)$ terminates $\forall i \in \{1, \ldots, n\}$ (**dk4**). From (dk3) and (dk4), we get (P4).
If $\mathbf{S_2} = \mathbf{dc}(\mathbf{S'_1}, \ldots, \mathbf{S'_n})$, we have :

$$constrain(\{\ldots\}_1, S_2) \longrightarrow_{\mathcal{CO}} dc(constrain(\{\ldots\}_1, S'_1), \ldots, constrain(\{\ldots\}_1, S'_n)).$$

Let us show the property $(P1)$. By induction hypothesis , for $i \in \{1, \ldots, n\}$, the evaluation of $constrain(\{\ldots\}_1, S'_i)$ with $\mathcal{CO}$ terminates and its normal form is an ELAN strategy.

Therefore $constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}} = dc(constrain(\{\ldots\}_1, S'_1)\downarrow_{\mathcal{CO}}, \ldots, constrain(\{\ldots\}_1, S'_n)\downarrow_{\mathcal{CO}})$ exists and is an ELAN strategy (**dc0**).
Let us now show (P2–P4). By definition of the dc operator, we have :

$$[S_2](t) \twoheadrightarrow \begin{cases} [S'_1](t) & \text{if } [S'_1](t) \neq \emptyset \\ \vdots & \\ [S'_n](t) & \text{if } [S'_n](t) \neq \emptyset \\ \emptyset & \text{if } \bigcup_{i=1}^{n} [S'_i](t) = \emptyset \end{cases} \quad (\mathbf{dc1})$$

From ($dc0$) and the definition of the dc operator, we get :

$$[constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t) \twoheadrightarrow \begin{cases} [constrain(\{\ldots\}_1, S'_1)\downarrow_{\mathcal{CO}}](t) & \text{if } [constrain(\{\ldots\}_1, S'_1)\downarrow_{\mathcal{CO}}](t) \neq \emptyset \\ \vdots & \\ [constrain(\{\ldots\}_1, S'_n)\downarrow_{\mathcal{CO}}](t) & \text{if } [constrain(\{\ldots\}_1, S'_n)\downarrow_{\mathcal{CO}}](t) \neq \emptyset \\ \emptyset & \text{if } \bigcup_{i=1}^{n} [constrain(\{\ldots\}_1, S'_i)\downarrow_{\mathcal{CO}}](t) = \emptyset \end{cases}$$

By induction hypothesis , we have $\forall i \in \{1, \ldots, n\} : [constrain(\{\ldots\}_1, S'_i)\downarrow_{\mathcal{CO}}](t) = [S'_i](t)$,
and therefore :

$$[constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t) \twoheadrightarrow \begin{cases} [S'_1](t) & \text{if } [S'_1](t) \neq \emptyset \\ \vdots & \\ [S'_n](t) & \text{if } [S'_n](t) \neq \emptyset \\ \emptyset & \text{if } \bigcup_{i=1}^{n} [S'_i](t) = \emptyset \end{cases} \quad (\mathbf{dc2})$$

From ($dc1$) and ($dc2$), we infer properties (P2–P4).
If $\mathbf{S_2} = \mathbf{first}(\mathbf{S'_1}, \ldots, \mathbf{S'_n})$, we have :

$$constrain(\{\ldots\}_1, S_2) \longrightarrow_{\mathcal{CO}} first(constrain(\{\ldots\}_1, S'_1), \ldots, constrain(\{\ldots\}_1, S'_n)).$$

Let us show the property ($P1$). By induction hypothesis , for $i \in \{1, \ldots, n\}$, the evaluation of $constrain(\{\ldots\}_1, S'_i)$ with $\mathcal{CO}$ terminates and its normal form is an ELAN strategy.
Therefore $constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}} = first(constrain(\{\ldots\}_1, S'_1)\downarrow_{\mathcal{CO}}, \ldots, constrain(\{\ldots\}_1, S'_n)\downarrow_{\mathcal{CO}})$ exists and is an ELAN strategy (**first0**).
Let us now show (P2–P4). By definition of the $first$ operator, we have :

$$[S_2](t) \twoheadrightarrow \begin{cases} [S'_1](t) & \text{if } [S'_1](t) \neq \emptyset \\ \vdots & \\ [S'_n](t) & \text{if } \bigcup_{i=1}^{n-1} [S'_i](t) = \emptyset \text{ and } [S'_n](t) \neq \emptyset \\ \emptyset & \text{if } \bigcup_{i=1}^{n} [S'_i](t) = \emptyset \end{cases} \quad (\mathbf{first1})$$

From ($first0$) and the definition of the $first$ operator, we get :

$$[constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t) \twoheadrightarrow \begin{cases} [constrain(\{\ldots\}_1, S'_1)\downarrow_{\mathcal{CO}}](t) & \text{if } [constrain(\{\ldots\}_1, S'_1)\downarrow_{\mathcal{CO}}](t) \neq \emptyset \\ \vdots & \\ [constrain(\{\ldots\}_1, S'_n)\downarrow_{\mathcal{CO}}](t) & \text{if } \bigcup_{i=1}^{n-1} [constrain(\{\ldots\}_1, S'_i)\downarrow_{\mathcal{CO}}](t) = \emptyset \\ & \text{and } [constrain(\{\ldots\}_1, S'_n)\downarrow_{\mathcal{CO}}](t) \neq \emptyset \\ \emptyset & \text{if } \bigcup_{i=1}^{n} [constrain(\{\ldots\}_1, S'_i)\downarrow_{\mathcal{CO}}](t) = \emptyset \end{cases}$$

By induction hypothesis , we have $\forall i \in \{1, \ldots, n\} : [constrain(\{\ldots\}_1, S'_i)\downarrow_{\mathcal{CO}}](t) = [S'_i](t)$,
and therefore :

$$[constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t) \twoheadrightarrow \begin{cases} [S'_1](t) & \text{if } [S'_1](t) \neq \emptyset \\ \vdots & \\ [S'_n](t) & \text{if } \bigcup_{i=1}^{n-1} [S'_i](t) = \emptyset \text{ and } [S'_n](t) \neq \emptyset \\ \emptyset & \text{if } \bigcup_{i=1}^{n} [S'_i](t) = \emptyset \end{cases} \quad (\mathbf{first2})$$

From ($first1$) and ($first2$), we infer properties (P2–P4).
If $\mathbf{S_2} = \mathbf{f}(\mathbf{S'_1}, \ldots, \mathbf{S'_n})$, with $f \in \mathcal{F}$, we have :

$$constrain(\{\ldots\}_1, S_2) \longrightarrow_{\mathcal{CO}} S_2$$

and Lemma 2 trivially holds.
If $\mathbf{S_2} = \mathbf{repeat}^*(\mathbf{S'_1})$, we have two cases :

- either $constrain(\{\ldots\}_1, S'_1) \longrightarrow^*_{\mathcal{CO}} fail$. In this case, $constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}} = id$, hence (P1). By induction hypothesis , we have $[S'_1](t) = \emptyset$, and then $[repeat^*(S'_1)](t) = \{t\} = [constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t)$, and both $[S_2](t)$ and $[constrain(\{\ldots\}_1, S_2)\downarrow_{\mathcal{CO}}](t)$ terminate ;
- or $constrain(\{\ldots\}_1, S_2) \longrightarrow_{\mathcal{CO}} S_2$, and then Lemma 2 trivially holds.

□

A.3 Exclusive strategies (Section 4.3)

Lemma 3. *Let S, S' be two strategies. We have :*

1. *The reduction of $mutex(S, S')$ with $\mathcal{E}xcl$ terminates and its normal form is either $true$ or $\sharp$.*
2. *If $mutex(S, S') \hookrightarrow^*_{\mathcal{E}xcl} true$, then S, S' are mutually exclusive.*

Proof: The proof is rather long and fastidious, because we proceed by structural induction on the strategies S and S'. Since we study 8 kinds of strategy, we have to perform a proof for 64 cases. Fortunately, some cases can be factorized.

1. **Case $\mathbf{S} = \mathbf{id}$ and $\mathbf{S'} = \mathbf{fail}$.**
   Then $mutex(S, S') \longrightarrow_{\mathcal{E}xcl} true$. We have $\forall t \in \mathcal{T}(\mathcal{F}) : [S'](t) = \emptyset$, therefore S' trivially excludes S, and S trivially excludes S'.
2. **Case $\mathbf{S} = \mathbf{id}$ and $\mathbf{S'} \neq \mathbf{fail}$.**
   Then $mutex(id, S') \longrightarrow_{\mathcal{E}xcl} false$, and Lemma 3 trivially holds.
3. **Case $\mathbf{S} = \{\ldots\}_\mathbf{1}$ and $\mathbf{S'} = \{\ldots\}_\mathbf{2}$.**
   Then $mutex(S, S') \longrightarrow_{\mathcal{E}xcl} \bigwedge_{(l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2) \in \{\ldots\}_1 \times \{\ldots\}_2} mutex(l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2)$.
   By definition of $\mathcal{E}xcl$, $\forall (l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2) \in \{\ldots\}_1 \times \{\ldots\}_2 : mutex(l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2) \longrightarrow_{\mathcal{E}xcl} true$ or $false$. Therefore the point 1 of Lemma 3 holds.
   If $\exists (l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2) \in \{\ldots\}_1 \times \{\ldots\}_2 : mutex(l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2) \longrightarrow_{\mathcal{E}xcl} false$, then $mutex(S, S') \longrightarrow^*_{\mathcal{E}xcl} false$, and point 2 is trivial.
   Let us now consider the case $mutex(S, S') \longrightarrow^*_{\mathcal{E}xcl} true$, i.e. $\forall (l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2) \in \{\ldots\}_1 \times \{\ldots\}_2 : mutex(l_1 \to r_1 \text{ if } c_1, l_2 \to r_2 \text{ if } c_2) \longrightarrow_{\mathcal{E}xcl} true(\mathbf{m0})$.
   Let $t \in \mathcal{T}(\mathcal{F})$, and show that S excludes S'.
   If $\nexists E \neq \emptyset : [S](t) \mapsto\!\!\!\!\!\rightarrow^* E$, then S trivially excludes S' on t.
   Let us now assume $\exists E \neq \emptyset : [S](t) \mapsto\!\!\!\!\!\rightarrow^* E$. By definition, we have

$$[S'](t) \mapsto\!\!\!\!\!\rightarrow \bigcup_{(l_2 \to r_2 \text{ if } c_2) \in \{\ldots\}_2} [l_2 \to r_2 \text{ if } c_2](t)$$

   and therefore the evaluation of $[S'](t)$ terminates. Let us now show, by contradiction, that $[S'](t) = \emptyset$. We then assume $[S'](t) \neq \emptyset$. We then have $\exists (l_2 \to r_2 \text{ if } c_2) \in \{\ldots\}_2 : [l_2 \to r_2 \text{ if } c_2](t) \neq \emptyset$. By definition of the conditional rewriting, $\exists \alpha_2 : Dom(\alpha_2) = \mathcal{V}ar(l_2) \wedge \alpha_2 l_2 = t \wedge \alpha_2 c_2 \longrightarrow^* true$.
   We are going to show that $\forall (l_1 \to r_1 \text{ if } c_1) \in \{\ldots\}_1 : [l_1 \to r_1 \text{ if } c_1](t) = \emptyset$, which entails that $S(t) \mapsto\!\!\!\!\!\rightarrow^* \emptyset$, which raises a contradiction.

Let $(l_1 \to r_1 \text{ if } c_1) \in \{\ldots\}_1$. From $(m0)$ and the definition of $\mathcal{E}xcl$, $/existsμ = mgu(l_1, l_2) : \mu c_1 \longrightarrow^* true \wedge \mu c_2 \longrightarrow^* true$(**m1**). Let us assume $[l_1 \to r_1 \text{ if } c_1](t) \neq \emptyset$. Then, by definition of the conditional rewriting, $\exists \alpha_1 : Dom(\alpha_1) = \mathcal{V}ar(l_1) \wedge \alpha_1 l_1 = t \wedge \alpha_1 c_1 \longrightarrow^* true$. Considering the substitution $\mu = \alpha_1 \wedge \alpha_2$, we have :

- $\mathcal{V}ar(l_1) \cap \mathcal{V}ar(l_2) = \emptyset$, therefore $\mu l_1 = (\alpha_1 \wedge \alpha_2) l_1 = \alpha_1 l_1 = t$. Symetrically, we get $\mu l_2 = \alpha_2 l_2 = t$, and hence $\mu l_1 = \mu l_2$;
- $\mathcal{V}ar(c_1) \subseteq \mathcal{V}ar(l_1)$, therefore $\mu c_1 = (\alpha_1 \wedge \alpha_2) c_1 = \alpha_1 c_1 \longrightarrow^* true$;
- $\mathcal{V}ar(c_2) \subseteq \mathcal{V}ar(l_2)$, therefore $\mu c_2 = (\alpha_1 \wedge \alpha_2) c_2 = \alpha_2 c_2 \longrightarrow^* true$.

The three points above raise a contradiction with $(m1)$, and then $[l_1 \to r_1 \text{ if } c_1](t) = \emptyset \forall (l_1 \to r_1 \text{ if } c_1) \in \{\ldots\}_1$, which contradicts $[S](t) \mapsto\!\!\!\!\twoheadrightarrow^* E \neq \emptyset$.

Consequently, $[S'](t) \neq \emptyset$ is impossible and, since the evaluation of $[S'](t)$ terminates, we get $[S'](t) = \emptyset$.

Symetrically, we can show that S' excludes S.

4. **Case $\mathbf{S = op(S_1, \ldots, S_m)}$ and $\mathbf{S'}$ any strategy**, with $op \in \{dk, dc, first\}$ and assuming Lemma 3 holds for each pair of strategies (S_i, S'), $i \in \{1, \ldots, n\}$.

   Then $mutex(S, S') \longrightarrow_{\mathcal{E}xcl} \bigwedge_{i=1}^{m} mutex(S_i, S')$.

   By induction hypothesis , $\forall i \in \{1, \ldots, m\} : mutex(S_i, S') \downarrow_{\mathcal{E}xcl}$ exists and is either $true$ or $false$. Therefore the evaluation of $mutex(S, S')$ with $\mathcal{E}xcl$ terminates and is either $true$ or $false$.

   If its normal form is $false$, then Lemma 3 trivially holds.

   Let us then assume that its normal form is $true$.

   Then we have $\forall i \in \{1, \ldots, m\} : mutex(S_i, S') \longrightarrow^*_{\mathcal{E}xcl} true$, and by induction hypothesis S_i and S' are mutually exclusive (**mdk0**).

   Let $t \in \mathcal{T}(\mathcal{F})$. We first show that S excludes S'.

   If $\nexists E \neq \emptyset : [S](t) \mapsto\!\!\!\!\twoheadrightarrow^* E$, then S trivially excludes S' on t.

   Let us now assume $\exists E \neq \emptyset : [S](t) \mapsto\!\!\!\!\twoheadrightarrow^* E$.

   By definition of the op operator, we have :

   - $[S](t) \mapsto\!\!\!\!\twoheadrightarrow \bigcup_{i=1}^{m} [S_i](t) \neq \emptyset$ if $op = dk$.
     By hypothesis, $E \neq \emptyset$, hence $\exists i \in \{1, \ldots, m\}, E_i : [S_i](t) \mapsto\!\!\!\!\twoheadrightarrow^* E_i \neq \emptyset$.
   - $\exists i \in \{1, \ldots, m\} : [S](t) \mapsto\!\!\!\!\twoheadrightarrow [S_i](t)$ if $op \in \{dc, first\}$.
     By hypothesis, $E \neq \emptyset$, hence $[S_i](t) \mapsto\!\!\!\!\twoheadrightarrow^* E_i \neq \emptyset$.

   From $[S_i](t) \mapsto\!\!\!\!\twoheadrightarrow^* E_i \neq \emptyset$ and $(mdk0)$, we get $[S'](t)$ terminates and evaluates in $\emptyset$.

   Let us now show that S' excludes S.

   If $\nexists E \neq \emptyset : [S'](t) \mapsto\!\!\!\!\twoheadrightarrow^* E$, then S' trivially excludes S on t.

   Let us now assume $\exists E \neq \emptyset : [S'](t) \mapsto\!\!\!\!\twoheadrightarrow^* E$. From $(mdk0)$, we get that $\forall i \in \{1, \ldots, m\} : [S_i](t)$ terminates and evaluates in $\emptyset$. Then, by definition of the op operator, we get : $[S](t) \mapsto\!\!\!\!\twoheadrightarrow \bigcup_{i=1}^{m} [S_i](t) \mapsto\!\!\!\!\twoheadrightarrow^* \emptyset$.

5. **Case $\mathbf{S = f(S_1, \ldots, S_m)}$ and $\mathbf{S' = \{\ldots\}_2}$**, with $f \in \mathcal{F}$.

   If $\exists (l_2 \to r_2 \text{ if } c_2) \in \{\ldots\}_2 : top(l_2) = f$, then $mutex(S, S') \longrightarrow_{\mathcal{E}xcl} false$, and Lemma 3 trivially holds.

   If $\nexists (l_2 \to r_2 \text{ if } c_2) \in \{\ldots\}_2 : top(l_2) = f$, then $mutex(S, S') \longrightarrow_{\mathcal{E}xcl} true$.

   Let $t \in \mathcal{T}(\mathcal{F})$. Let us first show that S excludes S'. If $\nexists E \neq \emptyset : [S](t) \mapsto\!\!\!\!\twoheadrightarrow^* E$, then S trivially excludes S' on t.

   Let us now assume $\exists E \neq \emptyset : [S](t) \mapsto\!\!\!\!\twoheadrightarrow^* E$. Necessarily, by definition of the congruence strategy, we have $top(t) = f$. Since no rule of $\{\ldots\}_2$ has the top symbol of its lhs equal to f, we have $[S'](t) = \emptyset$.

   Let us now show that S' excludes S.

   If $\nexists E \neq \emptyset : [S'](t) \mapsto\!\!\!\!\twoheadrightarrow^* E$, then S' trivially excludes S on t.

   Let us now assume $\exists E \neq \emptyset : [S'](t) \mapsto\!\!\!\!\twoheadrightarrow^* E$. From $[S'](t) \mapsto\!\!\!\!\twoheadrightarrow^* \bigcup_{(l_2 \to r_2 \text{ if } c_2) \in \{\ldots\}_2} [l_2 \to r_2 \text{ if } c_2](t)$ and $E \neq \emptyset$, we infer the existence of a rule $l_2 \to r_2$ if $c_2) \in \{\ldots\}_2$ that does not fail on t. Since $top(l_2) \neq f$, then $top(t) \neq f$ and then, by definition of the congruence strategy, $[S](t) = \emptyset$.

6. **Case $\mathbf{S = f(S_1, \ldots, S_n)}$ and $\mathbf{S' = g(S'_1, \ldots, S'_m)}$**, with $f, g \in \mathcal{F}$ and assuming Lemma 3 holds for each pair of strategies (S_i, S'_j), $(i, j) \in \{1, \ldots, n\} \times \{1, \ldots, m\}$.
   If $f \neq g$, then $mutex(S, S') \longrightarrow_{\mathcal{E}xcl} true$.
   Let us first show that S excludes S'. Given $t \in \mathcal{T}(\mathcal{F})$, $[S](t) \mapsto\!\!\!\!\!\rightarrow^* E \neq \emptyset$ implies that $top(t) = f$, and hence, by definition of the congruence strategy, we have $[S'](t) = \emptyset$. We can symetrically show that S' excludes S.
   Let us now consider the case where $f = g$.
   Then $mutex(S, S') \longrightarrow_{\mathcal{E}xcl} \bigvee_{i=1}^{n} mutex(S_i, S'_i)$. By induction hypothesis , the normal form of each $mutex(S_i, S'_i)$ exists and is either $true$ or $false$, and hence point 1 of Lemma 3 holds.
   If $\forall i \in \{1, \ldots, n\} : mutex(S_i, S'_i)\downarrow_{\mathcal{E}xcl} = false$, then $mutex(S, S') \longrightarrow^*_{\mathcal{E}xcl} false$, and Lemma 3 trivially holds.
   Let us now assume $\exists i \in \{1, \ldots, n\} : mutex(S_i, S'_i)\downarrow_{\mathcal{E}xcl} = true$. By induction hypothesis , strategies S_i, S'_i are mutually exclusive.
   Let $t \in \mathcal{T}(\mathcal{F})$. Let us show that S excludes S' on t.
   If $\nexists E \neq \emptyset : [S](t) \mapsto\!\!\!\!\!\rightarrow^* E$, then S trivially excludes S' on t.
   Let us now assume $\exists E \neq \emptyset : [S](t) \mapsto\!\!\!\!\!\rightarrow^* E$. Necessarily, denoting n the arity of symbol f, we have $\exists t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}) : t = f(t_1, \ldots, t_n)$ and $\exists E_i \neq \emptyset : [S_i](t_i) \mapsto\!\!\!\!\!\rightarrow^* E_i$. Then, by induction hypothesis, $[S'_i](t)$ terminates and evaluates in $\emptyset$, and therefore $[S'](t) = \emptyset$. We can symetrically show that S' excludes S on t.
7. **Case $\mathbf{S = f(S_1, \ldots, S_n)}$ and S neither a set of rules nor a congruence strategy**, with $f \in \mathcal{F}$.
   Then $mutex(S, S') \longrightarrow_{\mathcal{E}xcl} false$, and Lemma 3 trivially holds.
8. **Case $\mathbf{S = S_1; S_2}$ and $\mathbf{S'}$ any strategy**, assuming Lemma 3 holds for the pair of strategies (S_1, S').
   Then $mutex(S, S') \longrightarrow_{\mathcal{E}xcl} mutex(S_1, S')$. By induction hypothesis , $mutex(S_1, S')\downarrow_{\mathcal{E}xcl}$ exists and is either $true$ or $false$, therefore so is $mutex(S, S')\downarrow_{\mathcal{E}xcl}$.
   If $mutex(S_1, S')\downarrow_{\mathcal{E}xcl} = false$, then Lemma 3 trivially holds.
   If $FNar\mathcal{E}xclmutex(S_1, S') = true$ then, by induction hypothesis , S_1 and S' are mutually exclusive.
   Let $t \in \mathcal{T}(\mathcal{F})$. Let us first show that S excludes S'.
   If $\nexists existsE \neq \emptyset : [S](t) \mapsto\!\!\!\!\!\rightarrow^* E$, then S trivially excludes S'.
   Let us now assume $existsE \neq \emptyset : [S](t) \mapsto\!\!\!\!\!\rightarrow^* E$. Then, in particular, $\exists E_1 \neq \emptyset : [S_1](t) \mapsto\!\!\!\!\!\rightarrow^* E_1$. Since S_1 and S' are mutually exclusive, we get $[S'](t) = \emptyset$.
   Let us now show that S' excludes S.
   If $\nexists existsE \neq \emptyset : [S'](t) \mapsto\!\!\!\!\!\rightarrow^* E$, then S' trivially excludes S.
   Let us now assume $existsE \neq \emptyset : [S'](t) \mapsto\!\!\!\!\!\rightarrow^* E$. Since S_1 and S' are mutually exclusive, we get $[S_1](t) = \emptyset$, and then $[S](t) = \emptyset$.
9. **Case $\mathbf{S = repeat^*(S_1)}$ and $\mathbf{S' = fail}$.**
   Then $mutex(S, S') \longrightarrow_{\mathcal{E}xcl} true$. We have $\forall t \in \mathcal{T}(\mathcal{F}) : [S'](t) = \emptyset$, therefore S' trivially excludes S, and S trivially excludes S'.
10. **Case $\mathbf{S = repeat^*(S_1)}$ and $\mathbf{S' \neq fail}$.**
    Then $mutex(id, S') \longrightarrow_{\mathcal{E}xcl} false$, and Lemma 3 trivially holds.
11. **Case $\mathbf{S = \{\ldots\}_1}$ and $\mathbf{S' \neq \{\ldots\}_2}$.**
    Then $mutex(\{\ldots\}_1, S') \longrightarrow_{\mathcal{E}xcl} mutex(S', \{\ldots\}_1)$, and the case $mutex(S', \{\ldots\}_1)$ is covered by the previous cases.

□

Proposition 5. *Let $S_1, \ldots, S_n$ be strategies such that $\forall i, j \in \{1, \ldots, n\}, i \neq j : S_i$ and S_j are mutually exclusive. Then we have :*

1. *$dc(S_1, \ldots, S_n)$ is equivalent to $dk(S_1, \ldots, S_n)$.*

2. $first(S_1, \ldots, S_n)$ *is equivalent to* $dk(S_1, \ldots, S_n)$.

Proof: If $\forall i, j \in \{1, \ldots, n\}, i \neq j : S_i$ and S_j are mutually exclusive, then for any ground term t, there is at most one application of some S_i to t that does not fail.
Then, by definition of the dk operator, we have $[dk(S_1, \ldots, S_n)](t) \mapsto\!\!\!\!\rightarrow [S_i](t)$.
Likewise, by definition of the op operator, for $op \in \{dc, first\}$, we have $[op(S_1, \ldots, S_n)](t) \mapsto\!\!\!\!\rightarrow [S_i](t)$, hence Proposition 5. □

A.4 Equivalence of simplified strategies (Section 4.4)

Theorem 2. *Let* S, S' *be two strategies such that* $S \hookrightarrow^*_{\mathcal{SIMPL}} S'$. *Then* S *and* S' *are equivalent.*

Proof: It is sufficient to prove that the property holds for any rewrite step $S \hookrightarrow_{\mathcal{SIMPL}} S'$. We then perform a case study on these rewrite rules. We refer to the rules by their equation number in Table 5. For each of these rules, S' is the result of applying the rule to the strategy S, and t is any term. To prove equivalence, we show that $[S](t) = [S'](t)$ and that $[S](t)$ and $[S'](t)$ have the same termination behaviour, relying on the semantics of startegy operators given in Table 1.

Rule (1)
$S = fail; S_2$
$S' = fail.$

With the rules of Table 1, we have the reduction $[S](t) \mapsto\!\!\!\!\rightarrow [S_2]([fail](t)) \mapsto\!\!\!\!\rightarrow [S_2](\emptyset) \mapsto\!\!\!\!\rightarrow \emptyset$.
Then $[S](t)$ always terminates and fails, exactly like $S' = fail$. Therefore S and S' are equivalent.

Rule (2)
$S = id; S_2$
$S' = S_2.$

With the rules of Table 1, we have the reduction $[S](t) \mapsto\!\!\!\!\rightarrow [S_2]([id](t)) \mapsto\!\!\!\!\rightarrow [S_2](t)$.
Since $S' = S_2$, S and S' are equivalent.

Rule (3)
$S = S_1; id$
$S' = S_1.$

With the rules of Table 1, we have the reduction $[S](t) \mapsto\!\!\!\!\rightarrow [id]([S_1](t))$.
Obviously, $[S](t)$ terminates iff $[S_1](t)$ terminates, that is iff $[S'](t)$ terminates.
Moreover, for any $t' \in [S](t)$, the above rewriting step ensures that $t' \in [id]([S_1](t))$, i.e. $\exists t'' \in [S_1](t) : t' \in [id](t'')$, that is $t' = t''$, and hence $t' \in [S_1](t)$. We have shown $[S](t) \subseteq [S'](t)$.
Let $t' \in [S_1](t)$. By definition of id, we have $t' \in [id](t')$, and hence $t' \in [id]([S_1](t))$. By the above rewriting step, we get $t' \in [S](t)$. We have then also shown $[S'](t) \subseteq [S](t)$. Finally, S and S' are equivalent.

Rule (4)
$S = repeat^*(fail)$
$S' = id$

Trivially true, by definition of the $repeat^*$ operator.

Rule (5)

$$S = dk(S_1, \ldots, S_{k-1}, fail, S_{k+1}, \ldots, S_n)$$
$$S' = dk(S_1, \ldots, S_{k-1}, S_{k+1}, \ldots, S_n).$$

With the rules of Table 1, we have the reduction :
$[S](t) \mapsto\!\!\!\!\twoheadrightarrow \bigcup_{j \neq k} [S_j](t) \cup [fail](t) \mapsto\!\!\!\!\twoheadrightarrow \bigcup_{j \neq k} [S_j](t) \cup \emptyset$.
Since we also have the reduction $[S'](t) \mapsto\!\!\!\!\twoheadrightarrow \bigcup_{j \neq k} [S_j](t)$, then S and S' are equivalent.

Rule (6)

$$S = first(S_1, \ldots, S_{k-1}, fail, S_{k+1}, \ldots, S_n)$$
$$S' = first(S_1, \ldots, S_{k-1}, S_{k+1}, \ldots, S_n).$$

With the rules of Table 1, we have the reduction :

$$[S](t) \mapsto\!\!\!\!\twoheadrightarrow \begin{cases} \emptyset & \text{if } \bigcup_{j \neq k} [S_j](t) \cup [fail](t) = \emptyset \\ [S_1](t) & \text{if } [S_1](t) \neq \emptyset \\ \vdots & \\ [S_{k-1}](t) & \text{if } [S_{k-1}](t) \neq \emptyset \text{ and } \bigcup_{j=1}^{k-2} [S_j](t) = \emptyset \\ \mathbf{[fail](t)} & \mathbf{\text{if } [fail](t) \neq \emptyset \text{ and } \bigcup_{j=1}^{k-1} [S_j](t) = \emptyset} \\ [S_{k+1}](t) & \text{if } [S_{k+1}](t) \neq \emptyset \text{ and } \bigcup_{j=1}^{k-1} [S_j](t) \cup [fail](t) = \emptyset \\ \vdots & \\ [S_n](t) & \text{if } [S_n](t) \neq \emptyset \text{ and } \bigcup_{j=1, j \neq k}^{n-1} [S_j](t) \cup [fail](t) = \emptyset \end{cases}$$

However, the condition $[fail](t) \neq \emptyset$ is always false, whatever the term t. Then the result of the previous rewriting step is equivalent to :

$$\begin{cases} \emptyset & \text{if } \bigcup_{j \neq k} [S_j](t) \cup [fail](t) = \emptyset \\ [S_1](t) & \text{if } [S_1](t) \neq \emptyset \\ \vdots & \\ [S_{k-1}](t) & \text{if } [S_{k-1}](t) \neq \emptyset \text{ and } \bigcup_{j=1}^{k-2} [S_j](t) = \emptyset \\ [S_{k+1}](t) & \text{if } [S_{k+1}](t) \neq \emptyset \text{ and } \bigcup_{j=1}^{k-1} [S_j](t) = \emptyset \\ \vdots & \\ [S_n](t) & \text{if } [S_n](t) \neq \emptyset \text{ and } \bigcup_{j=1, j \neq k}^{n-1} [S_j](t) = \emptyset \end{cases}$$

Since we get the same result as reducing $[S'](t)$ in one step, S and S' are equivalent.

Rule (7)

$$S = dc(S_1, \ldots, S_{k-1}, fail, S_{k+1}, \ldots, S_n)$$
$$S' = dc(S_1, \ldots, S_{k-1}, S_{k+1}, \ldots, S_n).$$

With the rules of Table 1, we have the reduction :

$$[S](t) \mapsto\!\!\!\!\twoheadrightarrow \begin{cases} \emptyset & \text{if } \bigcup_{j \neq k} [S_j](t) \cup [fail](t) = \emptyset \\ [S_1](t) & \text{if } [S_1](t) \neq \emptyset \\ \vdots & \\ [S_{k-1}](t) & \text{if } [S_{k-1}](t) \neq \emptyset \\ \mathbf{[fail](t)} & \mathbf{\text{if } [fail](t) \neq \emptyset} \\ [S_{k+1}](t) & \text{if } [S_{k+1}](t) \neq \emptyset \\ \vdots & \\ [S_n](t) & \text{if } [S_n](t) \neq \emptyset \end{cases}$$

However, the condition $[fail](t) \neq \emptyset$ is always false, whatever the term t. Then the result of the previous rewriting step is equivalent to :

$$\begin{cases} \emptyset & \text{if } \bigcup_{j\neq k}[S_j](t)\cup[fail](t)=\emptyset \\ [S_1](t) & \text{if } [S_1](t)\neq\emptyset \\ \vdots & \\ [S_{k-1}](t) & \text{if } [S_{k-1}](t)\neq\emptyset \\ [S_{k+1}](t) & \text{if } [S_{k+1}](t)\neq\emptyset \\ \vdots & \\ [S_n](t) & \text{if } [S_n](t)\neq\emptyset \end{cases}$$

Since we get the same result as reducing $[S'](t)$ in one step, S and S' are equivalent.

Rule (8)

$$S = first(S_1,\ldots,S_{k-1},id,\ldots)$$
$$S' = first(S_1,\ldots,S_{k-1},id).$$

With the rules of Table 1, we have the reduction :

$$[S](t)\twoheadrightarrow\begin{cases} \emptyset & \text{if } \bigcup_{j\neq k}[S_j](t)\cup[id](t)=\emptyset \\ [S_1](t) & \text{if } [S_1](t)\neq\emptyset \\ \vdots & \\ [S_{k-1}](t) & \text{if } [S_{k-1}](t)\neq\emptyset \text{ and } \bigcup_{j=1}^{k-2}[S_j](t)=\emptyset \\ \mathbf{[id](t)} & \mathbf{\text{if } [id](t)\neq\emptyset \text{ and } \bigcup_{j=1}^{k-1}[S_j](t)=\emptyset} \\ [S_{k+1}](t) & \text{if } [S_{k+1}](t)\neq\emptyset \text{ and } \bigcup_{j=1}^{k-1}[S_j](t)\cup[id](t)=\emptyset \\ \vdots & \\ [S_n](t) & \text{if } [S_n](t)\neq\emptyset \text{ and } \bigcup_{j=1,j\neq k}^{n-1}[S_j](t)\cup[id](t)=\emptyset \end{cases}$$

However, we have $[id](t)=\{t\}\neq\emptyset$ for any term t, and then the condition $[id](t)=\emptyset$ is false. Then the result of the previous rewriting step is equivalent to :

$$\begin{cases} \emptyset & \text{if } \bigcup_{j\neq k}[S_j](t)\cup[id](t)=\emptyset \\ [S_1](t) & \text{if } [S_1](t)\neq\emptyset \\ \vdots & \\ [S_{k-1}](t) & \text{if } [S_{k-1}](t)\neq\emptyset \text{ and } \bigcup_{j=1}^{k-2}[S_j](t)=\emptyset \\ \mathbf{[id](t)} & \mathbf{\text{if } [id](t)\neq\emptyset \text{ and } \bigcup_{j=1}^{k-1}[S_j](t)=\emptyset} \end{cases}$$

Since we get the same result as reducing $[S'](t)$ in one step, S and S' are equivalent.

Rule (9)

$$S = first(S_1,\ldots,S_{k-1},repeat^*(S_k),\ldots)$$
$$S' = first(S_1,\ldots,S_{k-1},repeat^*(S_k)).$$

We proceed like for the rule (8), by noticing that $[repeat^*(S_k)](t)\neq\emptyset$ for any term t.

Rule (10)

$$S = f(S_1,\ldots,S_n);g(S'_1,\ldots,S'_m)$$
$$S' = \begin{cases} \emptyset & \text{if } f\neq g \\ f(S_1;S'_1,\ldots,S_n;S'_n) & \text{if } f=g \end{cases}$$

With the rules of Table 1, we have the reduction :
$[S](t)\twoheadrightarrow\bigcup_{t'\in[f(S_1,\ldots,S_n)](t)}[g(S'_1,\ldots,S'_m)](t')$.
Let us assume $f\neq g$. By definition of the congruence strategy, for any $t'\in[f(S_1,\ldots,S_n)](t)$, we necessarily have $top(t')=f$, and hence $[g(S'_1,\ldots,S'_m)](t')=\emptyset$, hence the equivalence of S and S' if $f\neq g$.
Let us now assume $f=g$. We then have the reduction :

$$[S'](t)\twoheadrightarrow\begin{cases} \emptyset & \text{if } top(t)\neq f \\ \bigcup_{v_1\in[S_1;S'_1](t_1),\ldots,v_n\in[S_n;S'_n](t_n)} f(v_1,\ldots,v_n) & \text{if } t=f(t_1,\ldots,t_n) \end{cases}$$

By definition of the composition, we then have the following reduction :
$$\mapsto\!\!\!\!\twoheadrightarrow \begin{cases} \emptyset & \text{if } top(t) \neq f \\ \bigcup_{v_1 \in \bigcup_{u_1 \in [S_1](t_1)} [S'_1](u_1), \ldots, v_n \in \bigcup_{u_n \in [S_n](t_n)} [S'_n](u_n)} f(v_1, \ldots, v_n) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$
which is equivalent to :
$$\begin{cases} \emptyset & \text{if } top(t) \neq f \\ \bigcup_{u_1 \in [S_1](t_1), \ldots, u_n \in [S_n](t_n)} \bigcup_{v_1 \in [S'_1](u_1), \ldots, v_n \in [S'_n](u_n)} f(v_1, \ldots, v_n) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$
Let us show that $[S](t)$ evaluates in the same application as above. With $f = g$, we have the reduction :
$[S](t) \mapsto\!\!\!\!\twoheadrightarrow \bigcup_{t' \in [f(S_1, \ldots, S_n)](t)} [f(S'_1, \ldots, S'_n)](t')$.
If $top(t) \neq f$, then $[f(S_1, \ldots, S_n)](t) = \emptyset$. Therefore the result of the above rewriting step is equivalent to :
$$\begin{cases} \emptyset & \text{if } top(t) \neq f \\ \bigcup_{t' \in [f(S_1, \ldots, S_n)](t)} [f(S'_1, \ldots, S'_n)](t') & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$
which is, by definition of the congruence strategy, equivalent to :
$$\begin{cases} \emptyset & \text{if } top(t) \neq f \\ \bigcup_{t' \in \bigcup_{u_1 \in [S_1](t_1), \ldots, [S_n](t_n)} f(u_1, \ldots, u_n)} [f(S'_1, \ldots, S'_n)](t') & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$
that is to :
$$\begin{cases} \emptyset & \text{if } top(t) \neq f \\ \bigcup_{u_1 \in [S_1](t_1), \ldots, [S_n](t_n)} [f(S'_1, \ldots, S'_n)](f(u_1, \ldots, u_n)) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$
The above result, by definition of the congruence strategy, is equivalent to :
$$\begin{cases} \emptyset & \text{if } top(t) \neq f \\ \bigcup_{u_1 \in [S_1](t_1), \ldots, u_n \in [S_n](t_n)} \bigcup_{v_1 \in [S'_1](u_1), \ldots, v_n \in [S'_n](u_n)} f(v_1, \ldots, v_n) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$
Therefore S and S' are equivalent.

Rule (11)

$S = dk(S_1, \ldots, S_n); S_0$
$S' = dk(S_1; S_0, \ldots, S_n; S_0)$.

With the rules of Table 1, we have the reduction :
$[S](t)$
$\mapsto\!\!\!\!\twoheadrightarrow [S_0]([dk(S_1, \ldots, S_n)](t))$
$\mapsto\!\!\!\!\twoheadrightarrow [S_0](\bigcup_{i=1}^{n} [S_i](t))$
$\mapsto\!\!\!\!\twoheadrightarrow \bigcup_{i=1}^{n} [S_0]([S_i](t))$ by distributivity of the application of a strategy over the union of strategies.
Besides, we also have the reduction $[S'](t) \mapsto\!\!\!\!\twoheadrightarrow \bigcup_{i=1}^{n} [S_i; S_0](t) \mapsto\!\!\!\!\twoheadrightarrow \bigcup_{i=1}^{n} [S_0]([S_i](t))$. Then $[S](t)$ and $[S'](t)$ both rewrite to the same application, and hence S and S' are equivalent.

Rule (12)

$S = S_0; dk(S_1, \ldots, S_n)$
$S' = dk(S_0; S_1, \ldots, S_0; S_n)$.

With the rules of Table 1, we have the reduction $[S](t) \mapsto\!\!\!\!\twoheadrightarrow [dk(S_1, \ldots, S_n)]([S_0](t)) \mapsto\!\!\!\!\mapsto \bigcup_{i=1}^{n} [S_i]([S_0](t))$.
Likewise, we have the reduction $[S'](t) \mapsto\!\!\!\!\twoheadrightarrow \bigcup_{i=1}^{n} [S_0; S_i](t) \mapsto\!\!\!\!\twoheadrightarrow \bigcup_{i=1}^{n} [S_i]([S_0](t))$. Then $[S](t)$ and $[S'](t)$ both rewrite to the same application, and hence S and S' are equivalent.

Rule (13)

$S = dk(S_1)$
$S' = S_1$

With the rules of Table 1, we have the reduction $[S](t) \mapsto\!\!\!\!\twoheadrightarrow [S_1](t)$, hence S and S' are equivalent.

Rule (14)

$$S = first(S_1)$$
$$S' = S_1$$

With the rules of Table 1, we have the reduction :

$$[S](t) \mapsto\!\!\!\!\rightarrow \begin{cases} \emptyset & \text{if } [S_1](t) = \emptyset \\ [S_1](t) & \text{if } [S_1](t) \neq \emptyset. \end{cases}$$

Therefore S and S' are equivalent.

Rule (15)

$$S = dc(S_1)$$
$$S' = S_1$$

Same as rule (14).

Rule (16)

$$S = dk(S_1, \ldots, S_{k-1}, dk(S'_1, \ldots, S'_m), S_{k+1}, \ldots, S_n)$$
$$S' = dk(S_1, \ldots, S_{k-1}, S'_1, \ldots, S'_m, S_{k+1}, \ldots, S_n)$$

With the rules of Table 1, we have the reduction :

$[S](t) \mapsto\!\!\!\!\rightarrow [dk(S'_1, \ldots, S'_m)](t) \bigcup_{i=1, i\neq k}^{n} [S_i](t) \mapsto\!\!\!\!\rightarrow \bigcup_{i=1}^{m} [S'_i](t) \bigcup_{i=1, i\neq k}^{n} [S_i](t)$.

Since we get the same result as reducing $[S'](t)$ in one step, S and S' are equivalent.

Rule (17)

$$S = first(S_1, \ldots, S_{k-1}, first(S'_1, \ldots, S'_m), S_{k+1}, \ldots, S_n)$$
$$S' = first(S_1, \ldots, S_{k-1}, S'_1, \ldots, S'_m, S_{k+1}, \ldots, S_n)$$

With the rules of Table 1, we have the reduction :

$$[S](t)$$
$$\mapsto\!\!\!\!\rightarrow \begin{cases} \emptyset & \text{if } [first(S'_1, \ldots, S'_m)](t) \bigcup_{i=1, i\neq k}^{n} [S_i](t) = \emptyset \\ [S_1](t) & \text{if } [S_1](t) \neq \emptyset \\ \vdots & \\ [S_{k-1}](t) & \text{if } [S_{k-1}](t) \neq \emptyset \text{ and } \bigcup_{i=1}^{k-2} [S_i](t) = \emptyset \\ [\mathbf{first}(\mathbf{S'_1}, \ldots, \mathbf{S'_m})](\mathbf{t}) & \text{if } [first(S'_1, \ldots, S'_m)](t) \neq \emptyset \text{ and } \bigcup_{i=1}^{k-1} [S_i](t) = \emptyset \\ [S_{k+1}](t) & \text{if } [S_{k+1}](t) \neq \emptyset \text{ and } [first(S'_1, \ldots, S'_m)](t) \bigcup_{i=1}^{k-1} [S_i](t) = \emptyset \\ \vdots & \\ [S_n](t) & \text{if } [S_n](t) \neq \emptyset \text{ and } [first(S'_1, \ldots, S'_m)](t) \bigcup_{i=1, i\neq k}^{n-1} [S_i](t) = \emptyset \end{cases}$$

By reducing $[first(S'_1, \ldots, S'_m)](t)$ with the rules of Table 1, we get :

$$\mapsto\!\!\!\!\rightarrow \begin{cases} \emptyset & \text{if } [\mathbf{first}(\mathbf{S'_1}, \ldots, \mathbf{S'_m})](\mathbf{t}) \bigcup_{i=1, i\neq k}^{n} [S_i](t) = \emptyset \\ [S_1](t) & \text{if } [S_1](t) \neq \emptyset \\ \vdots & \\ [S_{k-1}](t) & \text{if } [S_{k-1}](t) \neq \emptyset \text{ and } \bigcup_{i=1}^{k-2} [S_i](t) = \emptyset \\ \emptyset & \text{if } \bigcup_{\mathbf{i=1}}^{\mathbf{m}} [\mathbf{S'_i}](\mathbf{t}) = \emptyset \textbf{ and } [\mathbf{first}(\mathbf{S'_1}, \ldots, \mathbf{S'_m})](\mathbf{t}) \neq \emptyset \text{ and } \bigcup_{i=1}^{k-1} [S_i](t) = \emptyset \\ [S'_j](t) & \text{if } [\mathbf{S'_j}](\mathbf{t}) \neq \emptyset \text{ and } \bigcup_{i=1}^{j-1} [S'_i](t) = \emptyset \text{ and } [\mathbf{first}(\mathbf{S'_1}, \ldots, \mathbf{S'_m})](\mathbf{t}) \neq \emptyset \text{ and } \bigcup_{i=1}^{k-1} [S_i](t) = \emptyset \\ [S_{k+1}](t) & \text{if } [S_{k+1}](t) \neq \emptyset \text{ and } [\mathbf{first}(\mathbf{S'_1}, \ldots, \mathbf{S'_m})](\mathbf{t}) \bigcup_{i=1}^{k-1} [S_i](t) = \emptyset \\ \vdots & \\ [S_n](t) & \text{if } [S_n](t) \neq \emptyset \text{ and } [\mathbf{first}(\mathbf{S'_1}, \ldots, \mathbf{S'_m})](\mathbf{t}) \bigcup_{i=1, i\neq k}^{n-1} [S_i](t) = \emptyset \end{cases}$$

By noticing that :

- the condition $\bigcup_{\mathbf{i=1}}^{\mathbf{m}} [\mathbf{S'_i}](\mathbf{t}) = \emptyset$ **and** $[\mathbf{first}(\mathbf{S'_1}, \ldots, \mathbf{S'_m})](\mathbf{t}) \neq \emptyset$ is necessarily false ;
- since $[S'_j](t) \neq \emptyset \Rightarrow [first(S'_1, \ldots, S'_m)](t) \neq \emptyset$, the condition $[\mathbf{S'_j}](\mathbf{t}) \neq \emptyset$ **and** $[\mathbf{first}(\mathbf{S'_1}, \ldots, \mathbf{S'_m})](\mathbf{t}) \neq \emptyset$ is equivalent to the condition $[S'_j](t) \neq \emptyset$;

– the condition $[\mathbf{first}(\mathbf{S'_1},\ldots,\mathbf{S'_m})](\mathbf{t}) = \emptyset$ is equivalent to the condition $\bigcup_{i=1}^{m}[S'_i](t) = \emptyset$,

we get that the previous rewriting result is equivalent to :

$$\begin{cases} \emptyset & \text{if } \bigcup_{i=1}^{m}[S'_i](t)\bigcup_{i=1,i\neq k}^{n}[S_i](t) = \emptyset \\ [S_1](t) & \text{if } [S_1](t) \neq \emptyset \\ \vdots & \\ [S_{k-1}](t) & \text{if } [S_{k-1}](t) \neq \emptyset \text{ and } \bigcup_{i=1}^{k-2}[S_i](t) = \emptyset \\ [S'_j](t) & \text{if } [S'_j](t) \neq \emptyset \text{ and } \bigcup_{i=1}^{j-1}[S'_i](t) = \emptyset \text{ and } \bigcup_{i=1}^{k-1}[S_i](t) = \emptyset \\ [S_{k+1}](t) & \text{if } [S_{k+1}](t) \neq \emptyset \text{ and } \bigcup_{i=1}^{m}[S'_i](t)\bigcup_{i=1}^{k-1}[S_i](t) = \emptyset \\ \vdots & \\ [S_n](t) & \text{if } [S_n](t) \neq \emptyset \text{ and } \bigcup_{i=1}^{m}[S'_i](t)\bigcup_{i=1,i\neq k}^{n-1}[S_i](t) = \emptyset \end{cases}$$

Since we get the same result as reducing $[S'](t)$ in one step, S and S' are equivalent.

Rule (18)

$$S = dc(S_1,\ldots,S_{k-1},dc(S'_1,\ldots,S'_m),S_{k+1},\ldots,S_n)$$
$$S' = dc(S_1,\ldots,S_{k-1},S'_1,\ldots,S'_m,S_{k+1},\ldots,S_n)$$

With the rules of Table 1, we have the reduction :

$$[S](t) \rightarrow \begin{cases} \emptyset & \text{if } [dc(S'_1,\ldots,S'_m)](t)\bigcup_{i=1,i\neq k}^{n}[S_i](t) = \emptyset \\ [S_j](t) & \text{if } [S_j](t) \neq \emptyset \\ [\mathbf{dc}(\mathbf{S'_1},\ldots,\mathbf{S'_m})](\mathbf{t}) & \text{if } [dc(S'_1,\ldots,S'_m)](t) \neq \emptyset \end{cases}$$

By reducing $[dc(S'_1,\ldots,S'_m)](t)$ with the rules of Table 1, we get :

$$\rightarrow \begin{cases} \emptyset & \text{if } [\mathbf{dc}(\mathbf{S'_1},\ldots,\mathbf{S'_m})](\mathbf{t})\bigcup_{i=1,i\neq k}^{n}[S_i](t) = \emptyset \\ [S_j](t) & \text{if } [S_j](t) \neq \emptyset \\ \emptyset & \text{if } \bigcup_{\mathbf{i=1}}^{\mathbf{m}}[\mathbf{S'_i}](=)\emptyset \textbf{ and } [\mathbf{dc}(\mathbf{S'_1},\ldots,\mathbf{S'_m})](\mathbf{t}) \neq \emptyset \\ [S'_j](t) & \text{if } [\mathbf{S'_j}](\mathbf{t}) \neq \emptyset \textbf{ and } [\mathbf{dc}(\mathbf{S'_1},\ldots,\mathbf{S'_m})](\mathbf{t}) \neq \emptyset \end{cases}$$

By noticing that :

– the condition $\bigcup_{\mathbf{i=1}}^{\mathbf{m}}[\mathbf{S'_i}](\mathbf{t}) = \emptyset$ **and** $[\mathbf{dc}(\mathbf{S'_1},\ldots,\mathbf{S'_m})](\mathbf{t}) \neq \emptyset$ is necessarily false ;

– since $[S'_j](t) \neq \emptyset \Rightarrow [dc(S'_1,\ldots,S'_m)](t) \neq \emptyset$, the condition $[\mathbf{S'_j}](\mathbf{t}) \neq \emptyset$ **and** $[\mathbf{dc}(\mathbf{S'_1},\ldots,\mathbf{S'_m})](\mathbf{t}) \neq \emptyset$ is equivalent to the condition $[S'_j](t) \neq \emptyset$;

– the condition $[\mathbf{dc}(\mathbf{S'_1},\ldots,\mathbf{S'_m})](\mathbf{t}) = \emptyset$ is equivalent to the condition $\bigcup_{i=1}^{m}[S'_i](t) = \emptyset$,

we get that the previous rewriting result is equivalent to :

$$\begin{cases} \emptyset & \text{if } \bigcup_{i=1}^{m}[S'_i](t)\bigcup_{i=1,i\neq k}^{n}[S_i](t) = \emptyset \\ [S_j](t) & \text{if } [S_j](t) \neq \emptyset \\ [S'_j](t) & \text{if } [S'_j](t) \neq \emptyset \end{cases}$$

Since we get the same result as reducing $[S'](t)$ in one step, S and S' are equivalent.

Rule (19)

$$S = dk(\{\ldots\}_1,\ldots,\{\ldots\}_n)$$
$$S' = \bigcup_{i=1}^{n}\{\ldots\}_i$$

With the rules of Table 1, we have the reductions :
$[S](t) \twoheadrightarrow \bigcup_{i=1}^{n}[\{\ldots\}_i](t)$ and $[S'](t) \twoheadrightarrow [\bigcup_{i=1}^{n}\{\ldots\}_i](t)$. Obviously, S and S' are equivalent.

Rule (20)

$$S = f(\{\ldots\}_1,\ldots,\{\ldots\}_n)$$
$$S' = \bigcup_{(l_1\rightarrow r_1 \text{ if } c_1)\in\{\ldots\}_1,\ldots,(l_n\rightarrow r_n \text{ if } c_n)\in\{\ldots\}_n}\{f(l_1,\ldots,l_n)\rightarrow f(r_1,\ldots,r_n) \text{ if } \bigwedge_{i=1}^{n}c_i\}$$

With the rules of Table 1, we have the reduction :

$$[S](t) \twoheadrightarrow \begin{cases} \emptyset & \text{if } top(t) \neq f \\ \bigcup_{u_1\in[\{\ldots\}_1](t_1),\ldots,u_n\in[\{\ldots\}_n](t_n)} f(u_1,\ldots,u_n) & \text{if } t = f(t_1,\ldots,t_n) \end{cases}$$

By definition of application of labelled rewrite rules, the result above is equivalent to :

$$\begin{cases} \emptyset & \text{if } top(t) \neq f \\ \bigcup_{\substack{u_1 \in \{\alpha_1 r_1 | (l_1 \to r_1 \text{ if } c_1) \in \{\ldots\}_1, \alpha_1 l_1 = \alpha_1 t_1, \alpha_1 c_1 \longrightarrow^* true\}, \\ \ldots, \\ u_n \in \{\alpha_n r_n | (l_n \to r_n \text{ if } c_n) \in \{\ldots\}_n, \alpha_n l_n = \alpha_n t_n, \alpha_n c_n \longrightarrow^* true\}}} & f(u_1, \ldots, u_n) \quad \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

that is, more simply, to :

$$\begin{cases} \emptyset & \text{if } top(t) \neq f \\ \bigcup_{\substack{(l_1 \to r_1 \text{ if } c_1) \in \{\ldots\}_1, \alpha_1 : \alpha_1 l_1 = \alpha_1 t_1 \wedge \alpha_1 c_1 \longrightarrow^* true, \\ \ldots, \\ (l_n \to r_n \text{ if } c_n) \in \{\ldots\}_n, \alpha_n : \alpha_n l_n = \alpha_n t_n \wedge \alpha_n c_n \longrightarrow^* true}} & f(\alpha_1 r_1, \ldots, \alpha_n r_n) \quad \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

Let us show that $[S'](t)$ reduces to the same application. By definition of S', we have :
$[S'](t) = [\bigcup_{(l_1 \to r_1 \text{ if } c_1) \in \{\ldots\}_1, \ldots, (l_n \to r_n \text{ if } c_n) \in \{\ldots\}_n} \{f(l_1, \ldots, l_n) \to f(r_1, \ldots, r_n) \text{ if } \bigwedge_{i=1}^n c_i\}](t)$
If $top(t) \neq f$, then the application above reduces into $\emptyset$.
If $t = f(t_1, \ldots, t_n)$ then, with the rules of Table 1, the application reduces into :

$$\bigcup_{\substack{(l_1 \to r_1 \text{ if } c_1) \in \{\ldots\}_1, \alpha_1 : \alpha_1 l_1 = \alpha_1 t_1 \wedge \alpha_1 c_1 \longrightarrow^* true, \\ \ldots, \\ (l_n \to r_n \text{ if } c_n) \in \{\ldots\}_n, \alpha_n : \alpha_n l_n = \alpha_n t_n \wedge \alpha_n c_n \longrightarrow^* true}} f(\alpha_1 r_1, \ldots, \alpha_n r_n).$$

Therefore S and S' are equivalent.

Rule (21)

$$S = first(S_1, \ldots, S_i, \ldots, S_j, \ldots, S_n)$$
$$S' = first(S_1, \ldots, S_i, \ldots, \ldots, S_n)$$
$$\text{if } S_j \equiv S_i$$

With the rules of Table 1, we have the reduction :
$[S](t)$

$$\mapsto\!\!\!\!\gg \begin{cases} \emptyset & \text{if } \bigcup_{k=1, k \notin \{i,j\}}^n [S_k](t) \cup [S_i](t) \cup [S_j](t) = \emptyset \\ [S_m](t) & m < j, \text{ if } [S_m](t) \neq \emptyset \text{ and } \bigcup_{k=1}^{m-1} [S_k](t) = \emptyset \\ [S_j](t) & \text{if } [S_j](t) \neq \emptyset \text{ and } \bigcup_{k=1, k \neq i}^{m-1} [S_k](t) \cup [S_i](t) = \emptyset \\ [S_n](t) & n > j, \text{ if } [S_n](t) \neq \emptyset \text{ and } \bigcup_{k=1, k \notin \{i,j\}}^{n-1} [S_k](t) \cup [S_i](t) \cup [S_j](t) = \emptyset \end{cases}$$

If $S_j \equiv S_i$, the third condition is necessarily false, and the result above is equivalent to :

$$\begin{cases} \emptyset & \text{if } \bigcup_{k=1, k \notin \{i,j\}}^n [S_k](t) \cup [S_i](t) = \emptyset \\ [S_m](t) & m < j, \text{ if } [S_m](t) \neq \emptyset \text{ and } \bigcup_{k=1}^{m-1} [S_k](t) = \emptyset \\ [S_n](t) & n > j, \text{ if } [S_n](t) \neq \emptyset \text{ and } \bigcup_{k=1, k \notin \{i,j\}}^{n-1} [S_k](t) \cup [S_i](t) = \emptyset \end{cases}$$

which is also the first reduction step of $[S'](t)$. Hence S and S' are equivalent.

Rule (22)

$$S = dc(S_1, \ldots, S_i, \ldots, S_j, \ldots, S_n)$$
$$S' = dc(S_1, \ldots, S_i, \ldots, \ldots, S_n)$$
$$\text{if } S_j \equiv S_i$$

With the rules of Table 1, we have the reduction :
$[S](t)$

$$\mapsto\!\!\!\!\gg \begin{cases} \emptyset & \text{if } \bigcup_{k=1, k \notin \{i,j\}}^n [S_k](t) \cup [S_i](t) \cup [S_j](t) = \emptyset \\ [S_m](t) & m \notin \{i, j\}, \text{ if } [S_m](t) \neq \emptyset \\ [S_i](t) & \text{if } [S_i](t) \neq \emptyset \\ [S_j](t) & \text{if } [S_j](t) \neq \emptyset \end{cases}$$

If $S_j \equiv S_i$, the result above is equivalent to :

$$\begin{cases} \emptyset & \text{if } \bigcup_{k=1, k \notin \{i,j\}}^n [S_k](t) \cup [S_i](t) \cup [S_j](t) = \emptyset \\ [S_m](t) & m \notin \{i, j\}, \text{ if } [S_m](t) \neq \emptyset \\ [S_i](t) & \text{if } [S_i](t) \neq \emptyset \end{cases}$$

which is also the first reduction step of $[S'](t)$. Hence S and S' are equivalent.

Rule (23)

$$S = \{\ldots\}_1; \{\ldots\}_2$$
$$S' = \bigcup_{(l_1 \to r_1 \text{ if } c_1) \in \{\ldots\}_1, \ldots, (l_n \to r_n \text{ if } c_n) \in \{\ldots\}_n} comp(l_1 \to r_1 \text{ if } c_1, \ldots, l_n \to r_n \text{ if } c_n)\downarrow_{\mathcal{C}omp}$$

Straightforward from Proposition 2 and system $\mathcal{C}omp$.

Rule (24)

$$S = first(S_1, \ldots, S_n)$$
$$S' = first(S_1, \ldots, constrain(dk(S_1, \ldots, S_{n-1}), S_n)\downarrow_{\mathcal{CO}})$$

With the rules of Table 1, we have the reductions :

$$[S](t) \to \begin{cases} \emptyset & \text{if } \bigcup_{i=1}^{n} [S_i](t) = \emptyset \\ [S_1](t) & \text{if } [S_1](t) \neq \emptyset \\ [S_j](t) & \text{if } [S_j](t) \neq \emptyset \text{ and } \bigcup_{i=1}^{j-1} [S_i](t) = \emptyset \end{cases}$$

and

$$[S'](t) \to \begin{cases} \emptyset \text{ if } [S_1](t) \bigcup_{i=2}^{n} [constrain(dk(S_1, \ldots, S_{i-1}), S_i)\downarrow_{\mathcal{CO}}](t) = \emptyset \\ [S_1](t) \text{ if } [S_1](t) \neq \emptyset \\ [constrain(dk(S_1, \ldots, S_{j-1}), S_j)\downarrow_{\mathcal{CO}}](t) \text{ if } [constrain(dk(S_1, \ldots, S_{j-1}), S_j)\downarrow_{\mathcal{CO}}](t) \neq \emptyset \\ \quad \text{and } [S_1](t) \bigcup_{i=2}^{j-1} [constrain(dk(S_1, \ldots, S_{i-1}), S_i)\downarrow_{\mathcal{CO}}](t) = \emptyset \end{cases}$$

Thanks to Lemma 2, we can show by recurrence that the result of the above rewriting step is equivalent to :

$$\begin{cases} \emptyset & \text{if } \bigcup_{i=1}^{n} [S_i](t) = \emptyset \\ [S_1](t) & \text{if } [S_1](t) \neq \emptyset \\ [S_j](t) & \text{if } [S_j](t) \neq \emptyset \text{ and } \bigcup_{i=1}^{j-1} [S_i](t) = \emptyset \end{cases}$$

that is to the result of the rewriting step of $[S](t)$. Therefore S and S' are equivalent.

Rule (25)

$$S = repeat^*(S_1); S_2$$
$$S' = repeat^*(S_1); constrain(S_1, S_2)\downarrow_{\mathcal{CO}}$$

With the rules of Table 1, we have the reductions :
$[S](t) \twoheadrightarrow \bigcup_{t' \in [repeat^*(S_1)](t)} [S_2](t')$ and $[S'](t) \twoheadrightarrow \bigcup_{t' \in [repeat^*(S_1)](t)} [constrain(S_1, S_2)\downarrow_{\mathcal{CO}}](t')$.
By definition of the $repeat$ operator, we have $\forall t' \in [repeat^*(S_1)](t') : [S_1](t') = \emptyset$ and then, by Lemma 2, we get $[constrain(S_1, S_2)\downarrow_{\mathcal{CO}}](t') = [S_2](t')$, hence S and S' are equivalent.

Rule (26)

$$S = first(S_1, \ldots, S_n)$$
$$S' = dk(S_1, \ldots, S_n)$$
$$\text{if } \bigwedge_{i \neq j} mutex(S_i, S_j)\downarrow_{\mathcal{E}xcl}$$

Straightforward from Lemma 3 and Proposition 5.

Rule (27)

$$S = dc(S_1, \ldots, S_n)$$
$$S' = dk(S_1, \ldots, S_n)$$
$$\text{if } \bigwedge_{i \neq j} mutex(S_i, S_j)\downarrow_{\mathcal{E}xcl}$$

Straightforward from Lemma 3 and Proposition 5.
□

A.5 Termination

Proposition 1. *Given an* ELAN *strategy S, the reduction of $Rules(S)$ with $\mathcal{LR}$ terminates and results in a set of rules such that : if $Rules(S)\downarrow_{\mathcal{LR}}$ is ε-terminating, then S terminates.*

Proof: We proceed by structural induction on S.

- If $S = fail$, then $Rules(S)\downarrow_{\mathcal{LR}} = \emptyset$. Moreover, since S is terminating, the property is trivially true.
- If $S = id$, then $S\downarrow_{\mathcal{LR}} = \{x \to x\}$. Like in the previous case, S is terminating, and hence the property is trivially true.
- If $S = \{\ldots\}$, then $S\downarrow_{\mathcal{LR}} = \{\ldots\}$. Like in the previous two cases, S is terminating, and hence the property is trivially true.

We have shown that the property holds for any atomic strategy. Let us now assume the property for strategies $S_1, \ldots, S_n$, and let us show Proposition 1 by any combination S of $S_1, \ldots, S_n$.

- Case $\mathbf{S = dk(S_1, \ldots, S_n)}$. Let us assume that $Rules(S)\downarrow_{\mathcal{LR}}$ is ε-terminating. Since $Rules(S)\downarrow_{\mathcal{LR}} = \bigcup_{i=1}^{n} Rules(S_i)\downarrow_{\mathcal{LR}}$, each $Rules(S_i)\downarrow_{\mathcal{LR}}, i \in \{1, \ldots, n\}$, is ε-terminating. By induction hypothesis , each $S_i, i \in \{1, \ldots, n\}$ terminates. Let $t \in \mathcal{T}(\mathcal{F})$. By definition of the *dk* operator, we have
$$[S](t) \mapsto\!\!\!\!\!\rightarrow \bigcup_{i=1}^{n} [S_i](t)$$
with $[S_i](t)$ terminating for every $i \in \{1, \ldots, n\}$. Therefore $[S](t)$ terminates for any ground term t, hence S terminates.
- Case $\mathbf{S = dc(S_1, \ldots, S_n)}$. Let us assume that $Rules(S)\downarrow_{\mathcal{LR}}$ is ε-terminating. Since $Rules(S)\downarrow_{\mathcal{LR}} \bigcup_{i=1}^{n} Rules(S_i)\downarrow_{\mathcal{LR}}$, each $Rules(S_i)\downarrow_{\mathcal{LR}}, i \in \{1, \ldots, n\}$, is ε-terminating. By induction hypothesis , each $S_i, i \in \{1, \ldots, n\}$ terminates. Let $t \in \mathcal{T}(\mathcal{F})$. By definition of the *dc* operator, we have either $S(t) = \emptyset$, and in this case S terminates on t, or $\exists i \in \{1, \ldots, n\}$:
$$[S](t) \mapsto\!\!\!\!\!\rightarrow [S_i](t)$$
with $[S_i](t)$ terminating. Therefore $[S](t)$ terminates, for any ground term t.
- Case $\mathbf{S = first(S_1, \ldots, S_n)}$. Same case as $dc(S_1, \ldots, S_n)$.
- Case $\mathbf{S = f(S_1, \ldots, S_n), f} \in \mathcal{F}$. Let us assume that $Rules(S)\downarrow_{\mathcal{LR}}$ is ε-terminating. Since $Rules(S)\downarrow_{\mathcal{LR}} = \bigcup_{i=1}^{n} Rules(S_i)\downarrow_{\mathcal{LR}}$, each $Rules(S_i)\downarrow_{\mathcal{LR}}, i \in \{1, \ldots, n\}$, is ε-terminating. By induction hypothesis , each $S_i, i \in \{1, \ldots, n\}$ terminates. Let $t \in \mathcal{T}(\mathcal{F})$. By definition of the congruence, we have either $[S](t) = \emptyset$, in which case S terminates on t, or $t = f(t_1, \ldots, t_n)$ and
$$[S](t) \mapsto\!\!\!\!\!\rightarrow \bigcup_{u_1 \in [S_1](t_1), \ldots, u_n \in [S_n](t_n)} f(u_1, \ldots, u_n)$$
with $[S_i](t)$ terminating for every $i \in \{1, \ldots, n\}$. Therefore $[S](t)$ terminates, for any ground term t.
- Case $\mathbf{S = S_1; S_2}$. Let us assume that $Rules(S)\downarrow_{\mathcal{LR}}$ is ε-terminating. Since $Rules(S)\downarrow_{\mathcal{LR}} = Rules(S_1)\downarrow_{\mathcal{LR}} \cup Rules(S_2)\downarrow_{\mathcal{LR}}$, each $Rules(S_i)\downarrow_{\mathcal{LR}}, i \in \{1, 2\}$, is ε-terminating. By induction hypothesis , each $S_i, i \in \{1, 2\}$ terminates. Let $t \in \mathcal{T}(\mathcal{F})$. By definition of the composition, we have
$$[S](t) \mapsto\!\!\!\!\!\rightarrow \bigcup_{t' \in [S_1](t)} [S_2](t')$$
with $[S_1](t)$ and $[S_2](t)$ terminating. Hence $[S](t)$ terminates, for any ground term t.
- Case $\mathbf{S = repeat^*(S_1)}$. Since $Rules(S)\downarrow_{\mathcal{LR}} = Rules(S_1)\downarrow_{\mathcal{LR}} \cup \{x \to x\}$, $Rules(S)$ is not terminating, and the property is trivial.

□

In order to prove Theorem 1, we first show in Lemmas 4 and 5 a property linking application of a strategy and application of the rewrite rules involved in this strategy. We then deduce from this property Corollaries 1 and 2, that ensure termination of a strategy $repeat^*(S)$ if $Rules(S)\downarrow_{\mathcal{LR}}$ is terminating or is ε-terminating and S contains no congruence strategy, which is the trickiest part of Theorem 1.
For readibility, in the following, $Rules(S)\downarrow_{\mathcal{LR}}$ will be denoted $\{\ldots\}_S$, for any strategy S.

Lemma 4. *Let S be a strategy containing no repeat* symbol and no strategy congruence. Then we have $\forall t \in \mathcal{T}(\mathcal{F}), \forall i > 0 : \forall t' \in [S^i](t)$, there exists a ε-derivation chain of size greater than i from t to t' with $\{\ldots\}_S$.*

Proof: The proof relies on the idea that applying a strategy to a term consists in choosing a rewrite rule of the strategy to be applied to the term. If the strategy contains a finite composition of strategies, then the application of the strategy to the term consists in a finite sequence of applications of rewrite rules of the strategy to the term.
We proceed by induction on S.
- If $S = id$, then $\{\ldots\}_S = \{x \to x\}$. Given a ground term t, we have $\forall i > 0 : [S^i](t) = \{t\}$, and t can be obtained by i application of the rule $\{x \to x\}$ to t.
- If $S = fail$, then $\forall t \in \mathcal{T}(\mathcal{F}), \forall i > 0 : [S^i](t) = \emptyset$, and the property is trivially true.
- If $S = \{\ldots\}$, then $\{\ldots\}_S = \{\ldots\}$. The correspondance between application of S and rewriting with $\{\ldots\}_S$ is straightforward.

We have shown the property for any atomic strategy S. Let us now assume the property for strategies $S_1, \ldots, S_n$, and let us show Lemma 4 for any combination S of $S_1, \ldots, S_n$.
- If $S = dk(S_1, \ldots, S_n)$, then $\{\ldots\}_S = \bigcup_{i=1}^{n} \{\ldots\}_{S_i}$, and hence each rule of any $\{\ldots\}_i$ is also a rule of $\{\ldots\}_S$. Let $t \in \mathcal{T}(\mathcal{F})$. Let us show the property by recurrence on i.
  - Let us show the property for $i = 1$. Let $t' \in [S](t)$. By definition of the dk operator, $\exists j \in \{1, \ldots, n\} : t' \in [S_j](t)$, and hence, by induction hypothesis on S_j, there exists an ε-derivation chain of size greater than 1 from t to t' with $\{\ldots\}_{S_j}$, and hence also with $\{\ldots\}_S$.
  - Let us now assume the property for any $i \leq k, k \geq 1$. Let $t' \in [S^{k+1}](t)$. Then, by definition of the composition of strategies, there exists $t'' \in [S^k](t) : t' \in [S](t'')$. By definition of the dk operator, $\exists j \in \{1, \ldots, n\} : t' \in [S_j](t'')$. By recurrence hypothesis, there exists an ε-derivation chain of size greater than k from t to t'' with $\{\ldots\}_S$. By induction hypothesis on S_j, there exists an ε-derivation chain of size greater than 1 from t'' to t' with $\{\ldots\}_{S_j}$, and hence with $\{\ldots\}_S$. We then deduce the existence of an ε-derivation chain of size greater than $k+1$ from t to t' with $\{\ldots\}_S$.
- For $S = dc(S_1, \ldots, S_n)$ or $first(S_1, \ldots, S_n)$, we show the property like for $S = dk(S_1, \ldots, S_n)$.
- If $S = S_1; S_2$, then $\{\ldots\}_S = \{\ldots\}_{S_1} \cup \{\ldots\}_{S_2}$, and hence each rule of $\{\ldots\}_{S_1}, \{\ldots\}_{S_2}$ is also a rule of $\{\ldots\}_S$. Let $t \in \mathcal{T}(\mathcal{F})$. Let us show the property by recurrence on i.
  - Let us show the property for $i = 1$. Let $t' \in [S](t)$. By definition of the composition of strategies, $\exists t'' \in [S_1](t) : t' \in [S_2](t'')$. By induction hypothesis on S_1, there exists an ε-derivation chain of size greater than 1 from t to t'' with $\{\ldots\}_{S_1}$, and hence also with $\{\ldots\}_S$. Likewise, by induction hypothesis on S_2, there exists an ε-derivation chain of size greater than 1 from t'' to t' with $\{\ldots\}_{S_2}$, and hence also with $\{\ldots\}_S$. Therefore there exists an ε-derivation chain of size greater than 2 from t to t' with $\{\ldots\}_S$.
  - Let us now assume the property for any $i \leq k, k \geq 1$. Let $t' \in [S^{k+1}](t)$. Then, by definition of the composition of strategies, there exists $t'' \in [S^k](t) : t' \in [S](t'')$. By definition of the composition of strategies, $\exists t''' \in [S_1](t'') : t' \in [S_2](t''')$. By

induction hypothesis on S_1, S_2 and by recurrence hypothesis we show that there exists an ε-derivation chain of size greater than 1 from t to t' with $\{\ldots\}_S$

□

Lemma 5. *Let S be a strategy containing no repeat* symbol. Then we have $\forall t \in \mathcal{T}(\mathcal{F}), \forall i > 0 : \forall t' \in [S^i](t)$, there exists a derivation chain of size greater than i from t to t' with $\{\ldots\}_S$.*

Proof: The proof is similar as the one of Lemma 4, replacing ε-termination by termination, with the extra following case in the structural induction on S :

- If $S = f(S_1, \ldots, S_n), f \in \mathcal{F}$, then $\{\ldots\}_S = \bigcup_{i=1}^{n} \{\ldots\}_{S_i}$, and hence each rule of any $\{\ldots\}_i$ is also a rule of $\{\ldots\}_S$. Let $t \in \mathcal{T}(\mathcal{F})$. If $[S](t) = \emptyset$, then the property is trivial. Let us then assume $[S](t) \neq \emptyset$, that is $t = f(t_1, \ldots, t_n)$ with $\forall j \in \{j, \ldots, n\} : [S_j](t_j) \neq \emptyset$. Let us show the property by recurrence on i.
  - Let us show the property for $i = 1$. Let $t' \in [S](t)$. By definition of the congruence strategy, $\exists t'_1 \in [S_1](t_1), \ldots, t'_n \in [S_n](t_n) : t' = f(t'_1, \ldots, t'_n)$. For each $j \in \{1, \ldots, n\}$, by induction hypothesis on S_j, there exists a derivation chain of size greater than 1 from t_j to t'_j with $\{\ldots\}_{S_j}$, and hence also with $\{\ldots\}_S$. With the same rewritings, we then get a derivation chain of size greater than n, and hence greater than 1, from $t = f(t_1, \ldots, t_n)$ to $t' = f(t'_1, \ldots, t'_n)$ with $\{\ldots\}_S$.
  - Let us now assume the property for any $i \leq k, k \geq 1$. Let $t' \in [S^{k+1}](t)$. Then, by definition of the composition of strategies, there exists $t'' \in [S^k](t) : t' \in [S](t'')$. By definition of the congruence strategy, since $[S](t'') \neq \emptyset$, we have $\exists t''_1, \ldots, t''_n : t'' = f(t''_1, \ldots, t''_n)$ and $\forall j \in \{1, \ldots, n\} : [S_j](t''_j) \neq \emptyset$. By recurrence hypothesis, there exists a derivation chain greater than k from t to t''.
    Moreover, by definition of t' and the congruence strategy, we also have $\exists t'_1, \ldots, t'_n \in \mathcal{T}(\mathcal{F}) : t' = f(t'_1, \ldots, t'_n)$ and $\forall j \in \{1, \ldots, n\} : t'_j \in [S_j](t''_j)$. For each $j \in \{1, \ldots, n\}$, by induction hypothesis on S_j, there exists a derivation chain of size greater than 1 from t''_j to t'_j with $\{\ldots\}_{S_j}$, and hence with $\{\ldots\}_S$. We then deduce the existence of an derivation chain of size greater than $n * k$ from $t'' = f(t''_1, \ldots, t''_n)$ to $t' = f(t'_1, \ldots, t'_n)$ with $\{\ldots\}_S$. We then get a derivation chain of size greater that $(n + 1) * k$, and hence greater than $k + 1$, from t to t' with $\{\ldots\}_S$.

□

Corollary 1. *Let S be a strategy containing no repeat* symbol, t a ground term. If $\{\ldots\}_S$ is ε-terminating, then $\exists i > 0 : [S^i](t) = \emptyset$.*

Proof: A consequence of Lemma 4 is that given a ground term t, if $\nexists i > 0 : [S^i](t) = \emptyset$, then we can get an infinite ε-rewriting chain from t with $\{\ldots\}_S$. Therefore, if $\{\ldots\}_S$ is ε-terminating, such a chain cannot be obtained, and necessarily $\exists i > 0 : [S^i](t) = \emptyset$. □

Corollary 2. *Let S be a strategy containing no repeat* symbol and no strategy congruence, t a ground term. If $\{\ldots\}_S$ is terminating, then $\exists i > 0 : [S^i](t) = \emptyset$.*

Proof: The proof is the same as the one of Corollary 1, using Lemma 5 instead of Lemma 4. □

In addition to the previous two corrolaries, we will also need the following lemma to prove Theorem 1. It highlights a particular point of the semantics of symbol T in $\mathcal{T}ermin$: given a strategy S, the term $T(false, S)$ cannot reduce into $false$, since the first argument $false$ ensures we have lost the termination equivalence between S and the initial strategy.

Lemma 6. *Let S be a strategy. Then reduction of $T(false, S)$ with $\mathcal{T}ermin$ terminates and its normal form is either $true$ or $\sharp$.*

Proof: By structural induction on S. $\square$

Theorem 1. *Let S be a strategy. Then the reduction of $TERMIN(S)$ with $\mathcal{T}ermin$ terminates and its normal form is either $true$, $false$ or $\sharp$, and :*

- *if $TERMIN(S)\downarrow_{\mathcal{T}ermin} = true$, then S terminates ;*
- *if $TERMIN(S)\downarrow_{\mathcal{T}ermin} = false$, then S does not terminate.*

Proof: Since for any strategy S, $TERMIN(S)$ first rewrites into $T(true, S)$, we show the following more general property, for any $b \in \{false, true\}$:

- if $T(b, S)\downarrow_{\mathcal{T}ermin} = true$, then S terminates ;
- if $T(b, S)\downarrow_{\mathcal{T}ermin} = false$, then S does not terminate.

We proceed by structural induction on S. If $S = id, fail$ or $\{\ldots\}$, then S terminates and $T(b, S)\downarrow_{\mathcal{T}ermin} = true$. Let us now assume the property holds for strategies $S_1, \ldots, S_n$ and let us show it for any combination S of $S_1, \ldots, S_n$.

- If $S = dk(S_1, \ldots, S_n)$, then $T(b, S) \longrightarrow_{\mathcal{T}ermin} \bigwedge_{i=1}^{n} T(b, S_i)$. By induction hypothesis on each S_i, reduction of $T(b, S_i)$ with $\mathcal{T}ermin$ terminates and its normal form is either $true$, $false$ or $\sharp$. Therefore reduction of $T(b, S)$ terminates and its normal form $T(b, S)\downarrow_{\mathcal{T}ermin} = \bigwedge_{i=1}^{n} T\downarrow_{\mathcal{T}ermin}(b, S_i)$ is either $true$, $false$ or $\sharp$.
  - If $T(b, S)\downarrow_{\mathcal{T}ermin} = true$, then necessarily $\forall i \in \{1, \ldots, n\} : T(b, S_i)\downarrow_{\mathcal{T}ermin} = true$. By induction hypothesis on each S_i, S_i terminates. Therefore $S = dk(S_1, \ldots, S_n)$ also terminates.
  - If $T(b, S)\downarrow_{\mathcal{T}ermin} = false$, then necessarily $\exists i \in \{1, \ldots, n\} : T(b, S_i)\downarrow_{\mathcal{T}ermin} = false$. By induction hypothesis on S_i, S_i does not terminate, that is there exists $t \in \mathcal{T}(\mathcal{F})$ such that $[S_i](t)$ does not terminates. Since $[S](t)$ needs evaluation of $[S_i](t)$, then S does not terminate.
- If $S = dc(S_1, \ldots, S_n)$, then $T(b, S) \longrightarrow_{\mathcal{T}ermin} \bigwedge_{i=1}^{n} T(b, S_i)$. By induction hypothesis on each S_i, reduction of $T(b, S_i)$ with $\mathcal{T}ermin$ terminates and its normal form is either $true$, $false$ or $\sharp$. Therefore reduction of $T(b, S)$ terminates and its normal form $T(b, S)\downarrow_{\mathcal{T}ermin} = \bigwedge_{i=1}^{n} T\downarrow_{\mathcal{T}ermin}(b, S_i)$ is either $true$, $false$ or $\sharp$.
  - If $T(b, S)\downarrow_{\mathcal{T}ermin} = true$, then necessarily $\forall i \in \{1, \ldots, n\} : T(b, S_i)\downarrow_{\mathcal{T}ermin} = true$. By induction hypothesis on each S_i, S_i terminates. Therefore $S = dk(S_1, \ldots, S_n)$ also terminates.
  - If $T(b, S)\downarrow_{\mathcal{T}ermin} = false$, then necessarily $\exists i \in \{1, \ldots, n\} : T(b, S_i)\downarrow_{\mathcal{T}ermin} = false$. By induction hypothesis on S_i, S_i does not terminate, that is there exists $t \in \mathcal{T}(\mathcal{F})$ such that $[S_i](t)$ does not terminates. Since $[S](t)$ may need evaluation of $[S_i](t)$, then S does not terminate.
- If $S = f(S_1, \ldots, S_n)$, with $f \in \mathcal{F}$, then $T(b, S) \longrightarrow_{\mathcal{T}ermin} \bigwedge_{i=1}^{n} T(b, S_i)$. By induction hypothesis on each S_i, reduction of $T(b, S_i)$ with $\mathcal{T}ermin$ terminates and its normal form is either $true$, $false$ or $\sharp$. Therefore reduction of $T(b, S)$ terminates and its normal form $T(b, S)\downarrow_{\mathcal{T}ermin} = \bigwedge_{i=1}^{n} T\downarrow_{\mathcal{T}ermin}(b, S_i)$ is either $true$, $false$ or $\sharp$.
  - If $T(b, S)\downarrow_{\mathcal{T}ermin} = true$, then necessarily $\forall i \in \{1, \ldots, n\} : T(b, S_i)\downarrow_{\mathcal{T}ermin} = true$. By induction hypothesis on each S_i, S_i terminates. Therefore $S = dk(S_1, \ldots, S_n)$ also terminates.
  - If $T(b, S)\downarrow_{\mathcal{T}ermin} = false$, then necessarily $\exists i \in \{1, \ldots, n\} : T(b, S_i)\downarrow_{\mathcal{T}ermin} = false$. By induction hypothesis on S_i, S_i does not terminate, that is there exists $t \in \mathcal{T}(\mathcal{F})$ such that $[S_i](t)$ does not terminates. Since $[S](t)$ needs evaluation of $[S_i](t)$, then S does not terminate.

- If $S = first(S_1, \ldots, S_n)$, then $T(b,S) \longrightarrow_{\mathcal{T}ermin} T(b,S_1) \bigwedge_{i=2} nT(false, S_i)$. By induction hypothesis on S_1, reduction of $T(b,S_1)$ with $\mathcal{T}ermin$ terminates and its normal form is either $true$, $false$ or $\sharp$. By Lemma 6, for $i \in \{2,\ldots,n\}$, reduction of $T(false,S_i)$ with $\mathcal{T}ermin$ terminates and its normal form is either $true$ or $\sharp$. Therefore reduction of $T(b,S)$ with $\mathcal{T}ermin$ terminates and its normal form $T(b,S)\downarrow_{\mathcal{T}ermin} = T(b,S_1)\downarrow_{\mathcal{T}ermin} \bigwedge_{i=2}^{n} T(false,S_i)\downarrow_{\mathcal{T}ermin}$ is either $true$, $false$ or $\sharp$.
  - If $T(b,S)\downarrow_{\mathcal{T}ermin} = true$, then necessarily $T(b,S_1)\downarrow_{\mathcal{T}ermin} \bigwedge_{i=2}^{n} T(false,S_i)\downarrow_{\mathcal{T}ermin} = true$. By induction hypothesis on $S_1,\ldots,S_n$, each S_i terminates, and therefore $S = first(S_1,\ldots,S_n)$ terminates.
  - If $T(b,S)\downarrow_{\mathcal{T}ermin} = false$, then necessarily $T(b,S_1)\downarrow_{\mathcal{T}ermin} = false$ since, by Lemma 6, $\forall i \in \{2,\ldots,n\} : T(b,S_i)\downarrow_{\mathcal{T}ermin} \neq false$. By induction hypothesis on S_1, we get that S_1 is not terminating, and hence $\exists t \in \mathcal{T}(\mathcal{F})$ such that $[S_1](t)$ does not terminate. Since $[S](t)$ requires evaluation of $[S_1](t)$, then S does not terminate.
- If $S = S_1; S_2$, then $T(b,S) \longrightarrow_{\mathcal{T}ermin} T(b,S_1) \wedge T(false,S_2)$. By induction hypothesis on S_1, reduction of $T(b,S_1)$ with $\mathcal{T}ermin$ terminates and its normal form is either $true$, $false$ or $\sharp$. By Lemma 6, reduction of $T(false,S_2)$ with $\mathcal{T}ermin$ terminates and its normal form is either $true$ or $\sharp$. Therefore reduction of $T(b,S)$ with $\mathcal{T}ermin$ terminates and its normal form $T(b,S)\downarrow_{\mathcal{T}ermin} = T(b,S_1)\downarrow_{\mathcal{T}ermin} \wedge T(false,S_2)\downarrow_{\mathcal{T}ermin}$ is either $true$, $false$ or $\sharp$.
  - If $T(b,S)\downarrow_{\mathcal{T}ermin} = true$, then necessarily $T(b,S_1)\downarrow_{\mathcal{T}ermin} \wedge T(false,S_i)\downarrow_{\mathcal{T}ermin} = true$. By induction hypothesis on S_1, S_2, both S_1, S_2 terminate, and therefore $S = S_1; S_2$ terminates.
  - If $T(b,S)\downarrow_{\mathcal{T}ermin} = false$, then necessarily $T(b,S_1)\downarrow_{\mathcal{T}ermin} = false$ since, by Lemma 6, $T(b,S_2)\downarrow_{\mathcal{T}ermin} \neq false$. By induction hypothesis on S_1, we get that S_1 is not terminating, and hence $\exists t \in \mathcal{T}(\mathcal{F})$ such that $[S_1](t)$ does not terminate. Since $[S](t)$ requires evaluation of $[S_1](t)$, then S does not terminate.
- If $S = repeat^*(S_1)$, then let us distinguish reductions with $\mathcal{T}ermin$ of $T(false,S)$ and $T(true,S)$.
  - In any case, $T(false,S) \longrightarrow_{\mathcal{T}ermin} true$ or $\sharp$, and then reduction of $T(false,S)$ with $\mathcal{T}ermin$ terminates. If $T(false,S)\downarrow_{\mathcal{T}ermin} = true$, then $\{\ldots\}_{S_1}$ is either terminating or ε-terminating with S containing no congruence strategy. Then, by Corrolaries 1 and 2, $\forall t \in \mathcal{T}(\mathcal{F}), \exists i > 0 : [{S_1}^i](t) = \emptyset$, and therefore S terminates.
  - Reduction of $T(true,S)$ with $\mathcal{T}ermin$ depends on S_1.
    - If $S_1 = \{\ldots\}$, then $T(true,S)$ reduces into $false$, $true$ or $\sharp$ and hence terminates. Moreover, equivalence between ε-termination of $\{\ldots\}$ and termination of $repeat^*(\{\ldots\})$ is straightforward and ensures the property for $T(true,S)$.
    - If $S_1 \neq \{\ldots\}$, then $T(true,S)$ reduces into $true$ or $\sharp$ and hence terminates. If $T(true,S)\downarrow_{\mathcal{T}ermin} = true$, then necessarily $\{\ldots\}_{S_1}$ is ε-terminating or terminating with S_1 containing no congruence strategy. Then, by Corrolaries 1 and 2, $\forall t \in \mathcal{T}(\mathcal{F}), \exists i > 0 : [{S_1}^i](t) = \emptyset$, and therefore S terminates.

□

B Examples

Simplification of strategies is detailed in Example 6, and sketched in further examples, since detailed explanations would be similar. Table 6 emphasizes the importance of first simplifying strategies for proving their termination with our criterion.

Table 6. Examples – summary

	Initial strategy S	Simplified strategy S'	$TERMIN(repeat^*(S))$	$TERMIN(repeat^*(S'))$
Ex.2	$\{f(x) \to g(x), f(x) \to h(0,x)\}$; $\{h(1,x) \to f(1), h(x,y) \to y, f(1) \to h(1,1)\}$	$f(x) \to x$	$\sharp$	$true$ (LPO)
Ex.3	$first(\{g(x_1,1) \to h(x_1)$ if $x_1 > 1\}$, $\{g(x_3,x_4) \to f(x_3)$ if $x_3 > 2\})$	$\{g(x_1,1) \to h(x_1)$ if $x_1 > 1$, $g(x_3,x_4) \to f(x_3)$ if $x_3 > 2 \wedge x_4 \neq 1\}$	$true$ (LPO, Prop 3)	$true$ (LPO, Prop 3)
Ex.4	$[s_1]f(x,y) \to g(y,x)$ $[s_2]g(x,y) \to y$ $[s_3]g(x,y) \to f(y,x)$ $first(first(s_2,s_3), first(s_3,s_2,s_1))$ $first(dc(s_2,s_3), first(s_3,s_2,s_1))$ $first(first(s_3,s_2), first(s_3,s_2,s_1))$	 $s_1 \cup s_2$ $dk(dc(s_2,s_3), s_1)$ $s_1 \cup s_3$	 $\sharp$ $\sharp$ $\sharp$	 $true$ (LPO) $\sharp$ $false$
Ex.5	$S = repeat^*(first(\{f(x_1) \to g(x_1)$ if $x_1 > 3\}$; $first(\{g(x_2) \to h(x_2)$ if $x_2 > 1\}$, $\{g(4) \to g(4)\}, \{g(x_4) \to f(x_4)$ if $x_4 > 2\})$, $dk(\{f(x) \to g(x), f(x) \to h(0,x)$ if $x > 4\}$; $\{h(1,x) \to f(1), h(x,y) \to y, f(1) \to h(1,1)\}$, $f(x) \to f(f(x))$ if $x > 5)))$	$repeat^*(f(x) \to h(x)$ if $x > 3)$	$\sharp$	$true$ (LPO, Prop 3)
Ex.6	$first(\{h(x,y) \to f(y), f(a) \to g(b)\}$ $\{h(0,y) \to g(y), f(x) \to g(x), g(a) \to f(a)\})$	$\{h(x,y) \to f(y), f(a) \to g(b)$, $f(x) \to g(x)$ if $x \neq a, g(a) \to f(a)\}$	$\sharp$	$true$ (OSO, Prop 3)
Ex.7	$[r_1]f(x_1) \to g(x_1)$ if $x_1 > 3$ $[r_2]g(x_2) \to h(x_2)$ if $x_2 > 1$ $[r_3]g(4) \to g(4)$ $[r_4]g(x_4) \to f(x_4)$ if $x_4 > 2$ $r_1; first(r_2,r_3,r_4)$ $first(r_2, repeat^*(dk(r_3,r_4)))$	 $\{f(x) \to h(x)$ if $x > 3\}$ $first(r_2, id)$	 $\sharp$ $\sharp$	 $true$ (LPO, Prop 3) $\sharp$
Ex.8	$[r_{12}]s1 \to s2$ $[r_{13}]s1 \to s3$ $[r_{25}]s2 \to s5$ $[r_{32}]s3 \to s2$ $[r_{34}]s3 \to s4$ $[r_{41}]s4 \to s1$ $dk(r_{12}, r_{13}, r_{25}, r_{32}, r_{34}, r_{41})$	 $\{r_{12}, r_{13}, r_{25}, r_{32}, r_{34}, r_{41}\}$	 $\sharp$	 $false$

Example 2 (continued). We already showed in the paper that $S_{comp} = \{f(x) \to g(x), f(x) \to h(0,x)\}; \{h(1,x) \to f(1), h(x,y) \to y, f(1) \to h(1,1)\}$ simplifies into $S'_{comp} = \{f(x) \to x\}$.
Let us evaluate $TERMIN(repeat^*(S_{comp}))$:
$TERMIN(repeat^*(S_{comp})) \longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(S_{comp}))$
$\longrightarrow_{\mathcal{T}ermin} \sharp$
for $Rules(S_{comp})\downarrow_{\mathcal{LR}} = \{f(x) \to g(x), f(x) \to h(0,x), h(1,x) \to f(1), h(x,y) \to y, f(1) \to h(1,1)\}$, which is not ε-terminating, because of the infinite ε-chain $h(1,1) \longrightarrow f(1) \longrightarrow h(1,1) \longrightarrow \ldots$
Let us now evaluate $TERMIN(repeat^*(S'_{comp}))$:
$TERMIN(repeat^*(S'_{comp})) \longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(S'_{comp}))$
$\longrightarrow_{\mathcal{T}ermin} true$
for $Rules(S'_{comp})\downarrow_{\mathcal{LR}} = \{f(x) \to x\}$, which is terminating, hence ε-terminating.

Finally, $repeat^*(S_{comp})$ is proved terminating by combination of simplification process and termination criterion.

Example 3 (continued). We already showed in the paper that $S = first(\{g(x_1,1) \to h(x_1) \text{ if } x_1 > 1\}, \{g(x_3,x_4) \to f(x_3) \text{ if } x_3 > 2\})$ simplifies into $S' = \{g(x_1,1) \to h(x_1) \text{ if } x_1 > 1, g(x_3,x_4) \to f(x_3) \text{ if } x_3 > 2 \wedge x_4 \neq 1\}$.
Let us evaluate $TERMIN(repeat^*(S))$:
$TERMIN(repeat^*(S)) \longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(S))$
$\longrightarrow_{\mathcal{T}ermin} true$
for $Rules(S)\downarrow_{\mathcal{LR}} = \{g(x_1,1) \to h(x_1) \text{ if } x_1 > 1, g(x_3,x_4) \to f(x_3) \text{ if } x_3 > 2\}$, which can be proved terminating with any simplification ordering with the precedence $g \succ_{\mathcal{F}} h, f, >, 2$.
Remark that $Rules(S)\downarrow_{\mathcal{LR}}$ can also be proved ε-terminating by using Proposition 3, since $Rules(S)\downarrow^2_{\mathcal{LR}} \longrightarrow^*_{\mathcal{SIMPL}} \emptyset$.
Then $repeat^*(S)$ can be proved terminating directly, that is without using simplification.
Let us evaluate anyway $TERMIN(repeat^*(S'))$:
$TERMIN(repeat^*(S')) \longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(S'))$
$\longrightarrow_{\mathcal{T}ermin} true$
for $Rules(S)\downarrow_{\mathcal{LR}} = \{g(x_1,1) \to h(x_1) \text{ if } x_1 > 1, g(x_3,x_4) \to f(x_3) \text{ if } x_3 > 2 \wedge x_4 \neq 1\}$, which can be proved terminating with any simplification ordering with the precedence $g \succ_{\mathcal{F}} h, f, >, \wedge, \neq, 2, 1$. Remark that $Rules(S')\downarrow_{\mathcal{LR}}$ can also be proved ε-terminating by using Proposition 3, since $Rules(S')\downarrow^2_{\mathcal{LR}} \longrightarrow^*_{\mathcal{SIMPL}} \emptyset$.

Example 6. Let us consider the strategy

$$S = first(\ \{\ h(x,y) \to f(y),\ f(a) \to g(b)\},\ \{\ h(0,y) \to g(y),\ f(x) \to g(x),\ g(a) \to f(a)\ \}\ )$$

which is an extension of the strategy introduced at the end of Section 3 to motivate the need for first simplifying strategies.
– **Simplification of S** –
The simplification of S is given by :
$S \longrightarrow^{(24)} first(\{h(x,y) \to f(y), f(a) \to g(b)\}, \{f(x) \to g(x) \text{ if } x \neq a, g(a) \to f(a)\})$
$\longrightarrow^{(26)} dk(\{h(x,y) \to f(y), f(a) \to g(b)\}, \{f(x) \to g(x) \text{ if } x \neq a, g(a) \to f(a)\})$
$\longrightarrow^{(19)} \{h(x,y) \to f(y), f(a) \to g(b), f(x) \to g(x) \text{ if } x \neq a, g(a) \to f(a)\}$
Let us explain in detail the above applications of rules (24) and (26).

(rule (24)) Application of the rule (24) decomposes as follows :
$S \longrightarrow^{(24)} first(\{h(x,y) \to f(y), f(a) \to g(b)\},$
$constrain(\{h(x,y) \to f(y), f(a) \to g(b)\}, \{h(0,y) \to g(y), f(x) \to g(x), g(a) \to f(a)\})\downarrow_{\mathcal{CO}})$
The second argument of the $first$ strategy above is given by :
$constrain(\{h(x,y) \to f(y), f(a) \to g(b)\}, \{h(0,y) \to g(y), f(x) \to g(x), g(a) \to f(a)\})$
$\longrightarrow_{\mathcal{CO}} constrain(\{h(x,y) \to f(y), f(a) \to g(b)\}, h(0,y') \to g(y'))$
$\cup\ constrain(\{h(x,y) \to f(y), f(a) \to g(b)\}, f(x') \to g(x'))$
$\cup\ constrain(\{h(x,y) \to f(y), f(a) \to g(b)\}, g(a) \to f(a))$
$\longrightarrow_{\mathcal{CO}} h(0,y') \to g(y') \text{ if } not(true)$
$\cup\ f(x') \to g(x') \text{ if } not(x' = a)$
$\cup\ g(a) \to f(a)$

(rule (26)) Application of the rule (26) comes from the fact that $mutex(\{h(x,y) \to f(y), f(a) \to g(b)\}, \{f(x) \to g(x) \text{ if } x \neq a, g(a) \to f(a)\}) \longrightarrow^*_{\mathcal{E}xcl} true$. Indeed :

$$\begin{aligned}
&mutex(\{h(x,y) \to f(y), f(a) \to g(b)\}, \{f(x) \to g(x) \text{ if } x \neq a, g(a) \to f(a)\}) \\
&\longrightarrow_{\mathcal{E}xcl} mutex(h(x,y) \to f(y), f(x) \to g(x) \text{ if } x \neq a) \\
&\qquad \wedge mutex(h(x,y) \to f(y), g(a) \to f(a)) \\
&\qquad \wedge mutex(f(a) \to g(b), f(x) \to g(x) \text{ if } x \neq a) \\
&\qquad \wedge mutex(f(a) \to g(b), g(a) \to f(a)) \\
&\longrightarrow_{\mathcal{E}xcl} true \\
&\qquad \wedge true \\
&\qquad \wedge true \\
&\qquad \wedge true
\end{aligned}$$

– **Termination of repeat*(S)** –

Let us evaluate $TERMIN(repeat^*(S))$:

$$\begin{aligned}
TERMIN(repeat^*(S)) &\longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(S)) \\
&\longrightarrow_{\mathcal{T}ermin} \sharp
\end{aligned}$$

for $Rules(S)\downarrow_{\mathcal{LR}} = \{h(x,y) \to f(y), f(a) \to g(b), h(0,y) \to g(y), f(x) \to g(x), g(a) \to f(a)\}$, which is not ε-terminating, because of the infinite ε-chain $g(a) \longrightarrow f(a) \longrightarrow g(a) \longrightarrow \ldots$

Denoting $S' = \{h(x,y) \to f(y), f(a) \to g(b), f(x) \to g(x) \text{ if } x \neq a, g(a) \to f(a)\}$ the simplified form of S, let us now evaluate $TERMIN(repeat^*(S'))$:

$$\begin{aligned}
TERMIN(repeat^*(S')) &\longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(S')) \\
&\longrightarrow_{\mathcal{T}ermin} true
\end{aligned}$$

for $Rules(S')\downarrow_{\mathcal{LR}} = S'$, which can be proved terminating by using Proposition 3 and simplifying S'[3], since we have :

$$\begin{aligned}
S'; S'; S' &\longrightarrow_{\mathcal{SIMPL}} \{h(x,a) \to g(b), h(x,y) \to g(y) \text{ if } y \neq a, f(a) \to f(a) \text{ if } a \neq a, \\
&\qquad g(a) \to g(b), g(a) \to g(a) \text{ if } a \neq a\}; S' \\
&\longrightarrow \quad (fail \cup fail \cup \{h(x,a) \to g(b), h(x,y) \to g(y) \text{ if } y \neq a, g(a) \to g(b)\}); S'. \\
&\longrightarrow_{\mathcal{SIMPL}} \{h(x,a) \to f(a) \text{ if } a \neq a\} \\
&\longrightarrow \quad fail
\end{aligned}$$

Remark that S' can also be proved terminating by typing and using an Order-Sorted Ordering (OSO), as in [12]. The system S' can be typed without lack of generality in the following way. We define three sorts :

- Na for any term but a ;
- A for the constant a ;
- s for the most general sort.

and f, g, h are expressed by the typed operators :

$$\begin{aligned}
&f : Na \to s, f : A \to s \\
&g : Na \to s, g : A \to s \\
&h : s \times s \to s
\end{aligned}$$

Then S' is equivalent to the so-called *desambiguated* specification

$$\begin{aligned}
&(\forall y : A) h_{s \times s}(x,y) \to f_{A \to s}(y) \\
&(\forall y : Na) h_{s \times s}(x,y) \to f_{Na \to s}(y) \\
&f_{A \to s}(a) \to g_{Na \to s}(b) \\
&(\forall x : Na) f_{Na \to s}(x) \to g_{Na \to s}(x) \\
&g_{A \to s}(a) \to f_{A \to s}(a)
\end{aligned}$$

which can be proved terminating with an OSO having the precedence

$$\begin{aligned} f_{Na\to s} &\succ g_{Na\to s} \\ g_{A\to s} &\succ f_{A\to s} \\ f_{A\to s} &\succ g_{Na\to s} \\ h_{s\times s} &\succ f_{Na\to s} \\ h_{s\times s} &\succ f_{A\to s} \end{aligned}$$

Example 4 (extended with the study of new strategies). Let us consider the following three ELAN labelled rules :

```
[s_1] f(x,y) => g(y,x)
[s_2] g(x,y) => y
[s_3] g(x,y) => f(y,x)
```

and then let us the following strategies :

1. $S_1 = first(first(s_2, s_3), first(s_3, s_2, s_1))$.
2. $S_2 = first(dc(s_2, s_3), first(s_3, s_2, s_1))$.
3. $S_3 = first(first(s_3, s_2), first(s_3, s_2, s_1))$.

– **Simplification of $\mathbf{S_1}, \mathbf{S_2}, \mathbf{S_3}$** –

Let us simplify S_1 :

$$\begin{aligned} S_1 &\longrightarrow^{(17)} first(s_2, s_3, s_3, s_2, s_1) \\ &\longrightarrow^{(21)} first(s_2, s_3, s_3, s_1) \\ &\longrightarrow^{(21)} first(s_2, s_3, s_1) \\ &\longrightarrow^{(24)} first(s_2, fail, s_1) \\ &\longrightarrow^{(6)} \ first(s_2, s_1) \\ &\longrightarrow^{(26)} dk(s_2, s_1) \\ &\longrightarrow^{(19)} s_1 \cup s_2 \end{aligned}$$

Let us now simplify S_2 :

$$\begin{aligned} S_2 &\longrightarrow^{(17)} first(dc(s_2, s_3), s_3, s_2, s_1) \\ &\longrightarrow^{(24)} first(dc(s_2, s_3), fail, fail, s_1) \\ &\longrightarrow^{(6)} first(dc(s_2, s_3), fail, s_1) \\ &\longrightarrow^{(6)} first(dc(s_2, s_3), s_1) \\ &\longrightarrow^{(26)} dk(dc(s_2, s_3), s_1) \end{aligned}$$

Let us finally simplify S_3 :

$$\begin{aligned} S_3 &\longrightarrow^{(17)} first(s_3, s_2, s_3, s_2, s_1) \\ &\longrightarrow^{(21)} first(s_3, s_2, s_2, s_1) \\ &\longrightarrow^{(21)} first(s_3, s_2, s_1) \\ &\longrightarrow^{(24)} first(s_3, fail, s_1) \\ &\longrightarrow^{(6)} \ first(s_3, s_1) \\ &\longrightarrow^{(26)} dk(s_3, s_1) \\ &\longrightarrow^{(19)} s_1 \cup s_3 \end{aligned}$$

– **Termination of $\mathbf{repeat^*(S_1), repeat^*(S_2), repeat^*(S_3)}$** –

For $i \in \{1, 2, 3\}$, we have $TERMIN(repeat^*(S_i)) \longrightarrow^*_{\mathcal{T}ermin} \sharp$, for $Rules(S_i)\downarrow_{\mathcal{LR}} = s_1 \cup s_2 \cup s_3$, which is not ε-terminating, because of the infinite ε-chain $f(x, y) \longrightarrow g(y, x) \longrightarrow f(x, y) \longrightarrow \ldots$

Denoting S'_i the simplified forms of S_i, let us now evaluate $TERMIN(repeat^*(S'_i))$.

We first evaluate $repeat^*(S'_1) = repeat^*(s_1 \cup s_2)$:

$$\begin{aligned} TERMIN(repeat^*(S'_1)) &\longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(S'_1)) \\ &\longrightarrow_{\mathcal{T}ermin} true \end{aligned}$$

for $Rules(S'_1)\downarrow_{\mathcal{LR}} = S'_1$, which can be shown terminating with any simplification ordering with the precedence $f \succ g$.

We then evaluate $repeat^*(S_2') = repeat^*(dk(dc(s_2, s_3), s_1))$:
$TERMIN(repeat^*(S_2')) \longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(S_2'))$
$\longrightarrow_{\mathcal{T}ermin} \sharp$
for $Rules(S_2')\downarrow_{\mathcal{LR}} = s_1 \cup s_2 \cup s_3$, which is not ε-terminating, as previously noticed.
Finally, we evaluate $repeat^*(S_3') = repeat^*(s_1 \cup s_3)$:
$TERMIN(repeat^*(S_3')) \longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(S_3'))$
$\longrightarrow_{\mathcal{T}ermin} false$
for $Rules(S_3')\downarrow_{\mathcal{LR}} = s_1 \cup s_3$, which is not ε-terminating, because of the infinite ε-chain $f(x, y) \longrightarrow g(y, x) \longrightarrow f(x, y) \longrightarrow \ldots$.

We then proved that $repeat^*(S_1)$ terminates and $repeat^*(S_3)$ does not terminate. As for $repeat^*(S_2)$, our termination criterion cannot determine whether it terminates or not.

Example 7. Let us consider the following ELAN program :
$[r_1]\ f(x_1) \to g(x_1)$ if $x_1 > 3$
$[r_2]\ g(x_2) \to h(x_2)$ if $x_2 > 1$
$[r_3]\ g(4) \to g(4)$
$[r_4]\ g(x_4) \to f(x_4)$ if $x_4 > 2$
and let us study the termination of the following strategies :

1. $S_1 = r_1; first(r_2, r_3, r_4)$
2. $S_2 = first(s_2, repeat^*(dk(r_3, r_4)))$

Strategy S_1

Let us simplify S_1 :
$S_1 \longrightarrow^{(24)} r_1; first(r_2, fail, fail)$
$\longrightarrow^{(6)} r_1; first(r_2, fail)$
$\longrightarrow^{(6)} r_1; first(r_2)$
$\longrightarrow^{(17)} r_1; r_2$
$\longrightarrow^{(23)} \{f(x) \to h(x) \text{ if } x > 3\}$

Let us evaluate $TERMIN(repeat^*(S_1))$:
$TERMIN(repeat^*(S_1)) \longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(S_1))$
$\longrightarrow_{\mathcal{T}ermin} \sharp$
for $Rules(S_1)\downarrow_{\mathcal{LR}} = r_1 \cup r_2 \cup r_3 \cup r_4$, which is not ε-terminating, because of the infinite ε-chain $f(4) \longrightarrow g(4) \longrightarrow f(4) \longrightarrow \ldots$

Denoting S_1' the simplified form of S_1, let us now evaluate $TERMIN(repeat^*(S_1'))$:
$TERMIN(repeat^*(\{f(x) \to h(x) \text{ if } x > 3\})) \longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(\{f(x) \to h(x) \text{ if } x > 3\}))$
$\longrightarrow_{\mathcal{T}ermin} true$
for $Rules(S_1')\downarrow_{\mathcal{LR}} = S_1'$, which is terminating.

Strategy S_2

The simplification of S_2 consists of the rewriting step $S_2 \longrightarrow^{(24)} first(r_2, id)$. The $repeat^*$ strategy simplified to id, because $constrain(r_2, dk(r_3, r_4)) \longrightarrow^*_{\mathcal{CO}} fail$.

Let us evaluate $TERMIN(repeat^*(S_2))$:
$TERMIN(repeat^*(S_2)) \longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(S_2))$
$\longrightarrow_{\mathcal{T}ermin} \sharp$
for $Rules(S_2)\downarrow_{\mathcal{LR}} = r_2 \cup r_3 \cup r_4 \cup \{x \to x\}$, which is not ε-terminating.

Denoting S_2' the simplified form of S_2, let us now evaluate $TERMIN(repeat^*(S_2'))$:
$TERMIN(repeat^*(first(r_2, id))) \longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(first(r_2, id)))$
$\longrightarrow_{\mathcal{T}ermin} \sharp$
for $Rules(S_1')\downarrow_{\mathcal{LR}} = r_2 \cup \{x \to x\}$, which is not ε-terminating.

Example 8. This example is the Example 5.9, page 146, of the thesis of H. Cirstea [5], and describes transitions in a five states automata. The program contains in particular the ELAN labelled rules :

```
[r12] s1 => s2
[r13] s1 => s3
[r25] s2 => s5
[r32] s3 => s2
[r34] s3 => s4
[r41] s4 => s1
```

and the strategies :

$$follow = dk(r12, r13, r25, r32, r34, r41)$$
$$gen\_double = follow; follow$$

We study here the simplification of the strategy *gen_double*, and show that the strategy $repeat^*(follow)$ is not terminating.

– **Simplification of gen_double** –

Let us first study the simplification of the strategy *gen_double* :

$dk(r12, r13, r25, r32, r34, r41); dk(r12, r13, r25, r32, r34, r41)$
$\longrightarrow^{(12)} dk(dk(r12, r13, r25, r32, r34, r41); r12, \ldots, dk(r12, r13, r25, r32, r34, r41); r41)$
$\longrightarrow^{(11)*} dk(dk(r12; r12, \ldots, r41; r12), \ldots, dk(r12; r41, \ldots, r41; r41))$
$\longrightarrow^{(16)*} dk(r12; r12, \ldots, r41; r12, \ldots, r12; r41, \ldots, r41; r41)$

Let us summarize in a table the results of every argument ; the cell of the i^{th} line, named s_i and of the j^{th} column, named s_j contains the result of the simplification of the composition $s_i; s_j$:

	$r12$	$r13$	$r25$	$r32$	$r34$	$r41$
$r12$	$fail$	$fail$	$s1 \to s5$	$fail$	$fail$	$fail$
$r13$	$fail$	$fail$	$fail$	$s1 \to s2$	$s1 \to s4$	$fail$
$r25$	$fail$	$fail$	$fail$	$fail$	$fail$	$fail$
$r32$	$fail$	$fail$	$s3 \to s5$	$fail$	$fail$	$fail$
$r34$	$fail$	$fail$	$fail$	$fail$	$fail$	$s3 \to s1$
$r41$	$s4 \to s2$	$s4 \to s3$	$fail$	$fail$	$fail$	$fail$

We then get, after elimination of the $fail$:

$gen\_double \longrightarrow^*_{\mathcal{SIMPL}} dk(s1 \to s5, s1 \to s2, s1 \to s4, s3 \to s5, s3 \to s1, s4 \to s2, s4 \to s3)$
$\longrightarrow^{(19)} \{s1 \to s5, s1 \to s2, s1 \to s4, s3 \to s5, s3 \to s1, s4 \to s2, s4 \to s3\}$

– **Non termination of** $repeat^*(follow)$ –

Using the simplification rule (19), $follow$ is equivalent to $follow' = \{r12, r13, r25, r32, r34, r41\}$. Let us evaluate $TERMIN(repeat^*(follow'))$:

$TERMIN(repeat^*(follow')) \longrightarrow_{\mathcal{T}ermin} T(true, repeat^*(follow'))$
$\longrightarrow_{\mathcal{T}ermin} false$

for $follow'\downarrow_{\mathcal{LR}} = follow'$, which is not ε-terminating, because of the infinite ε-chain $s1 \longrightarrow s4 \longrightarrow s3 \longrightarrow s1 \longrightarrow \ldots$

Example 5 (detailed). Denoting S_2 the initial strategy of Example 2 where the rule $f(x) \to h(0, x)$ is replaced by the rule $f(x) \to h(0, x)$ if $x > 4$ and S_7 the first initial strategy of Example 7, the strategy studied in this example can be written

$$S = repeat^*(first(S_7, dk(S_2, f(x) \to f(f(x)) \text{ if } x > 5))).$$

From Examples 7 and 2, we deduce that S simplies with $\mathcal{SIMPL}$ in

$$S' = repeat^*(first(f(x) \to h(x) \text{ if } x > 3, dk(f(x) \to x \text{ if } x > 4, f(x) \to f(f(x)) \text{ if } x > 5))).$$

The above $first$ strategy, argument of the $repeat^*$ operator, simplifies in the strategy $first(f(x) \to h(x) \text{ if } x > 3, dk(fail, fail)))$ with rule (24). We then have the simplification $dk(fail, fail) \hookrightarrow^{(5)} dk(fail) \hookrightarrow^{(13)} fail$. Hence S' simplifies in

$$S'' = repeat^*(first(f(x) \to h(x) \text{ if } x > 3, fail)).$$

We finally have the following simplification :
$S'' \hookrightarrow^{(6)} repeat^*(first(f(x) \to h(x) \text{ if } x > 3))$
$\hookrightarrow^{(14)} repeat^*(f(x) \to h(x) \text{ if } x > 3)$
and we get that S simplifies in

$$S''' = repeat^*(f(x) \to h(x) \text{ if } x > 3).$$